MULTICS VIRTUAL MEMORY ANALYSIS AND METERING

STUDENT HANDBOOK
COURSE CODE F80A
COURSE DESCRIPTION

F80A Multics Virtual Memory Analysis and Metering

Duration: Five Days

Intended For: Personnel requiring analysis-level knowledge of Multics virtual memory implementation, metering and tuning. Especially useful for personnel concerned with maximizing system performance.

Synopsis: The Multics supervisor is logically divided into seven distinct subsystems. This course details the functions performed (and the data bases maintained) by five of these subsystems: Volume Management, Name and Address Space Management, Directory Control, Segment Control, and Page Control. Knowledge of the virtual memory implementation gives the student insight into the metering and tuning of the system. Other topics include overviews of the Multics system hardware, the Multics Supervisor, and a comparison with other memory management techniques. Question and answer periods are given daily to reinforce the material presented.

Objectives: Upon completion of this course, the student should be able to:

1. Understand the functions of the Multics supervisor subsystems, especially those subsystems which implement the Multics virtual memory.
2. Make optimal design choices when writing system applications to run in the Multics environment.
3. Evaluate and tune the system's performance by analyzing the system's virtual memory meters.

Prerequisites: Multics Subsystem Programming (F15D), Source Level Debugging & The Process Environment (F21) or equivalent experience.

Major Topics: System Hardware Components
Volume (Disk) Management
Name Space/Address Space Management
Directory, Segment and Page Control
Memory Management Techniques
Course Topic Map

to be

Inserted Here
## CONTENTS

### Topic I  Multics Design Philosophy
- Major Design Goals .................................. 1-1
- Virtual Memory Organization ....................... 1-3
- Selective, Controlled Sharing ..................... 1-5
- Security .............................................. 1-6
- Open-Ended, Modular System ....................... 1-8
- Decentralized Administration ...................... 1-9
- Flexible User Interfaces ......................... 1-11
- Continuous Operation ................................ 1-12
- Reliable File System ................................ 1-13
- Remote Access ........................................ 1-14
- Efficient Service to Large or Small Users ...... 1-15

### Topic II  Overview of the Operating System
- Structure of the Operating System ............... 2-1
- What is the Multics Supervisor .................... 2-3
- The Major Supervisor Subsystems ................ 2-4
- Name Space/Address Space Management ............ 2-6
- Directory Control .................................... 2-12
- Volume Management .................................. 2-14
- Segment Control ..................................... 2-17
- Page Control ........................................ 2-20
- Traffic Control ..................................... 2-23
- Fault and Interrupt Handling ..................... 2-27
- System Initialization ................................ 2-30
- File System Salvagers ................................ 2-33
- Metering & Tuning ................................... 2-36
- Initializer.SysDaemon ................................ 2-36

### Topic III  The Multics Environment
- What is a Process ................................... 3-1
- Cooperating Processes .............................. 3-4
- The PL/I Operators .................................. 3-7
- Interfaces to System Modules ..................... 3-9
- Deadlock Prevention ................................ 3-10
- Types of Locks ...................................... 3-15

### Topic IV  Name Space and Address Space Management
- Name/Address Space Overview ....................... 4-1
- Name/Address Space Terminology ................... 4-1
- Name/Address Space Concepts ....................... 4-5
- Name/Address Space Data Bases ..................... 4-7
- Reference Name Table (RNT) ......................... 4-12
- Known Segment Table (KST) ......................... 4-13
## CONTENTS (con't)

<table>
<thead>
<tr>
<th>Topic V</th>
<th>Directory Control</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Directory Control Overview</td>
<td>5-1</td>
</tr>
<tr>
<td></td>
<td>Directory Control Terminology</td>
<td>5-1</td>
</tr>
<tr>
<td></td>
<td>Directory Control Data Bases</td>
<td>5-3</td>
</tr>
<tr>
<td></td>
<td>Directory Segments</td>
<td>5-6</td>
</tr>
<tr>
<td></td>
<td>Directory Header</td>
<td>5-9</td>
</tr>
<tr>
<td></td>
<td>Directory Entries</td>
<td>5-11</td>
</tr>
<tr>
<td></td>
<td>Directory Control Commands</td>
<td>5-15</td>
</tr>
<tr>
<td></td>
<td>display_branch</td>
<td>5-15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic VI</th>
<th>Volume Management</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Volume Management Overview</td>
<td>6-1</td>
</tr>
<tr>
<td></td>
<td>The New Storage System</td>
<td>6-1</td>
</tr>
<tr>
<td></td>
<td>Volume Management Terminology</td>
<td>6-13</td>
</tr>
<tr>
<td></td>
<td>Volume Management Data Bases</td>
<td>6-15</td>
</tr>
<tr>
<td></td>
<td>Volume Label</td>
<td>6-15</td>
</tr>
<tr>
<td></td>
<td>Volume Map</td>
<td>6-19</td>
</tr>
<tr>
<td></td>
<td>Dumper Bit Map</td>
<td>6-22</td>
</tr>
<tr>
<td></td>
<td>VTOC Map</td>
<td>6-24</td>
</tr>
<tr>
<td></td>
<td>Physical Volume Table (PVT)</td>
<td>6-25</td>
</tr>
<tr>
<td></td>
<td>Logical Volume Table (LVT)</td>
<td>6-27</td>
</tr>
<tr>
<td></td>
<td>Physical Volume Hold Table</td>
<td>6-30</td>
</tr>
<tr>
<td></td>
<td>Volume Management Operations</td>
<td>6-32</td>
</tr>
<tr>
<td></td>
<td>Acceptance of Physical Volumes</td>
<td>6-32</td>
</tr>
<tr>
<td></td>
<td>Demounting of Physical Volumes</td>
<td>6-35</td>
</tr>
<tr>
<td></td>
<td>Logical Volume Management</td>
<td>6-37</td>
</tr>
<tr>
<td></td>
<td>Volume Management Commands</td>
<td>6-38</td>
</tr>
<tr>
<td></td>
<td>print_configuration_deck</td>
<td>6-38</td>
</tr>
<tr>
<td></td>
<td>list_vols</td>
<td>6-40</td>
</tr>
<tr>
<td></td>
<td>display_label</td>
<td>6-41</td>
</tr>
<tr>
<td></td>
<td>display_pvte</td>
<td>6-42</td>
</tr>
<tr>
<td></td>
<td>Volume Management Meters</td>
<td>6-43</td>
</tr>
<tr>
<td></td>
<td>disk_meters</td>
<td>6-43</td>
</tr>
<tr>
<td></td>
<td>device_meters</td>
<td>6-44</td>
</tr>
<tr>
<td></td>
<td>disk_queue</td>
<td>6-45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic VII</th>
<th>Segment Control</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Segment Control Overview</td>
<td>7-1</td>
</tr>
<tr>
<td></td>
<td>Segment Control Terminology</td>
<td>7-1</td>
</tr>
<tr>
<td></td>
<td>Segment Control Data Bases</td>
<td>7-4</td>
</tr>
<tr>
<td></td>
<td>Volume Table of Contents (VTOC)</td>
<td>7-5</td>
</tr>
<tr>
<td></td>
<td>Active Segment Table (AST)</td>
<td>7-11</td>
</tr>
</tbody>
</table>
## CONTENTS (con't)

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page Control</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Control Overview</td>
<td>6-1</td>
<td></td>
</tr>
<tr>
<td>Page Control Terminology</td>
<td>6-1</td>
<td></td>
</tr>
<tr>
<td>Page Control Data Bases</td>
<td>6-3</td>
<td></td>
</tr>
<tr>
<td>Page Tables</td>
<td>6-6</td>
<td></td>
</tr>
<tr>
<td>Core Map</td>
<td>6-6</td>
<td></td>
</tr>
<tr>
<td>System Segment Table (SST) Header</td>
<td>6-9</td>
<td></td>
</tr>
<tr>
<td>Other Data Bases</td>
<td>6-13</td>
<td></td>
</tr>
<tr>
<td>Services of Page Control</td>
<td>8-8</td>
<td></td>
</tr>
<tr>
<td>Page Fault Handling</td>
<td>8-9</td>
<td></td>
</tr>
<tr>
<td>Post Purging</td>
<td>8-14</td>
<td></td>
</tr>
<tr>
<td>Page Control Meters</td>
<td>8-25</td>
<td></td>
</tr>
<tr>
<td>file_system_meters</td>
<td>8-44</td>
<td></td>
</tr>
</tbody>
</table>

## Topic VIII

<table>
<thead>
<tr>
<th>Traffic Control</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Control Overview</td>
<td>9-1</td>
</tr>
<tr>
<td>Traffic Control Terminology</td>
<td>9-3</td>
</tr>
<tr>
<td>Traffic Control Data Bases</td>
<td>9-5</td>
</tr>
<tr>
<td>tc_data</td>
<td>9-5</td>
</tr>
<tr>
<td>Services of Traffic Control</td>
<td>9-6</td>
</tr>
<tr>
<td>Wait Locks</td>
<td>9-8</td>
</tr>
<tr>
<td>Processor Multiplexing</td>
<td>9-8</td>
</tr>
<tr>
<td>Traffic Control Meters</td>
<td>9-18</td>
</tr>
<tr>
<td>total_time_meters</td>
<td>9-18</td>
</tr>
<tr>
<td>traffic_control_meters</td>
<td>9-19</td>
</tr>
<tr>
<td>traffic_control_queue</td>
<td>9-21</td>
</tr>
<tr>
<td>work_class_meters</td>
<td>9-23</td>
</tr>
<tr>
<td>respons_meters</td>
<td>9-24</td>
</tr>
<tr>
<td>post_purge_meters</td>
<td>9-26</td>
</tr>
<tr>
<td>Traffic Control Commands</td>
<td>9-27</td>
</tr>
<tr>
<td>print_tuning_parameters</td>
<td>9-27</td>
</tr>
<tr>
<td>print_apt_entry</td>
<td>9-28</td>
</tr>
</tbody>
</table>
**CONTENTS (con't)**

<table>
<thead>
<tr>
<th>Topic X</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault and Interrupt Handling</td>
<td></td>
</tr>
<tr>
<td>Fault and Interrupt Handling Overview</td>
<td>10-1</td>
</tr>
<tr>
<td>Fault and Interrupt Data Bases</td>
<td></td>
</tr>
<tr>
<td>Fault and Interrupt Vectors</td>
<td>10-4</td>
</tr>
<tr>
<td>Fault Data Save Areas</td>
<td>10-6</td>
</tr>
<tr>
<td>Important Types of Faults</td>
<td>10-7</td>
</tr>
<tr>
<td>Fault/Interrupt Meters</td>
<td>10-12</td>
</tr>
<tr>
<td>sim_meters</td>
<td>10-12</td>
</tr>
<tr>
<td>interrupt_meters</td>
<td>10-13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic XI</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Initialization/Shutdown</td>
<td></td>
</tr>
<tr>
<td>System Initialization Overview</td>
<td>11-1</td>
</tr>
<tr>
<td>System Initialization Terminology</td>
<td>11-4</td>
</tr>
<tr>
<td>Initialization Data Bases</td>
<td>11-7</td>
</tr>
<tr>
<td>Environment Passed to Initialization</td>
<td>11-10</td>
</tr>
<tr>
<td>Collection 0</td>
<td>11-11</td>
</tr>
<tr>
<td>Collection 1</td>
<td>11-12</td>
</tr>
<tr>
<td>Collection 2</td>
<td>11-14</td>
</tr>
<tr>
<td>Collection 3</td>
<td>11-16</td>
</tr>
<tr>
<td>Normal Shutdown</td>
<td></td>
</tr>
<tr>
<td>File System Shutdown</td>
<td></td>
</tr>
<tr>
<td>Emergency Shutdown</td>
<td>11-18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic XII</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>File System Salvagers</td>
<td></td>
</tr>
<tr>
<td>Overview of Salvagers</td>
<td>12-1</td>
</tr>
<tr>
<td>Directory Salvager</td>
<td>12-1</td>
</tr>
<tr>
<td>Quota Salvaging</td>
<td>12-4</td>
</tr>
<tr>
<td>Physical Volume Scavenger</td>
<td>12-9</td>
</tr>
<tr>
<td>Physical Volume Salvager</td>
<td>12-11</td>
</tr>
<tr>
<td>sweep_pv</td>
<td>12-12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic XIII</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Initializer.SysDaemon Process</td>
<td>13-1</td>
</tr>
<tr>
<td>Initializer Overview</td>
<td></td>
</tr>
<tr>
<td>Services of Initializer</td>
<td>13-3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic XIV</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metering and Tuning</td>
<td></td>
</tr>
<tr>
<td>Meter and Tuning Overview</td>
<td>14-1</td>
</tr>
<tr>
<td>Analyzing Performance Problems</td>
<td>14-3</td>
</tr>
<tr>
<td>Detailed Problem Analysis</td>
<td>14-6</td>
</tr>
<tr>
<td>Output From Metering Commands</td>
<td>14-8</td>
</tr>
<tr>
<td>total_time_meters</td>
<td>14-8</td>
</tr>
<tr>
<td>interrupt_meters</td>
<td>14-9</td>
</tr>
<tr>
<td>file_system_meters</td>
<td>14-10</td>
</tr>
<tr>
<td>file_system_meters</td>
<td>14-12</td>
</tr>
<tr>
<td>device_meters</td>
<td>14-13</td>
</tr>
<tr>
<td>disk_meter</td>
<td>14-14</td>
</tr>
<tr>
<td>disk_queue</td>
<td>14-15</td>
</tr>
<tr>
<td>print_configuration_deck</td>
<td>14-16</td>
</tr>
<tr>
<td>list_vols</td>
<td>14-18</td>
</tr>
<tr>
<td>traffic_control_queue</td>
<td>14-19</td>
</tr>
<tr>
<td>traffic_control_meters</td>
<td>14-21</td>
</tr>
</tbody>
</table>
CONTENTS (con't)

print_tuning_parameters ... ... ... ... ... ... ... ... ... ... ... ... ... ... 14-23
work_class_meters ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 14-24
respons_meters ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 14-25
system_performance_graph ... ... ... ... ... ... ... ... ... ... ... ... 14-27
meter_gate ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 14-29

Topic XV Evolution of Memory Addressing/Management ... ... ... ... ... 15-1
Conventional Memory ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-1
  Structure ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-1
  Address Formation ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-1
  Characteristics ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-1
  Problems ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-4
Single Virtual Memory ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-7
  Structure ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-7
  Address Formation ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-7
  Characteristics ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-7
  Solved Problems ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-9
  Problems ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-10
Multiple Virtual Memories ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-11
  Structure ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-11
  Address Formation ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-11
  Characteristics ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-11
  Solved Problems ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-13
  Problems ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-14
Multics Virtual Memory ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-15
  Structure ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-15
  Address Formation ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-15
  Characteristics ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-15
  Solved Problems ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-17
  Problems ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... 15-19
TOPIC I

Multics Design Philosophy

Major Design Goals ........................................... 1-1
Virtual Memory Organization ................................ 1-3
Selective, Controlled Sharing .................................. 1-5
Security ........................................................................ 1-6
Open-Ended, Modular System .................................... 1-8
Decentralized Administration .................................... 1-9
Flexible User Interfaces ........................................... 1-11
Continuous Operation .............................................. 1-12
Reliable File System ............................................... 1-13
Remote Access ....................................................... 1-14
Efficient Service to Large or Small Users .................. 1-15
MAJOR DESIGN GOALS

1. MULTIPLEXED INFORMATION AND COMPUTING SERVICE (MULTICS)

2. MULTICS WAS ONE OF THE FIRST OPERATING SYSTEMS TO BE THOROUGHLY DESIGNED FROM THE TOP DOWN. THE DEVELOPERS:

   • STARTED WITH A SET OF GOALS
   • CREATED A SYSTEM WHICH WOULD SATISFY THESE GOALS
   • DEVELOPED GENERAL SOLUTIONS INSTEAD OF SPECIFIC SOLUTIONS (MAKING THE PRODUCT EXTENDABLE)
   • PRODUCED A Viable AND MARKETABLE PRODUCT
MAJOR DESIGN GOALS

These goals were carefully chosen to characterize a 'utility-grade' computer system, and are outlined below:

- Virtual Memory Organization
- Selective, Controlled Sharing
- Security
- Open-Ended, Modular System
- Decentralized Administration
- Flexible User Interfaces, End-User Orientation
- Continuous Operation
- Reliable File System
- Remote Access
- Efficient Service to Large and Small User
VIRTUAL MEMORY ORGANIZATION

* MOTIVATION

1. INFORMATION STORED ON-LINE IN LARGE INFORMATION UTILITIES OFTEN EXCEEDS THE SIZE OF AVAILABLE MAIN MEMORY

2. THIS INFORMATION SHOULD BE DIRECTLY (AND CONTINUOUSLY) ACCESSIBLE BY THE USER COMMUNITY

3. THE SIZE OF MAIN MEMORY SHOULD ONLY AFFECT PROCESSING TIME, NOT PROCESSING CAPABILITY

4. MAIN MEMORY MANAGEMENT SHOULD BE A TASK FOR THE OPERATING SYSTEM, NOT THE PROGRAMMER

* IMPLEMENTATION

5. ALL ON-LINE INFORMATION IS PROCESSOR ADDRESSABLE

6. ALL INFORMATION (PROCEDURE AND DATA) IS COMPARTMENTALIZED INTO UNITS CALLED "SEGMENTS" ALLOWING THE ASSOCIATION OF ATTRIBUTES WITH EACH SEGMENT(1)

---

(1) THROUGHOUT THIS DOCUMENT, GENERIC REFERENCES TO "SEGMENTS" INCLUDE DIRECTORIES AS WELL, SINCE DIRECTORIES ARE SIMPLY A SPECIAL KIND OF SEGMENT

Not To Be Reproduced
VIRTUAL MEMORY ORGANIZATION

- SEGMENTS ARE MADE PROCESS ADDRESSABLE AS THEY ARE REFERENCED

- ALL SEGMENTS ARE DIVIDED INTO AN INTEGRAL NUMBER OF 1024 WORD PAGES. THESE PAGES ARE BROUGHT INTO MAIN MEMORY IF AND ONLY IF THEY ARE REFERENCED (NEEDED) - AT THE TIME THEY ARE REFERENCED BY ANY PROCESS

- THE MULTICS HARDWARE INTERPRETS ALL ADDRESSES AS OFFSETS WITHIN A SPECIFIED SEGMENT (SEGNO OFFSET)

- THE HARDWARE MAKES NO DISTINCTION BETWEEN PROCEDURE AND DATA SEGMENTS. BOTH ARE PAGED IN THE SAME MANNER, BOTH ARE ADDRESSED IN THE SAME MANNER

- ALL COMPILERS PRODUCE LOAD MODULES - NO MODIFICATION IS REQUIRED TO EXECUTE PROCEDURE CODE
SELECTIVE, CONTROLLED SHARING

MOTIVATION

- USERS SHOULD BE ABLE TO USE COMMON PROCEDURE AND DATA SEGMENTS DIRECTLY (NOT COPIES)
- USERS SHOULD BE ABLE TO SHARE PRIVATE CODE IN A SELECTIVE MANNER

IMPLEMENTATION

- PURE, REENTRANT CODE IS ALWAYS GENERATED BY THE COMPILERS (ALLOWING SHARING OF PROCEDURE CODE IN A MULTI-PROCESS ENVIRONMENT)
- EVERYTHING THE USER TOUCHES (EXECUTE OR REFERENCE) WILL BE A SEGMENT HAVING ITS OWN ATTRIBUTES
- THE ACCESS ATTRIBUTES OF EACH SEGMENT ARE ESTABLISHED BY THE OWNER OF THAT SEGMENT
SECURITY

2 MOTIVATION

- In an environment of several coexisting processes, users must be protected from accidentally or intentionally interfering with each other.
- The supervisor must be protected from damage by users.
- Changes in access to information must be immediately effective.
- Disclosure of information should be allowed in a selective and controlled manner (versus all-or-none approach).
- Unwarranted denial of access to information must be prohibited.

2 IMPLEMENTATION

- Per-segment access control list (ACL) - specifying by whom and how the segment may be accessed.
- Ring protection mechanism - isolates segments and processes.
- Access isolation mechanism (AIM) - isolates segments according to categories and security levels.
- Passwords and audit trails.
SECURITY

USER

ACL

AIM

RING

SEGMENT

Jones.ProjA.a
Top Secret/NATO
Ring 4

rw *.ProjA.*
Secret/Air Force
Ring Brackets 1,7,7
OPEN-ENDED, MODULAR SYSTEM

2 MOTIVATION

- SOFTWARE SHOULD BE EASY TO MODIFY AND EXTEND

- THE OPERATING SYSTEM SHOULD BE MODULAR, AND THE MODULES SHOULD BE COMPREHENSIBLE

2 IMPLEMENTATION

- MODULAR DESIGN OF OPERATING SYSTEM AND USER PROGRAMS (COMPILERS, PROGRAMMING ENVIRONMENT ENCOURAGE MODULAR DESIGN)

- UNIFORM PROGRAMMING CONVENTIONS ARE FOLLOWED THROUGHOUT MOST SYSTEM CODE

- MORE THAN 92% OF THE OPERATING SYSTEM OBJECT CODE ORIGINATED FROM PL/I SOURCE

- DYNAMIC LINKING (ELIMINATES RE-COMPILING, RE-EDITING WHEN UN-BOUND MODULES ARE REPLACED)

- ON-LINE MODIFICATION, TESTING AND INSTALLATION OF SYSTEM MODULES (MULTICS INSTALLATION FACILITY - MIS)

- PATCH-FREE SYSTEM

- LIBRARY MANAGEMENT TOOLS AND LIBRARY CONVENTIONS
DECENTRALIZED ADMINISTRATION

MOTIVATION

- System resources must be effectively administered
- Resource allocation, accounting, registration, billing, etc., is too much for single individual
- Different groups of users have different administrative needs

IMPLEMENTATION

- Grouping of users, by function or management, into projects
- Three-level hierarchy of administration
  - System administrator: distributes resources and assigns attributes to projects
  - Project administrators: distributes resources and assigns attributes to users
  - Users: has full control over allocated resources, modified by assigned attributes
  - The project administrator may pass downward only those resources and attributes that have been given to the project
PROCESS ATTRIBUTES

SYSTEM ADMINISTRATION TABLE
(> sat1 > sat)
CREATED AND MAINTAINED
BY SYSTEM ADMINISTRATOR

PROCESS INITIALIZATION TABLE
([pd] > ptt)
ANSWERING SERVICE
TABLE ENTRY
(ANSWER_TABLE,
ABSENTEE_USER_TABLE,
DAEMON_USER_TABLE)
CREATED AT LOGIN
DESTROYED AT LOGOUT

PROJECT DEFINITION TABLE
(> sc1 > pdt> ABC.pdt)
CREATED AND MAINTAINED
BY PROJECT ADMINISTRATOR
FLEXIBLE USER INTERFACES

MOTIVATION

1. THE STANDARD USER ENVIRONMENT SHOULD BE EXTENSIVELY USER-MODIFIABLE

2. THE CAPABILITY SHOULD EXIST TO DEVELOP AND IMPOSE CLOSED SUBSYSTEMS WHICH CAN PROVIDE ANY DESIRED ENVIRONMENT

IMPLEMENTATION

1. USER HAS ABILITY TO CHANGE OR REPLACE CONTROL PROGRAMS IN THE USER’S RING

2. PROJECT ADMINISTRATOR CAN IMPOSE A CLOSED SUBSYSTEM ENVIRONMENT OR A DIFFERENT process_ overseer ON USERS

3. start_up.ec, ABBREV PROCESSOR, general_ready, ready_off, add_search_rules, CONDITION HANDLING, ETC.

4. OTHER TOOLS PROVIDE SIMULATION, ENCAPSULATION CAPABILITY (enter_lss, project_start_up_)

5. STANDARD INTERFACE FOR INTERACTIVE SUBSYSTEMS (ssu_) ENCOURAGES UNIFORM, FAMILIAR BEHAVIOUR OF USER SUBSYSTEMS.
CONTINUOUS OPERATION

§ MOTIVATION

§ UTILITY CONCEPT: SYSTEM SHOULD BE AVAILABLE ON DEMAND AT ALL TIMES

§ IMPLEMENTATION

§ ON-LINE SOFTWARE INSTALLATION

§ ON-LINE MAINTENANCE: MOVE MORE AND MORE BOS CAPABILITY INTO MULTICS (EG: RE-BOOT FNP FROM MULTICS)

§ ON-LINE FILE BACKUP AND RECOVERY

§ ON-LINE ACCOUNTING AND BILLING

§ DYNAMIC RECONFIGURATION

§ DYNAMIC FAILSOFT DECONFIGURATION OF FAILING HARDWARE

§ UNATTENDED SERVICE

§ AUTOMATIC REBOOT
RELIABLE FILE SYSTEM

* MOTIVATION

- MUST PROVIDE USERS SOME ASSURANCE THAT THEIR ON-LINE INFORMATION IS SAFE
- MUST PROVIDE CHECKPOINT CAPABILITY FOR RECOVERY FROM USER ERROR OR SYSTEM DISASTER

* IMPLEMENTATION

- AUTOMATIC BACKUP/RETRIEVAL FACILITY
- CONSOLIDATED STORAGE SYSTEM DUMPS
- PHYSICAL AND LOGICAL SAVE/RESTORE
- ALL STORAGE SYSTEM RECOVERY PROCEDURES RUN WHILE SYSTEM IS UP
- DAMAGE RECOVERY RUN AUTOMATICALLY FOLLOWING SYSTEM FAILURE
REMOTE ACCESS

• MOTIVATION

[ ] UTILITY CONCEPT: FULL ACCESS FROM ANY PHONE IN THE WORLD VIA ANY REMOTE DEVICE

[ ] WISH TO PROVIDE ONE "COMMAND LANGUAGE", TO SERVE ALL USERS, WHETHER LOCAL OR REMOTE, INTERACTIVE OR BATCH

• IMPLEMENTATION

[ ] MULTICS COMMUNICATION SYSTEM (MCS)

[ ] IN PRINCIPLE, ANY REMOTE DEVICE/TERMINAL IS CONNECTABLE

[ ] SINGLE COMMAND LANGUAGE

[ ] REMOTE JOB ENTRY (RJE) AND BULK I/O CAPABILITIES

[ ] DIRECT ATTACHMENTS TO PUBLIC DATA NETWORKS VIA X.25
EFFICIENT SERVICE TO LARGE OR SMALL USERS

MOTIVATION

- Utility concept: System should be available for, and capable of, any size task

- Running both large and small tasks together should not impact the efficiency of either

IMPLEMENTATION

- Dynamic resource allocation (don't have to pre-allocate or guess-timate resources required)

- Service on demand

- Dynamic system tuning to accommodate changing system workloads
STRUCTURE OF THE OPERATING SYSTEM

I SUBROUTINES (550)

I DESCRIBED IN THE MPM MANUALS "Multics Subroutines" (AG93) AND "Subsystems Writer's Guide" (OFTEN ABBREVIATED AS "SWG") (AK92)

I TOOLS (220)

I DESCRIBED IN THE MPM MANUAL "Multics Commands and Active Functions" (AG92)

I ADMINISTRATIVE ROUTINES (200)

I DESCRIBED IN THE MAM MANUALS "System Administrator" (AK50), "Registration & Accounting Administrator" (AS68), "Project Administrator" (AK51)

I OPERATOR COMMANDS (150)

I DESCRIBED IN THE MANUAL "Operator's Handbook" (AM81)
**WHAT IS THE MULTICS SUPERVISOR**

WHAT IS THE MULTICS SUPERVISOR?

A COLLECTION OF MANY LOGICAL SUBSYSTEMS WHICH IMPLEMENT THE FUNCTIONS OF MULTICS

THE PRIMARY PURPOSE OF MULTICS IS TO RUN PROGRAMS, WHICH ACCESS DATA, AND THUS THE MAJOR PURPOSE OF THE MULTICS SUPERVISOR IS TO MAKE THAT DATA ACCESSIBLE

THESE SUBSYSTEMS FALL INTO FOUR MAJOR GROUPS:

- THE FILE SYSTEM
- SUPPORT SERVICES FOR THE FILE SYSTEM
- MISCELLANEOUS SUPERVISOR SERVICES
- SUBSYSTEMS RELATED TO, BUT NOT STRICTLY PART OF THE SUPERVISOR

THESE DIVISIONS ARE SOMEWHAT ARTIFICIAL, BECAUSE THE SUBSYSTEMS ARE ALL INTIMATELY RELATED TO EACH OTHER. THE DIVISIONS REPRESENT A PARTICULAR VIEWPOINT OF SYSTEM FUNCTION.

A MULTICS SUBSYSTEM IS A SET OF PROGRAMS PERFORMING A SPECIFIC SERVICE FOR THE USER COMMUNITY - AND FOR THE OPERATING SYSTEM ITSELF

TOGETHER, ALL THESE SUBSYSTEMS IMPLEMENT THE FUNCTIONS DESCRIBED IN THE MPM SUBROUTINES AND SWG MANUALS, (ESSENTIALLY hcs_ AND THE VIRTUALS MEMORY).
THE MAJOR SUPERVISOR SUBSYSTEMS

MAJOR MULTICS SUPERVISOR SUBSYSTEMS: FOUR GROUPS OF ABOUT FOUR SUBSYSTEMS EACH

THE FILE SYSTEM - THOSE SUBSYSTEMS WHICH ARE CONCERNED WITH STORING DATA, MANAGING DATA, AND MAKING IT AVAILABLE TO USERS. FIVE MAJOR COMPONENTS:

- NAME SPACE / ADDRESS SPACE CONTROL
- DIRECTORY CONTROL
- VOLUME MANAGEMENT
- SEGMENT CONTROL
- PAGE CONTROL

SERVICES TO SUPPORT THE FILE SYSTEM, WHICH MULTIPLEX ITS FACILITIES BETWEEN DIFFERENT USERS, AND ENSURE ITS RELIABILITY. FOUR MAJOR COMPONENTS:

- TRAFFIC CONTROL
- FAULT AND INTERRUPT HANDLING
- SYSTEM INITIALIZATION
- THE FILE SYSTEM SALVAGERS
THE MAJOR SUPERVISOR SUBSYSTEMS

MISCELLANEOUS SUPERVISOR SERVICES - THESE ARE THINGS DONE IN THE SUPERVISOR FOR REASONS OF ACCESS CONTROL AND SHARING, BUT NOT DIRECTLY RELATED TO THE FILE SYSTEM.

BECAUSE THEY ARE NOT DIRECTLY RELATED, THEY WILL NOT BE COVERED IN ANY DETAIL.

MULTICS COMMUNICATIONS SYSTEM

RESOURCE CONTROL

USER DEVICE I/O - ioi_

LOW LEVEL SUPERVISOR I/O

RECONFIGURATION

SYSTEM ERROR HANDLING (syserr / verify_lock)

RELATED SUBSYSTEMS - THESE ARE NOT ACTUALLY PART OF THE SUPERVISOR, BUT ARE CLOSELY RELATED

METERING AND TUNING

THE Initializer.SysDaemon

THE MULTICS SUPERVISOR IS DESIGNED AROUND THE "LAYERED MACHINE" CONCEPT.
THE MAJOR SUPERVISOR SUBSYSTEMS

- Construct a simple set of operations called a "kernel" which implements the most fundamental (primitive) operations required.

- Construct a slightly more sophisticated set of operations which assumes and relies on the correct functioning of the kernel — another "layer".

- Construct a more sophisticated layer which assumes and relies on the correct functioning of the previous machines.

- Etc.

- The "layers" of the Multics supervisor partially map into the above subsystems.

- The following diagram represents this mapping:
THE MAJOR SUPERVISOR SUBSYSTEMS

THE MULTICS SUPERVISOR

COMPONENTS ARE ASYNCHRONOUSLY INVOKED
NAME SPACE/ADDRESS SPACE MANAGEMENT

FUNCTION

IMPLEMENT THE PER PROCESS VIRTUAL MEMORY

BASIC PHILOSOPHY

AS A NEWLY LOGGED IN USER ATTEMPTS TO TOUCH VARIOUS SEGMENTS A CONSIDERABLE AMOUNT OF MANAGEMENT INFORMATION MUST BE (TRANSPARENTLY) FOUND AND/OR COMPUTED BEFORE THE USER'S REFERENCE IS ACTUALLY ACCOMPLISHED

FOR EVERY SEGMENT REFERENCED BY THE USER, THE SUPERVISOR:

ASSIGNS A SEGMENT NUMBER (FOR REASON OF HARDWARE ADDRESSING), AND

RECORDS (REMEMBERS) THE MANAGEMENT INFORMATION (FOR REASON OF SOFTWARE EFFICIENCY AND CONTROL)

SUCH SEGMENTS ARE SAID TO BE "KNOWN TO THE PROCESS"

THE MANAGEMENT INFORMATION IS MAINTAINED ON A PER PROCESS BASIS IN THREE COMPLEMENTING AREAS: DSEG, KST, AND RNT
NAME SPACE/ADDRESS SPACE MANAGEMENT

MANAGES TWO DISTINCT SETS OF INFORMATION:

ADDRESS SPACE - CORRESPONDENCE BETWEEN SEGMENT NUMBERS AND THE SEGMENTS THEMSELVES

NAME SPACE - CORRESPONDENCE BETWEEN SEGMENT NUMBERS AND NAMES THE USER REFERS TO THEM BY

CALLS DIRECTORY CONTROL TO LOCATE SEGMENTS INITIALLY

NAME SPACE / ADDRESS SPACE MANAGEMENT IS INVOKED BY SUBROUTINE CALLS, AND BY LINKAGE FAULTS (THE "DYNAMIC LINKER")

PRINCIPAL USER INTERFACES

COMMAND LEVEL

initiate, terminate, terminate_segment, terminate_ref_name, terminate_single_ref_name, list_ref_name

THE COMMAND PROCESSOR ITSELF - WHICH USES THESE SERVICES TO LOCATE COMMANDS

SUBROUTINE LEVEL

hcs$INITIATE, hcs$nINITIATE_COUNT, hcs$TERMINATE_FILE, hcs$TERMINATE_SEGMENT, hcs$TERMINATE_NAME, hcs$TERMINATE_NONAME, term

Not To Be Reproduced 2-9 F80A
NAME SPACE/ADDRESS SPACE MANAGEMENT

MAJOR DATA BASES

- DESCRIPTOR SEGMENT (DSEG) - ONE PER PROCESS

  - SEGMENT DESCRIPTOR WORD (SDW) - ONE PER KNOWN SEGMENT
  - DEFINES THE USER'S ADDRESS SPACE TO THE HARDWARE

- KNOWN SEGMENT TABLE (KST) - ONE PER PROCESS

  - KNOWN SEGMENT TABLE ENTRY (KSTE) - ONE PER KNOWN SEGMENT (EXCEPT SUPERVISOR SEGMENTS)
  - DEFINES THE USER'S ADDRESS SPACE TO THE SUPERVISOR AND THE USER
  - EACH KSTE ASSOCIATES A USER'S SEGMENT NUMBER WITH THE SEGMENT CONTROL ATTRIBUTES OF THAT SEGMENT
  - THE SEARCH FOR AN AVAILABLE KSTE DETERMINES A SEGMENT'S NUMBER

- REFERENCE NAME TABLE (RNT) - ONE PER EACH RING IN EACH PROCESS

  - NOT A SEGMENT - KEPT AS A REGION ALLOCATED IN THE "LINKAGE AREA" FOR EACH RING
  - REFERENCE NAME TABLE ENTRY (RNTE) - ONE PER REFERENCE NAME
  - USED BY THE DYNAMIC LINKER TO IMPLEMENT THE "initiated_segments" SEARCH RULE
NAME SPACE/ADDRESS SPACE MANAGEMENT

| DEFINES THE USER'S NAME SPACE TO THE USER |
| NAME SPACE MAY BE DIFFERENT IN DIFFERENT RINGS OF THE SAME PROCESS |
DIRECTORY CONTROL

FUNCTION

DIRECTORY CONTROL IS A SET OF HARDCORE MODULES RESPONSIBLE FOR THE MAINTENANCE OF THE MULTICS DIRECTORY STRUCTURE -- IE: THE HIERARCHY

ITS TASKS INCLUDE CREATING, MANIPULATING AND INTERPRETING THE CONTENTS OF DIRECTORY SEGMENTS, TO INCLUDE:

ACCESS CONTROL Lists (ACL'S), NAMES, AND VTOCE POINTERS OF ENTRIES DESCRIBED THEREIN

ONLY DIRECTORY CONTROL IS ALLOWED TO ALTER THE CONTENTS OF DIRECTORY SEGMENTS

DIRECTORY CONTROL IMPICITLY RELIES UPON THE SERVICES OF OTHER SUBSYSTEMS SUCH AS SEGMENT CONTROL AND PAGE CONTROL, AND ALSO INVOKES THEM DIRECTLY BY SUBROUTINE CALL

DIRECTORIES ARE SIMPLY SEGMENTS TO THESE SUBSYSTEMS

DIRECTORY CONTROL IS INVOKED ONLY BY SUBROUTINE CALLS

PRINCIPAL USER INTERFACES

COMMAND LEVEL
DIRECTORY CONTROL

create, create_dir, link, set_acl, delete_acl, status, list,
add_name, rename

SUBROUTINE LEVEL

hcs_sappend_branch, hcs_sadd_acl_entries, hcs_sappend_link,
hcs_sdelete_acl_entries, hcs_sstatus_, hcs_schname_file

MAJOR DATA BASES

DIRECTORY SEGMENTS

CONTAIN THE ATTRIBUTES AND OTHER INFORMATION ABOUT THEIR
SEGMENTS (NEEDED TO FIND SEGMENTS, RETURN STATUS INFORMATION,
AND BUILD VTOCE'S AT SEGMENT CREATION)

THE DIRLOCKT_SEG

SEGMENT WHERE DIRECTORY LOCKING IS MANAGED
VOLUME MANAGEMENT

FUNCTION

VOLUME MANAGEMENT IS RESPONSIBLE FOR THE MANAGEMENT OF PHYSICAL AND LOGICAL VOLUMES

ITS TASKS INCLUDE:

- ACCEPTANCE AND DEMOUNTING OF PHYSICAL VOLUMES

- MAINTAINING THE ASSOCIATION BETWEEN PHYSICAL VOLUMES, LOGICAL VOLUMES, AND DISK DRIVES

- ENSURING THE INTEGRITY OF VOLUME CONTENTS

- MAKING VOLUME CONTENTS ACCESSIBLE TO PAGE CONTROL (PAGES) AND SEGMENT CONTROL (VTOC ENTRIES)

VOLUME MANAGEMENT IS INVOKED ONLY BY SUBROUTINE CALLS
MAJOR DATA BASES

PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM

PHYSICAL VOLUME TABLE ENTRY (PVTE) - ONE PER DISK DRIVE KNOWN TO THE SYSTEM

EACH PVTE IDENTIFIES A DRIVE'S DEVICE NUMBER, SUBSYSTEM NAME, DEVICE TYPE, AND INFORMATION ABOUT THE PHYSICAL VOLUME CURRENTLY MOUNTED

USED TO MAP REFERENCES TO PAGES OF SEGMENTS INTO AN I/O REQUEST TO THE CORRECT DISK DRIVE

LOGICAL VOLUME TABLE (LVT) - ONE PER SYSTEM

LOGICAL VOLUME TABLE ENTRY (LVTE) - ONE PER MOUNTED LOGICAL VOLUME

EACH LVTE CONTAINS THE LOGICAL VOLUME ID, POINTERS TO MEMBER PVTE'S, AIM CLASS LIMITS, ETC.

USED TO DETERMINE A USER'S ACCESS TO A LOGICAL VOLUME (PRIVATE OR PUBLIC) AND TO LOCATE MEMBER PHYSICAL VOLUMES

VOLUME HEADER - ONE PER PACK

VOLUME LABEL (REGISTRATION AND ACCEPTANCE INFORMATION)

VOLUME MAP (OCCUPIED/VACANT INFORMATION FOR VOLUME CONTENTS)

Not To Be Reproduced 2-15 F80A
VOLUME MANAGEMENT

RECORD STOCKS - ONE PER MOUNTED VOLUME

ONLINE CACHE OF INFORMATION ABOUT USED / UNUSED RECORDS ON THE VOLUME

THIS INFORMATION IS DERIVED FROM THE VOLUME MAP, BUT KEPT ONLINE TO AVOID THE NECESSITY OF REFERRING TO THE VOLUME MAP ON DISK EVERY TIME A RECORD IS ALLOCATED OR FREED

WHEN THE CACHE BECOMES COMPLETELY EMPTY OR COMPLETELY FULL, IT MUST BE UPDATED FROM/TO DISK - A PROTOCOL ENSURES THAT THE COPY ON DISK IS ALWAYS CONSISTENT

PROVIDED BY VOLUME MANAGEMENT, BUT USED BY PAGE CONTROL

VTOC STOCKS - ONE PER VOLUME

SIMILAR TO RECORD STOCKS, BUT MAINTAINS INFORMATION ABOUT USED / UNUSED VTOC ENTRIES ON THE VOLUME

PROVIDED BY VOLUME MANAGEMENT, BUT USED BY SEGMENT CONTROL

PHYSICAL VOLUME HOLD TABLE (PVHT) - ONE PER SYSTEM

RECORDS THE COMMENCEMENT OF COMPOUND I/O OPERATIONS UPON A PHYSICAL VOLUME

THIS INFORMATION PREVENTS A VOLUME FROM BEING DEMOUNTED WHILE SUCH AN OPERATION IS IN PROGRESS
SEGMENT CONTROL

FUNCTION

SEGMENT CONTROL IS RESPONSIBLE FOR THE MANAGEMENT OF LOGICAL MEMORY

ITS TASKS INCLUDE:

- MAINTAINING THE DISK RESIDENT MAPS OF SEGMENTS (IE: THEIR VTOCE'S)
- SEGMENT CREATION, TRUNCATION AND DELETION
- SEGMENT ACTIVATION AND DEACTIVATION (ASTE MULTIPLYING)
- SEGMENT CONTROL CAN BE INVOKED EITHER BY SUBROUTINE CALLS OR BY SEGMENT FAULTS

BASIC PHILOSOPHY OF ACTIVATION/DEACTIVATION

OF ALL SEGMENTS RESIDENT WITHIN THE SYSTEM'S MOUNTED PHYSICAL VOLUMES, ONLY A SMALL SUBSET WILL REQUIRE ACCESSING AT ANY ONE TIME. SUCH SEGMENTS WILL BE CALLED "ACTIVE SEGMENTS"

A PART OF MAIN MEMORY, CALLED THE "ACTIVE SEGMENT TABLE" (AST), WILL BE RESERVED TO HOLD MANAGEMENT INFORMATION FOR THESE ACTIVE SEGMENTS (IDENTITY, PVT INDEX, LOCATION OF PAGES, ETC.)
SEGMENT CONTROL

As segments fall into disuse, their "management information" in the AST will be replaced with information of other segments requiring activation.

USER INTERFACE

COMMAND LEVEL

create, delete, truncate, etc.

SUBROUTINE LEVEL

hcs_sappend_branch, hcs_sappend_branchx, hcs_sdelentry_seg, hcs_sdtentry_file, hcs_struncate_seg, hcs_struncate_file, hcs_sforce_write, etc

MAJOR DATA BASES

SYSTEM SEGMENT TABLE (SST) - one per system, shared with page control. One major component is "owned" by segment control:

ACTIVE SEGMENT TABLE (AST) - one per system

The AST is a list of active (currently being used) segments

ACTIVE SEGMENT TABLE ENTRY (ASTE) - one per active segment

Astes contain physical volume id's (PVID's) and VTOC index's (VTOCX's) of segments. Needed by segment control to find the segment on disk (hardware)
SEGMENT CONTROL

AST HASH TABLE

Allows efficient searching of AST's

Logically part of the AST, but elsewhere for historical reasons

DIRECTORY SEGMENTS

Contain locations and attributes of segments. Location information from directory segments is provided to segment control by directory control

VOLUME TABLE OF CONTENTS (VTOC) - ONE PER PHYSICAL VOLUME

VOLUME TABLE OF CONTENTS ENTRY (VTOCE) - ONE PER DISK-RESIDENT SEGMENT

Each VTOCE contains the segment's unique ID, current length, file map, etc (need to build ASTE'S and PT'S)

VTOCES are read and written only by segment control

VTOCE STOCKS - FROM VOLUME MANAGEMENT

Used when creating and deleting VTOCES for segments
PAGE CONTROL

FUNCTION

PAGE CONTROL is responsible for the management of physical memory to include the multiplexing of main memory frames, and the management of disk storage.

Its tasks include:

- Transferring the pages of segments between the memory devices, and recording the location of "the" copy of these pages.
- Reporting the status and file maps of segments to segment control.

Page control is largely coded in Multics Assembler Language (ALM).

Page control can be invoked either by subroutine calls or by page faults.

There are no explicit user interfaces to page control.
PAGE CONTROL

* BASIC PHILOSOPHY

- OF ALL THE SEGMENTS ACTIVE AT A GIVEN TIME, ONLY A SMALL SUBSET OF THEIR TOTAL PAGES WILL BE REQUIRED FOR ACCESSING.

- PAGES WILL BE READ INTO MAIN MEMORY AS THEY ARE REQUIRED.

- THE READING OF A PAGE INTO MAIN MEMORY WILL (PROBABLY) REQUIRE THE EVICTION OF A PREVIOUSLY REQUIRED PAGE.

- THE CHOICE OF A PAGE FOR EVICTION WILL BE BASED ESSENTIALLY UPON A "LEAST RECENTLY USED" CRITERIA.

- AN EVICTED PAGE NEED BE WRITTEN BACK TO DISK ONLY IF IT WAS MODIFIED DURING ITS RESIDENCY IN MAIN MEMORY.

* MAJOR DATA BASES

- PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM. PROVIDED BY VOLUME MANAGEMENT.

- PHYSICAL VOLUME TABLE ENTRY (PVTE) - ONE PER DISK DRIVE CONFIGURED.

- EACH PVTE CONTAINS:

  - THE DEVICE ID (DISK DRIVE ID) AND THE ID OF THE PHYSICAL VOLUME (DISK PACK) CURRENTLY MOUNTED.
PAGE CONTROL

- The number of records left unallocated on the physical volume, pointer to the record stock, etc.

- Record stocks - one per mounted physical volume, provided by volume management
  - Contains an in-memory cache of the in-use status of records on the volume, from the volume map, used when allocating or freeing pages
  - Accessed by a complex mechanism which uses normal page I/O but has a protocol to ensure synchronization of disk contents and record stock contents

- System segment table (SST) - one per system, shared with segment control. Contains the following five data bases used by page control:
  - System segment table (SST) header - one per system
    - Contains a large number of counters and pointers vital to the maintenance and metering of the storage system
    - Contains lockwords used to synchronize page control and segment control operations
  - Core map - the core_map segment - one per system
    - Core map entry (CME) - one per frame (1024 words) of configured main memory
    - Each CME represents a frame of main memory and identifies the current occupant of that frame
PAGE CONTROL

NOT PART OF THE SST SEGMENT ANY MORE, BUT LOGICALLY PART OF THE SST

ACTIVE SEGMENT TABLE (AST) - ONE PER SYSTEM

ACTIVE SEGMENT TABLE ENTRY (ASTE) - ONE PER ACTIVE SEGMENT

LIST OF ACTIVE (CURRENTLY BEING USED) SEGMENTS

PAGE TABLES (PT) - ONE PER ACTIVE SEGMENT, THE OTHER HALF OF EACH ASTE

PAGE TABLE WORD (PAGE PTW) - EITHER 4, 16, 64, OR 256 PER PAGE TABLE

EACH PTW DEFINES THE CURRENT LOCATION OF A PAGE OF THE SEGMENT: DISK, MAIN MEMORY ADDRESS, OR NULL
THE MULTICS FILE SYSTEM

DISK DRIVE
DISK PACK
DISK DRIVE
DISK PACK
DISK DRIVE
DISK PACK
...
FUNCTION

TRAFFIC CONTROL (OR THE "TRAFFIC CONTROLLER") IS RESPONSIBLE FOR MANAGING THE ASSIGNMENT OF PHYSICAL PROCESSORS TO MULTICS PROCESSES AND IMPLEMENTING THE SYSTEM'S WAIT/NOTIFY AND INTERPROCESS COMMUNICATION PRIMITIVES.

THE FUNCTIONS ASSUMED BY THE TRAFFIC CONTROLLER ARE KNOWN AS MULTIPROGRAMMING, MULTIPROCESSING, SCHEDULING, DISPATCHING, PROCESSOR MANAGEMENT, AND INTERPROCESS COMMUNICATION.

ITS MAJOR FUNCTION IS ALLOWING PROCESSES TO AWAIT THE COMPLETION OF FILE SYSTEM OPERATIONS, SUCH AS PAGE I/O.

TRAFFIC CONTROL CAN BE INVOKED BY SUBROUTINE CALLS AND INTERRUPTS.

THERE ARE NO IMPORTANT USER SUBROUTINE INTERFACES, BUT THERE ARE PRIVILEGED SUBROUTINE INTERFACES FOR PROCESS CREATION, ADJUSTMENT OF SCHEDULING PARAMETERS, ETC.

MAJOR DATA BASES

TC_DATA SEGMENT - ONE PER SYSTEM. CONTAINS THE FOLLOWING FOUR DATA BASES:

TC_DATA HEADER - ONE PER SYSTEM

CONTAINS VARIOUS METERS, COUNTERS AND POINTERS USED BY THE TRAFFIC CONTROLLER.
TRAFFIC CONTROL

ACTIVE PROCESS TABLE (APT) - ONE PER SYSTEM

ACTIVE PROCESS TABLE ENTRY (APTE) - ONE OCCUPIED PER ACTIVE PROCESS (TOTAL NUMBER IS DETERMINED BY CONFIG DECK)

EACH APTE CONTAINS VARIOUS ATTRIBUTES OF AN ACTIVE PROCESS INCLUDING THE PROCESS ID, STATE, THE VALUE OF ITS DESCRIPTOR BASE REGISTER (DBR), SCHEDULING PARAMETERS, AND A POINTER TO THE PROCESS'S ITT ENTRIES

THE APTE CONTAINS ALL INFORMATION THE SUPERVISOR NEEDS TO KNOW ABOUT A PROCESS WHEN THE PROCESS IS NOT RUNNING

INTERPROCESS TRANSMISSION TABLE (ITT) - ONE PER SYSTEM

ITT ENTRY - ONE OCCUPIED PER OUTSTANDING IPC WAKEUP

A QUEUE FOR TEMPORARILY STORING IPC WAKEUP INFORMATION (CHANNEL NAME, RANDOM DATA, PROCESS ID, ETC)

WORK CLASS TABLE (WCT) - ONE PER SYSTEM

WORK CLASS TABLE ENTRY (WCTE) - ONE PER WORKCLASS

EACH WCTE CONTAINS ADMINISTRATOR DEFINED PARAMETERS OF THE WORKCLASS, VARIOUS METERS AND POINTERS
FAULT AND INTERRUPT HANDLING

FUNCTION

- RESPONSIBLE FOR HANDLING ALL EXCEPTIONS IN A CPU WHETHER INTERNAL TO THE PROCESSOR (REFERRED TO AS FAULTS) OR EXTERNAL (REFERRED TO AS INTERRUPTS)

- ESTABLISHES THE SUPERVISOR ENVIRONMENT AT FAULT AND INTERRUPT TIME. SAVES THE MACHINE CONDITIONS AND TRANSFERS TO THE APPROPRIATE HANDLER

- MAJOR COMPONENTS: THE FAULT INTERCEPT MODULE (fim), WIRED-FAULT INTERCEPT MODULE (wired_fim), I/O INTERRUPT HANDLER (io_interrupt), sys_trouble, page_fault

MAJOR DATA BASES

- INTERRUPT VECTORS - ONE SET PER SYSTEM (WIRED)

- INTERRUPT PAIR (2 INSTRUCTIONS) - ONE PAIR PER DEFINED INTERRUPT TYPE

- LOCATED AT ABSOLUTE ADDRESS 0. A HARDWARE RECOGNIZED DATA BASE

- DESCRIBE WHERE TO SAVE THE CONTEXT, AND WHERE TO TRANSFER TO PROCESS THE INTERRUPT (ALWAYS io_interrupt)
FAULT AND INTERRUPT HANDLING

- **Fault Vectors** - One set per system (wired)
  - Vector pair (2 instructions) - one pair per defined fault type
  - Located at absolute address 100 (octal) immediately above the interrupt vectors. A hardware recognized data base
  - Describe where to save the context, and where to transfer to process the fault (fim, wired_fim, page_fault)

- **Process Data Segment (PDS)** - One per process (wired when eligible)
  - Contains process relevant info such as process ID, user ID, home/working/process directories, aim classification, initial ring, etc
  - Contains all information about the process needed by the supervisor code when the process is running
  - Contains save areas for context information about faults which can result in giving up the processor: page faults, segment faults, and all faults not handled by the supervisor

- **Processor Data Segment (PRDS)** - One per configured CPU (wired)
  - Serves as ring-zero stack for page control and traffic control
  - Also contains save areas for context information about faults which usually do not mean giving up the processor: connect faults and interrupts.
FAULT AND INTERRUPT HANDLING

FIM_TABLE Part of Fim not a real Table.

A table in the FIM program which describes the action to be taken for various types of faults.

43

41

46 causes linkage fault.

LINKS

a. F11

Compile time:
46 causes linkage fault calling dynamic link, 2nd.
The link changing 46 value in pointer to a valid pointer.

Linkage section shared among processes, so links snapped are per process. So first time
link is snapped, linkage section is copied per process.
Internal static is also in linkage section.
SYSTEM INITIALIZATION

FUNCTION

1. PREPARE THE SYSTEM TO OPERATE, STARTING FROM A COMPLETELY EMPTY MACHINE

2. READS IN SUPERVISOR PROGRAMS FROM SYSTEM TAPE, SNAPS LINKS BETWEEN SUPERVISOR COMPONENTS, VERIFIES AND INITIALIZES HARDWARE CONFIGURATION, SETS UP SYSTEM DATABASES, ACCEPTS STORAGE SYSTEM DISKS AND PREPARES THEM FOR USE BY THE FILE SYSTEM.

3. MOST PROGRAMS IN SYSTEM INITIALIZATION ARE DELETED AFTER INITIALIZATION IS COMPLETE. ONLY TEXT SECTION IS KEPT.

4. SUPERVISOR PROGRAMS ARE LOADED IN THREE "COLLECTIONS", EACH OF WHICH DEPENDS ON THE MECHANISMS SET UP BY THE PREVIOUS ONE.

MAJOR DATA BASES

1. THESE DATA BASES ARE ALL BUILT DURING THE PROCESS OF INITIALIZATION (EXCEPT FOR THE CONFIG DECK) AND KEPT AFTER INITIALIZATION IS FINISHED FOR DEBUGGING.

2. SEGMENT LOADING TABLE (SLT)

CONTAINS AN ENTRY DESCRIBING THE ATTRIBUTES OF EACH SEGMENT IN THE SUPERVISOR.
SYSTEM INITIALIZATION

NAME TABLE (>sl1>name_table)

CONTAINS A LIST OF NAMES FOR EACH OF THE SEGMENTS IN THE SUPERVISOR

DEFINITIONS SEGMENT (>sl1>definitions_)

CONTAINS THE DEFINITIONS SECTIONS FOR ALL THE SEGMENTS IN THE SUPERVISOR, WHICH ARE USED IN ORDER TO SNAP LINKS BETWEEN THE SUPERVISOR MODULES

CONFIG DECK (>sl1>config_deck)

CONTAINS A DESCRIPTION OF THE HARDWARE CONFIGURATION AND CERTAIN SOFTWARE PARAMETERS

PROVIDED TO SYSTEM INITIALIZATION BY BOS

SHUTDOWNS -- TERMINATES THE ACTIVITIES OF THE SYSTEM IN AN ORDERLY FASHION

TWO TYPES OF SHUTDOWN:

NORMAL -- REQUESTED BY THE INITIALIZER, RUNS IN THE USUAL SUPERVISOR ENVIRONMENT

EMERGENCY -- USED AFTER A CRASH, MUST MAKE THE SUPERVISOR ENVIRONMENT OPERABLE BEFORE PROCEEDING
SYSTEM INITIALIZATION

Both types exist primarily to shut down the file system -- that is, to write all data in memory into its proper home on disk.

Includes pages of segments, VTOCES, volume and VTOC maps.

Shutdown essentially runs the steps of initialization backwards, but with a lot of shortcuts.
FILE SYSTEM SALVAGERS

**FUNCTION**

- ENSURE THE CONSISTENCY OF THE FILE SYSTEM DATABASES AND PERFORM PERIODIC PREVENTIVE MAINTENANCE OPERATIONS

- THERE ARE SEVERAL SALVAGERS, EACH WITH A DIFFERENT FUNCTION


- SOME SALVAGING IS DONE AUTOMATICALLY, WHEN THE SYSTEM DETECTS AN INCONSISTENCY. OTHER SALVAGE OPERATIONS ARE EXPLICITLY REQUESTED, BY PRIVILEGED USERS.

- EXCEPT FOR SUPERVISOR BUGS, THE ONLY TIME DAMAGE OCCURS THAT REQUIRES SALVAGING TO FIX IS AFTER A CRASH WHERE EMERGENCY SHUTDOWN FAILS

**THE SALVAGERS:**

- DIRECTORY SALVAGER

  - CORRECTS INCONSISTENCIES IN DIRECTORY SEGMENTS BY REBUILDING THEM

  - THIS IS THE ONLY SALVAGER INVOKED AUTOMATICALLY IN USER PROCESSES: ANY ATTEMPT TO LEAVE RING ZERO WITH A DIRECTORY LOCKED FOR WRITING WILL CAUSE IT TO BE SALVAGED
FILE SYSTEM SALVAGERS

DIRECTORY SALVAGING ALSO RECLAIMS WASTED SPACE IN THE DIRECTORY, AND IS RUN PERIODICALLY TO COMPACT DIRECTORIES.

QUOTA SALVAGER

CORRECTS INCONSISTENCIES IN THE QUOTA SYSTEM.

PHYSICAL VOLUME SCAVenger

Fixes volume map & VTOCE map
Frees up records marked as being used.

Reconstructs record and VTOCE stock information from the VTOces on a volume, thereby reclaiming any records or VTOces which might have been lost.

Runs entirely online while the system is up for users (new in MR10.1)

This type of damage is usually benign, so running the scavenger can be delayed.

PHYSICAL VOLUME SALVAGER

Reconstructs record and VTOCE stock information.

Runs only during initialization, and therefore delays crash recovery.

Now used only for rare cases where there is not enough free space left for the scavenger to run. In these rare cases, it is invoked automatically by system initialization.

Sweep_pv

Fixes reverse connection failure.

Fixes disks where scav fixed incorrectly.
FILE SYSTEM SALVAGERS

- Deletes unused VTOC entries which have no directory entry pointing to them.

- Runs entirely in user ring, except for actually reading VTOC entries and directory entries.

- Purely a housekeeping function, and run only rarely.

Not To Be Reproduced 2-35  F80A
METERING & TUNING

While not a subsystem itself, metering and tuning is a policy and capability common to all of the supervisor's subsystems.

FUNCTION

METERING (consists of three activities)

1. Accumulating Data: This is performed throughout the supervisor by code which
   - records the number of times an event happens or a particular piece of code is executed; and/or
   - records the time required to perform a task
   - such data is stored in areas referred to as "metering cells"

2. Extracting Data: This is performed by numerous metering commands which (when invoked)
   - read and store the current values of relevant metering cells

3. Reporting the Data: This is performed by the meter commands which (when invoked)
   - compare current metering cell values with previously read values
   - perform the appropriate arithmetic computations upon the data in order to arrive at the desired statistic
   - arrange the data in a useful format (a report or diagram) and print it
METERING & TUNING

TUNING

CHANGING THE SYSTEM'S OPERATING PARAMETERS AND/OR CONFIGURATION BASED UPON THE DATA AND INSIGHTS FROM THE SYSTEM'S METERS

MAJOR DATA BASES

SST HEADER, TC_DATA HEADER, ETC.
INITIALIZER.SYSDAEMON

FUNCTION

THE SYSTEM'S INITIALIZATION, ADMINISTRATIVE AND CONTROL PROCESS (Initializer.SysDaemon.z), RESPONSIBLE FOR:

- INITIALIZING THE OPERATING SYSTEM AT BOOTLOAD, FOLLOWING SUCCESSFUL INITIALIZATION OF THE SUPERVISOR
- ANSWERING SERVICE (login and logout)
- PROCESS CREATION AND DESTRUCTION
- MESSAGE COORDINATOR (DAEMON COORDINATION)
- SYSTEM ADMINISTRATION FUNCTIONS
- SYSTEM ACCOUNTING FUNCTIONS

MAJOR DATA BASES, ALL KEPT IN >sc1

- ANSWER_TABLE
- ABSENTEE_USER_TABLE
- DAEMON_USER_TABLE
INITIALIZER.SYSDAEMON

- MASTER GROUP TABLE (MGT)
- CHANNEL DEFINITION TABLE (CDT)
- SYSTEM ADMINISTRATION TABLE (SAT)
- PERSON NAME TABLE (PNT)
- PROJECT DEFINITION TABLES (PDT'S)
# TOPIC III
The Multics Environment

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is a Process</td>
<td>3-1</td>
</tr>
<tr>
<td>Cooperating Processes</td>
<td>3-4</td>
</tr>
<tr>
<td>The PL/I Operators</td>
<td>3-7</td>
</tr>
<tr>
<td>Interfaces to System Modules</td>
<td>3-9</td>
</tr>
<tr>
<td>Deadlock Prevention</td>
<td>3-10</td>
</tr>
<tr>
<td>Types of Locks</td>
<td>3-15</td>
</tr>
</tbody>
</table>
A MULTICS PROCESS IS A WELL DEFINED COLLECTION OF SEGMENTS, EACH WITH DEFINED ACCESS, OVER WHICH A SINGLE EXECUTION POINT IS FREE TO ROAM (I.E., FETCH INSTRUCTIONS AND MAKE DATA REFERENCES)

THE ADDRESS SPACE OF A PROCESS IS THE ABOVE "COLLECTION OF SEGMENTS". SUCH SEGMENTS ARE SAID TO BE KNOWN TO THE PROCESS

EVERY LOGGED IN USER HAS A PROCESS

VERY IMPORTANT CONCEPT: THE MULTICS SUPERVISOR RUNS IN THE USER'S PROCESS (I.E.: IN THE USER'S ADDRESS SPACE), BUT IN A DIFFERENT RING

A PROCESS TAKES ON THE IDENTITY OF THE SOFTWARE IT IS EXECUTING WHEREEVER IT GOES

WHEN A USER WISHES TO CREATE A SEGMENT, IT IS THE USER'S PROCESS WHICH EXECUTES THE SUPERVISOR CODE hcs_sappend, CREATING THE SEGMENT

A PROCESS CAN BE VIEWED AS A CONTINUAL FLOW OF EXECUTION FLUCTUATING BETWEEN DIFFERENT RINGS: PRIMARILY RING FOUR AND RING ZERO
WHAT IS A PROCESS

PROCESS FLOW OF EXECUTION

* ALL PROCEDURE CODE (WHETHER SUPERVISOR OR USER CODE) MUST HAVE A STACK FRAME CONTAINING ITS ARGUMENTS AND ENVIRONMENT DATA.

* FOR REASONS OF SECURITY, MULTICS REQUIRES ONE STACK PER RING OF EXECUTION. WHEN EXECUTING RING "N" PROCEDURES, THERE WILL EXIST A RING "N" STACK CONTAINING STACK FRAMES FOR THESE PROCEDURES.
STACK PER RING

* PR7 IS THE STACK BASE POINTER REGISTER. POINTS TO CURRENT STACK. HARDWARE LOADED WITH DBR.stack *B + RING. THE VALUE OF DBR.stack CAN CHANGE WITH EACH RELEASE. IT IS 23 IN MR10.1.

* PR6 IS THE STACK FRAME POINTER REGISTER. POINTS TO CURRENT STACK FRAME.
COOPERATING PROCESSES

ALL ACTIVE PROCESSES (INTERACTIVE, ABSENTEE, AND DAEMONS) APPEAR TO BE AUTONOMOUS AND INDEPENDENT OF ONE ANOTHER.

IN REALITY, ALL PROCESSES ARE CONTINUALLY COOPERATING, COMPETING AND SHARING.

EXAMPLES OF COOPERATION

VOLUNTARY

THE SENDING AND ACCEPTING OF MESSAGES AND MAIL.

PREPLANNED BY SYSTEM PROGRAMMERS

EVERY PROCESS, BEFORE RELINQUISHING A PROCESSOR, Chooses THE MOST DESERVING REPLACEMENT AND EXECUTES THE CODE WHICH DISPATCHES THE CHOSEN PROCESS.

EVERY PROCESS, WHEN RUNNING, WILL SERVICE ALL INTERRUPTS FIELDED BY ITS PROCESSOR. THESE INTERRUPTS ARE GENERALLY THE REPLIES TO THE REQUESTS OF OTHER PROCESSES (IE: THE ARRIVAL OF A PAGE REQUESTED SOME TIME EARLIER).

PREPLANNED BY APPLICATION PROGRAMMERS

THE MULTICS TRANSACTION PROCESSOR IS COMPOSED OF MANY COOPERATING, INTER-DEPENDENT PROCESSES.

EXAMPLES OF COMPETITION

ALL PROCESSES COMPETE FOR PROCESSOR TIME AND MAIN MEMORY RESOURCES.
COOPERATING PROCESSES

| THIS COMPETITION IS HIGHLY REGULATED IN ORDER FOR ALL PROCESSES TO BE TREATED FAIRLY |
| THE COMPETITION IS ALSO SUBJECT TO VERSATILE ADMINISTRATIVE CONTROLS |
| EXAMPLES OF SHARING |
| BY DESIGN, A SIGNIFICANT PART OF THE ADDRESS SPACE OF ALL PROCESSES IS IDENTICAL (THE SUPERVISOR SEGMENTS) |
| BY DEFAULT, REFERENCES TO SEGMENT foo BY TWO DIFFERENT PROCESSES WILL RESULT IN REFERENCES TO THE SAME SEGMENT (LOGICALLY, PHYSICALLY, ACTUALLY AND ABSOLUTELY) |

...THERE IS NO SEPARATE ENTITY IN MULTICS LIKE AN EXECUTIVE DOING THINGS ON BEHALF OF THE USER. THE Initializer.SysDaemon IS NOT THE TIME-SHARE EXECUTIVE OF MULTICS
PROCESS A

PROCESS B

Changes address space which changes process in execution
THE PL/I OPERATORS

1. OPERATORS ARE LANGUAGE DEPENDENT PIECES OF CODE WHICH IMPLEMENT HARDWARE OR OPERATING-SYSTEM DEPENDENT FUNCTIONS SUCH AS CALLING AND SIGNALLING.

2. CURRENTLY THERE ARE OPERATORS FOR PL/I, COBOL, AND BASIC. ALM AND FORTRAN SHARE THE PL/I OPERATORS.

3. ALL OPERATORS IN MULTICS ARE PURE, SHARED AND RE-ENTRANT ALM CODE.

4. OPERATORS COULD BE GENERATED BY THE COMPILERS AND PLACED IN LINE WITH OTHER CODE, HOWEVER, THERE ARE DISADVANTAGES:

   a. SOME OPERATORS ARE TOO BULKY TO BE INCLUDED WITH EACH USE (SUCH AS COMPLICATED I/O STATEMENTS).

   b. SOME OPERATORS MIGHT CHANGE IN THE FUTURE (SUCH AS ENTRY AND RETURN SEQUENCES).

5. OPERATORS ARE SIMILAR TO QUICK INTERNAL PROCEDURES IMPLEMENTING WHAT IS OFTEN CALLED "LIBRARY FUNCTIONS" IN OTHER OPERATING SYSTEMS.
THE PL/I OPERATORS

In place of passing argument list, arguments are usually passed to the operators in the CPU's registers.

Instead of being called by a procedure call, a single transfer instruction is used (tsx0 or tsp3).

The PL/I operators implement the support functions for the PL/I environment.

Since a MULTICS process is a PL/I environment, the PL/I operators are vital to the MULTICS supervisor (and any other programs written in PL/I).
UNLIKE OTHER SUPERVISORS, THE MULTICS SUPERVISOR IS NOT SEQUENTIAL - THAT IS, THE CONCEPT OF "JOB FLOW" DOES NOT REALLY APPLY

INSTEAD, THE SUBSYSTEMS PERFORM ASYNCHRONOUSLY, BEING 'INVOKED' BY THOSE PROCESSES WHO REQUIRE THEIR SERVICES

THESE SUBSYSTEMS ARE INVOKED BY THE USER'S PROCESS IN ONE OF THREE WAYS:

- EXPLICITLY - VIA A SUBROUTINE CALL OR A COMMAND
- IMPLICITLY - VIA A FAULT
- IMPLICITLY - VIA AN INTERRUPT

CALL SIDE:
SERVICES PERFORMED AS A RESULT OF EXPLICIT SUBROUTINE CALLS. LOCKS ARE NORMALLY WAIT LOCKS.

FAULT SIDE:
SERVICES PERFORMED AS A RESULT OF FAULTS. LOCKS ARE NORMALLY WAIT LOCKS.

INTERRUPT SIDE:
SERVICE PERFORMED AS A RESULT OF INTERRUPTS. LOCKS ARE LOOP LOCKS.
WHAT IS DEADLOCK?

DEADLOCK CAN OCCUR IN ANY MULTI-PROGRAMMING ENVIRONMENT WHEN TWO OR MORE PROCESSES COMPETE RANDOMLY FOR SERIALLY REUSABLE RESOURCES.

THE CLASSIC EXAMPLE OF DEADLOCK IS THE "DEADLY EMBRACE".

DEADLY EMBRACE

- PROCESS A IS WAITING FOR A RESOURCE OWNED BY PROCESS B.
- PROCESS B IS WAITING FOR A RESOURCE OWNED BY PROCESS A.
DEADLOCK PREVENTION

DEADLOCK SOLUTIONS

DETECTION AND UNLOCKING

SOME SYSTEMS EMPLOY SCHEMES WHICH DETECT THE OCCURRENCE OF DEADLOCK AND "UNTANGLE" THE INVOLVED PROCESSES.

DETECTION SCHEMES ARE USUALLY DIFFICULT TO IMPLEMENT AND EXPENSIVE IN TERMS OF OVERHEAD.

THE ACT OF UNTANGLING THE INVOLVED PROCESSES USUALLY RESULTS IN AT LEAST ONE OF THEM LOSING RESOURCES, PRIORITY, OR EVEN ITS LIFE.

PREVENTION

MOST SYSTEMS ADOPT SOME FORM OF PREVENTION INSTEAD OF DETECTION.

PREVENTION SCHEMES NORMALLY TAKE ONE OF TWO FORMS:

CHECKING: WHEREBY REQUESTS FOR RESOURCES ARE SCREENED FOR DEADLOCK POTENTIAL PRIOR TO ACCEPTANCE.

IMPOSED POLICY: WHEREBY REQUESTS FOR MORE THAN ONE RESOURCE MUST BE MADE:

TOGETHER AS ONE TOTAL REQUEST BEFORE THE "JOB" OR "JOB-STEP" COMMENCES (ALL OR NOTHING); OR

SERIALLY, IN A FIXED, PRE-DEFINED ORDER.
DEADLOCK PREVENTION

MULTICS, IN GENERAL, ADOPTS THE FOLLOWING DEADLOCK PREVENTION SCHEME:

1. USER ASSIGNABLE RESOURCES (SUCH AS TAPE DRIVES, CARD PUNCHES, ETC)

   - WHEN THE USER IS INTERACTIVE NO POLICY IS ENFORCED. THE USER IS INFORMED IF THE RESOURCE IS BUSY AND MAY EITHER TRY AGAIN OR GIVE UP
   - WHEN THE USER IS NOT INTERACTIVE, THE "ALL" OR "NONE" APPROACH SHOULD BE USED. THE AVAILABILITY OF ALL REQUIRED RESOURCES BECOMES THE DETERMINING FACTOR IN SCHEDULING THE USER (SEE THE "RESOURCE CONTROL PACKAGE")
   - SHOULD A NON-INTERACTIVE USER ATTEMPT SERIAL REQUESTS FOR RESOURCES, A DEADLOCK SITUATION COULD POTENTIALLY ARISE AND EXIST UNTIL THE AUTOMATIC LOGOUT DUE TO INACTIVITY OCCURS

2. USER ACCESSIBLE RESOURCES (SUCH AS FILES, DATA BASES, ETC)

   - IN GENERAL, USER SEGMENTS IN THE HIERARCHY POSE NO DEADLOCK PROBLEM SINCE THEY ARE A SIMULTANEOUSLY USABLE RESOURCE (IE: THERE IS NO DEFAULT CONCURRENCY MECHANISM ASSOCIATED WITH USER SEGMENTS)
   - SEGMENTS MAY BE PROTECTED FROM POTENTIAL CONCURRENCY PROBLEMS THROUGH USE OF LOCK WORDS AND THE set_lock MECHANISM. THIS REQUIRES MUTUAL AGREEMENT AMONG ALL PROCESSES ACCESSING SUCH SEGMENTS
   - SOME SEGMENTS SUCH AS THOSE USED BY THE MULTICS DATA BASE MANAGER (MDBM), USE A "COMMITMENT/RollBACK" SCHEME
DEADLOCK PREVENTION

SUPERVISOR RESOURCES (SUCH AS HARDCORE DATABASE).

LOCKWORDS (OR SIMPLY "LOCKS") ARE USED IN MULTICS TO IMPLEMENT CONCURRENT ACCESS CONTROL IN THE MULTI-PROCESS ENVIRONMENT.

LOCKING CONCEPT

THE SUPERVISOR LOCKS ARE ARRANGED IN A PARTIAL ORDER AND A CODING CONVENTION PREVENTS WAITING ON A LOCK IF THE PROCESS HAS A HIGHER LOCK LOCKED.

THIS PARTIAL ORDER IS DETERMINED BY AN ANALYSIS OF THE OPERATING SYSTEM'S BEHAVIOR. FOR EXAMPLE: SINCE A PAGE FAULT MAY PROPERLY OCCUR WHILE A PROCESS HAS THE ACTIVE SEGMENT TABLE (AST) LOCKED, AND PAGE FAULT HANDLING REQUIRES THE LOCKING OF THE PAGE TABLE LOCK, THE PAGE TABLE LOCK MUST BE PLACED "HIGHER" IN THE PARTIAL ORDER THAN THE AST LOCK.

TO THE DEGREE THAT THE SYSTEM PROGRAMMERS OBEY THIS PARTIAL ORDER, A DEADLY EMBRACE CANNOT OCCUR WITHIN THE MULTICS SUPERVISOR.
MULTICS LOCKING HIERARCHY

CONNECT (LOOP)
GLOBAL APT (MULTI-READER (LOOP))
APT ENTRY (LOOP)
CORE QUEUE (LOOP)
DISK DATA (LOOP)
OCDCM (LOOP)
I/O MAILBOX (LOOP)
GLOBAL PAGE TABLE (LOOP/WAIT)
VTOC BUFFER SEGMENT (WAIT)
ACTIVE SEGMENT TABLE (WAIT)
DIRECTORY LOCK TABLE (WAIT)
ROOT DIRECTORY (MULTI-READER, WAIT)
LOWER DIRECTORY (MULTI-READER, WAIT)

MISCELLANEOUS: IOAT, IOI, SALV_DATA, RECONFIGURATION (WAIT)

SYSERR DATA (LOOP)
MCS QUEUE (LOOP)

DEADLOCK PREVENTION

3-14
TYPES OF LOCKS

LOCKS WITHIN MULTICS:

- ARE 36 BIT WORDS CONTAINING EITHER ZERO (UNLOCKED) OR A PROCESS_ID (LOCKED).

- CONTROL PROCESSES, NOT PROCESSORS.

- ARE MUTUALLY EXCLUSIVE LOCKS.

THE HARDWARE SUPPORTS SEVERAL INDIVISIBLE INSTRUCTIONS USED IN IMPLEMENTING THE LOCKING PRIMITIVES. FOR EXAMPLE:

STAC (STORE A CONDITIONAL)

- IF C(Y) = 0 THEN C(A) -> C(Y)

- TYPICAL USE: LOCKING. IF THE LOCKWORD (Y) IS UNLOCKED (=0) THEN LOCK THE LOCK BY STORING THE PROCESS_ID (WHICH IS IN A) INTO THE LOCKWORD.

- SPECIAL HARDWARE PROHIBITS SIMILAR REFERENCES BY OTHER PROCESSORS DURING THE TEST AND DATA TRANSFER WINDOW.

- ALSO STACQ (STORE A CONDITIONAL ON Q), LDAC (LOAD A AND CLEAR), LDQC (LOAD Q AND CLEAR), SZNC (SET Z AND N AND CLEAR).
TYPES OF LOCKS

Within the Multics Supervisor exists two types of locks: Loop locks and Wait locks.

Loop Locks

Loop locks are used when it would not be acceptable to give up the processor before locking the lock.

Loop locks typically protect the lowest level of critical supervisor databases: Traffic Control, Page Control, etc.

Simplified PL/I Analogy:

```pli
do while (lockword ^= 0);
end;
lockword = process_id;
<update data>
lockword = 0;
```
TYPES OF LOCKS

WAIT LOCKS (SIMPLIFIED)

A: PROCESS GIVES UP THE CPU
B: PROCESS DISPATCHED TO A CPU

Simplified PL/I Analogy:

```pli
do while (lockword ^= 0);
   <GIVE UP PROCESSOR>
   <WAIT FOR NOTIFICATION>
end;
Lockword = process_id;
<update data>
Lockword = 0
<SEND NOTIFICATION>
```

Most supervisor locks are wait locks.

In general, a process is allowed to give up its processor when it has wait locks locked.

The wait lock mechanism will be described in more detail in Topic 9.
TOPIC IV

Name Space and Address Space Management

Name/Address Space Overview ........................................ 4-1
Name/Address Space Terminology ..................................... 4-5
Name/Address Space Concepts ......................................... 4-7
Name/Address Space Data Bases ....................................... 4-12
  Reference Name Table (RNT) ........................................ 4-12
  Known Segment Table (KST) ......................................... 4-13
  Descriptor Segment (DSEG) ......................................... 4-14
Typical Address Space .................................................. 4-16
Name/Address Space Meters ............................................ 4-31
  system_link_meters .................................................. 4-31
  link_meters .......................................................... 4-32
Name/Address Space Commands ....................................... 4-33
  display_kst_entry .................................................... 4-33
NAME/ADDRESS SPACE OVERVIEW

FUNCTION

All online info is Processor Addressable.

They are made Processor addressable as they are referenced.

IMPLEMENT THE PER PROCESS VIRTUAL MEMORY

Known Segs

BASIC PHILOSOPHY

As a newly logged in user attempts to touch various segments a considerable amount of management information must be (transparently) found and/or computed before the user's reference is actually accomplished.

For every segment referenced by the user, the supervisor:

Assigns a segment number (for reason of hardware addressing), and

records (remembers) the management information (for reason of software efficiency and control).

Such segments are said to be "known to the process".

The management information is maintained on a per process basis in three complementing areas: DSEG, KST, and RNT.
NAME/ADDRESS SPACE OVERVIEW

- MANAGES TWO DISTINCT SETS OF INFORMATION:
  - ADDRESS SPACE - CORRESPONDENCE BETWEEN SEGMENT NUMBERS AND THE SEGMENTS THEMSELVES
  - NAME SPACE - CORRESPONDENCE BETWEEN SEGMENT NUMBERS AND NAMES THE USER REFERS TO THEM BY

- CALLS DIRECTORY CONTROL TO LOCATE SEGMENTS INITIALLY

- NAME SPACE / ADDRESS SPACE MANAGEMENT IS INVOKED BY SUBROUTINE CALLS, AND BY LINKAGE FAULTS (THE "DYNAMIC LINKER")

PRINCIPAL USER INTERFACES

- COMMAND LEVEL
  - initiate, terminate, terminate_segno, terminate_ref_name,
    terminate_single_ref_name, list_ref_name

  - THE COMMAND PROCESSOR ITSELF - WHICH USES THESE SERVICES TO LOCATE COMMANDS

- SUBROUTINE LEVEL
  - hcs_$initiate, hcs_$initiate_count, hcs_$terminate_file,
    hcs_$terminate_seg, hcs_$terminate_name,
    hcs_$terminate_noname, term

Not To Be Reproduced 4-2 F80A
NAME/ADDRESS SPACE OVERVIEW

MAJOR DATA BASES

- DESCRIPTOR SEGMENT (DSEG) - ONE PER PROCESS
  - SEGMENT DESCRIPTOR WORD (SDW) - ONE PER KNOWN SEGMENT
  - DEFINES THE USER'S ADDRESS SPACE TO THE HARDWARE

- KNOWN SEGMENT TABLE (KST) - ONE PER PROCESS
  - KNOWN SEGMENT TABLE ENTRY (KSTE) - ONE PER KNOWN SEGMENT
    (EXCEPT SUPERVISOR SEGMENTS)
  - DEFINES THE USER'S ADDRESS SPACE TO THE SUPERVISOR AND THE USER
  - EACH KSTE ASSOCIATES A USER'S SEGMENT NUMBER WITH THE SEGMENT
    CONTROL ATTRIBUTES OF THAT SEGMENT
  - THE SEARCH FOR AN AVAILABLE KSTE DETERMINES A SEGMENT'S
    NUMBER

- REFERENCE NAME TABLE (RNT) - ONE PER EACH RING IN EACH PROCESS
  - NOT A SEGMENT - KEPT AS A REGION ALLOCATED IN THE "LINKAGE
    AREA" FOR EACH RING
  - REFERENCE NAME TABLE ENTRY (RNTE) - ONE PER REFERENCE NAME
  - USED BY THE DYNAMIC LINKER TO IMPLEMENT THE
    "initiated_segments" SEARCH RULE
NAME/ADDRESS SPACE OVERVIEW

- DEFINES THE USER'S NAME SPACE TO THE USER

- NAME SPACE MAY BE DIFFERENT IN DIFFERENT RINGS OF THE SAME PROCESS

![Diagram showing the name/address space overview]
NAME/ADDRESS SPACE TERMINOLOGY

SEGMENT DESCRIPTOR WORD (SDW):
A TWO WORD PAIR USED BY THE HARDWARE WHEN REFERENCING A SEGMENT.

DESCRIPTOR SEGMENT (DSEG):
THE MOST FUNDAMENTALLY IMPORTANT SEGMENT IN A PROCESS.
CONTAINS AN ARRAY OF SDW'S DEFINING THE ADDRESS SPACE OF THE PROCESS.

ADDRESS SPACE:
THE SET OF ALL SEGMENTS (PROCEDURE AND DATA) FOR WHICH THE PROCESS HAS A SEGMENT NUMBER AND A CORRESPONDING SDW. THE ADDRESS SPACE EXPANDS AND CONTRACTS DURING A SEGMENT'S LIFE.

SEGMENT NUMBER:
AN OCTAL NUMBER 0-1777 (0-1023 DECIMAL) ASSIGNED UNIQUELY TO A SEGMENT. USED BY THE HARDWARE AS AN OFFSET INTO THE ARRAY OF SDW'S WHEN REFERENCING A SEGMENT.

MAKING KNOWN:
THE ACT OF ASSIGNING A SEGMENT NUMBER TO A SEGMENT, THEREBY ADDING IT TO THE ADDRESS SPACE. SEGMENTS MUST BE MADE KNOWN BEFORE THEY CAN BE REFERENCED.
NAME/ADDRESS SPACE TERMINOLOGY

NAME SPACE:

NAME/ADDRESS SPACE CONCEPTS

A MULTICS PROCESS IS A WELL DEFINED COLLECTION OF UNIQUE SEGMENTS, EACH WITH DEFINED ACCESS, OVER WHICH A SINGLE EXECUTION POINT IS FREE TO ROAM (I.E., FETCH INSTRUCTIONS AND MAKE DATA REFERENCES)

THE ADDRESS SPACE OF A PROCESS IS THE ABOVE "COLLECTION OF SEGMENTS". SUCH SEGMENTS ARE SAID TO BE KNOWN TO THE PROCESS

ALL SUPERVISOR (RING 0) SEGMENTS ARE PLACED INTO THE ADDRESS SPACE AT PROCESS CREATION TIME. THIS ADDRESS SPACE IS SAID TO BE "CLONED" FROM THE INITIALIZER ADDRESS SPACE

THE RING ZERO ADDRESS SPACE IN ANY PROCESS IS THE SAME AS THE INITIALIZER'S RING ZERO ADDRESS SPACE, EXCEPT FOR THE DSEG, KST, PDS, PRDS, AND STACK_0.

THE RING ZERO ADDRESS SPACE HAS NO RNT, BECAUSE IT IS SET UP DURING SYSTEM INITIALIZATION, AND DOES NOT CHANGE.

OTHER SEGMENTS ARE MADE KNOWN AND UNKNOWN DURING THE LIFE OF THE PROCESS

IMPICITLY BY THE DYNAMIC LINKER (LINKAGE FAULT) OR A SYSTEM COMMAND

print my_dir>my_seg
tester

EXPLICITLY BY COMMANDS OR SUBROUTINES THAT MANAGE THE ADDRESS SPACE

initiate my_prog call hcs$terminate_segno
NAME/ADDRESS SPACE CONCEPTS

Making a segment known is similar to declaring a variable in a PL/I program. It signifies intent, but not usage.

The presence of one or more pages of a segment in main memory implies that the segment is known to (and is being used by) at least one user.

Being known does not imply presence in main memory.

Note that this set of segments, the execution point, and the registers and indicators of the processor, uniquely defines the state of the process.
NAME SPACE AND ADDRESS SPACE MANIPULATION

1. PROCESS CREATION

2. INITIATE WITHOUT REFERENCE NAME

3. INITIATE WITH REFERENCE NAME

**THE rnt IS AN AREA WITHIN [pd]>[unique].area.linker**
NAME SPACE AND ADDRESS SPACE MANAGEMENT

PER PROCESS DATA BASES

PER SYSTEM DATA BASES

RNT
(USER'S NAME SPACE)

KST
(USER'S ADDRESS SPACE)

DSEG
(SYSTEM'S ADDRESS SPACE)

PAGE TABLES*

MAIN MEMORY

* THE ASTE'S ASSOCIATED WITH THE PAGE TABLES ARE NOT SHOWN IN THIS DIAGRAM
REFERENCE NAME TABLE (RNT)
A PAGED DATA BASE \( (pd + rnt) \) - ONE PER ACTIVE PROCESS
"INITIATED SEGMENTS" IN SEARCH RULES

REFERENCE NAME TABLE ENTRY (RNTE)
ONE PER REFERENCE NAME
NAME/ADDRESS SPACE DATA BASES

**KNOWN SEGMENT TABLE (KST)**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LOWEST SEGMENT NUMBER</td>
</tr>
<tr>
<td>1</td>
<td>HIGHEST SEGMENT NUMBER</td>
</tr>
<tr>
<td>2</td>
<td>HIGHEST SEGMENT NUMBER YET USED</td>
</tr>
<tr>
<td>3</td>
<td>NUMBER OF PRIVATE LOGICAL VOLUMES</td>
</tr>
<tr>
<td>4</td>
<td>TIME OF BOOTLOAD</td>
</tr>
<tr>
<td>5</td>
<td>NO. OF KST GARBAGE COLLECTIONS</td>
</tr>
<tr>
<td>6</td>
<td>NO. OF KST ENTRIES RECOVERED</td>
</tr>
<tr>
<td>7</td>
<td>FIRST FREE KSTE R_PTR</td>
</tr>
</tbody>
</table>

### TIME OF BOOTLOAD

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 64 WORD ARRAY FOR UID HASH TABLE

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FORWARD R_PTR</td>
</tr>
<tr>
<td>1</td>
<td>SEGMENT NUMBER</td>
</tr>
<tr>
<td>2</td>
<td>NUMBER OF TIMES INITIATED</td>
</tr>
<tr>
<td>3</td>
<td>THIS PER RING</td>
</tr>
<tr>
<td>4</td>
<td>BRANCH ENTRY POINTER</td>
</tr>
<tr>
<td>5</td>
<td>SEGMENT'S UNIQUE IDENTIFIER (UID)</td>
</tr>
<tr>
<td>6</td>
<td>DATE TIME BRANCH ENTRY MODIFIED (DTBMM)</td>
</tr>
</tbody>
</table>

### EXTENDED ACCESS

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### OTHER KSTE's

ONE PER INITIATED SEGMENT

### 256 WORD LIST OF PRIVATE LOGICAL VOLUME CONNECTIONS

### END OF KST
**SEGMENT DESCRIPTOR WORD (SDW)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Address</td>
<td>2047 WORDS</td>
</tr>
<tr>
<td>Offset</td>
<td>0 to 255 WORDS</td>
</tr>
<tr>
<td>Entry Point</td>
<td>PAGE TABLE</td>
</tr>
<tr>
<td>Page Frame</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Offset</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Directory</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Number</td>
<td>0 to 15</td>
</tr>
</tbody>
</table>

**ADDRESS OF PAGE TABLE**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Address</td>
<td>2047 WORDS</td>
</tr>
<tr>
<td>Offset</td>
<td>0 to 255 WORDS</td>
</tr>
<tr>
<td>Page Frame</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Offset</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Directory</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Number</td>
<td>0 to 15</td>
</tr>
</tbody>
</table>

**APAVED SEGMENT (DSEG)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Address</td>
<td>0 to 255 WORDS</td>
</tr>
<tr>
<td>Offset</td>
<td>0 to 255 WORDS</td>
</tr>
<tr>
<td>Entry Point</td>
<td>PAGE TABLE</td>
</tr>
<tr>
<td>Page Frame</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Offset</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Directory</td>
<td>0 to 15</td>
</tr>
<tr>
<td>Page Number</td>
<td>0 to 15</td>
</tr>
</tbody>
</table>

**NAME/ADDRESS SPACE DATA BASES**
NAME/ADDRESS SPACE DATA BASES

DESCRIPTOR SEGMENT (DSEG)

SEGMENTS IN THE MULTICS HIERARCHY

INITIATED WITH REFERENCE NAME(S)

IN USER'S RNT

NOT IN USER'S RNT

INITIATED WITHOUT REFERENCE NAME

IN USER'S KST

NOT IN USER'S KST

SUPERVISOR SEGMENTS

IN USER'S DSEG

NOT IN USER'S DSEG

MUST CONNECT USER TO CURRENTLY ACTIVE SEGMENT -OR-
RECONNECT BECAUSE OF ACL CHANGE, POOL PROMOTION, ETC.

VALID SDW

SEG FAULT

FAULTED SDW

NO SDW

ACTIVE SEGMENTS

IN THE AST

NOT IN THE AST

UNPAGED SUPERVISOR SEGMENTS

HAS A PAGE TABLE

DOES NOT HAVE A PAGE TABLE

IN MAIN MEMORY

NOT IN MAIN MEMORY

STATES OF SEGMENTS AND PAGES
TYPICAL ADDRESS SPACE

COLLECTION ZERO

I SEGMENTS WHICH MUST BE PRESENT TO RUN THE VERY FIRST LOADING PROGRAM

I SEGMENTS WHICH HAVE FIXED ABSOLUTE ADDRESSES TO INTERFACE WITH HARDWARE

I ALL DESCRIBED IN template_slit_cds

0 [pd]>dseg (ring 0)
The descriptor segment. The initializer's dseg comes from the system tape and is built during system initialization; all others are created by process creation.

1 fault_vector (ring 0, perm-wired)
Contains the interrupt vector, fault vector, and the ITS pairs for SCU and TRA instructions. Located at absolute locations 0-577. Used by the CPU hardware.

2 iom_mailbox (ring 0, perm-wired)
Mailboxes (communications areas) for up to four IOMs. Located at locations 1200-3377 absolute. Used by the IOM hardware.

3 >sl1>config_deck (ring 0, deciduous)
The online copy of the config deck. This is built from the config deck provided by BOS during system initialization, but it is not the copy BOS actually uses.

4 dn355_mailbox (ring 0)
Mailboxes (communications areas) for up to eight FNP's. Located at absolute locations 3400-6377. Used by the FNP hardware.

5 bos_toehold (ring 0, perm-wired)
The segment containing the tiny program used to switch between Multics and BOS at crash time. Located at absolute locations 10000-11777.
TYPICAL ADDRESS SPACE

6 flagbox (ring 0, perm-wired)
   A region inside the bos_toehold segment (yes, it really overlaps the toehold) used to access the BOS/Multics communication region.

7 >s11>slt (ring 0, deciduous)
10 >s11>name_table (ring 0, deciduous)
   The two primary databases of system initialization. The SLT contains one entry for every supervisor segment read from the system boot tape, containing all its attributes. The separate name_table is used to hold the names, because each segment may have several.

This marks the end of Collection Zero. All the rest of the segments in the address space are either read from the system tape or found in the online system.
TYPICAL ADDRESS SPACE

COLLECTION ONE

FIRST BATCH OF SEGMENTS READ FROM SYSTEM TAPE

ALL THE PROGRAMS AND DATABASES NEEDED TO MAKE PAGING RUN

ALL SEGMENTS WHICH MUST BE "perm-wired" -- PERMANENTLY ALLOCATED IN LOW MEMORY, WITHOUT PAGE TABLES

lot (ring 0)
The supervisor's linkage offset table. Used to find linkage sections. Built as the segments are read in from tape.

as_linkage (ring 0)
ws_linkage (ring 0)
The permanent supervisor combined linkage regions. The names mean "Active Supervisor Linkage" and "Wired Supervisor Linkage", respectively. The linkage sections of all permanent supervisor segments are put in one of these as the segments are read from the tape.

>definitions_ (ring 0, deciduous)
The segment containing all the definitions sections of supervisor programs. The definitions sections are placed here as the programs are read from the tape, and used by the hardcore prelinker.

sst_seg (ring 0, perm-wired)
The segment containing all ASTEs and page tables. Covered under Page and Segment Control. This segment is allocated at the very top end of the bootload SCU.

core_map (ring 0, perm-wired)
All the core map entries, describing all system memory. Covered under Page Control. This used to be part of the SST, but was moved out to make more room for page tables.

abs_seg (ring 0, abs-seg)
An abs-seg used for complex call-side operations in...
TYPICAL ADDRESS SPACE

page control, such as evict_page and reconfiguration.

20 abs_seg (ring 0, abs-seg)
An abs-seg used only by page control for checking page frame contents for zeros, and zeroing newly allocated page frames.

21 backup_abs_seg (ring 0, abs-seg)
An abs-seg used to access the segment being dumped in a Volume Dumper process. It is given an SDW which refers to any ordinary segment. This is a very special hardcore segment, because it has trailer entries, and it is special-cased by the trailer manipulation program, set faults. Normal hardcore segments never receive trailers, since they are never activated or deactivated.

22 fim_abs_seg (ring 0, abs-seg)
An abs-seg used by the FIM to do something.

23 isolts_abs_seg (ring 0, abs-seg)
An abs-seg used by ISOLTS, which gives it an SDW describing the low 64K of the SCU being used for ISOLTS testing.

24 volmap_abs_seg (ring 0, abs-seg)
An abs-seg used by page control to access record stocks. It is given the SDW of whichever volume's stock is needed.

25 bound_active_1 (ring 0)
26 bound_disk_util (ring 0)
27 bound_disk_util_wired (ring 0, perm-wired)
30 bound_error_active (ring 0)
31 bound_error_wired (ring 0, perm-wired)
32 bound_interceptors (ring 0, perm-wired)

This is where the FIM lives.

33 bound_io_wired (ring 0, perm-wired)
34 bound_lam_support (ring 0, perm-wired)
35 bound_page_control (ring 0, perm-wired)
36 bound_priv_1 (ring 0, perm-wired)
37 >sll>bound_sss_wired (ring 0, deciduous)

This is where pl1_operators_ and a whole host of other miscellaneous subroutines live. It is wired in a very special way, because not all of its contents need to be wired. In particular, only about half of pl1_operators_ needs to be wired, and there is a special hack in make_sdw.pl1 which finds the definition in the middle of pl1_operators_ which
TYPICAL ADDRESS SPACE

marks the end of the wired portion, and makes an
ASTE for bound_sss_wired which has all its pages up
to and including that definition wired, and the rest
unwired.

40 bound_tc_priv (ring 0, perm-wired)
41 bound_tc_wired (ring 0, perm-wired)
42 bound_unencacheable (ring 0, perm-wired)
43 dir_seg (ring 0, abs-seg)

An abs-seg now used only at process termination time
to loop through the dead process's KST in order to
flush any trailers it had for active segments.

44 disk_post_queue_seg (ring 0, perm-wired)
The segment where the core address queue lives: see
core_queue_man.aim. This is discussed under Page
Control.

45 disk_seg (ring 0, perm-wired)
The segment containing the disk DIM's databases:
device table, channel table, and I/O queues.

46 dn355_data (ring 0, perm-wired)
The segment containing software communications
regions for the FNPs. This is not where FNP buffers
are kept, but only the mailboxes that describe the
buffers, and some control information.

47 ds_seg (ring 0, abs-seg)
The segment used by setfaults when accessing another
process's DSEG in order to remove a trailer.
Covered under Segment Control.

50 emergency_shutdown (ring 0, perm-wired)
The procedure segment which starts an ESD. It is a
separate segment because BOS has to be able to find
it and transfer to it.

51 hardcore_sct_seg (ring 0)
The segment containing the static condition table
for ring zero; it's just like the one which is kept
in an outer ring stack header. The only static
handlers in ring zero are those used to invoke the
copy_on_write mechanism.

52 idle_dsegs (ring 0, perm-wired)
53 idle_pdses (ring 0, perm-wired)

Two similar segments: they contain the DSEG and PDS
segments for all the idle processes, all in a row.
When an idle process is constructed, the SDWs for
TYPICAL ADDRESS SPACE

its DSEG and PDS are set up to point into the middle of one of these segments.

54 init_processor (ring 0)
55 inzr_stk0 (ring 0)
The stack segment used by the Initializer during initialization and shutdown. During normal operation, the Initializer participates in ordinary ring zero stack sharing.

56 iobm_data (ring 0, perm-wired)
The database of iobm.pl1, the I/O Buffer Manager.

57 ioi_abs_seg (ring 0, abs-seg)
The abs-seg used by ioi_interrupt.pl1 to access a user ioi_buffer at interrupt time, for storing status information.

58 ioi_data (ring 0, perm-wired)
The database for ioi_ -- describes all user-accessible, or potentially user-accessible devices. See the programs in bound_io_active and bound_io_wired.

59 iom_data (ring 0, perm-wired)
Describes the configuration of the IOMs, and software information about IOM channels. Contains assignment information, software status queue location, and metering cells.

60 oc_data (ring 0, perm-wired)
The database for the ring zero operator's console mechanism. This is used by syser in ring zero, and by the Initializer to write on the system console (but not message coordinator consoles).

61 [pd]>pds (ring 0, deciduous)
The Process Data Segment. This contains all the miscellaneous information that makes a process unique to the supervisor, and need not be readily accessible to other processes. Most per-process variables are referenced symbolically, such as pds$processid, pds$page_fault_data (machine conditions for last page fault), etc.

62 >sll>prds (ring 0, deciduous)
Like the PDS, but per-processor. Contains the same sort of miscellaneous information, and is also used as the ring zero stack for certain types of faults (page faults, connects, and timer runouts) and all interrupts. There is a PRDS per processor, named
TYPICAL ADDRESS SPACE

>sl>cpu_A.prds, cpu_B.prds, etc. The PRDS segment in ring zero is changed at LDBR time to indicate the actual PRDS of the processor that the process is going to run on; thus, it's sort of an abs-seg. All PRDS's are wired, but have page tables.

65 >sl>pvt
   The Physical Volume Table. Described under Volume Management and Page Control.

66 rdisk_seg
   A PTW-type abs-seg (the only one where the PTW ever changes). Used only by the program read_disk, which does I/O to arbitrary pages on any disk, its aste.pvtx and PTW are switched around to indicate the right page, which is faulted on and (if needed) written back out by pc$cleanup.

67 restart_fault
70 return_to_ring_0
   These two procedures are used to implement the restarting of faults (such as a QUIT signal) from the user ring. When a fault occurs, a frame is pushed on the user ring stack, with its owner set to be return_to_ring_0_. Additionally, the machine conditions for the fault are saved in ring zero (in the PDS) so that when restart_fault is called to restart a possibly modified set of conditions, it can compare and validate.

71 scas
   The System Controller Addressing Segment. This segment has a page overlaid on a page of every system controller. No data is ever accessed through this segment; its page table is not even in the SST, but in the SCS. It is used only for certain privileged instructions which require an effective address in an particular SCU in order to read or set control registers in the SCU.

72 scs
   The System Configuration Segment. It describes most of the hardware configuration, and contains various control words used by privileged control instructions.

73 signaller
   The procedure which implements user ring fault signalling.
TYPICAL ADDRESS SPACE

74 >s11>sst_names_ (ring 0, deciduous)
The SST name table. This is a debugging feature; it contains (when in use) the primary name corresponding to every segment in the AST. It can be maintained online, by use of the PARM ASTK config card, but usually is not. It is always filled in by F8UMP after a crash if it was not already in use.

75 stack_0_data (ring 0, perm-wired)
Data segment describing the available segments for use in ring zero stack sharing. These segments, seen later on as segment 230, are recorded here and multiplexed among eligible processes.

76 stock_seg (ring 0, perm-wired)
The segment containing all in-core record and VTOCE stocks for use by page control and segment control, and covered under those topics.

77 >s11>sys_boot_info (ring 0, deciduous)
A data segment containing information about the I/O devices (tape and disk) used during bootload.

100 >s11>sys_info (ring 0, deciduous)
Contains assorted global wired information shared by the user ring and supervisor. Some is set during bootload, and some comes off the tape; none is modified after initialization is complete.

101 syserr_data (ring 0, perm-wired)
Data segment for the lowest level of the syserr mechanism. Syserr messages are built and queued here, and sent to the console. They are also copied out by the syserr logger hardcore process, into the syserr_log.

102 syserr_log (ring 0, abs-seg)
This segment overlays the LOG partition on some disk, which is used to reliably store syserr messages until they can be copied into the perm_syserr_log maintained in the Hierarchy.

103 tc_data (ring 0)
The traffic control data segment; contains all traffic control data. Covered under Traffic Control.

104 wired_hardcore_data (ring 0, perm-wired)
Miscellaneous data used by the wired supervisor.
TYPICAL ADDRESS SPACE

COLLECTION TWO

THE UNWIRED PORTION OF THE SUPERVISOR

READ IN BY COLLECTION ONE, DIRECTLY INTO PAGED SEGMENTS IN THE HARDCORE PARTITIONS

105 >s1>active_all_rings_data (ring 0, deciduous)
   Miscellaneous data shared between the unwired supervisor and the outer rings.

106 active_hardcore_data (ring 0)
   Miscellaneous data used by the unwired portion of the supervisor: system-wide locks, size constants for directory control, system search rule info, and metering for directory control and the dynamic linker.

107 >s1>admin_gate_ (ring 0, deciduous)

110 ast_lock_meter_seg (ring 0)
   A segment used to collect AST lock metering, normally off (enabled by the ast_lock_metering tuning parameter).

111 >s1>audit_gate_ (ring 0, deciduous)

112 bound_355_wired (ring 0)

113 bound_file_system (ring 0)

114 bound_HC_backup (ring 0)

115 bound_HC_reconfig (ring 0)

116 bound_HC_tuning (ring 0)

117 bound_imp_dim_ (ring 0)

120 bound_imp_status (ring 0)

121 bound_io_active (ring 0)

122 bound_mcs_util (ring 0)

123 bound.priv_mpX (ring 0)

124 bound_networkO_ (ring 0)

125 bound.priv_procs (ring 0)

126 bound_process_creation (ring 0)

127 bound_salvager (ring 0)

130 bound.scavenger (ring 0)

131 >s1>bound_sss_active_ (ring 0, deciduous)

132 bound_system_faults (ring 0)

133 boundtty_active (ring 0)

134 bound_vtoc_man (ring 0)

135 bound_x25_mpX (ring 0)
TYPICAL ADDRESS SPACE

136 dbm_seg (ring 0)
The segment used to hold the dumper bit maps for volumes being volume-dumped. The bitmaps are read from the volume header when a dump begins, and written back when finished.

137 dirlockt_seg (ring 0)
The segment used to keep track of all directory locks. Directory locks are kept in a supervisor segment, rather than in directories themselves.

140 \( \text{files} \text{dir} \text{seg} \) (ring 0, deciduous)
141 \( \text{files} \text{journal} \text{seg} \) (ring 0, deciduous)
142 \( \text{files} \text{error_table} \) (ring 0, deciduous)
143 fnp_dump_seg (ring 0)
Segment used for data buffering by FNP dump and patch operations (but not bootload).

144 hasp_mp
145 \( \text{files} \text{backup} \) (ring 0, deciduous)
146 \( \text{files} \text{seg} \) (ring 0, deciduous)
147 \( \text{files} \text{hpchs} \) (ring 0, deciduous)
150 ibm3270_mp
151 imp_data (ring 0)
152 imp_dim_buf (ring 0)
153 imp_tables (ring 0)
154 imp_wired_buffers (ring 0)
These four segments were used by the ring zero IMP DIM (part of the ARPAnet support), which has since been decommissioned.

155 initializer_abs_seg (ring 0, abs-seg)
An abs-seg used solely in order to copy a process's stack_0 segment into its process directory on process termination.

156 \( \text{files} \text{initializer} \text{gate} \) (ring 0, deciduous)
157 io_page_tables (ring 0)
The segment which contains page tables used for I/O if the IOM is operating in Paged mode. It is initialized by ioi_init and used only for ioi_1/0.

160 ioat (ring 0)
The I/O attach table. This is a largely obsolete database, the relic of an earlier I/O device attachment scheme. It is now used only for loading and dumping FNPs.

161 \( \text{files} \text{ioi} \) (ring 0, deciduous)
62 [path]kst_seg (ring 0, deciduous)
TYPICAL ADDRESS SPACE

The Known Segment Table. Described in Name & Address Space Management.

163 lvt (ring 0)
   The Logical Volume Table. Described in Volume Management.

164 >sll>mhcs_ (ring 0, deciduous)
165 ncp_tables_ (ring 0)
   Another part of the now-decommissioned ring zero ARPAnet support, no longer used.

166 >sll>net_ring0_admin_ (ring 0, deciduous)
167 >sll>net_ring0_sys_ (ring 0, deciduous)
170 >sll>net_ring0_user_ (ring 0, deciduous)
171 >sll>phcs_ (ring 0, deciduous)
172 polled_vip_mpx (ring 0)
173 pv_salv_seg (ring 0)
   This data segment is created in order to run the physical volume salvager (now rarely used). It contains various databases used by the salvager. It is created (by calling grab_aste.pl1) and destroyed for each volume salvage, in each process running a salvage, rather than being a shared segment.

174 salv_abs_seg_00 (ring 0, abs-seg)
175 salv_abs_seg_01 (ring 0, abs-seg)
176 salv_abs_seg_02 (ring 0, abs-seg)
177 salv_abs_seg_03 (ring 0, abs-seg)
200 salv_abs_seg_04 (ring 0, abs-seg)
   These five segments are used to overlay the VTOC of a volume being salvaged, and are set up and referenced by vm_vio.pl1. Covered under File System Salvagers.

201 salv_dir_space (ring 0)
202 salv_data (ring 0)
203 salv_temp_dir (ring 0)
   These three segments are used by the directory salvager when invoked as the online salvager, in response to a crawlout or bad_dir_condition. Only one instance of the online salvager may be running at a time, and this is ensured by a lock on salv_data. The online salvager does not interfere with demand directory salvages, however. Covered under File System Salvagers.

204 scavenger_data (ring 0)
   The segment containing the tables used when running the online volume scavenger. As many volumes as tables can be fit here may be scavenged at one time.
TYPICAL ADDRESS SPACE

The in-use portion of the segment is wired while a scavenge is being done. Covered under File System Salvagers.

205 $sl>shcs_  
206   str_seg   (ring 0, deciduous)

The segment trailer segment. The trailers in this segment record which processes have an SDW for any particular non-supervisor segment, and the backup_abs_seg (see above). Covered under Segment Control.

207 syserr_daemon_dseg (ring 0)
210 syserr_daemon_pds (ring 0)
211 syserr_daemon_stack (ring 0)

The DSEG, PDS, and ring zero stack of the syserr logging daemon. They are filled in when the daemon processes (which runs entirely in ring zero) is created. They are in the global system address space primarily for debugging and recordkeeping purposes.

212 $sl>system_privilege_  
213   $sl>tannd_  
214   template_pds   (ring 0, deciduous)

A template for the PDS, used when creating a process. It is a copy of the pds template which came off the system tape, but which was used to create the initializer’s PDS.

215 tty_area (ring 0)

An unwired database used by r i ng zero communications system, used primarily for saved metering information on each channel.

216 tty_buf (ring 0)

The important segment in the ring zero communications system. Contains the logical channel table, all communications I/O buffers, and all multiplexer databases.

217 tty_tables (ring 0)

An unwired database used by ring zero communications system to keep the tables used for input and output translation/conversion.

220 vtoc_buffer_seg (ring 0)

The segment containing all buffers for VTOC I/O, and a small amount of control information. Covered under Segment Control.
These segments are unused, since the first segment after the supervisor address space is the ring zero stack, and its segment number must be zero mod 8.

The ring zero stack. One is allocated from a pool (>s11>stack_0.016) whenever a process becomes eligible. When a process is not eligible (blocked, usually), it has no ring zero context, and needs no ring zero stack.

The rest of the stacks. There would be others if other rings were being used. A stack is created for each ring as it is needed; the segment number is automatically generated by the CALL6 instruction (from DBR.stack), and when a segment fault occurs on a stack not yet extant (pds$stacks(ring) is null), seg_fault.pl1 calls makestack.pl1 to create one.
TYPICAL ADDRESS SPACE

* NON-SUPERVISOR SEGMENTS

The remainder of a process address space.

Built by the normal name and address space management mechanisms as the process gets going and runs.

Different in different processes; this is only an example.

The root directory. The recursive nature of the initiate/segment fault mechanism ensures that this will be the first non-supervisor segment in the address space.

240 > (the root)

241 >pdd
242 >pdd
243 >ss>bound_process_init_
244 >udd
245 >udd>MED
246 >udd>MED>Sibert
247 [pd]>1BBBMjzkmcnbg.area.linker
250 >ss>bound_process_env_
251 >unb
252 >ss>bound_as_requests_
253 >ss>bound_info_rtns_
254 >ss>bound_command_loop_
255 >ss>bound_ipc_
256 >ss>bound_as_requests_
257 >ss>bound_error_handlers_
258 >ss>bound_ipc_
259 >ss>bound_info_rtns_
260 >ss>bound_command_loop_
261 >ss>bound_ipc_
262 >ss>bound_info_rtns_
263 >ss>bound_command_loop_
264 >ss>bound_ipc_
265 >ss>bound_info_rtns_
266 >ss>bound_command_loop_
267 >ss>bound_ipc_
268 >ss>bound_info_rtns_
269 >ss>bound_command_loop_
270 >ss>bound_ipc_
271 >ss>bound_info_rtns_
272 >ss>bound_ipc_
273 >ss>bound_info_rtns_

Not To Be Reproduced
TYPICAL ADDRESS SPACE

274  >scl>command_usage_counts
275  >scl>command_usage_counts>command_usage_list_
276  >scl>command_usage_counts>command_usage_totals_
277  >sss>bound_exec_com_
300  >ssl>error_table_
NAME/ADDRESS SPACE METERS

**system link meters**

- **SYSTEM_LINK_METERS** - RECORDS CPU TIME AND PAGING INFORMATION USED BY THE
  DYNAMIC LINKER IN ALL PROCESSES

### Linkage Meters:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Metering time</td>
<td>4:58:57</td>
</tr>
<tr>
<td>Total time in linker</td>
<td>0:50:53</td>
</tr>
<tr>
<td>Average time per link</td>
<td>6.01 msec.</td>
</tr>
<tr>
<td>Percentage of real time in linker</td>
<td>17.03</td>
</tr>
<tr>
<td>Percentage of CPU time in linker</td>
<td>4.66 %</td>
</tr>
</tbody>
</table>

#### Time slot (msec)

<table>
<thead>
<tr>
<th>Time slot (msec)</th>
<th>&lt;25</th>
<th>25-50</th>
<th>50-75</th>
<th>&gt;75</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calls</td>
<td>498469</td>
<td>8357</td>
<td>414</td>
<td>1100</td>
</tr>
<tr>
<td>Total time in slot</td>
<td>0:42:03</td>
<td>0:04:16</td>
<td>0:00:24</td>
<td>0:04:09</td>
</tr>
<tr>
<td>Percent total time</td>
<td>82.63</td>
<td>8.40</td>
<td>0.81</td>
<td>8.16</td>
</tr>
<tr>
<td>Percent total calls</td>
<td>98.06</td>
<td>1.64</td>
<td>0.08</td>
<td>0.22</td>
</tr>
<tr>
<td>Average time</td>
<td>5.06</td>
<td>30.68</td>
<td>59.63</td>
<td>226.59</td>
</tr>
<tr>
<td>Average page faults</td>
<td>0.18</td>
<td>2.29</td>
<td>6.34</td>
<td>5.65</td>
</tr>
</tbody>
</table>

#### Segment Search

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time</td>
<td>2.58</td>
</tr>
<tr>
<td>Average page faults</td>
<td>0.04</td>
</tr>
<tr>
<td>Percent time in slot</td>
<td>57.06</td>
</tr>
</tbody>
</table>

#### Get Linkage

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time</td>
<td>0.84</td>
</tr>
<tr>
<td>Average page faults</td>
<td>0.06</td>
</tr>
<tr>
<td>Percent time in slot</td>
<td>18.94</td>
</tr>
</tbody>
</table>

#### Definition Search

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time</td>
<td>0.24</td>
</tr>
<tr>
<td>Average page faults</td>
<td>0.02</td>
</tr>
<tr>
<td>Percent time in slot</td>
<td>5.26</td>
</tr>
</tbody>
</table>
NAME/ADDRESS SPACE METERS

**link meters**

- **Linkage Meters:** Records CPU time and paging information used by the dynamic linker in the process running it.

<table>
<thead>
<tr>
<th>Slot</th>
<th>Calls</th>
<th>Avg Time</th>
<th>Avg Pf</th>
<th>Tot Time</th>
<th>% Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;25</td>
<td>1245</td>
<td>4.689</td>
<td>0.3</td>
<td>5.838</td>
<td>76.1</td>
</tr>
<tr>
<td>25-50</td>
<td>34</td>
<td>30.201</td>
<td>2.9</td>
<td>1.027</td>
<td>13.4</td>
</tr>
<tr>
<td>50-75</td>
<td>6</td>
<td>62.226</td>
<td>6.2</td>
<td>0.373</td>
<td>4.9</td>
</tr>
<tr>
<td>&gt;75</td>
<td>2</td>
<td>216.545</td>
<td>6.0</td>
<td>0.433</td>
<td>5.6</td>
</tr>
<tr>
<td>Total</td>
<td>1287</td>
<td>5.961</td>
<td>0.4</td>
<td>7.671</td>
<td></td>
</tr>
</tbody>
</table>
NAME/ADDRESS SPACE COMMANDS

display kst entry

* DISPLAY_KST_ENTRY - DISPLAYS INFORMATION FROM A KST

! display_kst_entry >udd>Multics>Sibert

segno: 246 at 162 1270
usage: 7, 0, 0, 0, 0, 0, 0, 0
entryp: (245120750)
uid: 102401170050
dtbm: 446556324757
mode: 7 (0, -0, 0)
ex mode: 70000000000 (7, 7, 7) - 5MB
infcount: 3
hdr: 4
flags: dirsw write tms

rzd - Ring zero dump
rzd 142 270 10
segno offset 10007A1 words long
## TOPIC V
### Directory Control

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory Control Overview</td>
<td>5-1</td>
</tr>
<tr>
<td>Directory Control Terminology</td>
<td>5-3</td>
</tr>
<tr>
<td>Directory Control Data Bases</td>
<td>5-6</td>
</tr>
<tr>
<td>Directory Segments</td>
<td>5-6</td>
</tr>
<tr>
<td>Directory Header</td>
<td>5-9</td>
</tr>
<tr>
<td>Directory Entries</td>
<td>5-11</td>
</tr>
<tr>
<td>Directory Control Commands</td>
<td>5-15</td>
</tr>
<tr>
<td>display_branch</td>
<td>5-15</td>
</tr>
</tbody>
</table>
DIRECTORY CONTROL OVERVIEW

FUNCTION

DIRECTORY CONTROL IS A SET OF HARDCORE MODULES RESPONSIBLE FOR THE MAINTENANCE OF THE MULTICS DIRECTORY STRUCTURE -- IE: THE HIERARCHY

ITS TASKS INCLUDE CREATING, MANIPULATING AND INTERPRETING THE CONTENTS OF DIRECTORY SEGMENTS, TO INCLUDE:

ACCESS CONTROL LISTS (ACL'S), NAMES, AND VTOC POINTERs OF ENTRIES DESCRIBED THEREIN

ONLY DIRECTORY CONTROL IS ALLOWED TO ALTER THE CONTENTS OF DIRECTORY SEGMENTS

DIRECTORY CONTROL IMPLICITLY RELIES UPON THE SERVICES OF OTHER SUBSYSTEMS SUCH AS SEGMENT CONTROL AND PAGE CONTROL, AND ALSO INVOKES THEM DIRECTLY BY SUBROUTINE CALL

DIRECTORIES ARE SIMPLY SEGMENTS TO THESE SUBSYSTEMS

DIRECTORY CONTROL IS INVOKED ONLY BY SUBROUTINE CALLS

PRINCIPAL USER INTERFACES

COMMAND LEVEL
DIRECTORY CONTROL OVERVIEW

create, create_dir, link, set_acl, delete_acl, status, list, add_name, rename

SUBROUTINE LEVEL

hcs_sappend_branch, hcs_sadd_acl_entries, hcs_sappend_link, hcs_sdelete_acl_entries, hcs_sstatus_, hcs_schname_file

MAJOR DATA BASES

DIRECTORY SEGMENTS

CONTAIN THE ATTRIBUTES AND OTHER INFORMATION ABOUT THEIR SEGMENTS (NEEDED TO FIND SEGMENTS, RETURN STATUS INFORMATION, AND BUILD VTOCE'S AT SEGMENT CREATION)

THE DIRLOCKT_SEG

SEGMENT WHERE DIRECTORY LOCKING IS MANAGED
DIRECTORY CONTROL TERMINOLOGY

UNIQUE ID (UID): A 36-BIT ID (SERIAL NUMBER) ASSIGNED TO EVERY SEGMENT WHEN CREATED

SON: OF A DIRECTORY. AN IMMEDIATELY INFERIOR (SUBORDINATE) SEGMENT

SON'S LVID: OF A DIRECTORY. THE ID OF THE LOGICAL VOLUME ON WHICH THE DIRECTORY'S SONS RESIDE (AND WILL RESIDE)

GRANDSON: OF A DIRECTORY. A SEGMENT INFERIOR BY MORE THAN ONE HIERARCHICAL LEVEL

PARENT: OF A SEGMENT. THE "CONTAINING" DIRECTORY SEGMENT

ANCESTOR: OF A SEGMENT. THE PARENT, OR GRANDPARENT, OR GREAT GRANDPARENT, ETC.

BROTHER: OF A SEGMENT. ANOTHER SEGMENT HAVING THE SAME PARENT
DIRECTORY CONTROL TERMINOLOGY

BRANCH (1): IN A DIRECTORY. A DATA STRUCTURE, CONTAINED IN A DIRECTORY SEGMENT, THAT DESCRIBES AN IMMEDIATELY INFERIOR SEGMENT OR DIRECTORY (BUT NOT A LINK)

BRANCH (2): OF A DIRECTORY. REFERS TO THE ACTUAL SEGMENT OR DIRECTORY IMMEDIATELY INFERIOR TO THE DIRECTORY

ENTRY (1): IN A DIRECTOiTY. SAME AS BRANCH (1) BUT INCLUDES LINKS

ENTRY (2): OF A DIRECTORY. SAME AS BRANCH (2) BUT INCLUDES LINKS

STORAGE SYSTEM TERMINOLOGY
DIRECTORY CONTROL DATA BASES

DIRECTORY SEGMENTS

DIRECTORY SEGMENT STRUCTURE

Not To Be Reproduced 5-6 F80A
DIRECTORY CONTROL DATA BASES

DIRECTORY SEGMENTS

- DIRECTORY SEGMENTS ARE DISK RESIDENT (RLV) DATA BASES MAINTAINED BY DIRECTORY CONTROL

- ONE DIRECTORY SEGMENT PER DIRECTORY IN THE HIERARCHY

- DIRECTORY SEGMENTS CONTAIN A CATALOG OF STORAGE SYSTEM INFORMATION ABOUT OTHER SEGMENTS, DIRECTORIES AND LINKS

- ALL DIRECTORY SEGMENTS, BY CONVENTION, RESIDE ON THE ROOT LOGICAL VOLUME (RLV)

- DIRECTORY SEGMENTS ARE CREATED BY append MUCH LIKE NORMAL SEGMENTS ARE CREATED

- DIRECTORY SEGMENTS CONTAIN MANY INTER-RELATED COMPLEX DATA STRUCTURES TO INCLUDE THE FOLLOWING (NOT NECESSARILY CONTIGUOUS) REGIONS:

  - DIRECTORY HEADER

    - CONTAINS SELF DESCRIPTIVE INFORMATION LIKE UID, AIM CLASSIFICATION, ETC; AND POINTERS TO OTHER REGIONS

Not To Be Reproduced 5-7 F80A
DIRECTORY CONTROL DATA BASES

DIRECTORY SEGMENTS

<table>
<thead>
<tr>
<th>HASH TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used to quickly locate an entry, given its name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENTRY LIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains one descriptive data structure (an entry) for each segment, directory, or link immediately inferior to the directory (i.e., all entries)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PERSON_ID AND PROJECT_ID NAME LISTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains all person_id's/project_id's required to describe the ACL, the author, etc., of all segments and directories immediately inferior to the directory (i.e., all branches)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME LIST (ONE PER ENTRY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains all names currently assigned to the entry</td>
</tr>
<tr>
<td>Primary name is actually contained in entry structure itself</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACCESS CONTROL LIST (ONE PER ENTRY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains all ACL entries currently associated with the entry</td>
</tr>
</tbody>
</table>
## Directory Control Data Bases

### Directory Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Process ID of Last Modifier (Current Modifier)</td>
</tr>
<tr>
<td>1</td>
<td>Type of Object (Dir Header = 3)</td>
</tr>
<tr>
<td>2</td>
<td>6 Words Used by Salvager for Recording Date Time Checked and Errors Discovered</td>
</tr>
<tr>
<td>3</td>
<td>Directory's Unique Identifier (UID) +</td>
</tr>
<tr>
<td>4</td>
<td>Directory's Physical Volume ID (PVID) +</td>
</tr>
<tr>
<td>5</td>
<td>Sons' Logical Volume ID (LVID) +</td>
</tr>
<tr>
<td>6</td>
<td>Access Isolation Mechanism (AIM) Class +</td>
</tr>
<tr>
<td>7</td>
<td>Directory's VTOC X</td>
</tr>
<tr>
<td>8</td>
<td>Entry List Start R_PTR</td>
</tr>
<tr>
<td>9</td>
<td>Entry List End R_PTR</td>
</tr>
<tr>
<td>10</td>
<td>Person_ID List Start R_PTR</td>
</tr>
<tr>
<td>11</td>
<td>Person_ID List End R_PTR</td>
</tr>
<tr>
<td>12</td>
<td>Number of Seg Entries</td>
</tr>
<tr>
<td>13</td>
<td>Number of Link Entries</td>
</tr>
<tr>
<td>14</td>
<td>Allocation Area R_PTR</td>
</tr>
</tbody>
</table>

24 Words Used for Initial ACL Implementation

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Hash Table Size</td>
</tr>
<tr>
<td>46</td>
<td>Hash Table R_PTR</td>
</tr>
<tr>
<td>47</td>
<td>Number of Used Places in Hash Table</td>
</tr>
<tr>
<td>48</td>
<td>Depth of This Directory</td>
</tr>
<tr>
<td>49</td>
<td>Date Time Salvaged (DTS)</td>
</tr>
<tr>
<td>50</td>
<td>UID of Superior Master Dir</td>
</tr>
<tr>
<td>51</td>
<td>Modification Pseudo-Clock</td>
</tr>
<tr>
<td>52</td>
<td>12 Words Unused</td>
</tr>
<tr>
<td>53</td>
<td>Checksum (Not Used)</td>
</tr>
<tr>
<td>54</td>
<td>UID of Parent Dir</td>
</tr>
</tbody>
</table>

### Directory Header

A disk resident (LRV - Directory Segment) data base - one per directory

*Not wired. Not encacheable*

Not to be reproduced
DIRECTORY CONTROL DATA BASES

DIRECTORY HEADER

• The directory header is a disk resident data base contained at the beginning of a directory segment.

• One directory header per directory segment.

• The directory header contains self descriptive information such as:
  • The process id of the last process to modify the directory segment's contents.
  • The directory's UID, LVID, PVID, VTOC index, and AIM classification.
  • Relative pointers to the beginning and end of the entry list, person_id list, project_id list, and the hash table.
  • Hierarchy depth of the directory segment.
  • UID of the master directory and the parent directory.
  • Son's LVID - the id of the LV on which inferior non-directory segments reside (and will reside).

• The directory header is accessed at the beginning of directory query and update operations.
## DIRECTORY CONTROL DATA BASES

### DIRECTORY ENTRIES

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FORWARD R_PTR</td>
</tr>
<tr>
<td>1</td>
<td>BACKWARD R_PTR</td>
</tr>
<tr>
<td>2</td>
<td>TYPE OF ENTRY (SEG = 7)</td>
</tr>
<tr>
<td>3</td>
<td>SIZE OF ENTRY (127 WORDS)</td>
</tr>
<tr>
<td>4</td>
<td>ENTRY'S UNIQUE IDENTIFIER (UID)</td>
</tr>
<tr>
<td>5</td>
<td>DATE TIME ENTRY MODIFIED (OTEM)</td>
</tr>
<tr>
<td>6</td>
<td>NUMBER OF NAMES</td>
</tr>
<tr>
<td>7</td>
<td>NAME LIST START R_PTR</td>
</tr>
<tr>
<td>8</td>
<td>NAME LIST END R_PTR</td>
</tr>
<tr>
<td>9</td>
<td>AUTHOR'S PERSON_ID R_PTR</td>
</tr>
<tr>
<td>10</td>
<td>AUTHOR'S PROJECT_ID R_PTR</td>
</tr>
<tr>
<td>11</td>
<td>AUTHOR'S TAG</td>
</tr>
<tr>
<td>12</td>
<td>14 WORDS CONTAINING PRIMARY NAME</td>
</tr>
<tr>
<td>13</td>
<td>DATE TIME DUMPED (DTD)</td>
</tr>
<tr>
<td>14</td>
<td>BRANCH'S PHYSICAL VOLUME ID (PVID)</td>
</tr>
<tr>
<td>15</td>
<td>BRANCH'S VTOC INDEX</td>
</tr>
<tr>
<td>16</td>
<td>ENTRY POINT SOUND</td>
</tr>
<tr>
<td>17</td>
<td>ACCESS ISOLATION MECHANISM (AIM) CLASS</td>
</tr>
<tr>
<td>18</td>
<td>R(1)</td>
</tr>
<tr>
<td>19</td>
<td>R(2)</td>
</tr>
<tr>
<td>20</td>
<td>R(3)</td>
</tr>
<tr>
<td>21</td>
<td>XR(1)</td>
</tr>
<tr>
<td>22</td>
<td>XR(2)</td>
</tr>
<tr>
<td>23</td>
<td>XR(3)</td>
</tr>
<tr>
<td>24</td>
<td>ACL ENTRY COUNT</td>
</tr>
<tr>
<td>25</td>
<td>ACL START R_PTR</td>
</tr>
<tr>
<td>26</td>
<td>ACL END R_PTR</td>
</tr>
<tr>
<td>27</td>
<td>BC AUTHOR'S PERSON_ID R_PTR</td>
</tr>
<tr>
<td>28</td>
<td>BC AUTHOR'S PROJECT_ID R_PTR</td>
</tr>
<tr>
<td>29</td>
<td>BC AUTHOR'S TAG</td>
</tr>
<tr>
<td>30</td>
<td>BIT COUNT (IF DIR &gt; 0 IMPLIES MSF)</td>
</tr>
<tr>
<td>31</td>
<td>SONS' (NON-DIR') LOGICAL VOLUME ID (LVID)</td>
</tr>
<tr>
<td>32</td>
<td>CHECKSUM FROM DTD</td>
</tr>
<tr>
<td>33</td>
<td>UID OF PARENT DIRECTORY</td>
</tr>
</tbody>
</table>

---

### BRANCH ENTRY

A disk resident VTOC - directory segment/data base - one per branch in directory

(Not wired, not encacheable)

Any info not here is in VTOC

Not To Be Reproduced 5-11 F80A
DIRECTORY CONTROL DATA BASES

DIRECTORY ENTRIES

<table>
<thead>
<tr>
<th></th>
<th>FORWARD R_PTR</th>
<th>BACKWARD R_PTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TYPE OF ENTRY (LINK = 5)</td>
<td>SIZE OF ENTRY (69 WORDS)</td>
</tr>
<tr>
<td>1</td>
<td>LINK'S UNIQUE IDENTIFIER (UID) (NEVER VISIBLE TO USERS)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DATE TIME ENTRY MODIFIED (DTEM)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NUMBER OF NAMES</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>NAME LIST START R_PTR</td>
<td>NAME LIST END R_PTR</td>
</tr>
<tr>
<td>5</td>
<td>AUTHOR'S PERSON_ID R_PTR</td>
<td>AUTHOR'S PROJECT_ID R_PTR</td>
</tr>
<tr>
<td>6</td>
<td>AUTHOR'S TAG</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>34 WORDS CONTAINING PRIMARY NAME</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>DATE TIME DUMPED (DTD)</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>PATHNAME SIZE</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>42 WORDS CONTAINING THE ABSOLUTE PATHNAME OF THE LINK'S TARGET</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>UID CHECKSUM</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>UID OF PARENT DIRECTORY</td>
<td></td>
</tr>
</tbody>
</table>

LINK ENTRY

A DISK RESIDENT (RLV-DIRECTORY SEGMENT) DATA BASE – ONE PER LINK IN DIRECTORY

(NOT WIRED, NOT ENCACHEABLE)
DIRECTORY CONTROL DATA BASES

DIRECTORY ENTRIES

- The directory entry is a disk resident (RLV) data base contained within a directory segment.

- One directory entry (in the directory segment) for each immediately inferior entry in the hierarchy.

- Each directory entry is a data structure describing the attributes of a segment, directory or link.

- Directory entries come in two flavors:
  - Link entry, (38 or 72 words) containing:
    - Date time modified and dumped (by the hierarchy dumper, not volume dumper)
    - Relative pointers to the entry's name list and author's user id
    - Absolute pathname of the link's target
    - Uid of parent directory
DIRECTORY CONTROL DATA BASES

DIRECTORY ENTRIES

- BRANCH ENTRY, (38 WORDS) CONTAINING:
  - DATE TIME MODIFIED AND DUMPED (BY THE HIERARCHY DUMPER, NOT VOLUME DUMPER)
  - RELATIVE POINTERS TO THE ENTRY'S NAME LIST AND AUTHOR'S USER ID
  - BRANCH'S UID, PVID, AND VTOC INDEX
  - AIM CLASSIFICATION, ENTRY POINT BOUND, RING BRACKETS, AND RELATIVE POINTERS TO THE ACL
  - BRANCH'S BIT COUNT AND BIT COUNT AUTHOR
  - SON'S LVID (IF A DIRECTORY) AND PARENT'S UID
  - FLAGS DESCRIBING VARIOUS STATES AND PROPERTIES OF THE ENTRY SUCH AS: DIRECTORY, MASTER DIRECTORY, SECURITY OUT OF SERVICE, COPY AND SAFETY SWITCH, ETC
DIRECTORY CONTROL COMMANDS

display_branch

- DISPLAY_BRANCH - DISPLAYS BRANCHES IN THE DIRECTORY HIERARCHY

It is often useful on conjunction with DUMP_VTOCE old command

! display_branch >udd>Multics>Sibert

Branch for Sibert in >udd>Multics at 245|20742

UID 102401170050, is vtock 63 on root4 (of log vol. root)
Sibert is a directory.
Ring brackets (0 0 0)
Entry modified 02/23/83 1912.1 est Wed
Dumped 03/20/83 0955.6 est Sun
9 names.

- DISPLAY IS NOT COMPLETE, SO A RING_ZERO_DUMP OF THE SAME DATA IS INCLUDED

! ring_zero_dump >udd>Multics 20742 46 -ch

To find branch or segment to delete, if you can't delete with MV D or delete

020742 021310020604 000004000046 102401170050 44656324757 .....88.x(
020746 400000000011 020752021152 001720000532 172000000000 .....j...z
020752 021010000000 000056000016 020742020233 000000000000
020756 123151142145 162164404040 040040040040 040040040040 Sibert
020762 040040040040 040040040040 040040040040 040040040040
020766 000000000000 102401170050 446563247026 000000000000 ...B.X(...
020772 135244026001 000063000000 400000000000 000000000000 ]...3........
020776 000000000000 000770000012 02117021300 001720000532 ...........
021002 172000000000 225072707470 000000000000 000000000000 z...........
021006 033023254650 000000000000

To find branch or segment to delete, if you can't delete with MV D or delete

prz 245 20744 0

Not To Be Reproduced 5-15 UID

(End Of Topic)
## Volume Management

### Volume Management Overview
- The New Storage System
- Volume Management Terminology
- Volume Management Data Bases
  - Volume Label
  - Volume Map
  - Dumper Bit Map
  - VTOC Map
  - Physical Volume Table (PVT)
  - Logical Volume Table (LVT)
  - Physical Volume Hold Table
### Volume Management Operations
- Acceptance of Physical Volumes
- Demounting of Physical Volumes
- Logical Volume Management
### Volume Management Commands
- `print_configuration_deck`
- `list_vols`
- `display_label`
- `display_pvte`
### Volume Management Meters
- `disk_meters`
- `device_meters`
- `disk_queue`
FUNCTION

VOLUME MANAGEMENT IS RESPONSIBLE FOR THE MANAGEMENT OF PHYSICAL AND LOGICAL VOLUMES

ITS TASKS INCLUDE:

- ACCEPTANCE AND DEMOUNTING OF PHYSICAL VOLUMES
- MAINTAINING THE ASSOCIATION BETWEEN PHYSICAL VOLUMES, LOGICAL VOLUMES, AND DISK DRIVES
- ENSURING THE INTEGRITY OF VOLUME CONTENTS
- MAKING VOLUME CONTENTS ACCESSIBLE TO PAGE CONTROL (PAGES) AND SEGMENT CONTROL (VTOC ENTRIES)

VOLUME MANAGEMENT IS INVOKED ONLY BY SUBROUTINE CALLS
VOLUME MANAGEMENT OVERVIEW

MAJOR DATA BASES

- PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM
- PHYSICAL VOLUME TABLE ENTRY (PVTE) - ONE PER DISK DRIVE KNOWN TO THE SYSTEM
  - EACH PVTE IDENTIFIES A DRIVE'S DEVICE NUMBER, SUBSYSTEM NAME, DEVICE TYPE, AND INFORMATION ABOUT THE PHYSICAL VOLUME CURRENTLY MOUNTED
  - USED TO MAP REFERENCES TO PAGES OF SEGMENTS INTO AN I/O REQUEST TO THE CORRECT DISK DRIVE

- LOGICAL VOLUME TABLE (LVT) - ONE PER SYSTEM
- LOGICAL VOLUME TABLE ENTRY (LVTE) - ONE PER MOUNTED LOGICAL VOLUME
  - EACH LVTE CONTAINS THE LOGICAL VOLUME ID, POINTERS TO MEMBER PVTE'S, AIM CLASS LIMITS, ETC.
  - USED TO DETERMINE A USER'S ACCESS TO A LOGICAL VOLUME (PRIVATE OR PUBLIC) AND TO LOCATE MEMBER PHYSICAL VOLUMES

- VOLUME HEADER - ONE PER PACK
- VOLUME LABEL (REGISTRATION AND ACCEPTANCE INFORMATION)
- VOLUME MAP (OCCUPIED/VACANT INFORMATION FOR VOLUME CONTENTS)
VOLUME MANAGEMENT OVERVIEW

RECORD STOCKS - ONE PER MOUNTED VOLUME

ONLINE CACHE OF INFORMATION ABOUT USED / UNUSED RECORDS ON THE VOLUME

THIS INFORMATION IS DERIVED FROM THE VOLUME MAP, BUT KEPT ONLINE TO AVOID THE NECESSITY OF REFERRING TO THE VOLUME MAP ON DISK EVERY TIME A RECORD IS ALLOCATED OR FREED

WHEN THE CACHE BECOMES COMPLETELY EMPTY OR COMPLETELY FULL, IT MUST BE UPDATED FROM/TO DISK - A PROTOCOL ENSURES THAT THE COPY ON DISK IS ALWAYS CONSISTENT

PROVIDED BY VOLUME MANAGEMENT, BUT USED BY PAGE CONTROL

VTOCE STOCKS - ONE PER VOLUME

SIMILAR TO RECORD STOCKS, BUT MAINTAINS INFORMATION ABOUT USED / UNUSED VTOC ENTRIES ON THE VOLUME

PROVIDED BY VOLUME MANAGEMENT, BUT USED BY SEGMENT CONTROL

PHYSICAL VOLUME HOLD TABLE (PVHT) - ONE PER SYSTEM

RECORDS THE COMMENCEMENT OF COMPOUND I/O OPERATIONS UPON A PHYSICAL VOLUME

THIS INFORMATION PREVENTS A VOLUME FROM BEING DEMOUNTED WHILE SUCH AN OPERATION IS IN PROGRESS
THE NEW STORAGE SYSTEM

LOGICAL VOLUME A
- PHYSICAL VOLUME (PV): A DISK PACK
- LOGICAL VOLUME (LV): A COLLECTION OF ONE OR MORE PHYSICAL VOLUMES GIVEN A SINGLE LOGICAL IDENTITY
- VOLUME HEADER: CONTAINS THE LABEL, VOLUME MAP, VTOC MAP, AND VOLUME DUMP MAPS.
- VOLUME TABLE OF CONTENTS (VTOC): AN ARRAY OF ENTRIES (VTOCE's) DESCRIBING EACH SEGMENT ON THE PHYSICAL VOLUME
- PAGING REGION: THE REGION IN WHICH ALL RECORDS DESCRIBED IN THE VTOC RESIDE (I.E., PAGES OF STORAGE SYSTEM SEGMENTS)
- PARTITION: A REGION SET ASIDE FOR SOME PURPOSE OTHER THAN PAGES OF STORAGE SYSTEM SEGMENTS

LOGICAL VOLUME B
- PHYSICAL VOLUME (PV): A DISK PACK
- LOGICAL VOLUME (LV): A COLLECTION OF ONE OR MORE PHYSICAL VOLUMES GIVEN A SINGLE LOGICAL IDENTITY
- VOLUME HEADER: CONTAINS THE LABEL, VOLUME MAP, VTOC MAP, AND VOLUME DUMP MAPS.
- VOLUME TABLE OF CONTENTS (VTOC): AN ARRAY OF ENTRIES (VTOCE's) DESCRIBING EACH SEGMENT ON THE PHYSICAL VOLUME
- PAGING REGION: THE REGION IN WHICH ALL RECORDS DESCRIBED IN THE VTOC RESIDE (I.E., PAGES OF STORAGE SYSTEM SEGMENTS)
- PARTITION: A REGION SET ASIDE FOR SOME PURPOSE OTHER THAN PAGES OF STORAGE SYSTEM SEGMENTS
THE NEW STORAGE SYSTEM

Since Release 4.0, the Multics Storage System has been organized into physical and logical volumes having the following properties:

- A physical volume (PV) is a disk pack (mounted or not) containing:
  - A label identifying itself - including a physical volume ID (PVID)
  - A volume map describing which pages and VTOCs are in use and which are free.
  - A volume table of contents (VTOC) describing which segments are resident therein - and the exact location of each of their pages.
  - The pages of resident segments (assigned to records of 1024 words in size)
  - And optionally: contiguous regions called partitions, set aside for special use (FDUMP images, hardcore paging, etc)

- All pages of a segment reside on a given physical volume

- That is: each non-zero page of a segment is assigned to a record of the physical volume

- The pair of physical volume ID (PVID) and VTOC index uniquely identifies any segment in the storage system hierarchy

Not To Be Reproduced
THE NEW STORAGE SYSTEM

A LOGICAL VOLUME (LV) CONSISTS OF ONE OR MORE PHYSICAL VOLUMES, WHICH ARE:

- GIVEN ONE LOGICAL VOLUME ID (LVID)
- ALWAYS MOUNTED AS A SET
- OFFSPRING (SONS, GRANDSONS, ETC) OF A DIRECTORY (NORMALLY) RESIDE WITHIN A GIVEN LOGICAL VOLUME

IN OTHER WORDS, A SUB-TREE (NORMALLY) SPANS NO MORE THAN ONE LOGICAL VOLUME

DIRECTORY SEGMENTS ARE AN EXCEPTION TO THE ABOVE AS ALL DIRECTORY SEGMENTS ARE ASSIGNED TO A LOGICAL VOLUME OF THEIR OWN CALLED THE "ROOT LOGICAL VOLUME" (RLV)

THE PHYSICAL VOLUME CONTAINING THE ROOT DIRECTORY IS CALLED "THE ROOT PHYSICAL VOLUME" (RPV)

THE RLV IS SPECIAL BECAUSE IT MUST ALWAYS BE MOUNTED, AND IT CONTAINS ALL DIRECTORY SEGMENTS, BUT IT ALSO CONTAINS OTHER SEGMENTS

SHOULD THE GROWING OF A SEGMENT CAUSE A PHYSICAL VOLUME TO BECOME FULL, A "SEGMENT MOVE" IS AUTOMATICALLY INITIATED

THIS IS ONE OF THE MOST COMPLEX AND EXPENSIVE SERVICES PERFORMED BY THE SYSTEM - BUT HAPPENS VERY INFREQUENTLY

Not To Be Reproduced

6-6

F80A
SHOULD A LOGICAL VOLUME BECOME FULL:

USER AND SYSTEM ERROR MESSAGES ARE GENERATED

SPACE MUST BE OBTAINED ON THE LOGICAL VOLUME BY ADDING MORE PHYSICAL VOLUMES OR BY DELETING OR MOVING SEGMENTS FROM THE LOGICAL VOLUME

BECAUSE IT CONTAINS ALL DIRECTORY SEGMENTS, SPACE ON THE RLV IS CRITICAL: IF IT IS USED UP, THE SYSTEM MAY NOT BE ABLE TO CONTINUE OPERATION.

THE CHOICE OF WHICH PHYSICAL VOLUME TO USE WHEN CREATING A SEGMENT IS MADE IN SUCH A WAY AS TO TRY TO BALANCE THE ALLOCATED SPACE ON ALL THE PHYSICAL VOLUMES OF A LOGICAL VOLUME.
THE NEW STORAGE SYSTEM

VOLUME LABEL
(Registration Information)

VOLUME MAP
(Used Records Map)

VTOC HEADER AND DUMPER BIT MAP

BAD TRACK LIST

VTOCE's
(one per resident segment: ≈ 15,000 VTOCE's)
(five per record: ≈ 3,000 records)

Low Partitions (if any)
PAGES OF SEGMENTS
(one per record: ≈ 35,000 records)

Partitions (if any)
(special-use regions . . . such as FDUMP images)

38,247
(records)

PHYSICAL VOLUME FORMAT
(MSU 451)

Not To Be Reproduced 6-8 F30A
# Hierarchy to Storage System Mapping

## Physical Volume ID 14

<table>
<thead>
<tr>
<th>CONTENTS (Simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PV 14</code></td>
</tr>
<tr>
<td>HEADER INFO</td>
</tr>
<tr>
<td>&gt; file map</td>
</tr>
<tr>
<td>VTOCE 1 udd file map</td>
</tr>
<tr>
<td>VTOCE 2 udd file map</td>
</tr>
<tr>
<td>VTOCE 3 udd file map</td>
</tr>
<tr>
<td>VTOCE 4 udd file map</td>
</tr>
<tr>
<td>VTOCE 5 udd file map</td>
</tr>
<tr>
<td>VTOCE 6 udd file map</td>
</tr>
<tr>
<td>VTOCE 7 udd file map</td>
</tr>
<tr>
<td>VTOCE 8 udd file map</td>
</tr>
<tr>
<td>PHYSICAL VOLUME ID 14</td>
</tr>
</tbody>
</table>

## Physical Volume ID 29

<table>
<thead>
<tr>
<th>CONTENTS (Simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PV 29</code></td>
</tr>
<tr>
<td>HEADER INFO</td>
</tr>
<tr>
<td>my_seq file map</td>
</tr>
<tr>
<td>start_up.ac file map</td>
</tr>
<tr>
<td>Proj_A file map</td>
</tr>
<tr>
<td>test file map</td>
</tr>
<tr>
<td>test file map</td>
</tr>
<tr>
<td>Smith.mbx file map</td>
</tr>
<tr>
<td>PHYSICAL VOLUME ID 29</td>
</tr>
</tbody>
</table>

---

Not To Be Reproduced 6-9 F80A
THE NEW STORAGE SYSTEM

If one knows where the root directory is, all segments in the MULTICS hierarchy can be found (assuming the availability of all required physical volumes).

Major design point:

Many disk resident data bases (to include the pages of segments) are copied into main memory and written back to disk at such times as:

- System start-up/shut-down
- Physical volume mounting/demounting
- Segment activation/deactivation
- Page faults

While in main memory, the memory resident copy is considered to be the copy.

While in main memory, the disk resident copy is considered to be (and often is) wholly invalid.
## MULTICS DISK PACK STATISTICS (MSU 451)

* The above figures appear inconsistent if data formatting is not considered (e.g.: some disk space is not used)
LOGICAL VOLUME/MASTER DIRECTORY RELATIONSHIP

* NOTE: ALL DIRECTORY SEGMENTS RESIDE ON ONE DESIGNATED LOGICAL VOLUME CALLED THE ROOT LOGICAL VOLUME (RLV)
VOLUME MANAGEMENT TERMINOLOGY

MOUNT: TO PHYSICALLY PLACE A DISK PACK ON A DRIVE AND CYCLE UP
THE DRIVE. (PERFORMED BY THE OPERATOR, NOT BY
SOFTWARE)

ACCEPT: AFTER MOUNTING, TO ESTABLISH IN THE SUPERVISOR THE
BINDING BETWEEN THE DRIVE AND THE PHYSICAL VOLUME
MOUNTED

PUBLIC: A LOGICAL VOLUME ATTRIBUTE INDICATING THAT THE VOLUME
IS ATTACHED TO ALL PROCESSES (BY DEFAULT) WHEN ACCEPTED

PRIVATE: A LOGICAL VOLUME ATTRIBUTE INDICATING THAT THE VOLUME
IS ATTACHED ONLY TO REQUESTING PROCESSES (SUBJECT TO
ACCESS CONTROLS)

PARTITION: A REGION WITHIN A PHYSICAL VOLUME SET ASIDE FOR SPECIAL
USE

RECORD: A LOGICAL UNIT OF DISK SPACE, 1024 CONTIGUOUS WORDS IN
SIZE. (NUMBERED/ADDRESSED FROM ZERO)

SECTOR: A LOGICAL UNIT OF DISK SPACE, 64 CONTIGUOUS WORDS IN
SIZE. THE SMALLEST ADDRESSABLE UNIT OF DISK SPACE. A
RECORD CONTAINS 16 SECTORS
VOLUME MANAGEMENT TERMINOLOGY

PAGE: A 1024 WORD EXTENT OF DATA STARTING AT A 1024 WORD BOUNDARY OF A SEGMENT. SEGMENTS MUST BE AN INTEGER NUMBER OF PAGES IN SIZE. A PAGE CAN RESIDE IN ONE OR MORE OF THE FOLLOWING LOCATIONS:

MAIN MEMORY FRAME

DISK RECORD
VOLUME MANAGEMENT DATA BASES

VOLUME LABEL

320 WORDS (5 SECTORS) NOT USED (GCOS LABEL REGION)

8 WORDS: "Multics Storage System Volume"

8 WORDS CONTAINING MANUFACTURE'S SERIAL NUMBER

8 WORDS CONTAINING PHYSICAL VOLUME NAME

PHYSICAL VOLUME IDENTIFIER (PVID)

LOGICAL VOLUME IDENTIFIER (LVID)

PVID OF THE ROOT PHYSICAL VOLUME

TIME REGISTERED

VOLUME SIZE (NO. RECORDS)

VTOC SIZE (NO. RECORDS)

MAXIMUM AIM CLASSIFICATION IN VOLUME

MINIMUM AIM CLASSIFICATION IN VOLUME

18 WORDS NOT USED

Time stamp updated

Equal when volume format changed

If incorrect, then disk not mounted, t.e. : endless crash.

Not To Be Reproduced 6-15 F80A
VOLUME MANAGEMENT DATA BASES

VOLUME LABEL

6-0
VOLMAP VERSION (OVERLAYS OLD TIME-UNMOUNTED)
TIME SALVAGED
TIME OF BOOTLOAD
TIME UNMOUNTED

8 WORDS USED BY VOLUME DUMPER
48 WORDS UNUSED

7-0
VTOC INDEX OF "-" DIRECTORY
SHUTDOWN STATE
DISK_TABLE_VTOC INDEX
DISK_TABLE UNIQUE ID
E32 STATE
VOLUME MAP ORIGIN RECORD
VOLUME MAP SIZE
VTOC MAP ORIGIN RECORD
VTOC MAP SIZE
VOLUME MAP SECTION SIZE
VTOC ORIGIN RECORD
DUMPER BIT MAP RECORD
VOLUME INCONSISTENCY COUNT

52 WORDS UNUSED

8 - 3
NUMBER OF PARTITIONS
PARTITION 1 NAME
PARTITION 1 LOCATION
PARTITION 1 SIZE

188 WORDS DESCRIBING PARTITIONS 2 THRU 47

320 WORDS IS SECTORS NOT USED

15 - 63
VOLUME LABEL
A DISK RESIDENT DATA BASE - ONE PER PHYSICAL VOLUME

Not To Be Reproduced 6-16 F80A
THE VOLUME LABEL IS A DISK RESIDENT DATA BASE OCCUPYING THE FIRST MULTICS RECORD OF EACH STORAGE SYSTEM PHYSICAL VOLUME

ONE VOLUME LABEL PER PHYSICAL VOLUME

THE LABEL IS GENERATED BY init_disk_pack (init_empty_root IF RPV LABEL) AND CONTAINS REGISTRATION AND STATUS INFORMATION

THE LABEL IS INSPECTED WHEN THE VOLUME IS ACCEPTED AND UPDATED WHEN DEMOUNTED

THE LABEL IS DIVIDED INTO SIX SECTORS

GCOS REGION (SECTORS 0 TO 4)

SKIPPED OVER BY MULTICS TO AVOID ACCIDENTAL OVERWRITING OF GCOS PACKS AND ALLOW FOR FUTURE COMPATABILITY

PERMANENT REGION (SECTOR 5)

CONTAINS PERMANENT PER-PV INFORMATION (EG: MANUFACTURERS SERIAL NUMBER)
VOLUME MANAGEMENT DATA BASES

VOLUME LABEL

- DYNAMIC INFORMATION REGION (SECTOR 6)
  - CONTAINS INFORMATION RELATING TO THE MOST RECENT MOUNTING OF THE PV (E.G.: LAST MOUNT TIME)
  - ALLOWS THE STORAGE SYSTEM TO ENSURE THE INTEGRITY OF THE PV

- ROOT INFORMATION REGION (SECTOR 7)
  - DEFINED ONLY FOR THE ROOT PHYSICAL VOLUME
  - CONTAINS DYNAMIC INFORMATION ABOUT THE ENTIRE STORAGE SYSTEM (E.G.: SHUT DOWN STATE, PAGING DEVICE STATE, ETC)

- PARTITION MAP (SECTOR 8)
  - IDENTIFIES THE LOCATION AND LENGTH OF ANY RESIDENT PARTITIONS

- UNUSED (SECTORS 9 TO 13)
VOLUME MANAGEMENT DATA BASES

VOLUME MAP

VOLUME MAP
A DISK RESIDENT DATA BASE – ONE PER PHYSICAL VOLUME

Not To Be Reproduced 6-19 F80A
THE VOLUME MAP IS A DISK RESIDENT DATA BASE

- OCCUPIES RECORDS 1, 2, AND 3, IMMEDIATELY FOLLOWING THE VOLUME LABEL

- ONE VOLUME MAP PER PHYSICAL VOLUME


- THIS INFORMATION IS ALSO DERIVABLE FROM AN ANALYSIS OF THE VTOC (AT CONSIDERABLE EXPENSE) - THIS IS DONE WHEN THE VOLUME IS SCAVENGED OR SALVAGED.

RECORDS ARE TAKEN FROM THE VOLUME MAP DURING OPERATION AND PLACED IN THE RECORD STOCK

- RECORDS ARE ALLOCATED BY PAGE CONTROL FROM THE RECORD STOCK

- VOLUME MAP ON DISK IS ALWAYS CONSISTENT

- RECORDS MARKED FREE ON DISK ARE GUARANTEED TO BE FREE, AND SAFE TO RE-USE
VOLUME MANAGEMENT DATA BASES

VOLUME MAP

Never reverse

Records marked as allocated may not actually be in-use, if a crash occurred and destroyed the stock contents.

This situation is benign, and corrected by a scavenger or salvage at some convenient time.

Records freed are placed back in the stock.

If stock fills, it is written back to the volume map.

Consistency is ensured by complex protocol in page control.

Record stock mechanism replaces FSMAP segments in pre-MR10.0 systems.
### VOLUME MANAGEMENT DATABASES

**DUMPER BIT MAP**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Number of VTOCES</td>
</tr>
<tr>
<td>1</td>
<td>Number of Free VTOCES</td>
</tr>
<tr>
<td>2</td>
<td>Number of Words Used in Bit Map</td>
</tr>
<tr>
<td>3</td>
<td>Number of Last Record in VTOC</td>
</tr>
<tr>
<td>4</td>
<td>4 Words Unused</td>
</tr>
<tr>
<td>8</td>
<td>VTOC Bit Map Entry 0</td>
</tr>
<tr>
<td>1023</td>
<td>VTOC Bit Map Entry 1015</td>
</tr>
<tr>
<td>2047</td>
<td>VTOC Bit Map Entry 2039</td>
</tr>
</tbody>
</table>

### VOLUME DUMPER BIT MAP

**Records 4 and 5**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8 Words Unused</td>
</tr>
<tr>
<td>8</td>
<td>Volume Dumper Bit Map Entry 0</td>
</tr>
<tr>
<td>2047</td>
<td>Volume Dumper Bit Map Entry 2039</td>
</tr>
</tbody>
</table>

### VTOC MAP

**Record 6**
THE DUMPER BIT MAP DESCRIBES WHICH VTOCES ON THE VOLUME HAVE BEEN VOLUME DUMPED

ONE PER PHYSICAL VOLUME

OCCUPIES RECORDS 4 AND 5, IMMEDIATELY FOLLOWING VOLUME MAP

SEPARATE BIT MAPS FOR INCREMENTAL AND CONSOLIDATED VOLUME DUMPS
THE VTOC MAP DESCRIBES THE LOCATION AND SIZE OF THE VTOC, AND CONTAINS A BIT MAP OF VTOC ALLOCATIONS.

- OCCUPIES RECORD 6, IMMEDIATELY FOLLOWING THE DUMPER BIT MAP
- VTOC FOLLOWS, STARTING AT RECORD 8
- RECORD 7 IS UNUSED

VTOCES ARE TAKEN FROM THE VTOCE MAP AND PLACED IN AN ONLINE STOCK.

- VTOCE STOCK IS LIKE RECORD STOCK, BUT LESS CRITICAL
  - VTOCES ARE SELF-IDENTIFYING AS TO WHETHER THEY ARE IN USE OR NOT, SO IT IS NOT NECESSARY TO MAINTAIN PERFECT CONSISTENCY IF A CRASH OCCURS
  - VTOCE MAP IS ALSO REBUILT BY SCAVENGE OR SALVAGE OPERATIONS

- VTOCE MAP REPLACES VTOCE FREE LIST IN PRE-MR10.0 SYSTEMS
## VOLUME MANAGEMENT DATA BASES

<table>
<thead>
<tr>
<th>NUMBER OF ENTRIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIMUM NUMBER OF ENTRIES</td>
</tr>
<tr>
<td>NUMBER OF ENTRIES IN USE</td>
</tr>
</tbody>
</table>

### 5 WORDS FOR SHUTDOWN STATUS

| TIME OF BOOTLOAD |

### 9 WORDS OF INFORMATION ABOUT RPV AND RLV

### 21 WORDS OF GLOBAL INFORMATION AND METERING CELLS FOR VOLUME MAPS AND VTOC MAPS

### ARRAY OF PHYSICAL VOLUME TABLE ENTRIES

#### PHYSICAL VOLUME TABLE ENTRY (PVTE)
**ONE PER PHYSICAL VOLUME LOCATED IN PHYSICAL VOLUME TABLE**

<table>
<thead>
<tr>
<th>PHYSICAL VOLUME ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGICAL VOLUME ID</td>
</tr>
</tbody>
</table>

| DISK SUBSYSTEM NAME: dsk01 |

<table>
<thead>
<tr>
<th>DISK DRIVE TYPE</th>
<th>DISK DRIVE NUMBER</th>
</tr>
</thead>
</table>

### 9 WORDS FOR VOLUME MAP AND VTOC MAP INFORMATION

| VOLUME TROUBLE COUNT | SCAVENGER INFO POINTER |

---

**PHYSICAL VOLUME TABLE HEADER**

**A HARDCORE (WIRED) DATABASE - ONE PER SYSTEM**
THE PHYSICAL VOLUME TABLE (PVT) IS A HARDCORE, WIRED, PAGED DATA BASE MAINTAINED BY VOLUME MANAGEMENT

ONE PVT PER SYSTEM

THE PVT IS THE MOST IMPORTANT DATA BASE OF VOLUME MANAGEMENT, AND CONTAINS AN ARRAY OF PHYSICAL VOLUME TABLE ENTRIES (PVTE'S)

ONE PVTE PER DISK DRIVE KNOWN TO THE SYSTEM

EACH PVTE DESCRIBES:

A DISK DRIVE CONFIGURED TO THE SYSTEM

INCLUDING THE DEVICE NUMBER, DEVICE TYPE, SUBSYSTEM NAME AND OTHER INFORMATION NEEDED BY THE DISK DIM

THE PHYSICAL VOLUME CURRENTLY MOUNTED ON THE DISK DRIVE

INCLUDING THE PVID, LVID, AND OTHER INFORMATION TAKEN FROM THE VOLUME HEADER, VOLUME MAP AND VTOC MAP, NEEDED BY PAGE AND SEGMENT CONTROL (THE PV & LV NAMES ARE NOT RECORDED HERE)

RECORD AND VTOCE STOCKS ARE LOCATED FROM THE PVT, BUT KEPT IN THE stock_seg
## Logical Volume Table (LVT)

<table>
<thead>
<tr>
<th>Word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Max LVTE Index</td>
</tr>
<tr>
<td>1</td>
<td>High Water LVTE Index</td>
</tr>
<tr>
<td>2</td>
<td>First Free LVTE Pointer</td>
</tr>
<tr>
<td>3</td>
<td>4 Words Not Used</td>
</tr>
<tr>
<td>4-68</td>
<td>64 Words of LVTE Pointers Used as a Hash Table</td>
</tr>
</tbody>
</table>

- **Logical Volume ID (LVID)**
- **Minimum AIM Classification in Logical Volume**
- **Maximum AIM Classification in Logical Volume**
- **Allocation Algorithm Value**

### Logical Volume Table Entry (LVTE)

- One per mounted logical volume

### Logical Volume Management Data Bases

A hardcore data base - one per system

---

Not To Be Reproduced 6-27 F80A
THE LOGICAL VOLUME TABLE (LVT) IS A HARDCORE, PAGED DATA BASE MAINTAINED BY VOLUME CONTROL

ONE LVT PER SYSTEM

THE LVT CONTAINS AN ARRAY OF LOGICAL VOLUME TABLE ENTRIES (LVTE'S)

ONE LVTE FOR EACH MOUNTED LOGICAL VOLUME

EACH LVTE DESCRIBES THE LOGICAL VOLUME TO INCLUDE:

LVID AND AIM CLASSIFICATION

RELATIVE POINTER TO THE THREAD OF PVTE'S OF ACCEPTED PHYSICAL VOLUMES THAT ARE MEMBERS OF THE LOGICAL VOLUME
THE LVT IS REQUIRED AT THE FOLLOWING TIMES:

- SEGMENT CREATION
- SEGMENT MOVING TIME
- VOLUME MOUNTING AND DEMOUNTING
- INITIATION AND SEGMENT FAULT TIME (FOR PUBLIC/PRIVATE CHECK)
### PHYSICAL VOLUME HOLD TABLE

<table>
<thead>
<tr>
<th>WORD</th>
<th>PVT R_PTR (1)</th>
<th>APTE R_PTR (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### PHYSICAL VOLUME HOLD TABLE

An internal static array in get_pvtx - one per system

- The physical volume hold table (PVHT) is a hardcore data base maintained by volume management

- One PVHT per system

- The PVHT identifies the physical volume and the process ID of process that has started (and has not yet completed) compound operations upon the physical volume

- This information prevents a volume from being demounted while such an operation is in progress
INTERRUPTION OF A COMPOUND OPERATION CAUSES THE VOLUME TO BE MARKED AS CONTAINING AN INCONSISTENCY

FOR CRASH ANALYSIS, sst.pvthp CONTAINS A POINTER TO THIS TABLE
THE ACCEPTANCE OF PHYSICAL VOLUMES IS THE MOST IMPORTANT AND FUNDAMENTAL OPERATION OF VOLUME MANAGEMENT.

PHYSICAL VOLUME ACCEPTANCE IS ACCOMPLISHED BY CALLING initializer_gate_$accept_fs_disk

THE ROOT PHYSICAL VOLUME (RPV) IS ACCEPTED IN A SPECIAL FASHION DURING COLLECTION 2 OF BOOTLOAD.

THE RPV IS THE ONLY PV REQUIRED TO BOOTLOAD THE SYSTEM (MORE OF THE RLV WILL BE ACCEPTED BY RING ZERO DURING BOOTLOAD IF POINTED TO BY THE "ROOT" CONFIGURATION CARD).

ACCEPTANCE INCLUDES:

- VALIDATE THAT THE DISK PACK MOUNTED IS THE PACK REQUESTED BY THE OPERATOR OR REQUESTING PROCESS VIA label.pvid
- DETERMINE THAT THE DISK PACK MOUNTED IS IN FACT A MEMBER OF THIS HIERARCHY VIA label.root_pvid
- INITIALIZING THE APPROPRIATE PVTE WITH DATA FROM THE LABEL, VOLUME MAP, AND VTOC MAP
VOLUME MANAGEMENT OPERATIONS

ACCEPTANCE OF PHYSICAL VOLUMES

INITIALIZING THE INITIAL CONTENTS OF THE RECORD STOCK AND VTOC STOCKS

DETERMINING IF ANY VOLUME INCONSISTENCIES ARE PRESENT, AND LOGGING THIS INFORMATION

VOLUME INCONSISTENCIES ARE CAUSED BY EVENTS WHICH MAY MEAN THAT THE DISK RESIDENT COPY OF THE VOLUME MAP OR VTOC MAP IS INCONSISTENT:

- A crash without ESD - indicated by label.time_map_updated and label.time_unmounted being unequal, detected at acceptance time
- An inconsistency detected online, such as an invalid VTOC bit map or a reused address
- An I/O error when writing the volume map or VTOC map during normal operation

A count is kept in the label, and updated as necessary

Normally, inconsistencies are merely logged, and left for the site to take care of at some convenient time

If an RLV volume claims only a very small number of free pages, a volume salvage is done automatically to try to recover any lost due to the inconsistency, since a full RLV will cause system crashes.
VOLUME MANAGEMENT OPERATIONS

ACCEPTANCE OF PHYSICAL VOLUMES

<table>
<thead>
<tr>
<th>WRITING OUT THE LABEL TO UPDATE label.time_map_updated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTE: label.time_map_updated AND label.time_unmounted ARE NOW UNEQUAL</td>
</tr>
<tr>
<td>THIS INEQUALITY IMPLIES THAT THE VOLUME HAS NOT BEEN PROPERLY SHUT DOWN, AND WILL BE MARKED INCONSISTENT IF ACCEPTED AGAIN IN THIS STATE</td>
</tr>
<tr>
<td>MARKING THE PUTE AS &quot;IN USE&quot; (LAST STEP)</td>
</tr>
</tbody>
</table>

Not To Be Reproduced 6-34 F80A
VOLUME MANAGEMENT OPERATIONS

DEMOUNTING OF PHYSICAL VOLUMES

• The demounting of physical volumes involves reversing all of the steps taken at acceptance time.

• Demounting is complicated by the fact that the PV may be in use at the time.

• Demounting is accomplished by calling demount_pv ("the demounter").

• All volumes are demounted at shutdown time.

• Demounting includes:

  » Turning on pvte.being_demounted and waiting for all compound operations to terminate.

  » Deactivating all segments from the PV which are active. This includes:

    » Flushing main memory and paging device (if present) of all relevant pages.
VOLUME MANAGEMENT OPERATIONS

DEMENTURING OF PHYSICAL VOLUMES

1. UPDATING THE VTOC'S FROM THE ASTE'S AND PAGE TABLES

2. FLUSHING THE VTOC MANAGER'S BUFFER SEGMENT OF ALL RELEVANT VTOCE-PARTS

3. EMPTYING THE RECORD AND VTOCE STOCKS BACK INTO THE VOLUME MAP AND VTOC MAP

4. UPDATING THE VOLUME LABEL FROM THE PVTE, PARTICULARLY

   - label.time_unmounted, label.time_map_updated, AND
   - label.inconsistency_count

5. PHYSICALLY CYCLING DOWN THE DISK DRIVE

   NOT DONE AT SYSTEM SHUTDOWN, HOWEVER

ONLY ONE PV MAY BE DEMOUNTED AT A TIME
LOGICAL VOLUME MANAGEMENT INCLUDES:

- MAINTAINING THE LOGICAL VOLUME TABLE (LVT) TO REFLECT THE STATE OF THE LOGICAL VOLUMES

- MAINTAINING, IN THE KNOWN SEGMENT TABLE (KST) OF EACH PROCESS, A TABLE OF PRIVATE LOGICAL VOLUMES MOUNTED TO THE PROCESS

- ANSWERING THE QUESTION OF WHETHER OR NOT A GIVEN LOGICAL VOLUME IS MOUNTED TO THE CALLING PROCESS

- OR, IF A PUBLIC LV, MOUNTED AT ALL (TO THE SYSTEM)

- PROVIDING THE HEAD OF THE PVT CHAIN FOR A GIVEN LV, FOR THE SEGMENT CREATION FUNCTION
VOLUME MANAGEMENT COMMANDS

print configuration deck

PRINT CONFIGURATION DECK - DISPLAYS >sli>config_deck, WHICH CONTAINS INFORMATION ABOUT DISK LOCATIONS, THE RLV, AND PARTITIONS

.. ONLY THE PART OF THE CONFIG DECK RELEVANT TO VOLUME MANAGEMENT AND DISK CONFIGURATION IS SHOWN HERE


part bos dska 16.
part dump dska 16.
part log dska 16.

prph dska a 20. 2 451. 16.
chnl dska a 26. 2 b 24. 2 b 22. 2

prph dskb b 20. 2 0 16. 451. 16.
chnl dskb b 26. 2 a 24. 2 a 22. 2

prph dskc a 28. 2 501. 32.
chnl dskc a 30. 2 b 30. 2 b 28. 2

prph dske b 32. 2 451. 8.
chnl dske b 34. 2

prph dskf a 32. 2 501. 16.
chnl dskf a 34. 2

mpc mspa 451. a 20. 4 a 24. 4
mpc mspb 451. b 20. 4 b 24. 4
mpc mspc 607. a 28. 4
mpc spsd 607. b 28. 4
mpc mspe 451. b 32. 4
mpc mspf 607. a 32. 4
VOLUME MANAGEMENT COMMANDS

print configuration deck

* DISK CONFIGURATION CONFIG CARDS

  | ROOT
  |
  | IDENTIFIES THOSE VOLUMES IN THE ROOT LOGICAL VOLUME WHICH HAVE HQ PARTITIONS, USED BY THE SUPERVISOR FOR PAGING OF SUPERVISOR SEGMENTS

  | PART
  |
  | IDENTIFIES THE LOCATIONS OF CERTAIN IMPORTANT PARTITIONS
  |
  | ONLY PARTITIONS NECESSARY FOR MULTICS OPERATIONS ARE IDENTIFIED, NOT ALT PARTITIONS
  |
  | HC PARTITIONS ARE LOCATED BY THE ROOT CARD

  | PRPH DSKn, CHNL
  |
  | IDENTIFY PHYSICAL I/O CHANNEL PATHS FOR ACCESSING DISK DRIVES

  | MPC
  |
  | IDENTIFY PHYSICAL CONNECTIONS TO MICROPROGRAMMED DISK CONTROLLERS

Not To Be Reproduced 6-39 F80A
# VOLUME MANAGEMENT COMMANDS

## list vols

list_vols displays a table of online volumes, their location, and space utilization.

## Table: Volume Management Details

<table>
<thead>
<tr>
<th>Drive</th>
<th>Records</th>
<th>Left %</th>
<th>VTOCEs Left</th>
<th>Avg PV</th>
<th>PV</th>
<th>LV Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dskc_17</td>
<td>64504</td>
<td>54730</td>
<td>85</td>
<td>13440</td>
<td>11356</td>
<td>84</td>
</tr>
<tr>
<td>dskc_18</td>
<td>64504</td>
<td>55389</td>
<td>86</td>
<td>13440</td>
<td>11352</td>
<td>84</td>
</tr>
<tr>
<td>dskc_05</td>
<td>36428</td>
<td>5305</td>
<td>15</td>
<td>8400</td>
<td>2662</td>
<td>32</td>
</tr>
<tr>
<td>dskc_06</td>
<td>36428</td>
<td>4323</td>
<td>12</td>
<td>8400</td>
<td>2365</td>
<td>28</td>
</tr>
<tr>
<td>dskb_19</td>
<td>36428</td>
<td>4632</td>
<td>13</td>
<td>8400</td>
<td>2813</td>
<td>33</td>
</tr>
<tr>
<td>dskb_26</td>
<td>36429</td>
<td>13690</td>
<td>38</td>
<td>8400</td>
<td>4326</td>
<td>52</td>
</tr>
<tr>
<td>dskb_27</td>
<td>36429</td>
<td>4672</td>
<td>13</td>
<td>8400</td>
<td>2333</td>
<td>28</td>
</tr>
<tr>
<td>dskc_01</td>
<td>36308</td>
<td>4680</td>
<td>13</td>
<td>9000</td>
<td>450</td>
<td>5</td>
</tr>
<tr>
<td>dskc_03</td>
<td>36268</td>
<td>3588</td>
<td>10</td>
<td>5200</td>
<td>1017</td>
<td>11</td>
</tr>
<tr>
<td>dskc_04</td>
<td>36268</td>
<td>4500</td>
<td>12</td>
<td>9200</td>
<td>884</td>
<td>10</td>
</tr>
<tr>
<td>dskc_05</td>
<td>36268</td>
<td>4196</td>
<td>13</td>
<td>9200</td>
<td>864</td>
<td>9</td>
</tr>
<tr>
<td>dskb_17</td>
<td>36269</td>
<td>4221</td>
<td>12</td>
<td>9200</td>
<td>1002</td>
<td>11</td>
</tr>
<tr>
<td>dskb_18</td>
<td>36268</td>
<td>3840</td>
<td>11</td>
<td>9200</td>
<td>539</td>
<td>6</td>
</tr>
<tr>
<td>dskc_13</td>
<td>64504</td>
<td>294</td>
<td>0</td>
<td>13440</td>
<td>5539</td>
<td>41</td>
</tr>
<tr>
<td>dskc_14</td>
<td>64504</td>
<td>269</td>
<td>0</td>
<td>13440</td>
<td>5214</td>
<td>39</td>
</tr>
<tr>
<td>dskc_01</td>
<td>64503</td>
<td>4363</td>
<td>68</td>
<td>13440</td>
<td>10032</td>
<td>75</td>
</tr>
<tr>
<td>dskc_02</td>
<td>64503</td>
<td>4550</td>
<td>71</td>
<td>13440</td>
<td>9838</td>
<td>73</td>
</tr>
<tr>
<td>dskc_03</td>
<td>64503</td>
<td>4237</td>
<td>66</td>
<td>13440</td>
<td>9839</td>
<td>73</td>
</tr>
<tr>
<td>dskc_04</td>
<td>64503</td>
<td>4355</td>
<td>68</td>
<td>13440</td>
<td>5785</td>
<td>73</td>
</tr>
<tr>
<td>dskc_05</td>
<td>64504</td>
<td>58010</td>
<td>90</td>
<td>13440</td>
<td>12394</td>
<td>92</td>
</tr>
<tr>
<td>dskc_10</td>
<td>64504</td>
<td>56786</td>
<td>88</td>
<td>13440</td>
<td>12607</td>
<td>92</td>
</tr>
<tr>
<td>dskc_21</td>
<td>64503</td>
<td>23947</td>
<td>37</td>
<td>13440</td>
<td>6446</td>
<td>48</td>
</tr>
<tr>
<td>dskc_22</td>
<td>64503</td>
<td>2374</td>
<td>37</td>
<td>13440</td>
<td>6194</td>
<td>46</td>
</tr>
<tr>
<td>dskc_26</td>
<td>64503</td>
<td>23794</td>
<td>37</td>
<td>13440</td>
<td>6185</td>
<td>46</td>
</tr>
<tr>
<td>dskc_30</td>
<td>64503</td>
<td>21111</td>
<td>37</td>
<td>13440</td>
<td>6481</td>
<td>48</td>
</tr>
<tr>
<td>dskc_07</td>
<td>64503</td>
<td>11723</td>
<td>16</td>
<td>13440</td>
<td>6149</td>
<td>46</td>
</tr>
<tr>
<td>dskc_08</td>
<td>64503</td>
<td>11665</td>
<td>18</td>
<td>13440</td>
<td>6429</td>
<td>48</td>
</tr>
<tr>
<td>dskc_23</td>
<td>64504</td>
<td>9777</td>
<td>15</td>
<td>13440</td>
<td>6094</td>
<td>45</td>
</tr>
<tr>
<td>dskc_24</td>
<td>64504</td>
<td>11805</td>
<td>18</td>
<td>13440</td>
<td>6141</td>
<td>46</td>
</tr>
<tr>
<td>dskc_25</td>
<td>64504</td>
<td>11514</td>
<td>18</td>
<td>13440</td>
<td>7407</td>
<td>55</td>
</tr>
<tr>
<td>dskc_26</td>
<td>64504</td>
<td>12958</td>
<td>20</td>
<td>13440</td>
<td>7149</td>
<td>53</td>
</tr>
<tr>
<td>dskc_11</td>
<td>37089</td>
<td>8053</td>
<td>22</td>
<td>5100</td>
<td>3797</td>
<td>74</td>
</tr>
<tr>
<td>dskc_12</td>
<td>37562</td>
<td>6046</td>
<td>16</td>
<td>2735</td>
<td>957</td>
<td>35</td>
</tr>
<tr>
<td>dskc_07</td>
<td>37309</td>
<td>6825</td>
<td>18</td>
<td>4000</td>
<td>2107</td>
<td>53</td>
</tr>
<tr>
<td>dskc_08</td>
<td>36205</td>
<td>3827</td>
<td>11</td>
<td>7000</td>
<td>491</td>
<td>7</td>
</tr>
<tr>
<td>dskc_11</td>
<td>36205</td>
<td>8597</td>
<td>24</td>
<td>7000</td>
<td>4181</td>
<td>60</td>
</tr>
<tr>
<td>dskc_16</td>
<td>31283</td>
<td>2892</td>
<td>9</td>
<td>9000</td>
<td>3476</td>
<td>39</td>
</tr>
<tr>
<td>dskb_23</td>
<td>31208</td>
<td>4888</td>
<td>13</td>
<td>7000</td>
<td>455</td>
<td>7</td>
</tr>
<tr>
<td>dskb_24</td>
<td>36209</td>
<td>3097</td>
<td>9</td>
<td>7000</td>
<td>238</td>
<td>3</td>
</tr>
<tr>
<td>dskb_25</td>
<td>36350</td>
<td>4483</td>
<td>12</td>
<td>7000</td>
<td>223</td>
<td>3</td>
</tr>
</tbody>
</table>

Note: To Be Reproduced

---

6-40  
F80A
DISPLAY LABEL - DISPLAYS THE LABEL OF A STORAGE SYSTEM VOLUME BY READING IT FROM DISK

USED / FREE INFORMATION IS COPY ON DISK, AND THEREFORE OUT OF DATE WITH RESPECT TO THE PVTE

Label for Multics Storage System Volume rpv on dska_01 d451
PVID                      220531524345
Serial                 rpv
Logical Volume     root
LVID                    220531524466
Registered               01/28/81 1249.5
Dismounted               03/15/83  0741.9
Map Updated              03/15/83  0744.6
Salvaged                 10/01/82   0300.3
Bootload                 03/15/83  0743.5
Reloaded                  01/28/81  1510.1
Dumped                   03/17/83   2153.0
Incremental              03/16/83  2359.3
Consolidated             03/15/83  2353.0
Complete                 03/15/83
Inconsistencies         0
Minimum AIM              0:000000
Maximum AIM              7:777777

Volume contains Root (>) at vtocx 0
disk_table_ at vtocx 100 (uid 033022210261)

Volume Map from Label

<table>
<thead>
<tr>
<th>First Rec</th>
<th>(Octal)</th>
<th>Size</th>
<th>Label Region</th>
<th>VTOC Region</th>
<th>hc Partition</th>
<th>Paging Region</th>
<th>bos Partition</th>
<th>alt Partition</th>
<th>Total Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>3730</td>
<td>2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4016</td>
<td>7660</td>
<td>33901</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37917</td>
<td>112035</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38117</td>
<td>112345</td>
<td>141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>38258</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Not To Be Reproduced
VOLUME MANAGEMENT COMMANDS

display pvte

DISPLAY_PVTE - DISPLAYS THE PVT ENTRY OF A STORAGE SYSTEM VOLUME

PARTITION INFORMATION IS NOT DETAILED IN THE PVT, BUT USED/FREE INFORMATION IS COMPLETELY UP TO DATE

PVTE for Multics Storage System Volume rpv on dska_01 d451 at pvt|50
PVID 220531524345
LVID 220531524466

VTOCEs
Number 10000
Left 3323

Records
Number 33901
Left 3796
Inconsistencies 0

Volume Map
volmap_seg ASTE 15|4420
record stock 76|100
Page 0 - Base 7660
Free 3364
Page 1 - Base 103660
Free 3740
Page 2 - Base 203660
Free 0
vtoce stock 76|2400

ON: storage_system permanent hc_part_used
OFF: being_mounted being_demounted being_demounted2 scav_check_address device_inoperative vacating dmpr_in_use(incr) dmpr_in_use(cons) dmpr_in_use(comp)

Volume Map from PVTE

<table>
<thead>
<tr>
<th>First Rec (Octal)</th>
<th>Size</th>
<th>Label Region</th>
<th>Size</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>2000</td>
<td>2008</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>3740</td>
<td>33901</td>
<td>38258</td>
<td></td>
</tr>
<tr>
<td>4016</td>
<td>7660</td>
<td>38258</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37917</td>
<td>112035</td>
<td>6-42</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Not To Be Reproduced 6-42 F80A
VOLUME MANAGEMENT METERS

**disk meters**

**DISK_METERS - DISPLAYS I/O ACTIVITY TO DISK DRIVES**

[Image of a table showing disk meter statistics]

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Count</th>
<th>Waits</th>
<th>%Waits</th>
<th>Avg. Wait (ms.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>call locks</td>
<td>26005</td>
<td>217</td>
<td>0.83</td>
<td>0.259</td>
</tr>
<tr>
<td>run locks</td>
<td>112</td>
<td>0</td>
<td>0.00</td>
<td>0.000</td>
</tr>
<tr>
<td>interrupt locks</td>
<td>25998</td>
<td>239</td>
<td>0.92</td>
<td>0.208</td>
</tr>
<tr>
<td>allocations</td>
<td>26001</td>
<td>0</td>
<td>0.00</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drive</th>
<th>Reads</th>
<th>Writes</th>
<th>Seek Distance</th>
<th>ATB Reads</th>
<th>ATB Writes</th>
<th>ATB I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>269</td>
<td>67</td>
<td>214</td>
<td>4508</td>
<td>18102</td>
<td>3609</td>
</tr>
<tr>
<td>3</td>
<td>362</td>
<td>243</td>
<td>109</td>
<td>3350</td>
<td>4991</td>
<td>2004</td>
</tr>
<tr>
<td>4</td>
<td>309</td>
<td>131</td>
<td>184</td>
<td>3925</td>
<td>9258</td>
<td>2756</td>
</tr>
<tr>
<td>5</td>
<td>547</td>
<td>165</td>
<td>180</td>
<td>2217</td>
<td>7350</td>
<td>1703</td>
</tr>
<tr>
<td>6</td>
<td>631</td>
<td>165</td>
<td>161</td>
<td>1922</td>
<td>7350</td>
<td>1523</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>5843</td>
<td>2187</td>
<td>122</td>
<td>207</td>
<td>554</td>
<td>151</td>
</tr>
<tr>
<td>9</td>
<td>366</td>
<td>116</td>
<td>153</td>
<td>3313</td>
<td>10455</td>
<td>2516</td>
</tr>
<tr>
<td>11</td>
<td>3501</td>
<td>1431</td>
<td>200</td>
<td>346</td>
<td>847</td>
<td>245</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>7158</td>
<td>2508</td>
<td>135</td>
<td>169</td>
<td>483</td>
<td>125</td>
</tr>
</tbody>
</table>
**VOLUME MANAGEMENT METERS**

`device meters`

DEVICE METERS - DISPLAYS SUMMARY OF I/O ACTIVITY FOR ALL DISK SUBSYSTEMS

<table>
<thead>
<tr>
<th>Total metering time</th>
<th>0:20:13</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dska</strong></td>
<td><strong>dskb</strong></td>
</tr>
<tr>
<td>Prior Page I/O</td>
<td>18571</td>
</tr>
<tr>
<td>ATB</td>
<td>65.334</td>
</tr>
<tr>
<td>Other Page I/O</td>
<td>6525</td>
</tr>
<tr>
<td>ATB</td>
<td>185.949</td>
</tr>
<tr>
<td>ATB Page I/O</td>
<td>46.347</td>
</tr>
<tr>
<td>Prior VTOCE I/O</td>
<td>934</td>
</tr>
<tr>
<td>ATB</td>
<td>1299.061</td>
</tr>
<tr>
<td>ATB I/O</td>
<td>46.612</td>
</tr>
<tr>
<td>% Busy</td>
<td>76</td>
</tr>
<tr>
<td>Avg. Page Wait</td>
<td>47.289</td>
</tr>
<tr>
<td>Avg. Page Wait</td>
<td>176.082</td>
</tr>
<tr>
<td>Avg VTOCE Wait</td>
<td>41.138</td>
</tr>
<tr>
<td>Avg. Page I/O T</td>
<td>35.619</td>
</tr>
<tr>
<td>Avg. VTOCE I/O T</td>
<td>31.139</td>
</tr>
<tr>
<td>EDAC Corr. Errors</td>
<td>0</td>
</tr>
<tr>
<td>Errors</td>
<td>0</td>
</tr>
<tr>
<td>Fatal Errors</td>
<td>0</td>
</tr>
</tbody>
</table>
VOLUME MANAGEMENT METERS

**disk queue**

DISK QUEUE - DISPLAYS I/O QUEUE FOR A DISK SUBSYSTEM

ONLY ONE SUBSYSTEM SHOWN HERE TO CONSERVE SPACE

Connects = 2604781, 1359725, 677321, 309367, 123430, 40159, 10227, 1969.

<table>
<thead>
<tr>
<th>P</th>
<th>RW</th>
<th>VP</th>
<th>DV</th>
<th>SECTOR</th>
<th>MEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>24</td>
<td>1350330</td>
<td>2730400</td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>9</td>
<td>1020150</td>
<td>4432000</td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>16</td>
<td>1204130</td>
<td>3624600</td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>16</td>
<td>314370</td>
<td>2730600</td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>16</td>
<td>314430</td>
<td>3416600</td>
</tr>
</tbody>
</table>
TOPIC VII
Segment Control

Segment Control Overview .............................................. 7-1
Segment Control Terminology ......................................... 7-4
Segment Control Data Bases .......................................... 7-5
Volume Table of Contents (VTOC) .................................... 7-5
Active Segment Table (AST) .......................................... 7-11
Services of Segment Control ........................................... 7-25
Creating Segments ....................................................... 7-25
Segment Fault ............................................................. 7-28
Segment Activation ..................................................... 7-28
Segment Trailers ......................................................... 7-30
Boundsfault Handling .................................................. 7-32
Segment Deactivation .................................................. 7-35
Summary of Major Services ........................................... 7-37
Encacheability ............................................................. 7-38
Truncating Segments .................................................... 7-38
Deleting Segments ....................................................... 7-40
Other Services ............................................................ 7-42
Segment Control Meters ................................................ 7-44
file_system_meters .................................................... 7-44
vtoc_buffer_meters ..................................................... 7-46
Segment Control Commands ............................................. 7-57
print_aste_ptp ........................................................... 7-47
dump_vtoc ................................................................. 7-48
SEGMENT CONTROL OVERVIEW

FUNCTION

SEGMENT CONTROL IS RESPONSIBLE FOR THE MANAGEMENT OF LOGICAL MEMORY

ITS TASKS INCLUDE:

- MAINTAINING THE DISK RESIDENT MAPS OF SEGMENTS (IE: THEIR VTOC'S)

- SEGMENT CREATION, TRUNCATION AND DELETION

- SEGMENT ACTIVATION AND DEACTIVATION (ASTE MULTIPLEXING)

SEGMENT CONTROL CAN BE INVOKED EITHER BY SUBROUTINE CALLS OR BY SEGMENT FAULTS

BASIC PHILOSOPHY OF ACTIVATION/DEACTIVATION

OF ALL SEGMENTS RESIDENT WITHIN THE SYSTEM'S MOUNTED PHYSICAL VOLUMES, ONLY A SMALL SUBSET WILL REQUIRE ACCESSING AT ANY ONE TIME. SUCH SEGMENTS WILL BE CALLED "ACTIVE SEGMENTS"

A PART OF MAIN MEMORY, CALLED THE "ACTIVE SEGMENT TABLE" (AST), WILL BE RESERVED TO HOLD MANAGEMENT INFORMATION FOR THESE ACTIVE SEGMENTS (IDENTITY, PVT INDEX, LOCATION OF PAGES, ETC.)
SEGMENT CONTROL OVERVIEW

As segments fall into disuse, their "management information" in the AST will be replaced with information of other segments requiring activation.

USER INTERFACE

COMMAND LEVEL

create, delete, truncate, etc.

SUBROUTINE LEVEL

hcs_sappend_branch, hcs_sappend_branchx, hcs_sdelete_entry_seg, hcs_sdelete_entry_file, hcs_struncate_seg, hcs_struncate_file, hcs_sforce_write, etc.

MAJOR DATA BASES

SYSTEM SEGMENT TABLE (SST) - One per system, shared with page control. One major component is "owned" by segment control:

ACTIVE SEGMENT TABLE (AST) - One per system

The AST is a list of active (currently being used) segments

ACTIVE SEGMENT TABLE ENTRY (ASTE) - One per active segment

ASTEs contain physical volume ID's (PVID'S) and VTOC INDEX'S (VTOCK'S) of segments. Needed by segment control to find the segment on disk (hardware)
SEGMENT CONTROL OVERVIEW

| AST HASH TABLE
| ALLOWS EFFICIENT SEARCHING OF ASTE'S
| LOGICALLY PART OF THE AST, BUT ELSEWHERE FOR HISTORICAL REASONS

| DIRECTORY SEGMENTS
| CONTAIN LOCATIONS AND ATTRIBUTES OF SEGMENTS. LOCATION INFORMATION FROM DIRECTORY SEGMENTS IS PROVIDED TO SEGMENT CONTROL BY DIRECTORY CONTROL

| VOLUME TABLE OF CONTENTS (VTOC) - ONE PER PHYSICAL VOLUME
| VOLUME TABLE OF CONTENTS ENTRY (VTOCE) - ONE PER DISK-RESIDENT SEGMENT
| EACH VTOCE CONTAINS THE SEGMENT'S UNIQUE ID, CURRENT LENGTH, FILE MAP, ETC (NEED TO BUILD ASTE'S AND PT'S)
| VTOCES ARE READ AND WRITTEN ONLY BY SEGMENT CONTROL

| VTOCE STOCKS - FROM VOLUME MANAGEMENT
| USED WHEN CREATING AND DELETING VTOCES FOR SEGMENTS
SEGMENT CONTROL TERMINOLOGY

MULTIPLEXING: CONTROLLED SHARING OF A REUSABLE RESOURCE

VTOC: VOLUME TABLE OF CONTENTS (ONE PER PV). AN ARRAY OF VTOC'S IDENTIFYING ALL SEGMENTS RESIDENT ON THE PHYSICAL VOLUME

VTOCE: VOLUME TABLE OF CONTENTS ENTRY (ONE PER RESIDENT SEGMENT). CONTAINS IDENTIFICATION AND LOCATOR INFORMATION ABOUT A SEGMENT RESIDENT WITHIN THE PHYSICAL VOLUME

SEGMENT: A COLLECTION OF INFORMATION (PROCEDURE OR DATA) GROUPED TOGETHER UNDER THE SAME ACCESS CONTROL CONSTRAINTS. EACH SEGMENT IS GIVEN ONE OR MORE NAMES AND A COLLECTION OF ATTRIBUTES INCLUDING LENGTH, ACCESS PERMISSIONS, ETC

SEMI-PERMANENT ACTIVATION: A SEGMENT AND TURNING ON ITS ENTRY HOLD SWITCH (aste. ehs) PREVENTING NORMAL (ASTE CONTENTION) DEACTIVATION
SEGMENT CONTROL DATA BASES
VOLUME TABLE OF CONTENTS (VTOC)

FILE MAP CHECKSUM
10 WORDS OF QUOTA RELEVANT INFORMATION TO INCLUDE
THE TIME-RECORD PRODUCT AND USAGE COUNT

FILE MAP (0)
FILE MAP (1)
FILE MAP (22)
FILE MAP (223)
FILE MAP (224)
FILE MAP (225)
FILE MAP (254)
FILE MAP (255)
10 WORDS NOT USED

Connection Failure, directory entry + trace broken.
### Volume Table of Contents (VTOC)

#### Segment Control Data Bases

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>154-156</td>
<td>Date Time Dumped (DTDI)</td>
</tr>
<tr>
<td>156-158</td>
<td>Volume ID (1)</td>
</tr>
<tr>
<td>157-159</td>
<td>Volume ID (2)</td>
</tr>
<tr>
<td>158-160</td>
<td>Volume ID (3)</td>
</tr>
<tr>
<td>160-162</td>
<td>Volume ID (4)</td>
</tr>
<tr>
<td>161-163</td>
<td>UID of Master Directory</td>
</tr>
<tr>
<td>164-166</td>
<td>UID of Superior Directory (1)</td>
</tr>
<tr>
<td>165-167</td>
<td>UID of Superior Directory (2)</td>
</tr>
<tr>
<td>166-168</td>
<td>UID of Superior Directory (3)</td>
</tr>
<tr>
<td>167-169</td>
<td>UID of Superior Directory (4)</td>
</tr>
<tr>
<td>168-170</td>
<td>UID of Superior Directory (5)</td>
</tr>
<tr>
<td>170-172</td>
<td>UID of Superior Directory (6)</td>
</tr>
<tr>
<td>171-173</td>
<td>UID of Superior Directory (7)</td>
</tr>
<tr>
<td>172-174</td>
<td>UID of Superior Directory (8)</td>
</tr>
<tr>
<td>173-175</td>
<td>UID of Superior Directory (9)</td>
</tr>
<tr>
<td>174-176</td>
<td>UID of Superior Directory (10)</td>
</tr>
<tr>
<td>175-177</td>
<td>UID of Superior Directory (11)</td>
</tr>
<tr>
<td>176-178</td>
<td>UID of Superior Directory (12)</td>
</tr>
<tr>
<td>177-179</td>
<td>UID of Superior Directory (13)</td>
</tr>
<tr>
<td>178-180</td>
<td>UID of Superior Directory (14)</td>
</tr>
<tr>
<td>179-181</td>
<td>UID of Superior Directory (15)</td>
</tr>
<tr>
<td>180-182</td>
<td>8 Words Containing the Segment's Original Primary Name</td>
</tr>
</tbody>
</table>

#### Volume Table of Contents Entry (VTOCE)

A disk resident data base – one per segment.
THE "VOLUME TABLE OF CONTENTS" (VTOC) IS A DISK RESIDENT DATA BASE CONTAINING (OF INTEREST HERE) AN ARRAY OF ENTRIES KNOWN AS "VOLUME TABLE OF CONTENTS ENTRIES" (VTOCE'S)

ONE VTOC PER PHYSICAL VOLUME

ONE VTOCE PER SEGMENT

EACH VTOCE CONTAINS RESIDENCY INFORMATION (AND SOME ATTRIBUTE INFORMATION) OF A PARTICULAR SEGMENT

EACH VTOCE IS ADDRESSED BY INDEXING INTO THE ARRAY OF VTOCE'S

CONSEQUENTLY, THE PAIR OF PVID AND VTOC INDEX UNIQUELY IDENTIFIES ANY SEGMENT IN THE STORAGE SYSTEM HIERARCHY

EACH VTOCE IS 192 WORDS LONG AND IS DIVIDED INTO THREE LOGICAL PARTS:

ACTIVATION INFORMATION (16 WORDS)

CONTAINS ALL INFORMATION (EXCLUDING THE FILE MAP) NEEDED TO USE THE SEGMENT, OR MORE TECHNICALLY, TO ACTIVATE THE SEGMENT
SEGMENT CONTROL DATA BASES

VOLUME TABLE OF CONTENTS (VTOC)

INCLUDES: UID, CURRENT LENGTH, RECORDS USED, MAXIMUM LENGTH, RECORDS USED, ETC

ALL INFORMATION LIKELY TO CHANGE BECAUSE OF THE ACTIVATION

INCLUDES: DATE TIME MODIFIED AND USED, QUOTA CELLS (IF A DIRECTORY), ETC

FILE MAP (128 WORDS)

AN ARRAY OF 256 RECORD ADDRESS OR NULL ADDRESS DETAILING WHERE EACH PAGE OF THE SEGMENT RESIDES

A NULL ADDRESS (NOT TO BE CONFUSED WITH A NULLED ADDRESS -- DISCUSSED LATER) INDICATES THAT NO RECORD OF THE VOLUME IS ASSIGNED TO THAT PAGE OF THE SEGMENT


NOTE: PAGE CONTROL ENSURES THAT NO RECORD ADDRESS EVER APPEARS (OR REMAINS) IN THE FILE MAP UNLESS THE PAGE ACTUALLY APPEARS ON THE VOLUME

PERMANENT INFORMATION (48 WORDS)

CONTAINS ATTRIBUTES WHICH RARELY (IF EVER) CHANGE SUCH AS:

UID'S OF SUPERIOR DIRECTORIES, AIM CLASSIFICATION, DATE TIME DUMPED (BY PHYSICAL VOLUME DUMPER)
EACH VTOCE IS ALSO DIVIDED INTO THREE PHYSICAL PARTS:

FIRST SECTOR (WORDS 0-63):

CONTAINS ALL "ACTIVATION INFORMATION" AND THE FIRST PORTION OF THE FILE MAP

SECOND SECTOR (WORDS 64-127):

CONTAINS THE BULK OF THE FILE MAP

THIRD SECTOR (WORDS 128-191):

CONTAINS THE END OF THE FILE MAP AND ALL "PERMANENT INFORMATION"
SEGMENT CONTROL DATA BASES
VOLUME TABLE OF CONTENTS (VTOC)

VTOCE I/O

USING RECORD I/O (IN UNITS OF 1024 WORDS) TO ACCESS A VTOCE (192 WORDS) WOULD HAVE EXCESSIVE OVERHEAD FOR BUFFERS

FURTHERMORE, BECAUSE MOST SEGMENTS ARE SMALL, MOST VTOCE ACCESSING IS ONLY CONCERNED WITH ACTIVATION INFORMATION AND THE FIRST PORTION OF THE FILE MAP, I.E., THE FIRST SECTOR (64 WORDS)

TO TAKE ADVANTAGE OF THESE FACTS, VTOCE'S ARE ACCESSED VIA SECTOR I/O, NOT RECORD I/O

A LARGE MECHANISM KNOWN AS THE VTOC MANAGER (vtoc_man) EXIST TO EFFICIENTLY MANAGE THIS SECTOR I/O AND ITS BUFFERING

VTOCE I/O IS THE ONLY NON-PAGE I/O DONE TO DISK

VTOCE I/O IS DONE IN PARTS (SECTORS)

FOR A SEGMENT, OR A DIRECTORY WITHOUT TERMINAL QUOTA, PARTS ONE AND TWO CAN BE WRITTEN ENTIRELY FROM INFORMATION DERIVED FROM THE ASTE, AND NEED NOT BE READ IN FIRST

PART THREE MUST ALWAYS BE READ BEFORE BEING WRITTEN, AS MUST PART ONE FOR A DIRECTORY WITH QUOTA

ALL THREE PARTS OF A VTOCE ARE ALWAYS READ WHENEVER ANY PART IS REQUESTED, IN CASE THE OTHERS ARE NEEDED SOON AFTERWARDS
SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

A WIRED (STT) DATA BASE – ONE PER ACTIVE SEGMENT

ADDRESS

MAIN MEMORY ADDRESS

NULL ADDRESS – REPRESENTS PAGE OF ZEROES

A 16K PAGE TABLE (PT)

IMMEDIATELY FOLLOWS AN ASTE (ABOVE) – ONE PER ACTIVE SEGMENT

COMES IN 4, 16, 64 AND 256K FLAVORS

Not To Be Reproduced
THE ACTIVE SEGMENT TABLE (AST) IS A HARDCORE, WIRED, UNPAGED, DATA BASE LOCATED WITHIN THE SYSTEM SEGMENT TABLE (SST), AND CONSISTS OF AN ARRAY OF PAIRED ENTRIES KNOWN AS ACTIVE SEGMENT TABLE ENTRIES (ASTE'S) AND PAGE TABLES (PT'S).

ONE AST PER SYSTEM

ONE ASTE/PT PAIR PER ACTIVE SEGMENT

IN ORDER FOR A SEGMENT TO BE ACCESSED VIA THE HARDWARE, VTOCE INFORMATION MUST BE BROUGHT INTO MAIN MEMORY

THE 12 WORD ASTE (AND ITS ASSOCIATED PAGE TABLE) CAN BE THOUGHT OF AS THE MAIN MEMORY RESIDENT IMAGE OF THE VTOCE

SPECIFICALLY, THE ASTE CONTAINS:

THE VTOCE'S "ACTIVATION INFORMATION" SUCH AS THE SEGMENTS UID, CURRENT LENGTH, MAX LENGTH, DTU, DTM, QUOTA DATA

AND NON-VTOCE INFORMATION SUCH AS: PVT INDEX, VTOC INDEX, VARIOUS FLAGS AND POINTERS

THE PAGE TABLE CONTAINS THE RECORD ADDRESS (TAKEN FROM THE VTOCE'S "FILE MAP") OF EACH NONZERO PAGE OF THE SEGMENT
A segment having an AST and a page table is said to be active.

"Activating" a segment is the process of allocating (and filling in) an AST and a page table for the segment.

Conversely, "deactivating" a segment involves freeing its AST and page table for further use.

Being active does not imply that the segment is actually in use by any process.

Since the AST is a part of a single segment (having finite size), the number of AST/PT pairs is finite, implying:

Only a finite number of segments may be active at one time.

When a non-active segment is referenced, and there are no free AST's available, some segment must be deactivated.

This AST/PT multiplexing is the prime responsibility of segment control.

Being a finite (and a critical) system resource, the number of AST/PT pairs can dramatically affect the competition for AST's, and consequently system performance.
SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

TWO FEW ASTE/PT PAIRS WILL CAUSE "SEGMENT THRASHING"

TWO MANY ASTE/PT PAIRS WILL OCCUPY MAIN MEMORY FRAMES THAT
MIGHT BETTER BE UTILIZED FOR NORMAL PAGING TRAFFIC, PERHAPS
LEADING TO "PAGE THRASHING"

CONSEQUENTLY, THE NUMBER OF ASTE/PT PAIRS IS A CRITICAL
SYSTEM PARAMETER (SET ON THE SST CONFIG CARD)

OF THE TWO POSSIBILITIES, TOO MANY OR TOO FEW, TOO FEW IS
BY FAR THE WORSE
CONNECTING AND FAULTING SDW'S

PROCESS
Jones.ASP.m

PROCESS
Smith.FED.a

PROCESS
May.MED.a

PROCESS
Brown.FORD.a
SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

IN ORDER TO MAXIMIZE THE NUMBER OF ASTE/PT PAIRS WITHIN AN AST OF A GIVEN SIZE, ASTE/PT PAIRS COME IN FOUR FIXED SIZES:

1. ASTE + A 4 WORD PAGE TABLE (16 WORDS TOTAL)
   FOR 0-4K SEGMENTS

2. ASTE + A 16 WORD PAGE TABLE (28 WORDS TOTAL)
   FOR 5-16K SEGMENTS

3. ASTE + A 64 WORD PAGE TABLE (76 WORDS TOTAL)
   FOR 17-64K SEGMENTS

4. ASTE + A 256 WORD PAGE TABLE (268 WORDS TOTAL)
   FOR 65-256K SEGMENTS
### SEGMENT CONTROL DATA BASES

<table>
<thead>
<tr>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
<td>4 WD PT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 WD PT</td>
<td>16 WD PT</td>
<td>16 WD PT</td>
<td>16 WD PT</td>
<td>16 WD PT</td>
<td>16 WD PT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 WD PT</td>
<td>16 WD PT</td>
<td>16 WD PT</td>
<td>16 WD PT</td>
<td>16 WD PT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 WD PT</td>
<td>64 WD PT</td>
<td>64 WD PT</td>
<td>64 WD PT</td>
<td>64 WD PT</td>
<td>64 WD PT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
<th>ASTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 WD PT</td>
<td>256 WD PT</td>
<td>256 WD PT</td>
<td>256 WD PT</td>
<td>256 WD PT</td>
<td>256 WD PT</td>
</tr>
</tbody>
</table>

**ACTIVE SEGMENT TABLE (AST)**

A HARDCORE (SST) UN-PAGED DATA BASE – ONE PER SYSTEM

(NOT DRAWN TO SCALE)

Not To Be Reproduced 7-17 F80A
SEGMENT CONTROL DATA BASES
ACTIVE SEGMENT TABLE (AST)

* THE SIZE OF EACH OF THE FOUR ASTE POOLS IS DETERMINED AT SYSTEM INITIALIZATION BY THE SST CONFIG CARD AND IS A CRITICAL SYSTEM TUNING PARAMETER.

* SINCE THE FREQUENCY OF SMALL SEGMENTS IS HIGHER THAN THE FREQUENCY OF LARGE SEGMENTS, THE DISTRIBUTION OF ASTE'S IS NORMALLY AS FOLLOWS:

4K > 16K > 64K > 256K

ON SYSTEM-M, IN PHOENIX, A 6 CPU, 8MW MEMORY, 200 USER SYSTEM, THE ASTE DISTRIBUTION IS NORMALLY:
3500 1500 750 250

ON MIT-MULTICS, A 3 CPU, 3.5MW MEMORY, 110 USER SYSTEM, THE ASTE DISTRIBUTION IS NORMALLY:
1700 600 220 75

* A SEGMENT NORMALLY REMAINS ACTIVE (FOR >200 SECONDS) UNTIL FORCED TO GIVE UP ITS ASTE/PT PAIR TO ANOTHER SEGMENT (DEACTIVATION)
SEGMENr CONTROL DATA BASES
ACTIVE SEGMENT TABLE (AST)

* THIS DEACTIVATION CONSISTS OF:

1. MAKING THE SEGMENT INACCESSIBLE TO USER PROCESSES

   DONE BY "CUTTING TRAILERS", IN THE PROGRAM setfaults.pl1

2. A LIST OF ALL SEGMENTS CONNECTED TO (USING) THE SEGMENT IS
   KEPT FOR THIS PURPOSE

3. EVICTING ALL PAGES OF THE SEGMENT FROM MAIN MEMORY

   ONLY MODIFIED PAGES MUST BE WRITTEN BACK TO DISK. UNMODIFIED
   PAGES ARE SIMPLY OVERWRITTEN

4. UPDATING THE VTOCE BY WRITING THE (POSSIBLY MODIFIED) ACTIVATION
   INFORMATION BACK TO THE VTOCE

5. FREEING THE ASTE/PT PAIR

* SINCE ACTIVATING/DEACTIVATING SEGMENTS IS EXPENSIVE, THE CHOICE OF
  A SEGMENT FOR DEACTIVATION IS IMPORTANT, AND BELONGS TO THE SEGMENT
  REQUIRING ACTIVATION FURTHEST IN THE FUTURE
THE ALGORITHM WHICH Chooses A "BEST" SEGMENT FOR DEACTIVATION IS IMPLEMENTED IN THE PROGRAM get_aste, AND CONSIDERS SUCH FACTORS AS:

- THE PRESENCE OF ACTIVE INFERIORS (IF A DIRECTORY)
- THE NUMBER OF PAGES CURRENTLY IN MAIN MEMORY (SINCE WORK IS REQUIRED TO EVICT THE MODIFIED FRACTION OF SUCH PAGES AND BECAUSE THIS INDICATES "USED RECENTLY" IN SOME SENSE)
- IT LOOKS FIRST FOR A SEGMENT WITH NO INFERIORS, AND NO PAGES IN MEMORY, AND ALMOST ALWAYS SUCCEEDS - BUT IF IT FAILS, IT TRIES TO MAKE A "GOOD CHOICE", AND DEACTIVATES INFERIORS, AND/OR FLUSHES PAGES TO DISK IF NECESSARY

A

ASTE.used

ASTE.els Utility Hold Switch

ASTE.ons Each Page dir of active segment

AST.np Number Pages in main memory

Set by page control

ASTE.init If 0 not recently used so we can deactivate it.

First turns off init bits so that on 3rd pass we can deactivate segs in case 2nd pass didn't find any not in use.
TYPICAL "USED LIST"

A DOUBLE THREADED LIST OF SIMILAR OBJECTS GENERALLY IMPLEMENTING REPLACEMENT ALGORITHMS. CONTAINS BOTH FREE AND IN-USE OBJECTS, WITH FREE OBJECTS MAINTAINED AT THE HEAD OF THE LIST.
NOTE: WHILE INSPECTING THE ASTE'S, OPPORTUNITY IS TAKEN TO NOTICE ASTE'S WHOSE FILE MAPS HAVE CHANGED AND TO UPDATE THEIR VTOCE'S

KNOWN AS "AST TRICKLE"

THIS IS DONE TOTALLY AS A HEDGE AGAINST A FATAL CRASH, AS A SUCCESSFUL SHUTDOWN UPDATES ALL VTOCE'S OF ACTIVE SEGMENTS

THIS IS NOT DONE FOR PROCESS DIRECTORY SEGMENTS, SINCE THEIR CONTENTS ARE OF LITTLE USE AFTER A CRASH

THE "AST TRICKLE" IS ALSO FORCED TO OCCUR EVERY FIFTEEN MINUTES WHEN THE SYSTEM IS LIGHTLY LOADED, BECAUSE OTHERWISE VTOCES MIGHT REMAIN UNUPDATED FOR HOURS
AST HIERARCHY

MIRRORS A SUBSET OF THE STORAGE SYSTEM HIERARCHY

THE ROOT DIRECTORY (>) CANNOT BE DEACTIVATED

NO SEGMENT (EXCEPT THE ROOT) MAY BE ACTIVE UNLESS ITS PARENT IS ACTIVE. THE REASONS FOR THIS ARE:

PARENT MUST BE ACTIVATED IN ORDER TO FIND THE SON

ACTIVATION OF OTHER SONS IS EASIER IF THE PARENT REMAINS ACTIVE

THE QUOTA ACCOUNT AGAINST WHICH AN ACTIVE SEGMENT'S "RECORDS USED" IS TALLIED SHOULD BE IMMEDIATELY AVAILABLE WHEN A SEGMENT CHANGES SIZE. THE QUOTA ACCOUNT IS FOUND IN ONE OF THE ANCESTORS' ASTES

DATE TIME MODIFIED (DTM) FOR A DIRECTORY IS THE DTM OF THE LAST MODIFIED SEGMENT IN THE SUBTREE. DTM OF ALL ANCESTOR'S SHOULD BE IMMEDIATELY AVAILABLE WHEN A SEGMENT IS MODIFIED. DTM IS FOUND IN THE ASTE OF THE ANCESTORS. (USED BY THE HIERARCHY DUMPER)

SUCH UPDATES TO THE ASTE'S OF PARENTS ARE PERFORMED BY PAGE CONTROL

EACH ASTE HAS A POINTER TO ITS PARENT'S ASTE FOR THE ABOVE REASONS. (THIS POINTER IMPLEMENTS THE AST HIERARCHY)
SEGMENT CONTROL DATA BASES
ACTIVE SEGMENT TABLE (AST)

- ASTE'S MAY BE THREADED ONTO ONE OF SIX LISTS VIA THE RELATIVE POINTERS aste.fp and aste.bp

- FOUR USED LISTS: THREADS ALL FREE AND REPLACEABLE ASTE'S OF EACH POOL SIZE (sst.level.ausedp POINTS TO THE FIRST FREE ASTE IN THE LIST)

- INIT AND TEMP LISTS: USED AT SYSTEM INITIALIZATION TO RECEIVE (AND DELETE) INITIALIZATION AND TEMPORARY SEGMENTS

- THERE ALSO EXIST SEVERAL AUXILIARY LISTS SUCH AS THE HASH THREAD AND FATHER-SON, AND BROTHERS LISTS

- ALL ACTIVE SEGMENTS IN THE HIERARCHY ARE IN THE FOUR USED LISTS - EXCEPT FOR SEGMENTS IN THE HARDCORE PARTITION (THE PAGED SUPERVISOR), AND A SMALL CLASS OF SEGMENTS WHICH MAY NOT BE DEACTIVATED

- SEGMENTS ARE SOMETIMES UNTHEADED FROM THEIR USED LIST TEMPORARILY IN ORDER TO KEEP THEM OUT OF REACH WHILE SOME COMPLEX OPERATION IS PERFORMED
SERVICES OF SEGMENT CONTROL
CREATING SEGMENTS

SEGMENT CREATION IS PERFORMED BY THE PROCEDURE create_vtoce

| INPUT: A POINTER TO THE BRANCH ENTRY IN A DIRECTORY SEGMENT |

| OUTPUT: PVID AND VTOC INDEX OF THE CREATED SEGMENT |

create_vtoce MAY BE CALLED BY append (NORMAL SEGMENT CREATION) OR segment_mover (DUE TO PACK OVERFLOW)

PRINCIPAL STEPS OF create_vtoce:

| CREATE A LOCAL IMAGE OF THE VTOCE TO BE CREATED |
| FILL IN MOST ACTIVATION AND PERMANENT INFORMATION FROM THE BRANCH ENTRY |
| CREATE A NULL FILE MAP |
| DETERMINE THE UID PATH (UID'S OF SUPERIOR DIRECTORIES) |
SERVICES OF SEGMENT CONTROL

CREATING SEGMENTS

SELECT AN APPROPRIATE PV WITHIN THE LV SPECIFIED BY THE
sons_lvid OF THE DIRECTORY IN WHICH THE BRANCH ENTRY APPEARS

SELECTION GOAL IS TO EVENLY DISTRIBUTE SEGMENTS OVER ALL PV'S
OF THE LV, THEREBY REDUCING DISK CONTENTION

SELECTION ALGORITHM WALKS THE CHAIN (THROUGH
pvte.brother_pvtx) OF PV'S IN THE LV AND SELECTS THE PV
HAVING THE HIGHEST PERCENTAGE OF UNUSED RECORDS IN ITS PAGING
REGION

NO PV IS ACCEPTED IF pvte.vacating IS ON, SIGNIFYING THAT
sweep_pv IS TRYING TO VACATE, OR INHIBIT CREATION UPON, THE
PV

AN EXCEPTION IS MADE FOR PER PROCESS SEGMENTS
(entry.per_process IS ON)

SINCE SUCH SEGMENTS ARE ALL HEAVILY USED, A ROUND ROBIN
ALGORITHM EVENLY DISTRIBUTES THESE SEGMENTS ACROSS ALL
PV'S IN THE LV

INVOKE THE VTOC MANAGER (vtoc_man_alloc_and_put_vtoce) TO
ALLOCATE AND WRITE THE VTOCE IMAGE ON THE SELECTED PV

VTOC_MAN ATTEMPTS TO ALLOCATE A VTOCE FROM THE VTOCE STOCK
FOR THE VOLUME

IF THE STOCK IS EMPTY, IT REFILLS IT FROM THE VTOCE MAP ON
DISK (SEE vtoce_stock_man.pl1)

BECAUSE IT IS PERMISSIBLE TO TAKE PAGE FAULTS IN THE
VTOC_MAN_ENVIRONMENT, THE VTOCE STOCK IS ACCESSED WITHOUT
ANY SPECIAL PAGE CONTROL PROTOCOLS

RETURNS THE VTOC INDEX OF THE ALLOCATED VTOCE
SERVICES OF SEGMENT CONTROL
CREATING SEGMENTS

RETURN THE PVID AND VTOC INDEX OF THE NEW SEGMENT TO THE CALLER
(WHO RECORDS SAME IN entry.pvid AND entry.vtocx)
SERVICES OF SEGMENT CONTROL

SEGMENT ACTIVATION

SEGMENT ACTIVATION IS PERFORMED BY THE PROCEDURE "activate"

INPUT: A POINTER TO THE BRANCH ENTRY IN A DIRECTORY SEGMENT

OUTPUT: AN ASTE POINTER

activate IS PRINCIPALLY CALLED BY seg fault (OF ADDRESS/NAME SPACE MANAGEMENT) WHO HAS LOCATED THE SEGMENT'S BRANCH ENTRY, VALIDATED THE USER'S ACCESS, AND CHECKED THE PRESENCE OF THE LV

PRINCIPAL STEPS OF activate:

LOCK THE AST LOCK AND CHECK IF THE SEGMENT IS ALREADY ACTIVE. IF SO, UNLOCK THE AST AND RETURN ITS ASTE POINTER

IF THE SEGMENT IS NOT ACTIVE, UNLOCK THE AST AND READ IN ALL REQUIRED PARTS OF THE VTOCE AND COMPARE UID'S FOR CONNECTION FAILURE (IN WHICH CASE, DO NOT ACTIVATE AND RETURN AN ERROR)

ENSURE THAT THE SEGMENT'S PARENT IS ACTIVE

THIS IS DONE BY REFERENCING THE PARENT DIRECTORY (PERHAPS CAUSING A RECURSIVE SEGMENT FAULT AND ACTIVATION) AND SETTING THE ast.ehs BIT FOR IT TEMPORARILY WHILE ACTIVATION IS TAKING PLACE
SERVICES OF SEGMENT CONTROL

SEGMENT ACTIVATION

NOTE THAT THE ONLY EXPLICIT ACTION TAKEN TO ACTIVATE THE PARENT IS TO REFERENCE IT; THE POSSIBLE RECURSIVE SEGMENT FAULT TAKES CARE OF EVERYTHING AND ALLOWS THAT REFERENCE TO PROCEED.

OBTAIN AN ASTE FOR THE SEGMENT BY CALLING get_aste. THIS MAY INVOLVE DEACTIVATING SOME OTHER SEGMENT - BUT HOPEFULLY NOT THE PARENT! (ENSURED BY THE aste.ehs BEING ON)

THREAD THE ASTE INTO THE INFERIOR LIST OF THE PARENT'S ASTE (THIS WILL KEEP HIM ACTIVE), AND RESET THE PARENT'S aste.ehs

FILL IN THE ASTE WITH THE VTOCE'S ACTIVATION INFORMATION AND INITIAL FLAG VALUES

CALL pc$fill_page_table (PASSING THE VTOCE'S FILE MAP) TO INITIALIZE THE PAGE TABLE AND OTHER PAGE CONTROL INFORMATION

PLACE THE UID IN THE ASTE AND HASH IT INTO THE AST HASH TABLE

AFTER A SYSTEM FAILURE, ESD USES A ZERO UID AS A CUE THAT THE ASTE IS INVALID, AND DOES NOT INVOKE A VTOCE UPDATE FOR THE ASTE
SERVICES OF SEGMENT CONTROL

SEGMENT TRAILERS

WHEN A PROCESS IS USING A SEGMENT, AND IT HAS A VALID SDW FOR THAT SEGMENT, A RECORD MUST BE KEPT IN CASE IT IS NECESSARY TO REVOKE THAT SDW (WHEN THE SEGMENT IS DELETED, WHEN THE ASTE IS RE-USED, ETC)

THIS IS DONE BY THE SEGMENT TRAILER MECHANISM

EACH PROCESS WHICH HAS A VALID SDW FOR A SEGMENT HAS A "TRAILER ENTRY" WHICH RECORDS ITS PROCESS IDENTIFICATION AND THE SEGMENT NUMBER IT IS USING FOR THE SEGMENT

TRAILER ENTRIES ARE KEPT IN THE str_seg, A PAGED SUPERVISOR SEGMENT

THERE IS A LINKED LIST OF TRAILER ENTRIES FOR EACH SEGMENT; THE HEAD IS POINTED TO BY aste.strp

TRAILERS ARE ONLY KEPT FOR SEGMENTS WHICH MAY BE DEACTIVATED: ORDINARY SEGMENTS AND DIRECTORIES, BUT NOT SUPERVISOR SEGMENTS

A TRAILER IS ATTACHED FOR A SEGMENT BY seg_fault.pl1 BEFORE IT PLACES THE SDW INTO THE DSEG FOR THE PROCESS

TRAILERS ARE USED LATER BY setfaults.pl1 FOR:

DEACTIVATION - THE SDW IS ENTIRELY REVOKED (ZEROED)

ACCESS CHANGES, DELETION - THE SDW IS MADE INVALID, BUT ITS PAGE TABLE ADDRESS IS LEFT UNCHANGED, INDICATING TO seg_fault.pl1 THAT ALL IT MUST DO IS RECALCULATE THE ACCESS
SERVICES OF SEGMENT CONTROL

SEGMENT TRAILERS

- PROCESS TERMINATION - ALL SDWS A PROCESS HAD ARE REVOKED AT TERMINATION

- CACHE CONTROL - WHEN THE ENCACHABILITY OF A SEGMENT CHANGES, THE SDWS WHICH REFER TO IT MUST HAVE THEIR CACHE CONTROL BITS UPDATED

- TRAILERS ARE ONLY REMOVED WHEN THE ASSOCIATED SDW IS REVOKED COMPLETELY
SERVICES OF SEGMENT CONTROL
BOUNDSDFAULT HANDLING

BOUNDSD FAULT HANDLING

1. PAGE FAULT. PAGE 3 "CREATED" IN MAIN MEMORY
   AS A PAGE OF ZEROS (CALLED A "NEW PAGE")

2. BOUNDS FAULT. PROMOTE THE SEGMENT TO THE 16K
   ASTE/PT POOL...CONTINUE REFERENCE
   AS IN CASE #1

3. BOUNDS FAULT. ERROR: SIGNAL "out-of-bounds"
   CONDITION
SERVICES OF SEGMENT CONTROL

BOUNDSAFEULT HANDLING

**THE BOUNDSAFEULT HANDLER IS THE PROCEDURE "boundsfault", INVOKED BY THE FAULT INTERCEPTOR MODULE, FIM, WHEN THE BOUNDSAFEULT IS DETECTED BY THE APPENDING UNIT HARDWARE**

**BASIC STEPS OF BOUNDSAFEULT**

1. USING THE SEGMENT NUMBER IN THE (SAVED) MACHINE CONDITIONS, FIND AND LOCK THE PARENT DIRECTORY, AND FIND THE BRANCH ENTRY

2. LOCK THE AST AND FIND THE SEGMENT'S ASTE VIA get_ptrs_sgiven_segno. IF ATTEMPTED REFERENCE IS BEYOND THE MAXIMUM LENGTH (aste.msl) THEN CAUSE "out_of Bounds" TO BE SIGNALLED

3. MAKE THE SEGMENT INACCESSIBLE TO USERS BY "CUTTING TRAILERS"

4. TURN ON THE PARENT'S aste.ehs BIT AND CALL get_aste TO OBTAIN A LARGER ASTE

5. CALL PAGE CONTROL'S pcsmove_page_table TO MOVE ALL ASTE/PT INFORMATION TO THE NEW ASTE

6. RETHEARD ALL INFERIOR LIST AND PARENT POINTERS AFFECTED AND TURN OFF THE PARENT'S aste.ehs BIT

**NOTE:** IF THE SEGMENT IS A DIRECTORY, ALL FATHER POINTERS OF INFERIOR SEGMENTS MUST BE UPDATED
SERVICES OF SEGMENT CONTROL
BOUNDFAULT HANDLING

| THIS IS THE ONLY REASON FOR THE EXISTENCE OF INFERIOR LIST IN THE AST |
| REMOVE THE OLD ASTE FROM THE AST HASH TABLE AND HASH IN THE NEW |
| CALL put_aste TO FREE THE OLD ASTE |
| UNLOCK THE AST AND RETURN A ZERO STATUS CODE TO THE FIM |
SERVICES OF SEGMENT CONTROL

SEGMENT DEACTIVATION

- SEGMENT DEACTIVATION IS PERFORMED BY THE PROCEDURE "deactivate"

- INPUT: POINTER TO AN ASTE

- deactivate MAY BE CALLED BY:

- get_aste WHEN AN ASTE MUST BE FREED TO MAKE ROOM FOR A NEW SEGMENT

- delete_vtoce AS PART OF SEGMENT DELETING

- demount_pv IN ORDER TO UPDATE THE VTOCE'S (AND SEGMENTS) OF A DISK BEING DEMOUNTED

- PRINCIPAL STEPS OF deactivate:

- MAKE THE SEGMENT INACCESSIBLE TO USERS BY "CUTTING TRAILERS"

- CALL pcscleanup TO REMOVE ALL PAGES OF THE SEGMENT FROM BULK STORE AND MAIN MEMORY, WRITING ALL MODIFIED PAGES TO DISK
SERVICES OF SEGMENT CONTROL
SEGMENT DEACTIVATION

- UPDATE THE VTOCE FROM THE NOW QUIESCENT ASTE

- THREAD THE ASTE OUT OF PARENT'S INFERIOR LIST, AND OUT OF THE AST HASH TABLE

- CALL put aste TO CLEAR AND INITIALIZE THE ASTE/PT PAIR, AND THREAD THE ASTE AT THE HEAD OF THE APPROPRIATE USED LIST
SERVICES OF SEGMENT CONTROL

TRUNCATING SEGMENTS

SEGMENT TRUNCATION (IE: PAGE REMOVAL) IS PERFORMED BY THE PROCEDURE truncate_vtoce.

INPUT: A POINTER TO THE BRANCH ENTRY IN A DIRECTORY SEGMENT, AND A PAGE NUMBER FROM WHICH TO START TRUNCATING.

truncate_vtoce IS CALLED BY:

- truncate (DIRECTORY CONTROL) WHO HAS LOCATED THE SEGMENTS BRANCH ENTRY AND VALIDATED THE USER'S ACCESS.

- delete_vtoce (SEGMENT CONTROL) WHO REQUIRES TRUNCATION (FROM PAGE #0) PRIOR TO VTOCE DELETION.

PRINCIPAL STEPS OF truncate_vtoce:

- IF SEGMENT IS ACTIVE, CALL pc$truncate WHO MARKS THE DEVICE ADDRESS OF ALL PAGES ABOVE THE PAGE NUMBER SPECIFIED AS "NULLED" ADDRESSES (DISCUSSED IN "PAGE CONTROL" TOPIC).

- IF SEGMENT IS NOT ACTIVE:

  READ IN ALL REQUIRED PARTS OF THE VTOCE AND COMPARE UID'S FOR CONNECTION FAILURE (RETURN AN ERROR IF SO).
SERVICES OF SEGMENT CONTROL

TRUNCATING SEGMENTS

CALL get_pvtx$hold_pvtx TO PREVENT A DEMOUNT (IF CALLED BY truncate)

COPY ALL ADDRESSES OF PAGES TO BE TRUNCATED FROM THE FILE MAP AND REPLACE THEM WITH "NULL" ADDRESSES

FABRICATE A NEW VTOCE AND WRITE THE VTOCE BACK BY CALLING vtoc_man$put_vtoce

IF ANY REAL ADDRESSES WERE COPIED FROM THE FILE MAP, AWAIT THE SUCCESSFUL COMPLETION OF THE VTOCE WRITE BY CALLING vtoc_man$await_vtoce

CALL pc$deposit_list TO DEPOSIT (FREE) THESE REAL RECORD ADDRESSES

CALL get_pvtx$release_pvtx TO AGAIN PERMIT DEMOUNTING (IF CALLED BY truncate). THIS CREATES AN ENTRY IN THE PV HOLD TABLE (SEE TOPIC 6, VOLUME MANAGEMENT)
SERVICES OF SEGMENT CONTROL

DELETING SEGMENTS

SEGMENT DELETION IS PERFORMED BY THE PROCEDURE delete_vtoce

INPUT: A POINTER TO THE BRANCH ENTRY IN A DIRECTORY SEGMENT

delete_vtoce is called by delentry (OF DIRECTORY CONTROL FAME) WHO has located the segment's branch entry and validated the user's access

PRINCIPAL STEPS OF delete_vtoce:

CALL get_pvtx$hold_pvtx TO PREVENT A VOLUME DEMOUNT IN THE MIDDLE OF THE DELETION

IF ACTIVE, MAKE THE SEGMENT INACCESSIBLE TO USERS (SEE "ADDRESS AND NAME SPACE MANAGEMENT", TOPIC 5)

TRUNCATE THE SEGMENT TO ZERO LENGTH (SEE "TRUNCATING SEGMENTS" IN THIS TOPIC), FREEING ALL DISK, BULK STORE, AND MAIN MEMORY PAGES OCCUPIED BY THE SEGMENT

IF THE SEGMENT IS A DIRECTORY SEGMENT HAVING A QUOTA ACCOUNT, CALL THE QUOTA MOVE PRIMITIVE (quotaw$mq) TO RELINQUISH THE QUOTA TO ITS SUPERIOR
SERVICES OF SEGMENT CONTROL

DELETING SEGMENTS

NOTE: DIRECTORY CONTROL IS RESPONSIBLE FOR DELETING ALL INFERIOR SEGMENTS BEFORE REQUESTING DELETION OF THE DIRECTORY - ENSURING A CONSISTENT HIERARCHY AND RECOVERY OF ALL INFERIOR QUOTA ACCOUNTS

IF THE SEGMENT IS ACTIVE, DEACTIVATE IT, RELEASING ITS ASTE

FREE THE VTOCE WITH A CALL TO `vtoc_man$free_vtoce`

CALL `get_pvtx$release_pvtx` TO AGAIN PERMIT VOLUME DEMOUNTING
SERVICES OF SEGMENT CONTROL

OTHER SERVICES

OTHER SERVICES PERFORMED BY SEGMENT CONTROL INCLUDE:

SEGMENT MOVING

REQUIRED WHEN AN ATTEMPT IS MADE TO GROW A SEGMENT AND THERE IS NO MORE ROOM ON THE PHYSICAL VOLUME

THE ENTIRE SEGMENT MUST BE MOVED TO ANOTHER PV WITHIN THE SAME LV --- TRANSPARENT TO THE USER AND DIRECTORY CONTROL

THIS IS THE SINGLE MOST INVOLVED AND ESOTERIC SERVICE OF SEGMENT CONTROL

SEMI-PERMANENT ACTIVATION

ACTIVATING A SEGMENT INTO AN ASTE OF A GIVEN SIZE AND TURNING ON ITS ast.e.ehs (DONE BY grab_aste)

SERVICES FOR sweep_pv

LISTING THE VTOC OF A PACK (IE: REPORTING THE PATHNAMES OF ALL SEGMENTS OWNING VTOCE'S)

THE LOCATING AND DELETING OF ORPHAN VTOCE'S (VTOCE'S NOT DESCRIBED IN ANY BRANCH)

REBALANCING OR VACATING PACKS VIA DEMAND SEGMENT MOVING
SERVICES OF SEGMENT CONTROL

OTHER SERVICES

SERVICES AT DEMOUNT/SHUTDOWN TIME

- DEACTIVATION OF ALL SEGMENTS ON THE VOLUME BEING DEMOUNTED AND WRITING OUT THE LABEL, ETC

SERVICES FOR ADDRESS/NAME SPACE MANAGEMENT (SEE TOPIC 4)

- DESCRIPTOR SEGMENT (DSEG), PROCESS DATA SEGMENT (PDS) AND KNOWN SEGMENT TABLE (KST) MANAGEMENT

- SEGMENT FAULT HANDLING (seg_fault), CREATION, ENTRY HOLDING
SEGMENT CONTROL METERS

FILE SYSTEM METERS

FILE_SYSTEM_METERS - DISPLAYS MISCELLANEOUS METERING INFORMATION FOR THE FILE SYSTEM

I ONLY PARTS RELEVANT TO SEGMENT CONTROL INCLUDED HERE: SEE TOPIC B (PAGE CONTROL) FOR THE REST

Total metering time 0:20:02

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>ATB - AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activations</td>
<td>1043</td>
<td>1.153 sec.</td>
</tr>
<tr>
<td>segfault</td>
<td>969</td>
<td>1.241 sec.</td>
</tr>
<tr>
<td>makeknown</td>
<td>74</td>
<td>16.251 sec.</td>
</tr>
<tr>
<td>directories</td>
<td>96</td>
<td>12.527 sec.</td>
</tr>
<tr>
<td>Deactivations</td>
<td>1056</td>
<td>1.139 sec.</td>
</tr>
<tr>
<td>Demand deactivate attempts</td>
<td>3</td>
<td>400.857 sec.</td>
</tr>
<tr>
<td>Seg Faults</td>
<td>5080</td>
<td>0.237 sec.</td>
</tr>
<tr>
<td>fault</td>
<td>4311</td>
<td>0.279 sec.</td>
</tr>
<tr>
<td>call</td>
<td>769</td>
<td>1.564 sec.</td>
</tr>
<tr>
<td>activations</td>
<td>969</td>
<td>1.241 sec.</td>
</tr>
<tr>
<td>Bound Faults</td>
<td>220</td>
<td>5.466 sec.</td>
</tr>
<tr>
<td>Setfaults</td>
<td>4454</td>
<td>28.633 sec.</td>
</tr>
<tr>
<td>access</td>
<td>42</td>
<td>28.633 sec.</td>
</tr>
<tr>
<td>ASTE Trickle</td>
<td>139</td>
<td>8.652 sec.</td>
</tr>
<tr>
<td>Steps</td>
<td>4279</td>
<td>281.040 m sec.</td>
</tr>
<tr>
<td>Skips</td>
<td>3016</td>
<td>0.399 sec.</td>
</tr>
<tr>
<td></td>
<td>271</td>
<td>4.436 sec.</td>
</tr>
<tr>
<td></td>
<td>1083</td>
<td>1.110 sec.</td>
</tr>
<tr>
<td></td>
<td>1662</td>
<td>0.724 sec.</td>
</tr>
<tr>
<td>Searches</td>
<td>0</td>
<td>0.000 sec.</td>
</tr>
<tr>
<td>Cleanups</td>
<td>1056</td>
<td>1.139 sec.</td>
</tr>
<tr>
<td>Force writes</td>
<td>3</td>
<td>400.857 sec.</td>
</tr>
<tr>
<td>pages written</td>
<td>3</td>
<td>400.857 sec.</td>
</tr>
<tr>
<td>Lock AST</td>
<td>18422</td>
<td>0.065 sec.</td>
</tr>
</tbody>
</table>
SEGMENT CONTROL METERS

file system meters

<table>
<thead>
<tr>
<th>AST locked</th>
<th>AVE/lock</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.833 msec.</td>
<td>7.4</td>
<td></td>
</tr>
<tr>
<td>AST lock waiting</td>
<td>1.601 msec.</td>
<td>2.5</td>
</tr>
<tr>
<td>% / BY 100</td>
<td>SHOULD BE &lt; .8 %</td>
<td></td>
</tr>
</tbody>
</table>

AST Sizes

<table>
<thead>
<tr>
<th>Number</th>
<th>Need</th>
<th>Steps</th>
<th>Ave Steps</th>
<th>Lap Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1701</td>
<td>601</td>
<td>2341</td>
<td>2.9</td>
<td>873.8</td>
</tr>
<tr>
<td>64</td>
<td>221</td>
<td>645</td>
<td>3.2</td>
<td>1120.5</td>
</tr>
<tr>
<td>256</td>
<td>74</td>
<td>1139</td>
<td>5.5</td>
<td>233.3</td>
</tr>
<tr>
<td>154</td>
<td>4.5</td>
<td>416</td>
<td>8.1</td>
<td>577.9</td>
</tr>
</tbody>
</table>

Seg fault - refer SDW with valid bit off
put RTN in SDW
or
look in AST + active it
from reading VDCC

Sort command can cause thrashing for awhile.
Because it chooses 17 page size, putting it
in pool.

Error in segment replacing algorithm.
light load, lots of memory, all active
less in memory, hence it things it
is in use + has to go around for
awhile until it has to force it
out. Causes slow response time
on light load.
**SEGMENT CONTROL METERS**

**vtoc_buffer_meters**

#### VTOC_BUFFER_METERS - DISPLAYS VTOC BUFFER MANAGER ACTIVITY

<table>
<thead>
<tr>
<th>Routine</th>
<th># calls</th>
<th>ATB (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_vtoce</td>
<td>1346752</td>
<td>0.17</td>
</tr>
<tr>
<td>put_vtoce</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>alloc_and_put_vtoce</td>
<td>77378</td>
<td>3.04</td>
</tr>
<tr>
<td>free_vtoce</td>
<td>75664</td>
<td>3.11</td>
</tr>
<tr>
<td>await_vtoce</td>
<td>93370</td>
<td>2.52</td>
</tr>
<tr>
<td>GET BUFFERS</td>
<td>2732265</td>
<td>0.09</td>
</tr>
<tr>
<td>WAIT</td>
<td>946437</td>
<td>0.25</td>
</tr>
</tbody>
</table>

**Buffer Allocation**

<table>
<thead>
<tr>
<th>Type</th>
<th># ATB (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>steps</td>
<td>1354279</td>
</tr>
<tr>
<td>skips</td>
<td>278866</td>
</tr>
<tr>
<td>hot</td>
<td>240348</td>
</tr>
<tr>
<td>wait</td>
<td>38518</td>
</tr>
</tbody>
</table>

**Disk I/Os**

<table>
<thead>
<tr>
<th>Type</th>
<th># ATB (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reads</td>
<td>836941</td>
</tr>
<tr>
<td>writes</td>
<td>548573</td>
</tr>
</tbody>
</table>
SEGMENT CONTROL COMMANDS

**print aste ptp**

*DISPLAYolest*

- **PRINT_ASTE_PTP** - DISPLAYS INFORMATION FROM AN ASTE

ASTE for >udd>Multics>Sibert at 115244 in sst_seg
077550100664 041540056140 165706076310 102401170050
315015000063 640000044000 446736250032 446736250272
003720000000 003164000110 006200006000 073770000102

uid = 102401170050, vtocx = 63 on pvtx 15 (root4 = dskd_13)
max len 205, 6 recs used, 0 in core, cur len 6 (decimal)
Used 03/18/83 0116.0 est Fri
Modified 03/18/83 0116.2 est Fri
Par astep = 76310, Son = 56140, brother = 41540
Trailer thread = 165706
Aste for a directory.
Quota (S D) = (2000 0)
QUsed (S D) = (1652 72)

Flags: usedf init seg-tqsw fms

<table>
<thead>
<tr>
<th>PAGE</th>
<th>PT</th>
<th>DEVADD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>063540200041</td>
<td>63540</td>
</tr>
<tr>
<td>1</td>
<td>063546200041</td>
<td>63546</td>
</tr>
<tr>
<td>2</td>
<td>063547200041</td>
<td>63547</td>
</tr>
<tr>
<td>3</td>
<td>063550200041</td>
<td>63550</td>
</tr>
<tr>
<td>4</td>
<td>063567200041</td>
<td>63567</td>
</tr>
<tr>
<td>5</td>
<td>063571200041</td>
<td>63571</td>
</tr>
<tr>
<td>6</td>
<td>377020000001</td>
<td>null</td>
</tr>
<tr>
<td>17</td>
<td>377020000001</td>
<td>null</td>
</tr>
</tbody>
</table>

---

Not To Be Reproduced
SEGMENT CONTROL COMMANDS

dump vtoce

* DUMP_VTOCE - READS A VTOCE FROM DISK AND DISPLAYS ITS CONTENTS

vtoce Sibert (Directory), vtocx 63 on pvtx 15 (root4)-03/18/83 0116.7 est Fri

Uid = 102401170050, msl/csl/rec = 205  6  6
  Quota (S D) = (2000 0)
  Quota used (S D) = (1423 44)
  Quota received (S D) = (2000 0)
Created 08/16/81 1204.0 est Sun
Dumped 03/18/83 0106.7 est Fri
Used 03/17/83 1737.8 est Thu
Modified 03/15/83 2138.3 est Tue

Switches: fm_checksum_valid

Activation information:

0 000000000000 102401170050 315006006000 446734564570
4 446723404647 001000400000 06356063567 00372000000
10 002617000054 00372000000 0000000000 01536402005
14 000000000000 000652154327 446734634556 445673737671

File map:

20 063540063546 063547063550 06356063571 777020777020
24 777020777020 777020777020 777020777020 777020777020
30 777776777776 777776777776 777776777776 777776777776
=======
214 777776777776 777776777776 777776777776 777776777776

Permanent information:

220 000000000000 000000000000 000000000000 000000000000
=======
230 000000000000 000000000000 000000000000 446736227316
234 1261040000525 1261040000525 1261040000764 000000000000
240 777777777777 033022237767 033023254650 000000000000
244 000000000000 000000000000 000000000000 000000000000
=======
260 123151142145 162164040040 040040040040 040040040040
264 040040040040 040040040040 040040040040 040040040040
270 441200557477 135240026001 004370027106 000000000000
274 000000000000 000000000000 000000000000 000000000000

Not To Be Reproduced

7-48
(End Of Topic)
PAGE CONTROL OVERVIEW

FUNCTION

PAGE CONTROL IS RESPONSIBLE FOR THE MANAGEMENT OF PHYSICAL MEMORY TO INCLUDE THE MULTIPLEXING OF MAIN MEMORY FRAMES, AND THE MANAGEMENT OF DISK STORAGE.

ITS TASKS INCLUDE:

TRANSFERRING THE PAGES OF SEGMENTS BETWEEN THE MEMORY DEVICES, AND RECORDING THE LOCATION OF "THE" COPY OF THESE PAGES.

REPORTING THE STATUS AND FILE MAPS OF SEGMENTS TO SEGMENT CONTROL.

PAGE CONTROL IS LARGELY CODED IN MULTICS ASSEMBLER LANGUAGE (ALM).

PAGE CONTROL CAN BE INVOKED EITHER BY SUBROUTINE CALLS OR BY PAGE FAULTS.

THERE ARE NO EXPLICIT USER INTERFACES TO PAGE CONTROL.
PAGE CONTROL OVERVIEW

★ BASIC PHILOSOPHY

★ OF ALL THE SEGMENTS ACTIVE AT A GIVEN TIME, ONLY A SMALL SUBSET OF THEIR TOTAL PAGES WILL BE REQUIRED FOR ACCESSING

★ PAGES WILL BE READ INTO MAIN MEMORY AS THEY ARE REQUIRED

★ THE READING OF A PAGE INTO MAIN MEMORY WILL (PROBABLY) REQUIRE THE EVICTION OF A PREVIOUSLY REQUIRED PAGE

★ THE CHOICE OF A PAGE FOR EVICTION WILL BE BASED ESSENTIALLY UPON A "LEAST RECENTLY USED" CRITERIA

★ AN EVICTED PAGE NEED BE WRITTEN BACK TO DISK ONLY IF IT WAS MODIFIED DURING ITS RESIDENCY IN MAIN MEMORY

★ MAJOR DATA BASES

★ PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM. PROVIDED BY VOLUME MANAGEMENT

★ PHYSICAL VOLUME TABLE ENTRY (PVTE) - ONE PER DISK DRIVE CONFIGURED

★ EACH PVTE CONTAINS:

★ THE DEVICE ID (DISK DRIVE ID) AND THE ID OF THE PHYSICAL VOLUME (DISK PACK) CURRENTLY MOUNTED
THE NUMBER OF RECORDS LEFT UNALLOCATED ON THE PHYSICAL VOLUME, POINTER TO THE RECORD STOCK, ETC

RECORD STOCKS - ONE PER MOUNTED PHYSICAL VOLUME, PROVIDED BY VOLUME MANAGEMENT

CONTAINS AN IN-MEMORY CACHE OF THE IN-USE STATUS OF RECORDS ON THE VOLUME, FROM THE VOLUME MAP, USED WHEN ALLOCATING OR FREEING PAGES

ACCESSIBLE BY A COMPLEX MECHANISM WHICH USES NORMAL PAGE I/O BUT HAS A PROTOCOL TO ENSURE SYNCHRONIZATION OF DISK CONTENTS AND RECORD STOCK CONTENTS

SYSTEM SEGMENT TABLE (SST) - ONE PER SYSTEM. SHARED WITH SEGMENT CONTROL. CONTAINS THE FOLLOWING FIVE DATA BASES USED BY PAGE CONTROL:

SYSTEM SEGMENT TABLE (SST) HEADER - ONE PER SYSTEM

CONTAINS A LARGE NUMBER OF COUNTERS AND POINTERS VITAL TO THE MAINTENANCE AND METERING OF THE STORAGE SYSTEM

CONTAINS LOCKWORDS USED TO SYNCHRONIZE PAGE CONTROL AND SEGMENT CONTROL OPERATIONS

CORE MAP - THE core_map SEGMENT - ONE PER SYSTEM

CORE MAP ENTRY (CME) - ONE PER FRAME (1024 WORDS) OF CONFIGURED MAIN MEMORY

EACH CME REPRESENTS A FRAME OF MAIN MEMORY AND IDENTIFIES THE CURRENT OCCUPANT OF THAT FRAME
PAGE CONTROL OVERVIEW

NOT PART OF THE SST SEGMENT ANY MORE, BUT LOGICALLY PART OF THE SST

ACTIVE SEGMENT TABLE (AST) - ONE PER SYSTEM

ACTIVE SEGMENT TABLE ENTRY (ASTE) - ONE PER ACTIVE SEGMENT

LIST OF ACTIVE (CURRENTLY BEING USED) SEGMENTS

PAGE TABLES (PT) - ONE PER ACTIVE SEGMENT, THE OTHER HALF OF EACH ASTE

PAGE TABLE WORD (PAGE PTW) - EITHER 4, 16, 64, OR 256 PER PAGE TABLE

EACH PTW DEFINES THE CURRENT LOCATION OF A PAGE OF THE SEGMENT: DISK, MAIN MEMORY ADDRESS, OR NULL
PAGE CONTROL TERMINOLOGY

PAGING: THE PROCESS OF TRANSFERRING PAGES OF DATA BETWEEN DISK STORAGE AND MAIN MEMORY (CORE) TO ACHIEVE THE EFFECT OF ALL DATA BEING IN MEMORY ALL THE TIME

CORE: AN OBSOLETE TERM USED FREQUENTLY TO REFER TO MAIN MEMORY (WHICH IS MOS TECHNOLOGY, NOT CORE TECHNOLOGY)

PAGE FAULT: AN EXCEPTION CONDITION DETECTED BY THE PROCESSOR HARDWARE (IN THE APPENDING UNIT) WHEN AN ATTEMPT IS MADE TO USE A PTW SPECIFYING THAT ITS PAGE IS NOT IN MAIN MEMORY
**PAGE CONTROL DATA BASES**

**PAGE TABLES**

| ADDRESS | ADD_TYPE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALUE | VALU
PAGE CONTROL DATA BASES

PAGE TABLES

ALL PAGE TABLES ARE ASSOCIATED WITH, AND IMMEDIATELY FOLLOW AN ASTE IN THE AST REGION OF THE SST.

EACH PTW DESCRIBES THE STATUS OF ONE PAGE OF THE SEGMENT CURRENTLY IN POSSESSION OF THE ASSOCIATED ASTE, INCLUDING:

- THE DEVICE ADDRESS OF THE COPY OF THE PAGE
- PTW VALID INDICATOR AND FAULT NUMBER (FAULT #1)
- FLAGS INDICATING VARIOUS STATES AND PROPERTIES OF THE PAGE SUCH AS I/O IN PROGRESS, WIRED, USED, MODIFIED

THE ADDRESS PORTION OF EACH PTW IS INITIALIZED FROM THE SEGMENT'S VTOCE FILE MAP AT SEGMENT ACTIVATION TIME.

Not To Be Reproduced 8-7 F80A
PAGE TABLE DEVICE ADDRESSES

DISK PACK

PAGE 1

DISK RECORD NUMBER
NULL ADDRESS
MAIN MEMORY ADDRESS

A 4K PAGE TABLE

PAGE 3

MAIN MEMORY

"GHOST" PAGE OF ZEROES

PAGE 2
## PAGE CONTROL DATA BASES

### CORE MAP

<table>
<thead>
<tr>
<th>0</th>
<th>FORWARD REL POINTER</th>
<th>BACKWARD REL PTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DEVICE ADDRESS</td>
<td>TYPE</td>
</tr>
<tr>
<td>2</td>
<td>PTW OFFSET IN SST</td>
<td>A STE OFFSET IN SST</td>
</tr>
<tr>
<td>3</td>
<td>PIN COUNTER</td>
<td>PAGE SYNCHRONIZATION REL PTR</td>
</tr>
</tbody>
</table>

**CORE MAP ENTRY (CME)**

A HARDCORE DATABASE, LOCATED IN THE CORE MAP SEGMENT. ONE CME PER CONFIGURED FRAME OF MAIN MEMORY.

- THE CORE MAP IS A PERMANENTLY WIRED, UNPAGED, SEGMENT CONTAINING AN ARRAY OF CORE MAP ENTRIES (CME'S)

- ONE CORE MAP PER SYSTEM

- ONE CME PER ADDRESSABLE MAIN MEMORY FRAME

- IF THE CONFIGURATION HAS HOLES IN THE MEMORY ADDRESS ASSIGNMENTS, OR MEMORIES WHICH ARE TURNED OFF, THOSE CMES ARE PRESENT ANYWAY (BUT UNUSED) (0 TO HIGHEST FRAME ADDRESS)
Each CME describes the status of one page frame in main memory including:

- The disk address of the page currently occupying the frame
- Address of the ASTE and PTW of the occupant
- Flags indicating various states and properties of the frame and its occupant such as I/O in progress, notification requested, and PIN weight

The CME's are kept in a double-threaded circular list pointed to by sst.usedp.

- CME's for frames undergoing I/O are temporarily threaded out of the list.
- CME's for frames configured but not physically present are also threaded out but with thread word "7777777777" OCTAL.
- The remaining CME's represent main memory frames actively in use and subject to eviction by the page replacement algorithm.
PAGE CONTROL DATA BASES

SYSTEM SEGMENT TABLE (SST) HEADER

\* THE FIRST 512 WORDS OF THE SST IS CALLED THE SST HEADER AND CONTAINS:

\| A LARGE NUMBER OF GLOBAL VARIABLES VITAL TO THE OPERATION OF THE STORAGE SYSTEM AND ITS SUBSYSTEMS

\| NUMEROUS CELLS USED TO METER THE STORAGE SYSTEM

\* AMONG THOSE OF INTEREST TO PAGE CONTROL ARE THE FOLLOWING:

\| GLOBAL VARIABLES:

\| PAGE TABLE LOCK (sst.ptl)

\| NUMBER OF MAIN MEMORY FRAMES AVAILABLE FOR PAGING ACTIVITIES (sst.nused) AND NUMBER WIRED (sst.wired)

\| POINTERS TO THE BASE OF THE CME ARRAY AND TO THE CME OF THE "BEST" CANDIDATE PAGE FOR REPLACEMENT

\| PVT INDEX OF THE RPV (USED DURING INITIALIZATION)

\| METERS

\| THRASHING, POST-PURGE-TIME, PAGE FAULTS ON DIRECTORIES, RING 0 PAGE FAULTS, LOOP LOCK TIME, SEGMENT MOVES
PAGE CONTROL DATA BASES

SYSTEM SEGMENT TABLE (SST) HEADER

| PAGING METERS REPORTED BY file_system_meters SUCH AS STEPS, NEEDS, CEILING, SKIPS |
PAGE CONTROL DATA BASES

OTHER DATA BASES

Although basically data bases of volume management, the following contain information required by page control (as indicated)

Physical Volume Table (PVT) - One per system

- Information required by the disk DIM for I/O

- Information used by the disk record allocator/deallocator (free_store) such as:
  - The number of unallocated records left on the volume
  - The location of the record stock for the volume

Record stocks

- Record stocks are kept in a wired segment: stock_seg

- The record stock for a volume is a list of some of the records which are free on the volume

- When there are no more entries available in the stock, it is updated from the volume map

Not To Be Reproduced
IF THE STOCK BECOMES FULL, SOME OF ITS ENTRIES ARE UPDATED TO THE VOLUME MAP AND REMOVED FROM THE STOCK.

A COMPLEX MECHANISM (SEE volmap.alm, volmap_page.alm) MAKES IT POSSIBLE TO REFERENCE THE VOLUME MAP PAGES WHILE SATISFYING A PAGE FAULT.
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

* Within any demand paging environment, the choice of which page to evict is crucial to system performance.

* One of the better choices for eviction is the "least recently used" page...or (because of efficiency), the "least recently noticed as being used" page.

* The MULTICS page replacement algorithm (PRA), known in the literature as the "clock" algorithm was one of the first ever to be implemented.

* The version as it exists today is a direct descendant of Corbato's original algorithm (see Section 5 of the "MULTICS STORAGE SYSTEM PLM", AN61, for a bibliography).

* Pages are kept in a circular list, the core used list, implemented by the doubly threaded CME's.

* A pointer, maintained in the SST, (sst.usedp) points to the logical head of this list as follows:
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

THE CLOCK ALGORITHM

1. THE "REPLACER" SEARCHES FOR THE FIRST PAGE WHICH IS NEITHER WIRED NOR MODIFIED, AND HAS NOT BEEN USED SINCE LAST INSPECTED, MAKING THAT PAGE IMMEDIATELY AVAILABLE TO THE REQUESTOR.

2. THE "PURIFIER" THEN CATCHES UP, INITIATING WRITES FOR ALL "NOT USED BUT MODIFIED" PAGES PASSED OVER BY THE "REPLACER" AND TURNING OFF THE USED FLAG FOR ALL USED PAGES.
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

PAGE FAULT HANDLING IS THE MOST VISIBLE AND CRUCIAL SERVICE OF PAGE CONTROL

A PAGE FAULT OCCURS WHEN A USER REFERENCES A PAGE OF SOME SEGMENT THAT IS NOT IN MAIN MEMORY

OR MORE SPECIFICALLY: HARDWARE ATTEMPTS TO USE A PTW THAT INDICATES ITS PAGE IS NOT IN THE MAIN MEMORY

PAGE FAULT HANDLING IS IMPLEMENTED IN THE ALM PROGRAM page_fault WHICH IS INVOKED DIRECTLY BY THE FAULT VECTOR CODE (page_fault IS THE FAULT INTERCEPTOR FOR PAGE FAULTS)

THE PRINCIPAL STEPS OF page_fault ARE:

SAVE ALL MACHINE CONDITIONS, MASK AGAINST INTERRUPTS, AND ESTABLISH A STACK FRAME ON THE BASE OF THE PROCESSOR_DATA SEGMENT (PRDS), WHICH IS USED AS THE STACK FOR INTERRUPTS AND PAGE FAULTS

CHECK FOR ILLEGAL CONDITIONS AND CRASH IF SO
ATTEMPT TO LOCK THE PAGE TABLE LOCK (sst.ptl) AND WAIT IF UNSUCCESSFUL

LOCATE THE RESPONSIBLE PTW AND ITS ASTE. THIS IS OFTEN THE MOST DIFFICULT TASK

IT IS DIFFICULT BECAUSE IT REQUIRES FETCHING THE SDW FROM THE DSEG, WHICH IS, ITSELF, PAGED, AND NOT GUARANTEED TO BE IN MEMORY

CHECK FOR TWO WINDOW SITUATIONS INVOLVING SOME OTHER PROCESS HANDLING A PAGE FAULT FOR THE SAME PAGE:

IF PAGE IS NOW IN, THEN UNLOCK THE LOCK AND RESTART THE MACHINE CONDITIONS

IF PAGE IS BEING READ IN NOW, DEVELOP THE WAIT EVENT FOR THE PTW AND SKIP THE NEXT THREE STEPS

INVoke read_page TO FIND THE LEAST RECENTLY (NOTICED AS BEING) USED MAIN MEMORY FRAME, BEGIN THE PAGE-READING FUNCTION, AND DEVELOP THE WAIT EVENT

EXECUTE THE REPLACEMENT ALGORITHM’S WRITE-BEHIND (PURIFIER) FUNCTION, CAUSING PASSED OVER WRITE REQUESTS TO BE QUEUED

METER THE PAGE FAULT TO INCLUDE: TIME SPENT; MAIN MEMORY USAGE OF THIS PROCESS; RING ZERO, DIRECTORY, AND PER-PROCESS FAULTS
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

TRANSFER TO THE TRAFFIC CONTROLLER, WHO PLACES THE PROCESS IN THE WAIT STATE, UNLOCKS THE PAGE TABLE LOCK, AND ABANDONS THE ENVIRONMENT (SEE "TRAFFIC CONTROL", TOPIC 10)

WHEN THE PAGE READING I/O IS COMPLETE, THE EVENT WILL BE POSTED. THE WAITING PROCESS WILL BE GIVEN THE PROCESSOR AGAIN AND TRAFFIC CONTROLLER WILL TRANSFER THE FAULTING PROCESS TO page_fault$wait_return TO RESTART THE MACHINE CONDITIONS
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

LOGICAL VIEW OF SEGMENT A
(CL = 3)
(RU = 2)

PHYSICAL VOLUME

SEGMENT A's VTOCE FILE MAP

PAGE FAULT SCENARIO

SCENE 1: AFTER SEGMENT ACTIVATION

1. ASTE/PT PAIR FREED AND AlLOCATED TO SEGMENT A
2. VTOCE'S ACTIVATION DATA COPIED INTO ASTE
3. VTOCE'S FILE MAP DATA COPIED INTO PTW

Not To Be Reproduced
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

LOGICAL VIEW OF SEGMENT A (CL = 3) (RU = 2)

PHYSICAL VOLUME

SEGMENT A's VTOCE FILE MAP

ASSOCIATED CORE MAP ENTRIES

MAIN MEMORY FRAMES

PAGE FAULT SCENARIO

SCENE 2: AFTER PAGE FAULT ON PAGE NO. 2

1. FRAME 3702 FREED AND ALLOCATED TO PAGE NO. 2
2. PAGE NO. 2 COPIED INTO FRAME 3702
3. ADDRESS IN PTW NO. 2 COPIED TO CME 3702
4. ADDRESS IN PTW NO. 2 REPLACED WITH MAIN MEMORY ADDRESS
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

LOGICAL VIEW OF SEGMENT A
(CL = 3)
(RU = 2)

PHYSICAL VOLUME

SEGMENT A's VTOCE FILE MAP

ASSOCIATED CORE MAP ENTRIES

MAIN MEMORY FRAMES

PAGE FAULT SCENARIO

SCENE 3: AFTER PAGE FAULT ON PAGE NO. 1

1. FRAME 5614 FREED AND ALLOCATED TO PAGE NO. 1
2. FRAME 5614 CEROED BECAUSE OF PTW NO. 1's NULL ADDRESS
3. RECORD 2103 ALLOCATED TO PAGE NO.1
4. ADDRESS FOR RECORD 2103 WRITTEN INTO CME 5614 AS A NULLED ADDRESS

Not To Be Reproduced

8-22

F80A
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

LOGICAL VIEW OF SEGMENT A
(CL = 3) (RU = 2)

PHYSICAL VOLUME

SEGMENT A's VTOCE FILE MAP

SERVICE OF PAGE CONTROL

PAGE FAULT SCENARIO

SCENE 4A: PAGE NO.1 (UNMODIFIED) EVICTED FROM MAIN MEMORY

1. NULLED ADDRESS IN CME 5614 COPIED TO PTW NO.1
2. CME 5614 FREED
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

LOGICAL VIEW OF SEGMENT A
(CL = 3)
(RU = 2)

PHYSICAL VOLUME

SEGMENT A's VTOCE FILE MAP

PAGE FAULT SCENARIO

SCENE 5A: AFTER SEGMENT DEACTIVATION

1. PAGE NO. 2 EVICTED FROM MAIN MEMORY AFTER ADDRESS IN CME 3702 COPIED TO PTW NO. 2
2. PTW ADDRESSES WRITTEN TO VTOCE FILE MAP WITH NULLED ADDRESS IN PTW NO. 1
   CONVERTED TO A NULL ADDRESS
3. RECORD 2103 FREED
4. ASTE/PT PAIR FREED
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

LOGICAL VIEW OF SEGMENT A
(CL = 3)
(HU = 3)

PHYSICAL VOLUME

SEGMENT A's VTOCE FILE MAP

PAGE FAULT SCENARIO

SCENE 48: PAGE NO.1(MODIFIED) EVICTED FROM MAIN MEMORY
(AND DISK I/O KNOWN TO BE COMPLETE)

1. FRAME 5614 WRITTEN TO RECORD 2103
2. NULLED ADDRESS IN CME 5614 RESURRECTED AND COPIED TO PTW NO. 1
3. CME 5614 FREED

Not To Be Reproduced
SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

LOGICAL VIEW OF SEGMENT A
(CL = 3)
(RU = 3)

PHYSICAL VOLUME

SEGMENT A's
VTOCE FILE MAP

ASSOCIATED
CORE MAP
ENTRIES

MAIN MEMORY
FRAMES

PAGE FAULT SCENARIO

SCENE 58: AFTER SEGMENT DEACTIVATION

1. PAGE NO. 2 EVICTED FROM MAIN MEMORY
2. PTW ADDRESSES WRITTEN TO VTOCE FILE MAP
3. ASTE/PT PAIR FREED

Not To Be Reproduced
SERVICES OF PAGE CONTROL

POST PURGING

POST PURGING IS PERFORMED BY THE PROCEDURE post_purge

POST PURGING IS AN OPTIONAL SERVICE USED TO OPTIMIZE THE PAGE REPLACEMENT ALGORITHM

POST PURGING:

FAVORS THE REPLACING OF PAGES USED BY A PROCESS WHICH HAS JUST LOST ELIGIBILITY (SEE "TRAFFIC CONTROL", TOPIC 9)

IS A WORK CLASS SETTABLE ATTRIBUTE

POST PURGING IS LARGELY USELESS TODAY, BECAUSE MAIN MEMORY SIZE IS SO MUCH GREATER
FILE_SYSTEM_METERS - DISPLAYS MISCELLANEOUS METERING INFORMATION FOR THE FILE SYSTEM

ONLY PARTS RELEVANT TO PAGE CONTROL INCLUDED HERE; SEE TOPIC 7 (SEGMENT CONTROL) FOR THE REST

Total metering time 0:20:02

<table>
<thead>
<tr>
<th>#</th>
<th>ATB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Needc</td>
<td>62654</td>
</tr>
<tr>
<td>Ring 0 faults</td>
<td>19.194 msec.</td>
</tr>
<tr>
<td>PDIR faults</td>
<td>16.639 %</td>
</tr>
<tr>
<td>Level 2 faults</td>
<td>50.607 %</td>
</tr>
<tr>
<td>DIR faults - Direct</td>
<td>21.556 %</td>
</tr>
<tr>
<td>New Pages - Page</td>
<td>7.645 %</td>
</tr>
<tr>
<td>Volmap_seg</td>
<td>14.661 %</td>
</tr>
<tr>
<td>Zero pages</td>
<td>0.000 msec.</td>
</tr>
<tr>
<td>Laps</td>
<td>1561.779 msec</td>
</tr>
<tr>
<td>Steps</td>
<td>361453</td>
</tr>
<tr>
<td>Skip wired</td>
<td>3.728 msec.</td>
</tr>
<tr>
<td>used</td>
<td>3.327 msec.</td>
</tr>
<tr>
<td>mod</td>
<td>108.701 msec</td>
</tr>
<tr>
<td>fc pin</td>
<td>109719</td>
</tr>
<tr>
<td>cl pin</td>
<td>140336</td>
</tr>
<tr>
<td>Volmap_seg</td>
<td>37717</td>
</tr>
<tr>
<td>Max</td>
<td>23726</td>
</tr>
<tr>
<td>Total steps</td>
<td>3419</td>
</tr>
<tr>
<td>Average pages wired</td>
<td>5.770</td>
</tr>
</tbody>
</table>

Pin waits \* Page Replacement Algorithm

Pin wait = 3
Change by two

If used bit = 0
Then hook
AT PINWAIT

Pin wait gives
17 extra laps

Not To Be Reproduced 8-28
(End Of Topic)
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Control Overview</td>
<td>9-1</td>
</tr>
<tr>
<td>Traffic Control Terminology</td>
<td>9-3</td>
</tr>
<tr>
<td>Traffic Control Data Bases</td>
<td>9-5</td>
</tr>
<tr>
<td>tc_data</td>
<td>9-5</td>
</tr>
<tr>
<td>Services of Traffic Control</td>
<td>9-8</td>
</tr>
<tr>
<td>Wait Locks</td>
<td>9-8</td>
</tr>
<tr>
<td>Processor Multiplexing</td>
<td>9-9</td>
</tr>
<tr>
<td>Traffic Control Meters</td>
<td>9-18</td>
</tr>
<tr>
<td>total_time_meters</td>
<td>9-18</td>
</tr>
<tr>
<td>traffic_control_meters</td>
<td>9-19</td>
</tr>
<tr>
<td>traffic_control_queue</td>
<td>9-21</td>
</tr>
<tr>
<td>work_class_meters</td>
<td>9-23</td>
</tr>
<tr>
<td>respons_meters</td>
<td>9-24</td>
</tr>
<tr>
<td>post_purge_meters</td>
<td>9-26</td>
</tr>
<tr>
<td>Traffic Control Commands</td>
<td>9-27</td>
</tr>
<tr>
<td>print_tuning_parameters</td>
<td>9-27</td>
</tr>
<tr>
<td>print_apt_entry</td>
<td>9-28</td>
</tr>
</tbody>
</table>
TRAFFIC CONTROL OVERVIEW

FUNCTION

TRAFFIC CONTROL (OR THE "TRAFFIC CONTROLLER") IS RESPONSIBLE FOR MANAGING THE ASSIGNMENT OF PHYSICAL PROCESSORS TO MULTICS PROCESSES AND IMPLEMENTING THE SYSTEM'S WAIT/NOTIFY AND INTERPROCESS COMMUNICATION PRIMITIVES.

THE FUNCTIONS ASSUMED BY THE TRAFFIC CONTROLLER ARE KNOWN AS MULTIPROGRAMMING, MULTIPROCESSING, SCHEDULING, DISPATCHING, PROCESSOR MANAGEMENT, AND INTERPROCESS COMMUNICATION.

ITS MAJOR FUNCTION IS ALLOWING PROCESSES TO AWAIT THE COMPLETION OF FILE SYSTEM OPERATIONS, SUCH AS PAGE I/O.

TRAFFIC CONTROL CAN BE INVOKED BY SUBROUTINE CALLS AND INTERRUPTS.

THERE ARE NO IMPORTANT USER SUBROUTINE INTERFACES, BUT THERE ARE PRIVILEGED SUBROUTINE INTERFACES FOR PROCESS CREATION, ADJUSTMENT OF SCHEDULING PARAMETERS, ETC.

MAJOR DATA BASES

TC_DATA SEGMENT - ONE PER SYSTEM. CONTAINS THE FOLLOWING FOUR DATA BASES:

TC_DATA HEADER - ONE PER SYSTEM

CONTAINS VARIOUS METERS, COUNTERS AND POINTERS USED BY THE TRAFFIC CONTROLLER.

Not To Be Reproduced
TRAFFIC CONTROL OVERVIEW

ACTIVE PROCESS TABLE (APT) - ONE PER SYSTEM

ACTIVE PROCESS TABLE ENTRY (APTE) - ONE OCCUPIED PER ACTIVE PROCESS (TOTAL NUMBER IS DETERMINED BY CONFIG DECK)

EACH APTE CONTAINS VARIOUS ATTRIBUTES OF AN ACTIVE PROCESS INCLUDING THE PROCESS ID, STATE, THE VALUE OF ITS DESCRIPTOR BASE REGISTER (DBR), SCHEDULING PARAMETERS, AND A POINTER TO THE PROCESS'S ITT ENTRIES.

THE APTE CONTAINS ALL INFORMATION THE SUPERVISOR NEEDS TO KNOW ABOUT A PROCESS WHEN THE PROCESS IS NOT RUNNING.

INTERPROCESS TRANSMISSION TABLE (ITT) - ONE PER SYSTEM

ITT ENTRY - ONE OCCUPIED PER OUTSTANDING IPC WAKEUP

A QUEUE FOR TEMPORARILY STORING IPC WAKEUP INFORMATION (CHANNEL NAME, RANDOM DATA, PROCESS ID, ETC)

WORK CLASS TABLE (WCT) - ONE PER SYSTEM

WORK CLASS TABLE ENTRY (WCTE) - ONE PER WORKCLASS

EACH WCTE CONTAINS ADMINISTRATOR DEFINED PARAMETERS OF THE WORKCLASS, VARIOUS METERS AND POINTERS.
PROCESS:

AN ADDRESS SPACE AND AN EXECUTION POINT WITHIN THAT ADDRESS SPACE

MULTIPROGRAMMING:

PERTAINING TO THE CONCURRENT EXECUTION OF TWO OR MORE PROGRAMS BY INTERLEAVING THEIR EXECUTION

MULTIPROCESSING:

PERTAINING TO THE SIMULTANEOUS EXECUTION OF TWO OR MORE PROGRAMS BY A MULTIPROCESSOR SYSTEM (PARALLEL PROCESSING)

A SINGLE PROCESS IS NOT ON MORE THAN 1 CPU

ELIGIBLE:

AN ADJECTIVE DESCRIBING THOSE PROCESSES ACTIVELY COMPETING FOR A PROCESSOR. ALL PROCESSES ARE EITHER ELIGIBLE OR INELIGIBLE

SCHEDULING:

PERTAINS TO THE ACT OF CHOOSING AND PROMOTING AN "INELIGIBLE" PROCESS TO "ELIGIBLE" STATUS

DISPATCHING:

PERTAINS TO THE ACT OF CHOOSING AND PLACING AN ELIGIBLE PROCESS IN THE "RUNNING" STATE (IE: EXECUTING INSTRUCTIONS ON A PROCESSOR)
TRAFFIC CONTROL TERMINOLOGY

WORK CLASS: A WELL DEFINED SET OF USERS (USUALLY CONSISTING OF ONE OR MORE PROJECTS) HAVING COMMON PERFORMANCE PARAMETERS. THE USER COMMUNITY MAY BE DIVIDED INTO NO MORE THAN 16 WORK CLASSES.

WORKING SET: THE SET OF PAGES A PROCESS TOUCHES DURING A GIVEN INTERVAL. THE SIZE OF A CURRENT WORKING SET IS PREDICTIVE OF FUTURE MEMORY REQUIREMENTS.

Diagram:

- Memory
- OS/360
- DEB
- TSO
- DEB2
- TSO2
- Other

Working set: memory units referred to as Frankstons

- Loading: getting processes working in memory
- Time slice or quantum: amount of VCPU you can use while your eligible. Once you use it your turn is over.
- Prempt: one process on CPU voluntarily gives it up even though they need it.
- VCPU - CPU billed to users, doesn't count 1 interrupts & most faults.
TRAFFIC CONTROL DATA BASES

TC_DATA

tc_data HEADER

WORK CLASS TABLE (WCT)
(0:16)

ACTIVE PROCESS TABLE (APT)
(0:CONFIGURED)

INTERPROCESS TRANSMISSION TABLE (ITT)
(0:CONFIGURED)

TC_DATA
A WIRED DATA BASE — ONE PER SYSTEM
## TRAFFIC CONTROL DATA BASES

### TC DATA

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PROCESS IDENTIFIER</td>
</tr>
<tr>
<td>1</td>
<td>TIME SINCE TL CHANGED (TS)</td>
</tr>
<tr>
<td>2</td>
<td>TIME SINCE LAST INTERACTION (TI)</td>
</tr>
<tr>
<td>3</td>
<td>NUMBER OF PAGE FAULTS</td>
</tr>
<tr>
<td>4</td>
<td>EXECUTION STATE</td>
</tr>
<tr>
<td>5</td>
<td>EXECUTION EVENT</td>
</tr>
<tr>
<td>6</td>
<td>EXECUTION THREAD</td>
</tr>
<tr>
<td>7</td>
<td>IPC EVENT THREAD</td>
</tr>
<tr>
<td>8</td>
<td>IPS MESSAGE(S)</td>
</tr>
<tr>
<td>9</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>10</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>11</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>12</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>13</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>14</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>15</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>16</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>17</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>18</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
<tr>
<td>19</td>
<td>R_PTR TO ASTE OF OSEG</td>
</tr>
</tbody>
</table>

**NOT TO BE REPRODUCED**
TRAFFIC CONTROL DATA BASES

TC DATA

20
WAKEUP EVENT FOR ALARM CLOCK MANAGER

21
ALARM TIME THREAD

22
ALARM TIME

23
EVENT CHANNEL TO NOTIFY ON PROCESS TERMINATION

24
WORKING SET ESTIMATE

25
MAXIMUM VALUE FOR TE (TEMAX)

26
DEADLINE SCHEDULING TIME

28
APTE LOCK

30
CPU MONITOR

31
PAGING MEASURE

32
ACCESS ISOLATION MECHANISM (AIM) CLASS

34
DESCRIPTOR BASE REGISTER VALUE (DBR)

35
VIRTUAL CPU TIME

36
ITT MESSAGES SENT AND NOT READ OUT

37
ITT MESSAGES RECEIVED AND NOT READ OUT

39
8 WORDS FOR RESPONSE TIME METERING

40
SAVED VALUE OF TETE

41
PROCS REQUIRED

12 WORDS OF PADDING (TO 64 WORDS):

ACTIVE PROCESS TABLE (APT)
A HARDCORE TC_DATA DATA BASE = ONE PER SYSTEM
CONSISTS OF AN ARRAY OF APTE'S AS ABOVE

ACTIVE PROCESS TABLE ENTRY (APTE)
ONE PER ACTIVE PROCESS

Not To Be Reproduced 9-7
SERVICES OF TRAFFIC CONTROL

WAIT LOCKS

WAIT LOCKS

WAIT EVENT

ADD EVENT

DELETE EVENT

NOTIFY EVENT
SERVICES OF TRAFFIC CONTROL
PROCESSOR MULTIPLEXING

- Since the number of active processes generally exceeds the number of processors (often 50:1) the processors must be multiplexed.

- Processor multiplexing is the primary responsibility of the traffic controller.

- The MULTICS architecture dictates the following axioms:

  1. All processors are symmetrical.

  2. An interrupt is seen by all processors and is serviced by the processor that claims it first.

     (Not actually true, due to hardware connection limitations)

  3. There are no master or slave processors. Only a processor designated to perform bootload and shutdown (the "bootload processor").

     Bootload processor can be changed at any time by dynamic reconfiguration.

  4. A processor may be "in" at most one process at a time.

  5. A process may execute on one and only one processor at a time. The process may, however, "randomly" migrate from one processor to another.
SERVICES OF TRAFFIC CONTROL
PROCESSOR MULTIPLEXING

A PROCESSOR WILL, AT ALL TIMES, BE "IN" A PROCESS

PROCESSES MAY EXIST IN ONE OF SIX STATES - ONE OF WHICH IS "RUNNING" (IE: EXECUTING ON A PROCESSOR)

THE TRAFFIC CONTROLLER CODE PERFORMS ALL STATE CHANGES. THE STATE CHANGES FALL INTO TWO CATEGORIES: SELF IMPOSED AND EXTERNALLY IMPOSED.

SELF IMPOSED STATE CHANGES (THE PROCESS MUST HAVE A PROCESSOR)

RUNNING -> READY (#1 -> #2)

THE PROCESS WAS TOLD (BY EITHER A CONNECT FAULT FROM ANOTHER PROCESSOR OR A TIMER RUNOUT) TO GIVE UP ITS PROCESSOR

THE PROCESS IS NOW WAITING FOR NO OTHER RESOURCE THAN A PROCESSOR

RUNNING -> WAITING (#1 -> #3)

THE PROCESS ISSUED A REQUEST FOR A "SYSTEM EVENT" (EG: A PAGE FAULT OR A WAIT LOCK)

"SYSTEM EVENTS" OCCUR AFTER A PREDICTABLY SHORT PERIOD OF TIME AND ARE HANDLED BY THE WAIT/NOTIFY MECHANISM

THE PROCESS IS NOW WAITING FOR A NOTIFY INDICATING THE COMPLETION OF THE EVENT (EG: THE ARRIVAL OF THE PAGE IN MAIN MEMORY)
SERVICES OF TRAFFIC CONTROL
PROCESSOR MULTIPLEXING

RUNNING -> PAGE TABLE LOCK WAITING (#1 -> #6)

THE PROCESS ATTEMPTED TO LOCK THE PAGE-TABLE LOCK AND
FOUND IT ALREADY LOCKED

THE PROCESS IS NOW WAITING FOR A NOTIFY INDICATING THE
UNLOCKING (A SYSTEM EVENT)

RUNNING -> BLOCKED (#1 -> #4)

THE PROCESS ISSUED A REQUEST FOR A "USER EVENT" (EG: A
READ FROM THE TERMINAL)

"USER EVENTS" OCCUR AFTER A PREDICTABLY LONG PERIOD OF
TIME AND ARE HANDLED BY THE BLOCK/WAKEUP MECHANISM

THE PROCESS IS NOW WAITING FOR A WAKE UP INDICATING THE
COMPLETION OF THE EVENT (EG: A LINE_FEED GENERATES A
WAKE_UP) ARE HANDLED BY THE BLOCK/WAKE_UP MECHANISM

A PROCESS GOING BLOCKED GIVES UP ITS RING ZERO STACK

RUNNING -> STOPPED (#1 -> #5)

THE PROCESS EXECUTED THE logout OR new_proc COMMAND,
EVENTUALLY CALLING hcs_$stop_process

THE PROCESS IS PROHIBITED FROM RUNNING AGAIN AND WILL
QUICKLY BE DESTROYED BY THE INITIALIZER
SERVICES OF TRAFFIC CONTROL
PROCESSOR MULTIPLEXING

| EXTERNALLY IMPOSED STATE CHANGES (THE STATE IS CHANGED BY ANOTHER PROCESS) |
| READY -> RUNNING (#2 -> #1) |
| THE PROCESS (WHICH WAS WAITING FOR NO OTHER RESOURCE THAN A PROCESSOR) WAS CHOSEN AS SUCCESSOR BY A PROCESS RELINQUISHING A PROCESSOR |
| THE PROCESS IS NOW EXECUTING |

| BLOCKED -> READY (#4 -> #2) |
| SOME OTHER PROCESS SENT A WAKEUP INDICATING THE COMPLETION OF THE EVENT THIS PROCESS WAS WAITING ON, AND CHANGE THE STATE OF THIS PROCESS |
| THE PROCESS IS NOW WAITING FOR NO OTHER RESOURCE THAN A PROCESSOR |

| WAITING -> READY (#6 -> #2) |
| PTL-WAITING -> READY (#3 -> #2) |
| SOME OTHER PROCESS SENT A NOTIFY INDICATING THE COMPLETION OF THE EVENT THIS PROCESS WAS WAITING ON, AND CHANGED THE STATE OF THIS PROCESS |
| THE PROCESS IS NOW WAITING FOR NO OTHER RESOURCE THAN A PROCESSOR |
TRAFFIC CONTROL STATES

ELIGIBLE PROCESSES (13)

INELIGIBLE PROCESSES (87)

* ALL FIGURES ARE BASED ON METERS FROM A 100 USER, 3 CPU, 2.5M MEMORY, MAXE = 16 SYSTEM.
SERVICES OF TRAFFIC CONTROL
PROCESSOR MULTIPLEXING

TRAFFIC CONTROL SCENARIO

SCENE 1: CPU "C" TAKES AN INTERRUPT

1. EVENT 3331 OCCURES. CPU "C" TAKES THE INTERRUPT AND STORES MACHINE CONDITIONS

2. May.F01 NOTIFIES Kelly.EREG BY CHANGING HIS STATE TO READY (r)

3. May.F01 CALLS get-processor WHICH LOOKS FOR AN EXECUTING (x) PROCESS LOWER THAN Kelly.EREG, FINDS IDLE_A, AND SENDS CPU "A" A CONNECT FAULT (cioc INSTRUCTION)

4. May.F01 RESUMES PRIOR ACTIVITY BY RESTORING MACHINE CONDITIONS

Not To Be Reproduced 9-14 FS0A
SERVICES OF TRAFFIC CONTROL
PROCESSOR MULTIPLEXING

TRAFFIC CONTROL SCENARIO
SCENE 2: CPU "A" RECEIVES CONNECT FAULT

1. CPU "A" RECEIVES CONNECT FAULT AND STORES MACHINE CONDITIONS

2. Idle_A BEGINS PRE-EMPTION BY CHANGING STATE TO READY (r)

3. Idle_A CALLS get-work WHICH LOOKS FOR THE HIGHEST READY (r) PROCESS, FINDS Kelly.EREG, CHANGES HIS STATE TO EXECUTING (x) AND CPU TAG TO "A", AND LOADS HIS DBR

4. Kelly.EREG RESUMES PRIOR ACTIVITY BY RESTORING MACHINE CONDITIONS

Not To Be Reproduced 9-15 F80A
WORKING SETS AND PAGE THRASHING

PAGE REFERENCE STRING

CASE # 1: MINIMAL PAGE FAULTS

CASE # 2: UNNECESSARY PAGE FAULTS

WORKING SET = 7 PAGES

- CASE # 1: THRASHING = \( \frac{0}{7} = 0\% \)

- CASE # 2: THRASHING = \( \frac{3}{10} = 30\% \)
## IDLE CATEGORIES

<table>
<thead>
<tr>
<th></th>
<th>ELIGIBLE</th>
<th>NOT ELIGIBLE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RUNNING</td>
<td>WAITING</td>
<td>READY</td>
</tr>
<tr>
<td>ZERO IDLE</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NMP IDLE</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LOADING IDLE</td>
<td>1</td>
<td>2*</td>
<td>0</td>
</tr>
<tr>
<td>MP IDLE</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>WORKCLASS IDLE</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Example:** 2 CPUs, MAX=3

*Last process being loaded.

---

SERVICES OF TRAFFIC CONTROL
PROCESSOR MULTIPLEXING

Not To Be Reproduced 9-17
TOTAL TIME METERS - OVERVIEW OF HOW THE SYSTEM IS USING ITS RESOURCES, ALSO MEASURED AGAINST NON-IDLE TIME

<table>
<thead>
<tr>
<th>Total metering time</th>
<th>0:20:36</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Page Faults</td>
<td>6.55</td>
</tr>
<tr>
<td>% PC Loop Locks</td>
<td>0.39</td>
</tr>
<tr>
<td>% PC Queue</td>
<td>0.86</td>
</tr>
<tr>
<td>% Seg Faults</td>
<td>1.74</td>
</tr>
<tr>
<td>% Bound Faults</td>
<td>0.15</td>
</tr>
<tr>
<td>% Interrupts</td>
<td>9.02</td>
</tr>
<tr>
<td>% Other Faults</td>
<td>8.45</td>
</tr>
<tr>
<td>% Network</td>
<td>4.54</td>
</tr>
<tr>
<td>% TC Loop Locks</td>
<td>0.20</td>
</tr>
<tr>
<td>% Post Purging</td>
<td>0.36</td>
</tr>
<tr>
<td>% MP Idle</td>
<td>0.70</td>
</tr>
<tr>
<td>% Work Class Idle</td>
<td>0.98</td>
</tr>
<tr>
<td>% Loading Idle</td>
<td>0.16</td>
</tr>
<tr>
<td>% NMP Idle</td>
<td>8.84</td>
</tr>
<tr>
<td>% Zero Idle</td>
<td>0.05</td>
</tr>
<tr>
<td>% Other Overhead</td>
<td>0.03</td>
</tr>
<tr>
<td>% Virtual CPU Time</td>
<td>62.53</td>
</tr>
</tbody>
</table>

Not To Be Reproduced 9-18 F80A
TRAFFIC CONTROL METERS

Traffic control meters

- TRAFFIC_CONTROL_METERS - DISPLAY THE STATE OF THE SCHEDULER

- OUTPUT COMES IN THREE PARTS, SHOWN OUT OF ORDER HERE:

  - QUEUE LENGTHS AND RESPONSE TIME - THESE ARE WEIGHTED AVERAGES OVER THE LAST FIFTEEN SECONDS.

  - ACTIVITIES VERSUS DEPTH - HOW DEEP THE TRAFFIC CONTROLLER HAD TO SEARCH TO FIND A SCHEDULABLE PROCESS

  - MISCELLANEOUS COUNTERS AND FREQUENCIES OF VARIOUS EVENTS

<table>
<thead>
<tr>
<th>Total metering time</th>
<th>0:20:34</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave queue length</td>
<td>16.52</td>
</tr>
<tr>
<td>Ave eligible</td>
<td>13.31</td>
</tr>
<tr>
<td>Response time</td>
<td>0.264 sec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEPTH</th>
<th>%PF</th>
<th>TBPF</th>
<th>%GTW</th>
<th>TBS</th>
<th>%CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12.0</td>
<td>22.8</td>
<td>10.8</td>
<td>11.6</td>
<td>8.1</td>
</tr>
<tr>
<td>2</td>
<td>11.8</td>
<td>21.0</td>
<td>9.2</td>
<td>12.2</td>
<td>7.4</td>
</tr>
<tr>
<td>3</td>
<td>10.8</td>
<td>24.9</td>
<td>8.7</td>
<td>14.1</td>
<td>8.0</td>
</tr>
<tr>
<td>4</td>
<td>10.2</td>
<td>27.9</td>
<td>8.5</td>
<td>15.2</td>
<td>8.4</td>
</tr>
<tr>
<td>5</td>
<td>9.5</td>
<td>29.8</td>
<td>8.3</td>
<td>15.5</td>
<td>8.4</td>
</tr>
<tr>
<td>6</td>
<td>8.4</td>
<td>33.5</td>
<td>7.8</td>
<td>16.5</td>
<td>8.4</td>
</tr>
<tr>
<td>7</td>
<td>7.4</td>
<td>36.3</td>
<td>7.3</td>
<td>16.8</td>
<td>8.0</td>
</tr>
<tr>
<td>8</td>
<td>30.0</td>
<td>48.6</td>
<td>39.5</td>
<td>16.8</td>
<td>43.3</td>
</tr>
</tbody>
</table>

Response time is a good indication of response time.

Not To Be Reproduced 9-19 F80A
<table>
<thead>
<tr>
<th>COUNTER</th>
<th>TOTAL</th>
<th>ATB</th>
<th>#/INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactions</td>
<td>7977</td>
<td>0.155 sec</td>
<td>1.525</td>
</tr>
<tr>
<td>Loadings</td>
<td>12161</td>
<td>0.102 sec</td>
<td></td>
</tr>
<tr>
<td>Blocks</td>
<td>14082</td>
<td>0.088 sec</td>
<td></td>
</tr>
<tr>
<td>Wakeups</td>
<td>36078</td>
<td>0.034 sec</td>
<td></td>
</tr>
<tr>
<td>Schedulings</td>
<td>12591</td>
<td>0.098 sec</td>
<td>1.578</td>
</tr>
<tr>
<td>Lost priority</td>
<td>1</td>
<td>1234.756 sec</td>
<td></td>
</tr>
<tr>
<td>Priority boosts</td>
<td>0</td>
<td>0.000 sec</td>
<td></td>
</tr>
<tr>
<td>I/O boosts</td>
<td>578</td>
<td>2.136 sec</td>
<td></td>
</tr>
<tr>
<td>Wait Page</td>
<td>127040</td>
<td>9.719 msec</td>
<td>15.926</td>
</tr>
<tr>
<td>Wait PTL</td>
<td>75691</td>
<td>16.313 msec</td>
<td>9.489</td>
</tr>
<tr>
<td>Wait Other</td>
<td>31912</td>
<td>38.693 msec</td>
<td>4.001</td>
</tr>
<tr>
<td>Total Waits</td>
<td>234643</td>
<td>5.262 msec</td>
<td>29.415</td>
</tr>
<tr>
<td>Notify Page</td>
<td>128954</td>
<td>9.575 msec</td>
<td></td>
</tr>
<tr>
<td>Notify PTL</td>
<td>75691</td>
<td>16.313 msec</td>
<td></td>
</tr>
<tr>
<td>Notify Other</td>
<td>25330</td>
<td>48.747 msec</td>
<td></td>
</tr>
<tr>
<td>Total Notifies</td>
<td>229975</td>
<td>5.369 msec</td>
<td></td>
</tr>
<tr>
<td>Get Processor</td>
<td>245856</td>
<td>5.022 msec</td>
<td></td>
</tr>
<tr>
<td>Pre-empteds</td>
<td>94235</td>
<td>13.103 msec</td>
<td>11.813</td>
</tr>
<tr>
<td>Getwork</td>
<td>338802</td>
<td>3.644 msec</td>
<td></td>
</tr>
<tr>
<td>Retry getwork</td>
<td>4988</td>
<td>0.248 sec</td>
<td></td>
</tr>
<tr>
<td>Extra notifies</td>
<td>2949</td>
<td>0.419 sec</td>
<td></td>
</tr>
<tr>
<td>Last EN event</td>
<td>0000000000071</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last NTO event</td>
<td>033022237767</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ALARM_CLOCK_METERS - DISPLAYS INFORMATION ABOUT THE USER ALARM TIMER FACILITY (HARDCORE INTERFACE FOR timer_manager_)

Total metering time 0:20:31
No. alarm clock sims. 2171
Simulation lag 5.245 msecs.
Max. lag 1.7340314e4 msecs.
TRAFFIC CONTROL METERs

- Traffic control queue displays the current contents of the scheduler queues, useful for getting an idea of what the user processes are doing.

FIRST PART OF OUTPUT IS ELIIGIBLE QUEUE: T 스케줄링의 내용을 보는 데 유용하지만 사용자의 프로세스가 무엇을 하고 있는지 알 수 있습니다.

Flags = 13, elapsed time = 124.7 sec, active events = 377

- Traffic control queue displays the current contents of the scheduler queues, useful for getting an idea of what the user processes are doing.

FIRST PART OF OUTPUT IS ELIIGIBLE QUEUE: T 스케줄링의 내용을 보는 데 유용하지만 사용자의 프로세스가 무엇을 하고 있는지 알 수 있습니다.

Flags = 13, elapsed time = 124.7 sec, active events = 377

- Traffic control queue displays the current contents of the scheduler queues, useful for getting an idea of what the user processes are doing.

FIRST PART OF OUTPUT IS ELIIGIBLE QUEUE: T 스케줄링의 내용을 보는 데 유용하지만 사용자의 프로세스가 무엇을 하고 있는지 알 수 있습니다.

Flags = 13, elapsed time = 124.7 sec, active events = 377

- Traffic control queue displays the current contents of the scheduler queues, useful for getting an idea of what the user processes are doing.

FIRST PART OF OUTPUT IS ELIIGIBLE QUEUE: T 스케줄링의 내용을 보는 데 유용하지만 사용자의 프로세스가 무엇을 하고 있는지 알 수 있습니다.

Flags = 13, elapsed time = 124.7 sec, active events = 377

- Traffic control queue displays the current contents of the scheduler queues, useful for getting an idea of what the user processes are doing.

FIRST PART OF OUTPUT IS ELIIGIBLE QUEUE: T 스케줄링의 내용을 보는 데 유용하지만 사용자의 프로세스가 무엇을 하고 있는지 알 수 있습니다.

Flags = 13, elapsed time = 124.7 sec, active events = 377
TRAFFIC CONTROL METERS

traffic control queue

SECOND PART IS REALTIME, INTERACTIVE, AND ALL WORKCLASS QUEUES:

<table>
<thead>
<tr>
<th>WORKCLASS</th>
<th>QUEUE</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKCLASS 2</td>
<td>credits = 576 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 3</td>
<td>credits = 242 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 4</td>
<td>credits = 2601 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 5</td>
<td>credits = 4000 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 6</td>
<td>credits = -563 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 7</td>
<td>credits = 3962 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 8</td>
<td>credits = 3934 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 9</td>
<td>credits = 2216 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 10</td>
<td>credits = 4000 ms.</td>
<td></td>
</tr>
<tr>
<td>WORKCLASS 11</td>
<td>credits = 4000 ms.</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{Used to give out correct time to work classes} \]
### Traffic Control Meters

**Work Class Meters**

Display the various workclass parameters, and show what resources each workclass is consuming.

<table>
<thead>
<tr>
<th>WC</th>
<th>%GUAR</th>
<th>%MAX</th>
<th>TCP</th>
<th>V/ELIG</th>
<th>PW</th>
<th>IRESP</th>
<th>IQUANT</th>
<th>RESP QUANT</th>
<th>P</th>
<th>M</th>
<th>R</th>
<th>LCG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>3</td>
<td>0.12</td>
<td>3</td>
<td>0.26</td>
<td>2.10</td>
<td>0.26</td>
<td>2.10</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>Init</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>0.09</td>
<td>1</td>
<td>0.25</td>
<td>0.75</td>
<td>0.50</td>
<td>1.00</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>RTime</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7</td>
<td>0.44</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.09</td>
<td>1</td>
<td>P</td>
<td>Q</td>
<td>SysAdm</td>
<td>DPR</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>32</td>
<td>0.49</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0.75</td>
<td>P</td>
<td>Q</td>
<td>SysProg</td>
<td>SysDev</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>9</td>
<td>0.26</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.50</td>
<td>1.00</td>
<td>P</td>
<td>Q</td>
<td>SEngr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>20</td>
<td>0.46</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.09</td>
<td>1</td>
<td>P</td>
<td>Q</td>
<td>HEngr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>16</td>
<td>0.25</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0.75</td>
<td>P</td>
<td>Q</td>
<td>MktUS</td>
<td>MktFor</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>4</td>
<td>0.36</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.50</td>
<td>1.00</td>
<td>P</td>
<td>Q</td>
<td>DS-CC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>6</td>
<td>0.18</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.09</td>
<td>1</td>
<td>P</td>
<td>Q</td>
<td>OffAuto</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>4</td>
<td>0.62</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0.75</td>
<td>P</td>
<td>Q</td>
<td>Misc</td>
<td>Mfg</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>3</td>
<td>0.55</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.50</td>
<td>1.00</td>
<td>P</td>
<td>Q</td>
<td>Other</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>4</td>
<td>0.16</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.09</td>
<td>1</td>
<td>P</td>
<td>Q</td>
<td>Special</td>
<td></td>
</tr>
</tbody>
</table>

**TCP** percent (%GUAR) control non-realtime work classes.

Not to be reproduced 9-23
**TRAFFIC CONTROL METERS**

RESPONSE METERS DISPLAY RESPONSE TIME, BASED ON TERMINAL INTERACTIONS, ON A PER-WORKCLASS BASIS

<table>
<thead>
<tr>
<th>WC Times/---</th>
<th>---</th>
<th>Response Times by VCPU Range</th>
<th>Load Control Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>Avg Time Sent</td>
<td>VCPU Range</td>
<td># Avg</td>
</tr>
<tr>
<td>0</td>
<td>86 2.70</td>
<td>0.00 0.50</td>
<td>113 0.04 0.42</td>
</tr>
<tr>
<td>92 0.15</td>
<td>0.50 1.00</td>
<td>3 0.55 5.51</td>
<td>10.00</td>
</tr>
<tr>
<td>1.00 10.00</td>
<td>3 2.43 14.96</td>
<td>6.15</td>
<td></td>
</tr>
<tr>
<td>119 0.12 0.91</td>
<td>7.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>35 11.78</td>
<td>0.00 0.50</td>
<td>34 0.11 0.96</td>
</tr>
<tr>
<td>39 0.21</td>
<td>0.50 1.00</td>
<td>2 0.83 4.06</td>
<td>4.87</td>
</tr>
<tr>
<td>2</td>
<td>593 14.90</td>
<td>0.00 0.50</td>
<td>620 0.05 0.45</td>
</tr>
<tr>
<td>612 0.15</td>
<td>0.50 1.00</td>
<td>28 0.71 3.61</td>
<td>5.10</td>
</tr>
<tr>
<td>1.00 10.00</td>
<td>39 1.77 8.39</td>
<td>4.74</td>
<td></td>
</tr>
<tr>
<td>637 0.17 1.07</td>
<td>6.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2456 6.38</td>
<td>0.00 0.50</td>
<td>2993 0.06 0.66</td>
</tr>
<tr>
<td>2622 0.17</td>
<td>0.50 1.00</td>
<td>117 0.68 4.22</td>
<td>6.16</td>
</tr>
<tr>
<td>1.00 10.00</td>
<td>66 2.46 13.47</td>
<td>5.49</td>
<td></td>
</tr>
<tr>
<td>10.00 99.99</td>
<td>10 56.85 99.99</td>
<td>3.33</td>
<td></td>
</tr>
<tr>
<td>3186 0.33 1.65</td>
<td>4.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>581 15.82</td>
<td>0.00 0.50</td>
<td>663 0.05 0.71</td>
</tr>
<tr>
<td>530 0.21</td>
<td>0.50 1.00</td>
<td>13 0.64 4.64</td>
<td>7.30</td>
</tr>
<tr>
<td>1.00 10.00</td>
<td>8 3.47 32.07</td>
<td>5.22</td>
<td></td>
</tr>
<tr>
<td>684 0.11 1.15</td>
<td>10.88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>133 29.51</td>
<td>0.00 0.50</td>
<td>148 0.06 0.83</td>
</tr>
<tr>
<td>141 0.17</td>
<td>0.50 1.00</td>
<td>3 0.69 4.64</td>
<td>6.70</td>
</tr>
<tr>
<td>1.00 10.00</td>
<td>5 3.15 10.24</td>
<td>3.25</td>
<td></td>
</tr>
<tr>
<td>10.00 99.99</td>
<td>2 26.69 48.84</td>
<td>1.83</td>
<td></td>
</tr>
<tr>
<td>158 0.51 1.81</td>
<td>3.55</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**First Place**

To be reproduced.

9-24 F804

---

Not To Be Reproduced
### Traffic Control Meters

#### Response Meters

<table>
<thead>
<tr>
<th></th>
<th>1180</th>
<th>11.65</th>
<th>0.00</th>
<th>0.50</th>
<th>577</th>
<th>0.05</th>
<th>1.13</th>
<th>20.91</th>
<th>MktUS MktFor MktEd</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1211</td>
<td>0.58</td>
<td>0.50</td>
<td>1.00</td>
<td>24</td>
<td>0.64</td>
<td>4.95</td>
<td>7.74</td>
<td>10.00 11.88 6.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td>10.00</td>
<td>16</td>
<td>1.89</td>
<td>11.88</td>
<td>6.29</td>
<td>10.00 11.88 6.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1017</td>
<td>0.10</td>
<td>1.39</td>
<td>14.38</td>
<td>10.00 11.88 6.29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>259</th>
<th>14.65</th>
<th>0.00</th>
<th>0.50</th>
<th>292</th>
<th>0.08</th>
<th>1.53</th>
<th>20.17</th>
<th>DS-CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>287</td>
<td>0.98</td>
<td>0.50</td>
<td>1.00</td>
<td>9</td>
<td>0.71</td>
<td>9.03</td>
<td>12.66</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td>10.00</td>
<td>17</td>
<td>2.04</td>
<td>14.76</td>
<td>7.24</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>319</td>
<td>0.24</td>
<td>2.81</td>
<td>11.86</td>
<td>10.00 99.99 9.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>73</th>
<th>2.69</th>
<th>0.00</th>
<th>0.50</th>
<th>79</th>
<th>0.06</th>
<th>0.31</th>
<th>5.45</th>
<th>OffAuto</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>74</td>
<td>0.05</td>
<td>0.50</td>
<td>1.00</td>
<td>2</td>
<td>0.60</td>
<td>6.40</td>
<td>10.62</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td>10.00</td>
<td>1</td>
<td>3.20</td>
<td>6.73</td>
<td>2.10</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>82</td>
<td>0.11</td>
<td>0.54</td>
<td>4.94</td>
<td>10.00 99.99 9.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>94</th>
<th>41.91</th>
<th>0.00</th>
<th>0.50</th>
<th>80</th>
<th>0.11</th>
<th>1.23</th>
<th>11.07</th>
<th>Misc Mfg</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>96</td>
<td>0.30</td>
<td>0.50</td>
<td>1.00</td>
<td>11</td>
<td>0.69</td>
<td>4.14</td>
<td>6.91</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td>10.00</td>
<td>13</td>
<td>2.99</td>
<td>12.62</td>
<td>4.30</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>104</td>
<td>0.53</td>
<td>3.05</td>
<td>5.74</td>
<td>10.00 99.99 9.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>99.99</th>
<th>0.00</th>
<th>0.50</th>
<th>10</th>
<th>0.11</th>
<th>1.45</th>
<th>13.82</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>8</td>
<td>0.21</td>
<td>1.00</td>
<td>10.00</td>
<td>1</td>
<td>3.88</td>
<td>39.11</td>
<td>10.09</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>0.45</td>
<td>4.88</td>
<td>10.89</td>
<td>10.00 99.99 9.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>417</th>
<th>12.71</th>
<th>0.00</th>
<th>0.50</th>
<th>445</th>
<th>0.03</th>
<th>0.50</th>
<th>15.32</th>
<th>Special</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>420</td>
<td>0.16</td>
<td>0.50</td>
<td>1.00</td>
<td>5</td>
<td>0.57</td>
<td>3.61</td>
<td>6.29</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td>10.00</td>
<td>4</td>
<td>1.65</td>
<td>11.40</td>
<td>6.90</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>454</td>
<td>0.05</td>
<td>0.63</td>
<td>11.93</td>
<td>10.00 99.99 9.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>5554</th>
<th>11.13</th>
<th>0.00</th>
<th>0.50</th>
<th>6454</th>
<th>0.07</th>
<th>0.77</th>
<th>11.40</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>6192</td>
<td>0.29</td>
<td>0.50</td>
<td>1.00</td>
<td>217</td>
<td>0.68</td>
<td>4.54</td>
<td>6.70</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td>10.00</td>
<td>173</td>
<td>2.31</td>
<td>13.11</td>
<td>5.68</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>48.77</td>
<td>99.99</td>
<td>3.32</td>
<td>10.00 99.99 9.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6557</td>
<td>0.24</td>
<td>1.51</td>
<td>6.39</td>
<td>10.00 99.99 9.45</td>
</tr>
</tbody>
</table>

86797 calls to meter_response_time 283 invalid transitions.
Overhead = 0.09% (0.052 ms./call)
TRAFFIC CONTROL METERS

post purge meters

POST_PURGE_METERS - DISPLAY THE STATE OF POST-PURGE ACTIVITY.

CONSIDERABLY MORE DETAILED METERS ARE KEPT BY RING ZERO, BUT NOT REPORTED AND NOT PARTICULARLY INTERESTING.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total metering time</td>
<td>0:20:29</td>
</tr>
<tr>
<td>Post purge time</td>
<td>1.41 msec. (0.36% of system)</td>
</tr>
<tr>
<td>Ave list size</td>
<td>17.76 entries</td>
</tr>
<tr>
<td>Ave working set</td>
<td>5.59 pages</td>
</tr>
<tr>
<td>Working set factor</td>
<td>0.50</td>
</tr>
<tr>
<td>Working set addend</td>
<td>0</td>
</tr>
<tr>
<td>Thrashing percentage</td>
<td>1.34 %</td>
</tr>
<tr>
<td>Ave post in core</td>
<td>10.65 (59.94 %)</td>
</tr>
</tbody>
</table>
TRAFFIC CONTROL COMMANDS

print tuning parameters also CTP

PRINT TUNING_PARAMETERS - PRINT VALUES FOR SYSTEM CONTROL PARAMETERS. MOST CONTROL THE SCHEDULER

Current system tuning parameters:

tefirst 0.5 seconds
test 1. seconds	timax 8. seconds
priority_sched_inc 80. seconds
min_eligible 2.
max_eligible 20.
max_batch_elig 0
working_set_factor 0
working_set_addend 0
deadline_mode 0.04 seconds
int_q_enabled 2. seconds to get logged w out
post_purge 0.25
pre_emt_sample_time 4. seconds
on_notify 0.005 seconds
np_at_notify 30. seconds
np_at_pltnotify 724
process_initial_quantum 0
quit_priority 0
realtime_io_priority 0
realtime_io_deadline 0
realtime_io_quantum 0
notify_timeout_interval 0
notify_timeout_severity 0
write_limit 0

These are "INTERNAL", normally never changed, and only printed if the -all control argument is given

stack truncation on
stack truncation always off
stk_trunc_block_avg_factor 0.25
trap_invalid_masked off
meter_irl locking off
checksum_filemap on
TRAFFIC CONTROL COMMANDS

print apt entry display.txt

PRINT_APT_ENTRY - INTERPRETS AND DUMPS AN APT ENTRY

! print_apt_entry Sibert -dump

Sibert.Multics.a b.126 at 10300 in tc data, >pdd>!BblCpbBbBBBBBB
PID:010300356001 TRM:000447007410 407777000460
Running for 0.067825 (since 01:28:53).
Usage: cpu 8:40.8; vcpu 6:04.6; pf 23622.
te/s/i/x: 0.411 0.000 0.647 32.000.
Flags: loaded,eligible,mbz11,dbr_loaded.
Alarm in 31.244 (at 01:29:24).

0 005400003000 014225000001 000000056106 010300356001
4 000001443141 000000000000 000002356631 000172044000
10 000000000000 000000000000 005200013740 000000002076
14 00300777777 115641364232 000000000000 003702747134
20 000000000000 001360000002 0000000111567 4575723766
24 000000000000 000000000000 011700111567 457761705103
30 000447007410 407777000460 000000000000 000001720440
34 000000000000 000111567 457736532472 003320000000 00000000000
40 000000000000 00000325430 000000000000 000000000000
44 035117540004 001775100023 000000000000 002556711030
50 000000000000 000000000000 000000000004 000000004441
54 000000000000 000111567 457572506651 000000000000 0142740070015
60 000000000000 002557051053 000001720440 776000000000
64 000000000000 000000000000 000000000000 000000000000
70 000000000000 000000000000 000000000000 000000000000
74 000000000000 000000000000 000000000000 000000000000
## TOPIC X

### Fault and Interrupt Handling

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault and Interrupt Handling Overview</td>
<td>10-1</td>
</tr>
<tr>
<td>Fault and Interrupt Data Bases</td>
<td>10-4</td>
</tr>
<tr>
<td>Fault and Interrupt Vectors</td>
<td>10-4</td>
</tr>
<tr>
<td>Fault Data Save Areas</td>
<td>10-6</td>
</tr>
<tr>
<td>Important Types of Faults</td>
<td>10-7</td>
</tr>
<tr>
<td>Fault/Interrupt Meters</td>
<td>10-12</td>
</tr>
<tr>
<td>fim_meters</td>
<td>10-12</td>
</tr>
<tr>
<td>interrupt_meters</td>
<td>10-13</td>
</tr>
</tbody>
</table>
FAULT AND INTERRUPT HANDLING OVERVIEW

FUNCTION

- RESPONSIBLE FOR HANDLING ALL EXCEPTIONS IN A CPU WHETHER INTERNAL TO THE PROCESSOR (REFERRED TO AS FAULTS) OR EXTERNAL (REFERRED TO AS INTERRUPTS)

- ESTABLISHES THE SUPERVISOR ENVIRONMENT AT FAULT AND INTERRUPT TIME, SAVES THE MACHINE CONDITIONS AND TRANSFERS TO THE APPROPRIATE HANDLER

- MAJOR COMPONENTS: THE FAULT INTERCEPT MODULE (fim), WIRED-FAULT INTERCEPT MODULE (wired_fim), I/O INTERRUPT HANDLER (io_interrupt), sys_trouble, page_fault

MAJOR DATA BASES

- INTERRUPT VECTORS - ONE SET PER SYSTEM (WIRED)

- INTERRUPT PAIR (2 INSTRUCTIONS) - ONE PAIR PER DEFINED INTERRUPT TYPE

- LOCATED AT ABSOLUTE ADDRESS 0. A HARDWARE RECOGNIZED DATA BASE

- DESCRIBE WHERE TO SAVE THE CONTEXT, AND WHERE TO TRANSFER TO PROCESS THE INTERRUPT (ALWAYS iointerrupt)
FAULT AND INTERRUPT HANDLING OVERVIEW

FAULT VECTORS - ONE SET PER SYSTEM (WIRED)

VECTOR PAIR (2 INSTRUCTIONS) - ONE PAIR PER DEFINED FAULT TYPE

LOCATED AT ABSOLUTE ADDRESS 100 (OCTAL) IMMEDIATELY ABOVE THE INTERRUPT VECTORS. A HARDWARE RECOGNIZED DATA BASE

DESCRIBE WHERE TO SAVE THE CONTEXT, AND WHERE TO TRANSFER TO PROCESS THE FAULT (fim, wired_fim, page_fault)

PROCESS DATA SEGMENT (PDS) - ONE PER PROCESS (WIRED WHEN ELIGIBLE)

CONTAINS PROCESS RELEVANT INFO SUCH AS PROCESS ID, USER ID, HOME/WORKING/PROCESS DIRECTORIES, AIM CLASSIFICATION, INITIAL RING, ETC

CONTAINS ALL INFORMATION ABOUT THE PROCESS NEEDED BY THE SUPERVISOR CODE WHEN THE PROCESS IS RUNNING

CONTAINS SAVE AREAS FOR CONTEXT INFORMATION ABOUT FAULTS WHICH CAN RESULT IN GIVING UP THE PROCESSOR: PAGE FAULTS, SEGMENT FAULTS, AND ALL FAULTS NOT HANDLED BY THE SUPERVISOR

PROCESSOR DATA SEGMENT (PRDS) - ONE PER CONFIGURED CPU (WIRED)

SERVES AS RING-ZERO STACK FOR PAGE CONTROL AND TRAFFIC CONTROL

ALSO CONTAINS SAVE AREAS FOR CONTEXT INFORMATION ABOUT FAULTS WHICH USUALLY DO NOT MEAN GIVING UP THE PROCESSOR: CONNECT FAULTS AND INTERRUPTS.
FAULT AND INTERRUPT HANDLING OVERVIEW

FIM_TABLE

A TABLE IN THE FIM PROGRAM WHICH DESCRIBES THE ACTION TO BE TAKEN FOR VARIOUS TYPES OF FAULTS
FAULT AND INTERRUPT DATA BASES

FAULT AND INTERRUPT VECTORS

ALL FAULTS AND INTERRUPTS ARE HANDLED IN A CENTRALIZED FASHION

FOR EACH FAULT OR INTERRUPT, THERE ARE TWO INSTRUCTIONS:

1. AN scu INSTRUCTION, TO STORE THE ABSOLUTELY ESSENTIAL DATA NEEDED TO RESTART FROM THE FAULT
2. A tra INSTRUCTION, TO TRANSFER TO THE APPROPRIATE FAULT HANDLER

ASSOCIATED WITH EACH OF THESE INSTRUCTIONS, THERE IS A POINTER

1. AN scu DATA POINTER, POINTING TO ONE OF SIX REGIONS WHERE FAULT DATA GOES
2. A FAULT HANDLER POINTER, INDICATING ONE OF THE PROCEDURES USED TO HANDLE FAULTS

THESE INSTRUCTIONS AND POINTERS ARE STORED IN THE fault_vector

THE DATA STORED BY scu IS ONLY THE "CONTROL UNIT DATA" - EACH HANDLER MUST IMMEDIATELY SAVE THE PROGRAM VISIBLE REGISTERS, THE EIS POINTER & LENGTH DATA, AND SO FORTH. THIS DATA IS ALWAYS STORED IN THE SAME FORMAT, THE MACHINE CONDITIONS STRUCTURE (mc.incl.pll)
FAULT AND INTERRUPT DATA BASES

FAULT AND INTERRUPT VECTORS

A HARDWARE RECOGNIZED DATA BASE LOCATED
AT ABSOLUTE ADDRESS 0-577

Not To Be Reproduced
FAULT AND INTERRUPT DATA BASES

FAULT DATA SAVE AREAS

There are six regions where machine conditions are stored.

They are selected to minimize the number of times when fault data must be moved. Usually, once fault data has been stored in a particular place, it can be restored directly from there, but sometimes it must be moved to another place.

- prds$interrupt_data - used for interrupts, only
- prds$fim_data - used for faults such as connect faults, which will be handled entirely using the wired ring zero stack (PRDS)
- prds$sys_trouble_data - used for the fault that crashed the system. No machine conditions are ever stored here directly, only moved here.
- prds$fim_data - used for faults which will be handled in ring zero, using the ring zero stack, where page faults might be taken while the other fault is being handled.
- prds$page_fault_data - used for page faults and timer runouts (which indicate the end of a quantum) - both are events which almost always result in giving up the processor, but which are handled on the wired ring zero stack (PRDS)
- prds$signal_data - used for faults which will be signalled out for the user ring to handle. If an error occurs processing some fault in ring zero, its' fault data is moved here before signalling.

There are also special stack frames created by the FIM used for fault signalling, and the machine conditions are copied there.
FAULT AND INTERRUPT DATA BASES

IMPORTANT TYPES OF FAULTS

CERTAIN FAULTS DESERVE SPECIAL DISCUSSION, AS THEY ARE USED TO IMPLEMENT IMPORTANT SUPERVISOR SERVICES

LINKAGE FAULT

OCCUR WHEN A POINTER CONTAINING 46 OCTAL IN THE LOW SIX BITS OF THE FIRST WORD IS USED

USED TO IMPLEMENT DYNAMIC LINKING (SEE NAME/ADDRESS SPACE MANAGEMENT, TOPIC 4)

USES pds$fim_data, IS HANDLED BY fim alm, WHICH INVOKES link_man.pl1, HANDLED ENTIRELY ON THE stack_0

SEGMENT FAULT

OCCURS WHEN AN NON-ACTIVE SEGMENT IS REFERENCED (SEE SEGMENT CONTROL, TOPIC 7)

USES pds$fim_data, IS HANDLED BY fim alm, WHICH INVOKES seg_fault.pl1, HANDLED ENTIRELY ON THE stack_0

A SEGMENT FAULT MAY OCCUR WHILE ANOTHER IS BEING HANDLED, AND IT WILL BE HANDLED RECURSIVELY

PAGE FAULT

OCCURS WHEN A PAGE NOT IN MEMORY IS REFERENCED (SEE PAGE CONTROL, TOPIC 8)
USES `pds$page_fault_data`, IS HANDLED DIRECTLY BY `page_fault.alm`, AND IS HANDLED ENTIRELY ON THE PRDS.

TIMER RUNOUT OCCURS WHEN THE TIMER REGISTER IS DECREMENTED THROUGH ZERO, INDICATING THAT THE RUNNING PROCESS HAS NOW OVERSTAYED ITS WELCOME, AND SHOULD LOSE ELIGIBILITY.

TIMER RUNOUT FAULTS ARE ALSO USED INTERNAL TO TRAFFIC CONTROL TO IMPLEMENT "PRE-EMPT" SAMPLING; IN THIS MODE, NOW THE DEFAULT, THE TIMER GOES OFF EVERY FEW MILLISECONDS AND THE PROCESS CHECKS TO SEE WHETHER A HIGHER PRIORITY PROCESS WANTS THE PROCESSOR; THIS DOES NOT MAKE THE RUNNING PROCESS INELIGIBLE, HOWEVER, UNLESS ITS QUANTUM HAS ALSO RUN OUT.

A PROCESS RUNNING IN RING ZERO NEED NEVER GIVE UP ELIGIBILITY - WHEN A TIMER RUNOUT HAPPENS, IT REMEMBERS, AND SETS THE RING ALARM REGISTER, WHICH WILL CAUSE A RING ALARM FAULT LATER ON WHEN IT LEAVES RING ZERO.

USES `pds$page_fault_data`, IS HANDLED DIRECTLY BY `pxss.alm`, AND IS HANDLED ENTIRELY ON THE PRDS.

RING ALARM FAULT

OCCURS WHEN A PROCESS RETURNS TO AN OUTER RING FROM AN INNER RING, AND THE RING ALARM REGISTER HAS BEEN SET.

(1) USED TO DEFER ACTION ON TIMER RUNOUTS AND CONNECT FAULTS (SEE BELOW) UNTIL A PROCESS LEAVES RING ZERO.

(2) USED TO ENSURE THAT THE SOFTWARE VALIDATION LEVEL (SET WITH `cu$.level_set`) IS NEVER LOWER THAN THE RING OF EXECUTION - SETTING THE VALIDATION LEVEL ALSO SETS THE RING ALARM REGISTER, AND THE RING ALARM FAULT CAUSES THE VALIDATION LEVEL TO BE RESET.
FAULT AND INTERRUPT DATA BASES

IMPORTANT TYPES OF FAULTS

USES pds$page_fault_data, IS HANDLED DIRECTLY BY ring_alarmealm AND IS HANDLED ENTIRELY ON THE PRDS.

IS ACTUALLY A SUBTYPE OF ACCESS VIOLATION

CONNECT FAULT

**Communication between CPUs usually to clear associative memory when PWS are changed.**

OCCURS WHEN ONE PROCESSOR SENDS A "connect" TO ANOTHER, USING A cioc INSTRUCTION

RESEMBLES A SOFTWARE SENDABLE INTERRUPT; ALTHOUGH PROCESSORS CAN SEND INTERRUPTS TO EACH OTHER, LIMITATIONS OF THE HARDWARE MAKE CONNECT FAULTS EASIER TO USE

USED FOR ALL INTERPROCESSOR SIGNALLING -

(1) TO CAUSE ANOTHER PROCESSOR TO SELECTIVELY CLEAR ITS CACHE OR ASSOCIATIVE MEMORY

(2) TO PRE-EMPT A PROCESS RUNNING ON ANOTHER CPU (WHEN PRE-EMPT SAMPLING IS NOT IN USE)

(3) TO INFORM ANOTHER PROCESSOR THAT THE SYSTEM IS CRASHING

(4) TO INFORM ANOTHER PROCESSOR THAT IT IS BEING REMOVED FROM THE CONFIGURATION

USES prds$fim_data, IS SOMETIMES HANDLED BY wired_fim.alm, AND IS HANDLED ENTIRELY ON THE PRDS.

FOR TYPE 1 CONNECTS (CACHE CLEAR), THE FAULT IS HANDLED VERY SPECIALY BY CODE WHICH IS ACTUALLY EXECUTED FROM WITHIN THE PRDS (SEE fast_connect_init.alm), AND RUNS SOMEWHAT FASTER. THE COMPLICATED CASES ARE LEFT TO WIRED_FIM

CELLS IN THE Scs ARE USED TO DISTINGUISH BETWEEN THE DIFFERENT TYPES OF CONNECT FAULTS; A PROCESSOR SETS THE APPROPRIATE CELLS BEFORE SENDING THE CONNECT.
OTHER FAULTS

- PARITY - Run automatic parity error logging and diagnosis (to chip level, for cache) routines

- O.P. not complete, command, shutdown, startup, store, trouble - run hardware error logging routines

- Overflow, underflow - can be set up to automatically set a specified (very large or very small) value and restart without interrupting the running program

- Derail - used when crashing the system or voluntarily returning to BOS to begin execution in BOS. An ordinary signallable fault at all other times (and used that way by the gtss emulator). Some systems used to have supervisor do something.

- Execute - used to force a system crash

- Access violation - can automatically log access violations for security audits

- Others - handled by fim.aim, which maps the hardware faults onto the Multics environment condition names.

Considerable interpretation is sometimes required; for instance, a null pointer is actually signalled by the hardware as an access violation, out of bounds on DSEG, fault

INTERRUPT

Interrupts are used to announce the completion of all I/O operations, and also (rarely) used during dynamic reconfiguration to start processors
FAULT AND INTERRUPT DATA BASES

IMPORTANT TYPES OF FAULTS

USES prds$interrupt_data, IS HANDLED DIRECTLY BY
iom_interrupt.alm OR init_processor.alm, AND IS HANDLED
ENTIRELY ON THE PRDS.
**FAULT/INTERRUPT METERS**

`fim meters`

**FIM_METERS - COUNTS FOR ALL FAULT PROCESSING**

```
Total metering time: 0:20:28

<table>
<thead>
<tr>
<th>fault type</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>shutdown</td>
<td>0</td>
</tr>
<tr>
<td>store</td>
<td>0</td>
</tr>
<tr>
<td>mmel</td>
<td>0</td>
</tr>
<tr>
<td>fault_tag_1</td>
<td>2</td>
</tr>
<tr>
<td>timer_runout</td>
<td>12907</td>
</tr>
<tr>
<td>command</td>
<td>5</td>
</tr>
<tr>
<td>derail</td>
<td>0</td>
</tr>
<tr>
<td>lockup</td>
<td>0</td>
</tr>
<tr>
<td>connect</td>
<td>742501</td>
</tr>
<tr>
<td>parity</td>
<td>0</td>
</tr>
<tr>
<td>illegal_procedure</td>
<td>0</td>
</tr>
<tr>
<td>op_not_complete</td>
<td>0</td>
</tr>
<tr>
<td>startup</td>
<td>0</td>
</tr>
<tr>
<td>overflow</td>
<td>0</td>
</tr>
<tr>
<td>divide_check</td>
<td>0</td>
</tr>
<tr>
<td>execute</td>
<td>0</td>
</tr>
<tr>
<td>segment_fault</td>
<td>6697</td>
</tr>
<tr>
<td>page_fault</td>
<td>151772</td>
</tr>
<tr>
<td>directed_fault_2</td>
<td>0</td>
</tr>
<tr>
<td>directed_fault_3</td>
<td>0</td>
</tr>
<tr>
<td>access_violation</td>
<td>21120</td>
</tr>
<tr>
<td>mme2</td>
<td>0</td>
</tr>
<tr>
<td>mme3</td>
<td>0</td>
</tr>
<tr>
<td>mme4</td>
<td>0</td>
</tr>
<tr>
<td>linkage_fault</td>
<td>21058</td>
</tr>
<tr>
<td>fault_tag_3</td>
<td>0</td>
</tr>
<tr>
<td>trouble</td>
<td>0</td>
</tr>
</tbody>
</table>
```

1 CPU uses connect faults in lightly loaded use idle processes.
## FAULT/INTERRUPT METERS

### interrupt meters

### INTERRUPT_METERS - COUNTS & TIMING FOR ALL I/O INTERRUPTS

Total metering time 0:20:28

<table>
<thead>
<tr>
<th>IOM Ch</th>
<th>Int</th>
<th>Avg Time</th>
<th>% CPU</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 6</td>
<td>11</td>
<td>2.043</td>
<td>0.00</td>
<td>IOM A special</td>
</tr>
<tr>
<td>A 10</td>
<td>379</td>
<td>1.553</td>
<td>0.01</td>
<td>prtb</td>
</tr>
<tr>
<td>A 13</td>
<td>9253</td>
<td>3.436</td>
<td>0.65</td>
<td>fnp c</td>
</tr>
<tr>
<td>A 14</td>
<td>19</td>
<td>0.788</td>
<td>0.00</td>
<td>opc</td>
</tr>
<tr>
<td>A 16</td>
<td>26248</td>
<td>1.347</td>
<td>0.72</td>
<td>tapa</td>
</tr>
<tr>
<td>A 17</td>
<td>128</td>
<td>1.328</td>
<td>0.00</td>
<td>tapa</td>
</tr>
<tr>
<td>A 18</td>
<td>13097</td>
<td>5.383</td>
<td>1.43</td>
<td>fnp b</td>
</tr>
<tr>
<td>A 20</td>
<td>21041</td>
<td>0.860</td>
<td>0.37</td>
<td>dska</td>
</tr>
<tr>
<td>A 21</td>
<td>2881</td>
<td>0.860</td>
<td>0.05</td>
<td>dska</td>
</tr>
<tr>
<td>A 22</td>
<td>8486</td>
<td>0.877</td>
<td>0.15</td>
<td>dskb</td>
</tr>
<tr>
<td>A 23</td>
<td>257</td>
<td>0.964</td>
<td>0.01</td>
<td>dskb</td>
</tr>
<tr>
<td>A 24</td>
<td>17896</td>
<td>0.895</td>
<td>0.33</td>
<td>dskb</td>
</tr>
<tr>
<td>A 25</td>
<td>2573</td>
<td>0.943</td>
<td>0.05</td>
<td>dskb</td>
</tr>
<tr>
<td>A 26</td>
<td>10486</td>
<td>0.857</td>
<td>0.18</td>
<td>dska</td>
</tr>
<tr>
<td>A 27</td>
<td>255</td>
<td>0.899</td>
<td>0.00</td>
<td>dska</td>
</tr>
<tr>
<td>A 28</td>
<td>24717</td>
<td>0.845</td>
<td>0.43</td>
<td>dska</td>
</tr>
<tr>
<td>A 29</td>
<td>2319</td>
<td>0.849</td>
<td>0.04</td>
<td>dska</td>
</tr>
<tr>
<td>A 30</td>
<td>9915</td>
<td>0.845</td>
<td>0.17</td>
<td>dska</td>
</tr>
<tr>
<td>A 31</td>
<td>215</td>
<td>0.767</td>
<td>0.00</td>
<td>dske</td>
</tr>
<tr>
<td>B 14</td>
<td>236</td>
<td>0.877</td>
<td>0.00</td>
<td>fnp d</td>
</tr>
<tr>
<td>B 15</td>
<td>6917</td>
<td>3.351</td>
<td>0.47</td>
<td>fnp f</td>
</tr>
<tr>
<td>B 18</td>
<td>3547</td>
<td>3.543</td>
<td>0.26</td>
<td>fnp a</td>
</tr>
<tr>
<td>B 19</td>
<td>13667</td>
<td>3.435</td>
<td>0.96</td>
<td>fnp e</td>
</tr>
<tr>
<td>B 20</td>
<td>22013</td>
<td>0.881</td>
<td>0.39</td>
<td>dska</td>
</tr>
<tr>
<td>B 21</td>
<td>4745</td>
<td>0.863</td>
<td>0.08</td>
<td>dska</td>
</tr>
<tr>
<td>B 22</td>
<td>6054</td>
<td>0.861</td>
<td>0.11</td>
<td>dska</td>
</tr>
<tr>
<td>B 23</td>
<td>60</td>
<td>0.925</td>
<td>0.00</td>
<td>dska</td>
</tr>
<tr>
<td>B 24</td>
<td>15946</td>
<td>0.867</td>
<td>0.28</td>
<td>dska</td>
</tr>
<tr>
<td>B 25</td>
<td>989</td>
<td>0.841</td>
<td>0.02</td>
<td>dska</td>
</tr>
<tr>
<td>B 26</td>
<td>12757</td>
<td>0.863</td>
<td>0.22</td>
<td>dska</td>
</tr>
<tr>
<td>B 27</td>
<td>1020</td>
<td>0.903</td>
<td>0.02</td>
<td>dska</td>
</tr>
<tr>
<td>B 28</td>
<td>5178</td>
<td>0.838</td>
<td>0.09</td>
<td>dska</td>
</tr>
<tr>
<td>B 29</td>
<td>30</td>
<td>0.653</td>
<td>0.00</td>
<td>dska</td>
</tr>
<tr>
<td>B 30</td>
<td>16582</td>
<td>0.823</td>
<td>0.28</td>
<td>dska</td>
</tr>
<tr>
<td>B 31</td>
<td>825</td>
<td>0.841</td>
<td>0.01</td>
<td>dska</td>
</tr>
<tr>
<td>B 32</td>
<td>1764</td>
<td>0.825</td>
<td>0.03</td>
<td>dske</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chan</th>
<th>262506</th>
<th>1.464</th>
<th>7.82</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ovhd</td>
<td>258932</td>
<td>0.230</td>
<td>1.21</td>
</tr>
<tr>
<td>Total</td>
<td>258932</td>
<td>1.713</td>
<td>9.03</td>
</tr>
</tbody>
</table>

Not To Be Reproduced 10-13

(End Of Topic)
TOPIC XI
System Initialization/Shutdown

System Initialization Overview ...................................... 11-1
System Initialization Terminology .................................... 11-4
Initialization Data Bases .............................................. 11-7
Environment Passed to Initialization ................................. 11-10
Collection 0 .......................................................... 11-11
Collection 1 ......................................................... 11-12
Collection 2 .......................................................... 11-14
Collection 3 .......................................................... 11-16
Normal Shutdown ..................................................... 11-17
File System Shutdown .................................................. 11-18
Emergency Shutdown .................................................... 11-20
SYSTEM INITIALIZATION OVERVIEW

* FUNCTION

- PREPARE THE SYSTEM TO OPERATE, STARTING FROM A COMpletely EMPTY MACHINE

- READS IN SUPERVISOR PROGRAMS FROM SYSTEM TAPE, SNAPS LINKS BETWEEN SUPERVISOR COMPONENTS, VERIFIES AND INITIALizes HARDWARE CONFIGURATION, SETS UP SYSTEM DATABASES, ACCEPTs STORAGE SYSTEM DISKS AND PREPAREs THEM FOR USE BY THE FILE SYSTEM

- MOST PROGRAMS IN SYSTEM INITIALIZATION ARE DELETED AFTER INITIALIZATION IS COMPLETE.

- SUPERVISOR PROGRAMS ARE LOADED IN THREE "COLLECTIONS", EACH OF WHICH DEPENDS ON THE MECHANISMS SET UP BY THE PREVIOUS ONE

* MAJOR DATABASES

- THESE DATABASES ARE ALL BUILT DURING THE PROCESS OF INITIALIZATION (EXCEPT FOR THE CONFIG DECK) AND KEPT AFTER INITIALIZATION IS FINISHED

- SEGMENT LOADING TABLE (>sll>slt)

- CONTAINS AN ENTRY DESCRIBING THE ATTRIBUTES OF EACH SEGMENT IN THE SUPERVISOR
SYSTEM INITIALIZATION OVERVIEW

NAME TABLE (>sl1>name_table)

CONTAINS A LIST OF NAMES FOR EACH OF THE SEGMENTS IN THE SUPERVISOR

DEFINITIONS SEGMENT (>sl1>definitions_)

CONTAINS THE DEFINITIONS SECTIONS FOR ALL THE SEGMENTS IN THE SUPERVISOR, WHICH ARE USED IN ORDER TO SNAP LINKS BETWEEN THE SUPERVISOR MODULES

CONFIG DECK (>sl1>config_deck)

CONTAINS A DESCRIPTION OF THE HARDWARE CONFIGURATION AND CERTAIN SOFTWARE PARAMETERS

PROVIDED TO SYSTEM INITIALIZATION BY BOS

SHUTDOWN -- TERMINATES THE ACTIVITIES OF THE SYSTEM IN AN ORDERLY FASHION

TWO TYPES OF SHUTDOWN:

NORMAL -- REQUESTED BY THE INITIALIZER, RUNS IN THE USUAL SUPERVISOR ENVIRONMENT

EMERGENCY -- USED AFTER A CRASH, MUST MAKE THE SUPERVISOR ENVIRONMENT OPERABLE BEFORE PROCEEDING
SYSTEM_INITIALIZATION_OVERVIEW

Both types exist primarily to shut down the file system -- that is, to write all data in memory into its proper home on disk.

Includes pages of segments, VTOCES, volume and VTOC maps.

 Shutdown essentially runs the steps of initialization backwards, but with a lot of shortcuts.
SYSTEM INITIALIZATION TERMINOLOGY

CONFIG DECK: A SET OF CARDS OR CARD IMAGES USED TO INFORM THE SOFTWARE ABOUT THE OPERATIONAL READINESS OF THE HARDWARE PRESENT, SWITCH SETTINGS AND SPECIFICATIONS OF SOME SOFTWARE DATA BASES (SIZE, LOCATION, ETC)

BOS: THE BOOTLOAD, OPERATING SYSTEM. A SIMPLE OPERATING SYSTEM (A ONE CPU, UNPAGED ENVIRONMENT) OF 8 SEGMENTS OCCUPYING THE FIRST 16K OF MAIN MEMORY. BOS RESIDES IN THE BOS PARTITION WHEN "MULTICS" IS RUNNING

MST: THE MULTICS SYSTEM TAPE CONTAINS PRECISELY ENOUGH INFORMATION (PROCEDURES AND DATA BASES) TO BRING A BARE HARDWARE SYSTEM TO MULTICS COMMAND LEVEL (ACTUALLY, ONLY ENOUGH OF COMMAND LEVEL TO PERFORM A RELOAD)

BOOT: THE OPERATIONAL PROCEDURE OF READING THE SEGMENTS FROM THE MST AND EXECUTING THE PROCEDURE/SEGMENTS THEREIN

WARM/COLD BOOT: BOOTING THE SYSTEM WITH/WITHOUT A HIERARCHY PRESENT

HARDCORE PARTITION: A RLV PARTITION FOR PAGING HARDCORE SEGMENTS CREATED DURING BOOTLOAD

Not To Be Reproduced 11-4 F80A
SYSTEM INITIALIZATION TERMINOLOGY

DECIDUOUS SEGMENT:
A SEGMENT READ IN AS PART OF THE BOOTLOAD TAPE AND PLACED INTO THE HIERARCHY. DECIDUOUS SEGMENTS ARE PART OF THE INITIALIZER'S HARDCORE ADDRESS SPACE AND RESIDE ENTIRELY IN THE HARDCORE PARTITION. THEY ARE PUT INTO THE HIERARCHY (>s11) IN ORDER TO BE ACCESSIBLE FROM THE USER RINGS

NON DECIDUOUS HARDCORE SEGMENT:
A PAGED HARDCORE SEGMENT NOT IN THE HIERARCHY (AND THUS HAS NO PATHNAME)
<table>
<thead>
<tr>
<th>ALL OTHER SEGMENTS</th>
<th>ORINARY SEGMENTS:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&gt; sss &gt; bound_pl1_</td>
</tr>
<tr>
<td></td>
<td>&gt; udd &gt; MED &gt; Kaiser &gt; Kaiser.mbx</td>
</tr>
<tr>
<td></td>
<td>&gt; udd &gt; Site Sa &gt; Homan &gt; test_pl1</td>
</tr>
<tr>
<td></td>
<td>[pd] &gt; stack_4</td>
</tr>
<tr>
<td></td>
<td>DsRER.PROCESS, per-bootload, ring 0 only)</td>
</tr>
<tr>
<td></td>
<td>&gt; udd &gt; Site Sa</td>
</tr>
<tr>
<td></td>
<td>&gt; udd &gt; MED</td>
</tr>
<tr>
<td></td>
<td>&gt; sss &gt; udd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COLLECTION # 3</th>
<th>NON-SUPERVISOR PROGRAMS:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&gt; sl1 &gt; bound_command_loop_</td>
</tr>
<tr>
<td></td>
<td>&gt; sl1 &gt; bound_rcp_</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MST SEGMENTS</th>
<th>NON-SUPERVISOR AND INIT-SEGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NON-HIERARCHY</td>
</tr>
<tr>
<td></td>
<td>DISK OVERLAY ABS-SEGS:</td>
</tr>
<tr>
<td></td>
<td>volmap_abs_seg</td>
</tr>
<tr>
<td></td>
<td>syserr_log</td>
</tr>
<tr>
<td></td>
<td>NON-DECIDUOUS PAGED:</td>
</tr>
<tr>
<td></td>
<td>bound_file_system</td>
</tr>
<tr>
<td></td>
<td>bound_system_faults</td>
</tr>
<tr>
<td></td>
<td>str_seg</td>
</tr>
<tr>
<td></td>
<td>UNPAGED:</td>
</tr>
<tr>
<td></td>
<td>sst_seg</td>
</tr>
<tr>
<td></td>
<td>tc_data</td>
</tr>
<tr>
<td></td>
<td>pxss</td>
</tr>
<tr>
<td></td>
<td>bound_page_control</td>
</tr>
<tr>
<td></td>
<td>bound_to_wired</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HIERARCHY</th>
<th>SEGMENT CATEGORIES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Not To Be Reproduced
INITIALIZATION DATA BASES

The term initialization refers to the actions required to create the Multics environment given the existence of a configuration deck, and hardware containing no other data than firmware and BOS.

Initialization is accomplished by an orderly loading and processing of the segments residing on the Multics system tape (MST).

The segments of the MST may be divided into three categories:

1. Initialization segments
   - Procedures used only for initialization and subsequently discarded

2. Supervisor segments
   - Data bases used during initialization that ultimately become data bases of Initializer.SysDaemon.z
   - Procedures and data bases that constitute Multics hardcore supervisor in its entirety

3. Non-supervisor segments
   - The segments of collection three are precisely the non-supervisor segments of the MST. These segments are loaded directly into >system_library_1, and are not part of the ring zero supervisor.
BOOTLOAD PROCESSOR

ONE PROCESSOR (THE BOOTLOAD CPU) PERFORMS ALL OF INITIALIZATION RUNNING EXCLUSIVELY IN RING ZERO.

IN THE MOST OF INITIALIZATION (COLLECTION ONE AND MOST OF COLLECTION TWO), THERE ARE NO PROCESSES, AS SUCH. THE ENVIRONMENT WHICH RUNS THERE EVENTUALLY BECOMES THE Initializer.SysDaemon PROCESS.

NOTE: SINCE THE Initializer.SysDaemon DOES NOT LOGIN LIKE OTHER USERS, IT DOES NOT APPEAR IN THE NORMAL USER TABLE, CONSEQUENTLY IS NOT VISIBLE TO THE who COMMAND.

STRATEGY OF INITIALIZATION: BOOTSTRAPPING.

THE FIRST PROCEDURES RUN IN AN ENVIRONMENT DEVOID OF ALL SOFTWARE ASSISTANCE.

EACH NEW MECHANISM (SEGMENTATION, STACKS, SYMBOLIC LINKING, PAGING, ETC) IS MADE OPERATIVE AS SOON AS POSSIBLE TO ENRICH THE ENVIRONMENT IN WHICH FURTHER MECHANISMS ARE MADE OPERATIVE.

MANY MECHANISMS HAVE SUBSYSTEMS THAT CONTROL THEM AND THESE SUBSYSTEMS ARE NORMALLY INITIALIZED BY CALLING A SPECIAL ENTRY POINT IN THE SUBSYSTEM WHICH PERFORM SUCH TASKS AS:

CREATING TABLES WHOSE SIZES ARE DETERMINED BY DATA SPECIFIED IN THE "CONFIG" DECK.

THREADING OF RELEVANT LISTS.
INITIALIZATION DATA BASES

SEGMENTS ON THE MST ARE ARRANGED IN SUCH AN ORDER THAT THE EARLIER SEGMENTS ALLOW AS MANY MECHANISMS AS POSSIBLE TO BE USED IN LOADING AND PROCESSING OF THE LATER SEGMENTS

FOR THIS PURPOSE (AND BECAUSE THE SIZE OF THE MST IS POTENTIALLY LARGER THAN MAIN MEMORY), THE MST IS DIVIDED INTO FOUR PARTS KNOWN AS COLLECTION ZERO, ONE, TWO AND THREE

INITIALIZATION CAN BE VIEWED AS THE LOADING AND PROCESSING OF COLLECTION ONE, COLLECTION TWO, AND COLLECTION THREE, IN TURN

THE ADDRESS SPACE OF INITIALIZATION (MINUS THE INITIALIZATION SEGMENTS) BECOMES THE GLOBAL SUPERVISOR ADDRESS SPACE OF MULTICS

THIS ADDRESS SPACE IS "CLONED" TO BECOME THE INITIAL ADDRESS SPACE OF NEWLY CREATED PROCESSES BY DUPLICATING THE DSEG
ENVIRONMENT PASSED TO INITIALIZATION

THE FIRST SEGMENT OF COLLECTION ONE IS THE BOUND SEGMENT bound_bootload_1

AT THE TIME CONTROL IS TRANSFERRED TO bound_bootload_1, IT IS REQUIRED THAT BOS HAS INITIALIZED MAIN MEMORY AS FOLLOWS:

- THE IOM MAILBOX MUST BE AT LOCATION 1400 AND CONTAIN THE CHANNEL AND DEVICE NUMBER OF THE TAPE DRIVE ON WHICH THE MST IS MOUNTED
- THE CONFIG DECK (AS PRODUCED BY BOS) MUST RESIDE AT LOCATION 14000 THRU 15777
- THE BOS TOEHOOL AND FLAGBOX MUST BE AT LOCATION 10000 THRU 11777
- THIS ONE PAGE CONSTITUTES ALL KNOWLEDGE THAT MULTICS HAS OF BOS

TRANSFERRING CONTROL TO THE START OF THE BOS TOEHOOL WILL CAUSE:

- FIRST 64K OF MAIN MEMORY TO BE FLUSHED OUT TO THE BOS PARTITION
- THE BOS OPERATING SYSTEM TO BE READ INTO THE FIRST 64K OF MAIN MEMORY
- CONTROL GIVEN TO BOS

- THE REMAINDER OF MAIN MEMORY MUST CONTAIN ZEROES
COLLECTION 0

* THIS CONSISTS OF THE SEGMENTS WHICH ARE DEFINED TO BE PRESENT IN THE INITIAL ADDRESS SPACE

THESE SEGMENTS ARE EMPTY, AND OVERLAY SPECIFIC REGIONS OF MAIN MEMORY. THEY ARE DEFINED SO THAT bound_bootload_1 CAN KNOW WHAT SEGMENT NUMBERS TO USE FOR WHAT DATA

THE COLLECTION ZERO SEGMENTS ARE dseg, fault_vector, iom_mailbox, config_deck, dn355_mailbox, bos_toehold, flagbox, slt, and name_table

SEE SECTION 5, NAME / ADDRESS SPACE MANAGEMENT, FOR THEIR DESCRIPTIONS
COLLECTION 1

COLLECTION 1 CONTAINS ALL OF THE PROCEDURES AND DATA BASES NECESSARY TO MAKE PAGING OPERATIVE

BASIC STEPS OF COLLECTION 1 LOADING AND INITIALIZATION:

| bound_bootload_1 GAINS CONTROL FROM BOS, IN ABSOLUTE MODE, AND PERFORMS THE FOLLOWING: |
| LOADS THE REMAINDER OF ITSELF INTO MAIN MEMORY |
| ESTABLISHES INTERIM FAULT AND INTERRUPT VECTORS |
| Initializes the initialization DSEG, and enters appending mode |
| Reads the remainder of Collection 1 into main memory (including a segment named bootstrap2) |
| bound_bootload_1 TRANSFERS TO bootstrap2 |

bootstrap2 PERFORMS THE FOLLOWING:

| Creates a stack frame in the segment "inzr_stk0" |
| Call the appropriate procedures to prelink the segments of Collection 1 |
| Sets up the PL/I environment and calls the first PL/I procedure "initializer" |
initializer (A SUPERVISOR SEGMENT), ACTUALLY CALLS REAL_INITIALIZER TO DO THE REAL WORK. ALL THE REST OF INITIALIZATION TAKES PLACE VIA CALLS IN real_initializer.

FOR DEBUGGING PURPOSES, THERE IS A MECHANISM IN real_initializer WHICH CAN BE USED TO STOP AT ANY OF THOSE CALLS BY SETTING A VALUE IN THE PROCESSOR SWITCHES.

THE MAJOR INITIALIZATIONS PERFORMED IN THE REST OF COLLECTION ONE ARE:

- INITIALIZE THE SCU, CLOCK, AND CPU CONTROL MECHANISMS, CHECKING THE SWITCHES AND THE ADDRESSABILITY OF MEMORY
- INITIALIZE FAULT AND INTERRUPT PROCESSING. INITIALIZE THE CONSOLE AND SYSERR MECHANISMS
- INITIALIZE PRIMITIVE TRAFFIC CONTROL (WAIT FOR SINGLE EVENTS). INITIALIZE THE SST
- INITIALIZE AND CHECK THE DISK CONFIGURATION. AT THIS POINT, IT BECOMES POSSIBLE TO TAKE PAGE FAULTS, AND ALL FURTHER DISK I/O IS DONE BY PAGING
- CHECK THE ROOT VOLUMES SPECIFIED ON THE ROOT CARD, AND INITIALIZE THE HARDCORE PARTITION MECHANISM. THIS ALSO INCLUDES CREATING THE PARTITIONS AND VTOC IN A COLD BOOT
- MAKE SEGMENTS PAGED -- AT THIS POINT, ALL PAGED SUPERVISOR SEGMENTS ARE COPIED INTO THE HARDCORE PARTITION, AND ACCESSED BY THE NORMAL PAGE FAULT MECHANISM. COLLECTION ONE ENDS HERE
COLLECTION 2

* COLLECTION TWO - THE REST OF SUPERVISOR INITIALIZATION

READ IN THE REST OF THE SUPERVISOR SEGMENTS FROM TAPE AND INITIALIZE THE REST OF THE SUPERVISOR DATABASES

WHEN COLLECTION TWO IS FINISHED, THE INITIALIZATION ENVIRONMENT HAS BECOME Initializer PROCESS, AND IT CALLS OUT TO RING ONE TO START UP THE ANSWERING SERVICE

THE FOLLOWING MAJOR STAGES TAKE PLACE IN COLLECTION TWO:

COLLECTION TWO IS READ FROM TAPE (STILL USING A SPECIAL PROCEDURE, tape_reader, WHICH DOUBLE-BUFFERS)

THE AND CONDITION SIGNALLING AND HIGHER LEVEL FAULT MECHANISMS ARE INITIALIZED

HIGH LEVEL FILE SYSTEM MECHANISMS ARE INITIALIZED: THE VTOC MANAGER, VOLUME DUMPER BIT MAP, SCAVENGER, SEGMENT TRAILERS, LOGICAL VOLUME MANAGEMENT; DIRECTORY LockING

TRAFFIC CONTROL IS FURTHER INITIALIZED, AND AN IDLE PROCESS IS CREATED FOR THE RUNNING CPU

SYSERR LOGGING (TO THE SYSERR PARTITION) IS INITIALIZED. ALL SYSERR MESSAGES GENERATED AFTER THIS POINT WILL GO IN THE LOG

THE FILE SYSTEM ON THE RPV IS "ACCEPTED", AND THE ROOT DIRECTORY INSPECTED (OR CREATED, IF THIS IS A COLD BOOT)

AFTER THIS POINT, THE CONTENTS OF THE HARDCORE PARTITIONS ARE FIXED, AND ALL FURTHER RECORD ALLOCATION AND FREEING IS DONE FROM THE PAGING REGIONS OF THE FILE SYSTEM VOLUMES

Not To Be Reproduced 11-14 F80A
DECIDUOUS SEGMENTS ARE SPLICED INTO THE HIERARCHY

STACK SHARING IS INITIALIZED

THE REST OF THE SEGMENTS, COMPRISING COLLECTION THREE, ARE READ, DIRECTLY INTO THEIR PLACES IN THE HIERARCHY (>s11)

THE USER VISIBLE SUPERVISOR I/O MECHANISM (oci) IS INITIALIZED, AND THE SPECIAL SUPERVISOR TAPE READER IS SHUT DOWN

TRAFFIC CONTROL IS FULLY INITIALIZED BY CREATING IDLE PROCESSES FOR THE OTHER CPUs AND STARTING THEM

ALL INITIALIZATION SEGMENTS ARE NOW DISCARDED, AND WHAT HAS BECOME THE Initializer.SysDaemon PROCESS CALLS init_proc AND BEGINS EXECUTION IN RING ONE

AT THIS POINT, SUPERVISOR INITIALIZATION IS FINISHED
COLLECTION THREE IS NOT PROPERLY A "COLLECTION" AT ALL, BUT JUST A TERM USED TO DESCRIBE THE BEGINNING OF Initializer AND ANSWERING SERVICE INITIALIZATION

MANY THINGS HAPPEN BETWEEN FIRST LEAVING RING ZERO AND LOGGING IN THE FIRST USER. MOST HAVE NOTHING TO DO WITH THE FILE SYSTEM. THE FEW MOST INTERESTING ONES ARE:

VOLUME ACCEPTANCE - THE NON-ROOT VOLUMES ARE INSPECTED AND ACCEPTED INTO THE HIERARCHY. THEIR CONFIGURATION CAN BE CHANGED BY USE OF OPERATOR COMMANDS

delete old pdds - THE PROCESS DIRECTORY AND DECIDUOUS SEGMENTS LEFT OVER FROM THE PREVIOUS BOOTLOAD(S) ARE DELETED

THIS MUST BE DONE AFTER THE HIERARCHY IS FULLY AVAILABLE, SINCE NON-RLV VOLUMES MAY HAVE BEEN USED FOR PROCESS DIRS

VOLUME SCAVENGER AND QUOTA SALVAGER PROCESSES MAY BE STARTED AT THIS TIME, IF THERE ARE INCONSISTENCIES IN THE HIERARCHY CAUSED BY PREVIOUS CRASHES
NORMAL SHUTDOWN

When the system is shut down, it must be done in an orderly manner.

This is accomplished, more or less, by running the steps in initialization backwards:

- Crawlouts are disabled. Once shutdown begins, it can't be stopped.
- The Initializer switches to running on the bootload CPU, traffic control is disabled, and the "other CPUs are stopped and deleted.
- Locking is disabled at this point, since there is now only one process and one processor running.
- All the disk drives are exercised, to determine if any are broken and cannot be shut down.
- Any volume scavenges in progress are stopped and abandoned.
- At this point, normal shutdown is ready to shut down the file system, and all the normal mechanisms are assumed to be operating.
- The Initializer switches to inzr_stk0 (where it all started) and calls shutdown_file_system.

Not To Be Reproduced 11-17 P80A
FILE SYSTEM SHUTDOWN

FILE SYSTEM SHUTDOWN CONSISTS OF FORCING ALL DATA OUT OF MEMORY TO ITS HOME ON DISK:

1. ALL PAGES ARE WRITTEN

2. SEGMENTS ARE DEACTIVATED, AND THEIR VTOCES UPDATED

3. VOLUMES ARE DEMOUNTED, AND THEIR LABELS UPDATED

FILE SYSTEM SHUTDOWN TAKES THE FOLLOWING STEPS:

1. ALL MODIFIED PAGES ARE WRITTEN TO DISK. THIS IS DONE SEVERAL TIMES DURING THE COURSE OF SHUTDOWN

2. STACK_0 SEGMENTS ARE DEACTIVATED AND DISCARDED

3. THE deactivate_for_demount PROEDURE IS CALLED TO DEACTIVATE ALL OTHER SEGMENTS—AND UPDATE THEIR VTOCES

   THIS IS DONE BY WALKING THE AST HIERARCHY FROM THE BOTTOM UP, DEACTIVATING A SEGMENT, ITS SIBLINGS, AND ITS PARENTS, ETC.

   THIS IS DONE TO ENSURE CONSISTENT QUOTA VALUES IN VTOCES AFTER SHUTDOWN, BECAUSE QUOTA MUST BE UPDATED FROM THE BOTTOM UP

4. ALL VOLUMES ARE DEMOUNTED, THEIR VOLUME AND VTOC MAPS UPDATED, AND LABELS CHANGED TO INDICATE SUCCESSFUL DEMOUNT

   THE ORDER IS NOT IMPORTANT, EXCEPT THAT THE RPV GOES LAST
FILE SYSTEM SHUTDOWN

MEMORY IS FLUSHED AGAIN

IF ANY DRIVES WERE INOPERATIVE, THIS IS ANNOUNCED, AND THE RPV IS NOT DEMOUNTED.

THIS MAKES IT POSSIBLE TO FIX THE BROKEN DRIVE AND DO AN EMERGENCY SHUTDOWN TO Finish SHUTDOWN.

IF THERE WERE NO PROBLEMS, THE RPV IS DEMOUNTED AND MEMORY IS FLUSHED ONE LAST TIME. ALL RELEVANT INFORMATION IS NOT ON DISK.

AT THE END OF FILE SYSTEM SHUTDOWN, ALL CONSOLE MESSAGES ARE ALLOWED TO COMPLETE, AND THE SYSTEM RETURNS TO BOS.
EMERGENCY SHUTDOWN

- EMERGENCY SHUTDOWN IS DONE AFTER A CRASH OR SHUTDOWN FAILURE

- LIKE NORMAL SHUTDOWN, THE MISSION OF ESD IS TO SHUT DOWN THE FILE SYSTEM, AND IT DOES THIS BY MAKING THE SYSTEM WORK WELL ENOUGH TO CALL shutdown_file_system

- UNLIKE NORMAL SHUTDOWN, IT CANNOT ASSUME THAT NORMAL MECHANISMS ARE OPERATIONAL, AND MUST MAKE THEM WORK FIRST

- AFTER THE SUPERVISOR IS MADE OPERATIONAL, EMERGENCY SHUTDOWN TRANSFERS TO THE NORMAL FILE SYSTEM SHUTDOWN

- THE FOLLOWING STEPS ARE TAKEN TO REANIMATE THE SUPERVISOR:

  - EMERGENCY SHUTDOWN STARTS OUT RUNNING IN ABSOLUTE MODE

  - IT ENTERS APPENDING MODE, FINDS THE PRDS FOR THE PROCESSOR IT IS RUNNING ON, SETS IT UP AS ITS STACK

  - ALL CRITICAL LOCKS (PAGE TABLE, APT) ARE FORCIBLY UNLOCKED. TRAFFIC CONTROL IS DISABLED. THE PROCESSOR RUNNING IS NOW THE ONLY ONE, AND LOCKS ARE UNNECESSARY

  - THE CONSOLE AND SYSERR MECHANISMS ARE RESET

  - SUPERVISOR I/O SUPPORT, IN PARTICULAR THE DISK DIM, IS REINITIALIZED

  - THE STATE OF PAGE TABLES AND THE CORE MAP IS MADE CONSISTENT

  - THIS IS DONE BY pc_recover_sst AND CAN BE DONE ONLY BECAUSE ALL OF PAGE CONTROL IS CODED TO FOLLOW PROTOCOLS ABOUT THE ORDER IN WHICH TO UPDATE RELATED DATA
EMERGENCY SHUTDOWN

BECAUSE OF THIS, IF PAGE CONTROL IS INTERRUPTED AT ANY POINT, IT IS POSSIBLE TO DETERMINE WHAT IT WAS DOING AND COMPLETE THE OPERATION.

THIS STEP IS CRUCIAL TO BEING ABLE TO TAKE PAGE FAULTS LATER IN SHUTDOWN.

BECAUSE THERE IS NO GUARANTEE THAT THE PROCESS WHICH CRASHED THE SYSTEM (AND THUS, THE ADDRESS SPACE WHERE ESD IS RUNNING) IS NOT DEFECTIVE, ESD REBUILDS ITS PDS FROM THE template_pds.

ESD SWITCHES TO THE inzr_stk0 AND CALLS wired_shutdown.

ALL THE ABOVE STEPS WERE DONE IN ALM. wired_shutdown IS A PL/I PROCEDURE.

THE VTOC BUFFER IS CHECKED TO SEE WHETHER ANY OPERATIONS WERE IN PROGRESS. IF SO, THOSE VOLUMES ARE MARKED (IN THE PVT) TO INDICATE THAT THEY MAY HAVE INCONSISTENCIES.

MEMORY IS FLUSHED.

THE VTOC MANAGER IS REINITIALIZED.

AT THIS POINT, THE SUPERVISOR SHOULD BE WORKING WELL ENOUGH TO RUN shutdown_file_system. FROM NOW ON, EMERGENCY SHUTDOWN FOLLOWS THE SAME PATH AS NORMAL SHUTDOWN.

IF EMERGENCY SHUTDOWN FAILS, FOR TAKING A FAULT OR SOME OTHER REASON, IT CAN BE RETRIED INDEFINITELY.

SHUTDOWN IS MARKED COMPLETE IN VOLUME LABELS, AND THESE ARE NOT UPDATED UNTIL THE VERY END. THUS, NO HARM CAN COME FROM RETRYING ARBITRARILY.

ESD TRIES TO CORRECT ALL THE PROBLEMS THAT MIGHT ARISE. BECAUSE OF THE COMPLEXITY OF THE SUPERVISOR, IT DOES NOT ALWAYS GET ALL OF THEM.
TOPIC XII
File System Salvagers

Overview of Salvagers ........................................ 12-1
Directory Salvager ........................................... 12-4
Quota Salvaging .............................................. 12-7
Physical Volume Scavenger ................................... 12-9
Physical Volume Salvager .................................... 12-11
sweep_pv ..................................................... 12-12
OVERVIEW OF SALVAGERS

FUNCTION

ENSURE THE CONSISTENCY OF THE FILE SYSTEM DATABASES AND PERFORM PERIODIC PREVENTIVE MAINTENANCE OPERATIONS

THERE ARE SEVERAL SALVAGERS, EACH WITH A DIFFERENT FUNCTION


SOME SALVAGING IS DONE AUTOMATICALLY, WHEN THE SYSTEM DETECTS AN INCONSISTENCY. OTHER SALVAGE OPERATIONS ARE EXPLICITLY REQUESTED, BY PRIVILEGED USERS.

EXCEPT FOR SUPERVISOR BUGS, THE ONLY TIME DAMAGE OCCURS THAT REQUIRES SALVAGING TO FIX IS AFTER A CRASH WHERE EMERGENCY SHUTDOWN FAILS.

THE SALVAGERS:

DIRECTORY SALVAGER

CORRECTS INCONSISTENCIES IN DIRECTORY SEGMENTS BY REBUILDING THEM

THIS IS THE ONLY SALVAGER INVOKED AUTOMATICALLY IN USER PROCESSES: ANY ATTEMPT TO LEAVE RING ZERO WITH A DIRECTORY LOCKED FOR WRITING WILL CAUSE IT TO BE SALVAGED
OVERVIEW OF SALVAGERS

DIRECTORY SALVAGING ALSO RECLAIMS WASTED SPACE IN THE DIRECTORY, AND IS RUN PERIODICALLY TO COMPACT DIRECTORIES.

QUOTA SALVAGER

CORRECTS INCONSISTENCIES IN THE QUOTA SYSTEM.

PHYSICAL VOLUME SCAVENGER

RECONSTRUCTS RECORD AND VTOCE STOCK INFORMATION FROM THE VTOCES ON A VOLUME, THEREBY RECLAIMING ANY RECORDS OR VTOCES WHICH MIGHT HAVE BEEN LOST.

RUNS ENTIRELY ONLINE WHILE THE SYSTEM IS UP FOR USERS (NEW IN MR10.1).

THIS TYPE OF DAMAGE IS USUALLY BENIGN, SO RUNNING THE SCAVENGER CAN BE DELAYED.

PHYSICAL VOLUME SALVAGER

RECONSTRUCTS RECORD AND VTOCE STOCK INFORMATION.

RUNS ONLY DURING INITIALIZATION, AND THEREFORE DELAYS CRASH RECOVERY.

NOW USED ONLY FOR RARE CASES WHERE THERE IS NOT ENOUGH FREE SPACE LEFT FOR THE SCAVENGER TO RUN. IN THESE RARE CASES, IT IS INVOKED AUTOMATICALLY BY SYSTEM INITIALIZATION.

Sweep_pv
OVERVIEW OF SALVAGERS

DELETES UNUSED VTOC ENTRIES WHICH HAVE NO DIRECTORY ENTRY POINTING TO THEM

RUNS ENTIRELY IN USER RING, EXCEPT FOR ACTUALLY READING VTOC ENTRIES AND DIRECTORY ENTRIES

PURELY A HOUSEKEEPING FUNCTION, AND RUN ONLY RARELY.
DIRECTORY SALVAGER

The directory salvager is used for two main purposes: damage correction and storage reclamation.

Directory damage occurs as needed.

Damage can occur in a directory for several reasons:

- Disk I/O error writing bad data to a directory segment
- Supervisor bug
- Crash without ESD, where the directory was updated, but not fully written to disk

Actual damage to directories is comparatively rare, but:

- If a fault occurs for any reason while a user has a directory locked, the system assumes the directory could be at fault, and salvages it.

This sort of salvaging is done by the online_salvager program, which is invoked dynamically by verify_lock if a process attempts to leave ring zero ("crawl out") with a directory locked user message.

Ordinarily, there is nothing wrong, and the salvager just checks over the directory.

If the directory is damaged, it is "repaired" as well as possible.
DIRECTORY SALVAGER

**DIRECTORY SALVAGER**

Unfortunatley, the most common forms of damage destroy the directory header so that no repair is possible, and the directory must be reinitialized as empty.

Not really true, only for process dirs.

**STORAGE RECLAMATION**

- Directory space cannot always be efficiently re-used, and unusable space accumulates as entries are created and deleted.

- It is not practical for the system to compact directories when this happens, since this could cause major resource consumption, and cause expense for random processes.

- Instead, the directory salvager is periodically run in Salvager.SysDaemon processes to rebuild all the directories in the hierarchy, collecting all free space.

- This is done by calls to hphcs_ssalv_directory.

- If the Salvager processes detect damage, it is corrected as well.

- Demand directory salvaging is usually also instructed to look for and correct "connection failures."

  - A connection failure is a directory branch which indicates a free VTOCE, or a VTOCE with a different UID than the branch. This means that the directory and VTOC are inconsistent.

  - Connection failures are usually deleted by demand salvaging (this is an option).

  - Online salvaging does not delete connection failures because it is not always possible to identify them properly during an online salvage.

Not To Be Reproduced 12-5 F80A
DIRECTORY SALVAGER

DEMAND DIRECTORY SALVAGING CAN ALSO BE USED BY SYSTEM MAINTENANCE PERSONNEL WHEN THERE APPEAR TO BE DIRECTORY PROBLEMS.
QUOTA SALVAGING CORRECTS INCONSISTENCIES IN THE HIERARCHY OF QUOTA-USED VALUES

THESE INCONSISTENCIES ARISE AFTER A CRASH WHERE ESD FAILS:

| A SEGMENT (>udd>a>b) IS DELETED, FREEING PAGES, AND THE ASTES OF ITS PARENT (>udd>a) AND ITS PARENT'S PARENT (>udd) ARE UPDATED TO REFLECT THIS DECREASE IN QUOTA USED. |
| THE VTOCE FOR >udd>a IS WRITTEN TO DISK DURING NORMAL OPERATION. THE VTOCE FOR >udd IS NOT |
| THE SYSTEM CRASHES, AND NO ESD IS PERFORMED |
| WHEN THE SYSTEM COMES BACK UP, THE VTOCE FOR >udd IS INACCURATE: IT CLAIMS THAT RECORDS ARE STILL IN USE, WHEN IN FACT THEY WERE FREED |
| IF A NEW SEGMENT (>udd>a>c) IS NOW CREATED, IT MAY SPURIOUSLY CAUSE A RECORD QUOTA OVERFLOW ON >udd. |

QUOTA SALVAGING IS PERFORMED ENTIRELY ONLINE, WHILE THE SYSTEM IS RUNNING, USUALLY IN A Salvager.SysDaemon PROCESS

THE FUNDAMENTAL PRINCIPLE BEHIND THE QUOTA SALVAGER IS THAT QUOTA INCONSISTENCIES CANNOT ARISE DURING NORMAL OPERATION: IF QUOTA IS CONSISTENT, OR INCONSISTENT BY n RECORDS, IT WILL STAY THAT WAY UNLESS EXPLICITLY CORRECTED

THE QUOTA SALVAGER PROCESS WALKS THE HIERARCHY FROM THE BOTTOM UP, USING do subtree, CORRECTING INCONSISTENCIES IN A DIRECTORY AND ALL ITS SIBLINGS, THEN IN THEIR PARENT, AND SO ON UP
AN hphcs ENTRY IS CALLED TO CORRECT EACH DIRECTORY, ON THE ASSUMPTION THAT ALL ITS CHILDREN ARE CONSISTENT.

THE ACTUAL CORRECTION MECHANISM INVOLVES COMPLICATED MANIPULATION OF VARIOUS LOCKS. REFER TO THE Multics Storage System PLM (AN61) FOR A DESCRIPTION.
PHYSICAL VOLUME SCAVENGER

IT IS ALSO PRIMARILY USEFUL AFTER A CRASH WITHOUT ESD

IT REBUILDS THE VOLUME MAP AND VTOC MAP FOR A VOLUME BY EXAMINING ALL THE VTOCES AND ASTES FOR SEGMENTS ON THE VOLUME.

IT RUNS WHILE THE VOLUME IS IN USE BY USERS, AND NO INTERRUPTION OF SERVICE OCCURS.

DAMAGE TO THE MAPS OCCURS AFTER A CRASH WHEN THE STOCKS, IN MEMORY, DO NOT GET PROPERLY UPDATED TO THE MAPS ON DISK.

BECAUSE OF THE STOCK MANAGEMENT POLICIES, THIS IS ALWAYS BENIGN: RECORDS MAY BE MARKED IN-USE THAT ARE NOT, BUT IT IS NOT POSSIBLE FOR THE MAP TO INDICATE A RECORD AS FREE (REUSABLE) WHEN IT BELONGS TO A SEGMENT ON THE VOLUME.

FOR RECONSTRUCTING THE VOLUME MAP, THE SCAVENGER WORKS BY AN INTERACTION WITH THE PAGE CONTROL RECORD ALLOCATION MECHANISM.

IT BUILDS A MAP, IN scavenger_data, THAT SAYS WHAT VTOCE OWNS EACH RECORD ON THE VOLUME, AND THEN RESOLVES THE INCONSISTENCIES BETWEEN THAT AND THE MAP ON DISK.

THE DATABASE IS WIRED FOR THE DURATION OF THE SCAVENGE, AND CAN BE QUITE LARGE (64K FOR AN MSU0501).

WHILE A SCAVENGER IS RUNNING, PAGE CONTROL KEEPS THE DATABASE UP TO DATE AS IT ALLOCATES AND FREES RECORDS.

FOR VTOCES, THE PROBLEM IS MUCH SIMPLER: THE VTOC IS SIMPLY SCANNED, AND ALL IN-USE VTOCES ARE RECORDED.
PHYSICAL VOLUME SCAVENGER

vtoc man also keeps the table up to date as it frees and allocates vtoces on the volume.

A vtoce that appears in-use may not necessarily be part of the hierarchy -- sweep_pv is run to take care of that.

After the new maps are constructed, they are written to disk.

The scavenger runs entirely in ring zero, but it can be safely interrupted and restarted from the beginning at any time.

The scavenger accesses the vtoc using vtoc man.

A started per svuec

The scavenger takes about fifteen minutes per volume, but the system is up while it does so.

Esp-1ps from not ps

X repair quota > 2 usually done in couple of minutes fixes quota.

X repair salvquota > 2 -compact -check-vtoce

Sweep_pv all pvs - gc - force

1 delete connection failures.
1p's directory entry but no vtoc.
Message: svq fail svf contents not in volume table or contents.

Good Esp

Can do & repair quota but not necessary.
PHYSICAL VOLUME SALVAGER

- THIS DOES THE SAME JOB AS THE SCAVENGER, BUT CAN RUN ONLY WHEN THE VOLUME IS UNAVAILABLE TO USERS

- BECAUSE IT HAS THE VOLUME ALL TO ITSELF, IT IS MUCH SIMPLER:
  - IT SCANS THE VTOC, BUILDING THE SAME SORT OF DATABASES AS THE SCAVENGER
  - WHEN FINISHED, IT RESOLVES INCONSISTENCIES, AND WRITES THE MAPS BACK TO DISK
  - THE VOLUME SALVAGER ACCESSES THE VTOC USING salv_abs_seg_NN

- THE VOLUME SALVAGER TAKES ABOUT A MINUTE AND A HALF PER VOLUME, BUT ALL VOLUMES MUST BE SALVAGED BEFORE THE SYSTEM CAN RUN AGAIN

- AFTER A CRASH WITHOUT ESD, ALL VOLUMES MOUNTED AT THE TIME OF THE CRASH ARE PRESUMED TO BE INCONSISTENT

- THE VOLUME SALVAGER IS LARGELY OBSOLETE TODAY. IT IS RETAINED TO DEAL WITH RARE SITUATIONS WHICH MAKE IT IMPOSSIBLE TO BRING THE SYSTEM UP FAR ENOUGH TO RUN THE SCAVENGER WITHOUT FIRST REBUILDING THE FILE MAP TO FREE UNUSED SPACE
SWEEP PV

sweep pv ELIMINATES ORPHAN VTOCES

AN ORPHAN VTOCE IS ONE WHICH APPEARS TO DESCRIBE A VALID SEGMENT, BUT DOES NOT APPEAR IN ANY DIRECTORIES

ORPHANS ARE USUALLY CREATED BY CRASHES OR DIRECTORY DAMAGE WHERE THE DIRECTORY COULD NOT BE REPAIRED

ORPHANS ARE ALSO SOMETIMES CALLED "REVERSE CONNECTION FAILURES"

sweep pv WORKS BY INSPECTING EVERY VTOCE ON A VOLUME, AND ATTEMPTING TO FIND ITS PARENT

THERE IS A "UID PATHNAME" IN PART 3 OF ALL VTOCES WHICH CONTAINS THE UIDS OF ALL ITS PARENT DIRECTORIES

BY STARTING FROM THE ROOT, AND SEARCHING EACH DIRECTORY IN THE PATH FOR THE UID OF THE NEXT ONE, THE VTOCE CAN BE FOUND

IF IT CAN'T BE FOUND, IT IS AN ORPHAN, AND SWEEP_pv DELETES IT

IT IS POSSIBLE TO "ADOPT" ORPHANS INTO A DIFFERENT PLACE IN THE HIERARCHY IF IT IS IMPORTANT TO RECOVER THEIR CONTENTS, USING THE adopt_seg TOOL

THE UID PATHNAME OF A VTOCE CAN BE INTERPRETED MANUALLY BY USING THE vtoc_pathname TOOL

sweep pv -adopt -> not really a good idea

Not To Be Reproduced 12-12 (End Of Topic)
# TOPIC XIII

The Initializer.SysDaemon Process

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initializer Overview</td>
</tr>
<tr>
<td>Services of Initializer</td>
</tr>
</tbody>
</table>
INITIALIZER OVERVIEW

FUNCTION

| THE SYSTEM'S INITIALIZATION, ADMINISTRATIVE AND CONTROL PROCESS (Initializer.SysDaemon.z), RESPONSIBLE FOR:
| INITIALIZING THE OPERATING SYSTEM AT BOOTLOAD, FOLLOWING SUCCESSFUL INITIALIZATION OF THE SUPERVISOR
| ANSWERING SERVICE (login and logout)
| PROCESS CREATION AND DESTRUCTION
| MESSAGE COORDINATOR (DAEMON COORDINATION)
| SYSTEM ADMINISTRATION FUNCTIONS
| SYSTEM ACCOUNTING FUNCTIONS

MAJOR DATA BASES, ALL KEPT IN >sc1

| ANSWER_TABLE
| ABSENTEE_USER_TABLE
| DAEMON_USER_TABLE
INITIALIZER OVERVIEW

- MASTER GROUP TABLE (MGT)
- CHANNEL DEFINITION TABLE (CDT)
- SYSTEM ADMINISTRATION TABLE (SAT)
- PERSON NAME TABLE (PNT)
- PROJECT DEFINITION TABLES (PDT'S)
SERVICES OF INITIALIZER

THE Initializer.SysDaemon process performs various services necessary for the supervisor. Many of them are performed by the initializer primarily for convenience, since it is easier to control resources in a controlled environment such as a single process than to create a mechanism which can be run in all processes.

SYSTEM INITIALIZATION AND SHUTDOWN

THE Initializer process is created from the environment that initializes the supervisor.

When the system is shut down, the Initializer coordinates the orderly cessation of system activity and calls hphcs_$shutdown to shut down the supervisor.

LOGICAL VOLUME MANAGEMENT

THE Initializer process handles all requests from user processes to mount and unmount private logical volumes.

It also makes the necessary checks to accept a volume into the hierarchy or remove it for demounting.

RESOURCE CONTROL

THE Initializer does all the control, assignment, and access checking of resources (such as I/O devices) controlled by RCP.

PROCESS CREATION AND DESTRUCTION
SERVICES OF INITIALIZER

THE Initializer creates and destroys all processes, schedules absentee processes, and handles "console" I/O from daemon processes.

An important service of process management is process dir volume management, in which the Initializer picks the logical volumes used for process directories.

The Initializer also handles all process accounting and load control.

Communications

The Initializer manages loading / dumping of FNPs and software communications multiplexers.

It also handles allocation of communication channels, both for processes logging in and requests made through dial_manager.

Dynamic Reconfiguration

The Initializer runs the dynamic reconfiguration software, and updates the system load limits when the configuration changes.

Actually, any process with hphcs access can use the reconfiguration commands, but only the Initializer can keep all the accounting databases up to date.

Not To Be Reproduced (End Of Topic)
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meter and Tuning Overview</td>
<td>14-1</td>
</tr>
<tr>
<td>Analyzing Performance Problems</td>
<td>14-3</td>
</tr>
<tr>
<td>Detailed Problem Analysis</td>
<td>14-6</td>
</tr>
<tr>
<td>Output From Metering Commands</td>
<td>14-8</td>
</tr>
<tr>
<td>total_time_meters</td>
<td>14-8</td>
</tr>
<tr>
<td>interrupt_meters</td>
<td>14-9</td>
</tr>
<tr>
<td>file_system_meters</td>
<td>14-10</td>
</tr>
<tr>
<td>file_system_meters</td>
<td>14-12</td>
</tr>
<tr>
<td>device_meters</td>
<td>14-13</td>
</tr>
<tr>
<td>disk_meters</td>
<td>14-14</td>
</tr>
<tr>
<td>disk_queue</td>
<td>14-15</td>
</tr>
<tr>
<td>print_configuration_deck</td>
<td>14-16</td>
</tr>
<tr>
<td>list_vols</td>
<td>14-18</td>
</tr>
<tr>
<td>traffic_control_queue</td>
<td>14-19</td>
</tr>
<tr>
<td>traffic_control_meters</td>
<td>14-21</td>
</tr>
<tr>
<td>print_tuning_parameters</td>
<td>14-23</td>
</tr>
<tr>
<td>work_class_meters</td>
<td>14-24</td>
</tr>
<tr>
<td>respons_meters</td>
<td>14-25</td>
</tr>
<tr>
<td>system_performance_graph</td>
<td>14-27</td>
</tr>
<tr>
<td>meter_gate</td>
<td>14-29</td>
</tr>
</tbody>
</table>
METER AND TUNING OVERVIEW

While not a subsystem itself, metering and tuning is a policy and capability common to all of the supervisor's subsystems.

Function

Metering (consists of three activities)

1. Accumulating Data: This is performed throughout the supervisor by code which
   - Records the number of times an event happens or a particular piece of code is executed; and/or
   - Records the time required to perform a task
   - Such data is stored in areas referred to as "metering cells"

2. Extracting Data: This is performed by numerous metering commands which (when invoked)
   - Read and store the current values of relevant metering cells

3. Reporting the Data: This is performed by the meter commands which (when invoked)
   - Compare current metering cell values with previously read values
   - Perform the appropriate arithmetic computations upon the data in order to arrive at the desired statistic
   - Arrange the data in a useful format (a report or diagram) and print it
METER AND TUNING OVERVIEW

TUNING

CHANGING THE SYSTEM'S OPERATING PARAMETERS AND/OR
CONFIGURATION BASED UPON THE DATA AND INSIGHTS FROM THE
SYSTEM'S METERS

MAJOR DATA BASES

SST HEADER, TC_DATA HEADER, ETC.
MULTICS IS VERY HEAVILY INSTRUMENTED WITH MANY METERING COMMANDS

MOST SOLVABLE PERFORMANCE PROBLEMS SHOW UP QUITE DIRECTLY IN THE METERS

IF THE SYSTEM IS TOO SLOW, THERE ARE GOOD RULES OF THUMB TO FOLLOW TO LOOK FOR WHEN PROBLEM

NO TWO SYSTEMS ARE IDENTICAL - THE MOST IMPORTANT INFORMATION YOU HAVE IS HOW THE METERS ARE DIFFERENT FROM THE WAY THEY WERE WHEN THE SYSTEM WAS WORKING BETTER

NOT ALL TUNING PROBLEMS CAN BE RESOLVED WITH SOFTWARE. OFTEN IT SIMPLY INDICATES THAT THERE IS NOT ENOUGH MEMORY, OR NOT ENOUGH DISK CAPACITY. IT IS OFTEN DIFFICULT TO DETERMINE WHAT HARDWARE CHANGES WOULD BE MOST COST-EFFECTIVE

total_time_meters - THE FIRST STEP

total_time_meters MAY INDICATE EXCESSIVE TIME SPENT IN SEVERAL AREAS:

PAGE FAULTS - TOO MANY PAGE FAULTS MEAN NOT ENOUGH MEMORY, TOO MANY ELIGIBLE PROCESSES (max_eligible), WHICH CAUSE THRASHING, INSUFFICIENT DISK CAPACITY, OR INEFFICIENT APPLICATIONS

LOOK TO file_system_meters, device_meters, AND disk_meters FOR MORE HELP
ANALYZING PERFORMANCE PROBLEMS

Segment faults - too many almost always mean that the AST pools (sst config card) are too small

Look to file_system_meters (AST Pool grace time) for more help.

Interrupt - usually means excessive interrupt activity either for FNPS or because of excessive paging, but may indicate hardware problems.

Look to interrupt_meters to localize it, then to file_system_meters and system_comm_meters.

Other fault - generally indicates too many connect or timer runouts faults, indicating excessive traffic control activity.

Look to the tuning parameters (ptp) and traffic_control_meters for more help.

MP idle - indicates too much traffic control activity, usually because there are too many eligible processes (max_eligible) and/or not enough memory.

Look to the tuning parameters (ptp), file_system_meters, device_meters, disk_meters, and traffic_control_meters for more help.

Work class idle - this is CPU time wasted because governed workclasses were not permitted to use it, and no other takers wanted it.

Frequently not a problem, but may indicate a need to readjust workclasses (work_class_meters).

NMP idle, zero idle - indicate that CPU time has gone to waste because no processes wanted it. NMP idle means that there are more processors than processes that want CPU time, and zero idle means there are no processes at all.
ANALYZING PERFORMANCE PROBLEMS

NORMALLY INSIGNIFICANT QUANTITIES - THE FOLLOWING NUMBERS SHOULD ALWAYS BE VERY SMALL; LARGE VALUES PROBABLY INDICATE A HARDWARE OR SOFTWARE PROBLEM

PC LOOP LOCKS
BOUND FAULTS
TC LOOP LOCKS
POST PURGING
LOADING IDLE
OTHER OVERHEAD
DETAILED PROBLEM ANALYSIS

THE MORE DETAILED METERING COMMANDS CAN BE USED TO PIN DOWN A PROBLEM

DETAILED DESCRIPTIONS CAN BE FOUND IN THE Multics Metering Manual (AN52)

THE DETAILED METERS SHOULD ALSO BE CHECKED IF THERE IS NO OBVIOUS PROBLEM SHOWN BY total_time_meters

AGAIN, THE REALLY IMPORTANT THING IS TO BE ABLE TO COMPARE AGAINST PREVIOUS DATA FOR YOUR SITE

SOME GOOD PLACES TO LOOK ARE:

interrupt_meters

UNUSUALLY LONG DISK INTERRUPTS MAY INDICATE LOCKING PROBLEMS - SEE disk_meters

TOO MUCH TOTAL TIME SPENT WITH DISK INTERRUPTS USUALLY MEANS TOO MANY PAGE FAULTS, WHICH WILL BE SHOWN IN MORE DETAIL BY file_system_meters

FNP INTERRUPTS ARE TYPICALLY MUCH LONGER THAN OTHERS. TOO MUCH FNP INTERRUPT TIME MAY INDICATE A BAD FRONT END CHANNEL, BUT THIS IS RARE
DETAILED PROBLEM ANALYSIS

file_system_meters

A short page lap time means that pages are not staying in memory long enough; that is, there is not enough memory.

Frequent page claim runs indicate that the write limit is too low.

If the average page fault duration is high, it can mean that the write limit is too high, and that disk allocation locks are a limitation; see disk_meters to check.

Short ast pool grace times mean that the pool sizes should be increased. This is usually also indicated by too high a frequency of segment faults.

disk_meters

When the ATB I/O for a drive is too low, the drive is a serious bottleneck.

This often happens for RLV drives, because of directories.

If PARM DIRW is on, it should be turned off.

Too high a percentage of allocation locks indicate that too many writes are being queued simultaneously, and making it impossible to get a queue entry for reading.

This usually means that WRITE_LIMIT is too high.

device_meters

If the subsystem busy percentage is too close to the disk channel capacity, it usually means that there are too few physical disk channels.
**OUTPUT FROM METERING COMMANDS**

**total time meters**

* TOTAL TIME METER - OVERVIEW OF HOW THE SYSTEM IS USING ITS RESOURCES, ALSO MEASURED AGAINST NON-IDLE TIME

<table>
<thead>
<tr>
<th>Total metering time</th>
<th>0:20:36</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>%NI</td>
</tr>
<tr>
<td>Page Faults</td>
<td>6.55</td>
</tr>
<tr>
<td>PC Loop Locks</td>
<td>0.39</td>
</tr>
<tr>
<td>PC Queue</td>
<td>0.86</td>
</tr>
<tr>
<td>Seg Faults</td>
<td>1.74</td>
</tr>
<tr>
<td>Bound Faults</td>
<td>0.15</td>
</tr>
<tr>
<td>Interrupts</td>
<td>9.02</td>
</tr>
<tr>
<td>Other Fault</td>
<td>8.45</td>
</tr>
<tr>
<td>Getwork</td>
<td>4.54</td>
</tr>
<tr>
<td>TC Loop Locks</td>
<td>0.20</td>
</tr>
<tr>
<td>Post Purging</td>
<td>0.36</td>
</tr>
<tr>
<td>MP Idle</td>
<td>0.70</td>
</tr>
<tr>
<td>Work Class Idle</td>
<td>0.98</td>
</tr>
<tr>
<td>Loading Idle</td>
<td>0.16</td>
</tr>
<tr>
<td>NMP Idle</td>
<td>8.84</td>
</tr>
<tr>
<td>Zero Idle</td>
<td>0.00</td>
</tr>
<tr>
<td>Other Overhead</td>
<td>0.03</td>
</tr>
<tr>
<td>Virtual CPU Time</td>
<td>62.53</td>
</tr>
</tbody>
</table>

**Page Faults**
- Check AVE first
- Check file system next
- Check disk I/O

**Seg Faults**
- Check file system next
- Check for low disk I/O

**Interrupts**
- Check file system next
- Check disk I/O

**Getwork**
- Check file system next
- Check for low disk I/O

**MP Idle**
- Check file system next
- Check for low disk I/O

---

Not To Be Reproduced
### OUTPUT FROM METERING COMMANDS

**interrupt meters**

#### INTERRUPT_METERS - COUNTS & TIMING FOR ALL I/O INTERRUPTS

Total metering time 0:20:28

<table>
<thead>
<tr>
<th>IOM Ch</th>
<th>Int</th>
<th>Avg Time</th>
<th>% CPU</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>2.043</td>
<td>0.00</td>
<td>IOM A special</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>1.553</td>
<td>0.01</td>
<td>prtb</td>
</tr>
<tr>
<td>A</td>
<td>13</td>
<td>3.436</td>
<td>0.65</td>
<td>fnp c</td>
</tr>
<tr>
<td>A</td>
<td>14</td>
<td>0.788</td>
<td>0.00</td>
<td>opc</td>
</tr>
<tr>
<td>A</td>
<td>16</td>
<td>1.347</td>
<td>0.72</td>
<td>tapa</td>
</tr>
<tr>
<td>A</td>
<td>17</td>
<td>1.328</td>
<td>0.00</td>
<td>tapa</td>
</tr>
<tr>
<td>A</td>
<td>18</td>
<td>5.383</td>
<td>1.43</td>
<td>fnp b</td>
</tr>
<tr>
<td>A</td>
<td>20</td>
<td>0.860</td>
<td>0.37</td>
<td>dska</td>
</tr>
<tr>
<td>A</td>
<td>21</td>
<td>0.860</td>
<td>0.05</td>
<td>dska</td>
</tr>
<tr>
<td>A</td>
<td>22</td>
<td>0.877</td>
<td>0.15</td>
<td>dskb</td>
</tr>
<tr>
<td>A</td>
<td>23</td>
<td>0.964</td>
<td>0.01</td>
<td>dskb</td>
</tr>
<tr>
<td>A</td>
<td>24</td>
<td>0.895</td>
<td>0.33</td>
<td>dskb</td>
</tr>
<tr>
<td>A</td>
<td>25</td>
<td>0.943</td>
<td>0.05</td>
<td>dskb</td>
</tr>
<tr>
<td>A</td>
<td>26</td>
<td>0.857</td>
<td>0.18</td>
<td>dska</td>
</tr>
<tr>
<td>A</td>
<td>27</td>
<td>0.899</td>
<td>0.00</td>
<td>dska</td>
</tr>
<tr>
<td>A</td>
<td>28</td>
<td>0.845</td>
<td>0.43</td>
<td>dskc</td>
</tr>
<tr>
<td>A</td>
<td>29</td>
<td>0.849</td>
<td>0.04</td>
<td>dskc</td>
</tr>
<tr>
<td>A</td>
<td>30</td>
<td>0.845</td>
<td>0.17</td>
<td>dskc</td>
</tr>
<tr>
<td>A</td>
<td>31</td>
<td>0.767</td>
<td>0.00</td>
<td>dskc</td>
</tr>
<tr>
<td>B</td>
<td>14</td>
<td>0.877</td>
<td>0.00</td>
<td>fnp d</td>
</tr>
<tr>
<td>B</td>
<td>15</td>
<td>3.351</td>
<td>0.47</td>
<td>fnp f</td>
</tr>
<tr>
<td>B</td>
<td>18</td>
<td>3.543</td>
<td>0.26</td>
<td>fnp a</td>
</tr>
<tr>
<td>B</td>
<td>19</td>
<td>3.435</td>
<td>0.96</td>
<td>fnp e</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>0.881</td>
<td>0.39</td>
<td>dskb</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>0.863</td>
<td>0.08</td>
<td>dskb</td>
</tr>
<tr>
<td>B</td>
<td>22</td>
<td>0.861</td>
<td>0.11</td>
<td>dska</td>
</tr>
<tr>
<td>B</td>
<td>23</td>
<td>0.925</td>
<td>0.00</td>
<td>dska</td>
</tr>
<tr>
<td>B</td>
<td>24</td>
<td>0.867</td>
<td>0.28</td>
<td>dska</td>
</tr>
<tr>
<td>B</td>
<td>25</td>
<td>0.841</td>
<td>0.02</td>
<td>dska</td>
</tr>
<tr>
<td>B</td>
<td>26</td>
<td>0.863</td>
<td>0.22</td>
<td>dskb</td>
</tr>
<tr>
<td>B</td>
<td>27</td>
<td>0.903</td>
<td>0.02</td>
<td>dskb</td>
</tr>
<tr>
<td>B</td>
<td>28</td>
<td>0.838</td>
<td>0.09</td>
<td>dskc</td>
</tr>
<tr>
<td>B</td>
<td>29</td>
<td>0.653</td>
<td>0.00</td>
<td>dskc</td>
</tr>
<tr>
<td>B</td>
<td>30</td>
<td>0.823</td>
<td>0.28</td>
<td>dskc</td>
</tr>
<tr>
<td>B</td>
<td>31</td>
<td>0.841</td>
<td>0.01</td>
<td>dskc</td>
</tr>
<tr>
<td>B</td>
<td>32</td>
<td>0.825</td>
<td>0.03</td>
<td>dske</td>
</tr>
</tbody>
</table>

Chan      262506     1.464    7.82
Ovhd       258932     0.230    1.21
Total      258932     1.713    9.03
OUTPUT FROM METERING COMMANDS

file system meters

`FILE_SYSTEM_METERS` - DISPLAYS MISCELLANEOUS METERING INFORMATION FOR THE FILE SYSTEM

I Only parts relevant to segment control included here; see topic 8 (page control) for the rest.

Total metering time 0:20:02

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>ATB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activations</td>
<td>1043</td>
<td>1.153 sec.</td>
</tr>
<tr>
<td>segfault</td>
<td>969</td>
<td>1.241 sec. 92.905% of all</td>
</tr>
<tr>
<td>makeknown</td>
<td>74</td>
<td>16.251 sec. 7.095% of all</td>
</tr>
<tr>
<td>directories</td>
<td>96</td>
<td>12.527 sec. 9.204% of all</td>
</tr>
<tr>
<td>Deactivations</td>
<td>1056</td>
<td>1.139 sec.</td>
</tr>
<tr>
<td>Demand deactivate attempts</td>
<td>3</td>
<td>400.857 sec.</td>
</tr>
<tr>
<td>Seg Faults</td>
<td>5080</td>
<td>0.237 sec.</td>
</tr>
<tr>
<td>fault</td>
<td>4311</td>
<td>0.279 sec. 84.862% of Seg Faults</td>
</tr>
<tr>
<td>call</td>
<td>769</td>
<td>1.564 sec. 15.138% of Seg Faults</td>
</tr>
<tr>
<td>activations</td>
<td>969</td>
<td>1.241 sec. 19.075% of Seg Faults</td>
</tr>
<tr>
<td>Bound Faults</td>
<td>120</td>
<td>5.466 sec.</td>
</tr>
<tr>
<td>Setfaults</td>
<td>4484</td>
<td>268.191 msec.</td>
</tr>
<tr>
<td>access</td>
<td>42</td>
<td>28.633 sec. 0.937% of setfaults</td>
</tr>
<tr>
<td>ASTE Trickle</td>
<td>139</td>
<td>8.652 sec.</td>
</tr>
<tr>
<td>Steps</td>
<td>4779</td>
<td>281.040 msec.</td>
</tr>
<tr>
<td>Skips</td>
<td>3016</td>
<td>0.399 sec. 70.484% of Steps</td>
</tr>
<tr>
<td>ens</td>
<td>271</td>
<td>4.438 sec. 8.983% of Skips</td>
</tr>
<tr>
<td>mem</td>
<td>1083</td>
<td>1.110 sec. 35.909% of Skips</td>
</tr>
<tr>
<td>init</td>
<td>1662</td>
<td>0.724 sec. 55.106% of Skips</td>
</tr>
<tr>
<td>Searches</td>
<td>0</td>
<td>0.000 sec.</td>
</tr>
<tr>
<td>Cleanups</td>
<td>1056</td>
<td>1.139 sec. 0.1% of real time</td>
</tr>
<tr>
<td>Force writes</td>
<td>3</td>
<td>400.857 sec.</td>
</tr>
<tr>
<td>pages written</td>
<td>3</td>
<td>400.857 sec.</td>
</tr>
<tr>
<td>Lock AST</td>
<td>18422</td>
<td>0.065 sec.</td>
</tr>
</tbody>
</table>
## OUTPUT FROM METERING COMMANDS

### file system meters

<table>
<thead>
<tr>
<th>Metric</th>
<th>AVE/lock</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>AST locked</td>
<td>4.833 msec.</td>
<td>7.4</td>
</tr>
<tr>
<td>AST lock waiting</td>
<td>1.601 msec.</td>
<td>2.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AST Sizes</td>
<td>4, 16, 64, 256</td>
</tr>
<tr>
<td>Number</td>
<td>1701, 601, 221, 74</td>
</tr>
<tr>
<td>Need</td>
<td>819, 202, 208, 34</td>
</tr>
<tr>
<td>Steps</td>
<td>2341, 645, 1139, 154</td>
</tr>
<tr>
<td>Ave Steps</td>
<td>2.9, 3.2, 5.5, 4.5</td>
</tr>
<tr>
<td>Lap Time(sec)</td>
<td>873.8, 1120.5, 233.3, 577.9</td>
</tr>
</tbody>
</table>
FILE_SYSTEM_METERS - DISPLAYS MISCELLANEOUS METERING INFORMATION FOR THE FILE SYSTEM

ONLY PARTS RELEVANT TO PAGE CONTROL INCLUDED HERE; SEE TOPIC 7 (SEGMENT CONTROL) FOR THE REST

Total metering time 0:20:02

<table>
<thead>
<tr>
<th>#</th>
<th>ATB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Needc</td>
<td>62654</td>
</tr>
<tr>
<td>Ring 0 faults</td>
<td></td>
</tr>
<tr>
<td>PDIR faults</td>
<td></td>
</tr>
<tr>
<td>Level 2 faults</td>
<td></td>
</tr>
<tr>
<td>DIR faults</td>
<td></td>
</tr>
<tr>
<td>New Pages</td>
<td></td>
</tr>
<tr>
<td>Volmap_seg</td>
<td>0</td>
</tr>
<tr>
<td>Zero pages</td>
<td>770</td>
</tr>
<tr>
<td>Laps</td>
<td>105</td>
</tr>
<tr>
<td>Steps</td>
<td>361483</td>
</tr>
<tr>
<td>Skip wired</td>
<td>11057</td>
</tr>
<tr>
<td>Skip used</td>
<td>109719</td>
</tr>
<tr>
<td>Skip mod</td>
<td>140336</td>
</tr>
<tr>
<td>Skip fc pin</td>
<td>37717</td>
</tr>
<tr>
<td>Skip cl pin</td>
<td>23726</td>
</tr>
</tbody>
</table>

3419 pages, 139 wired.
Average steps 5.770
## OUTPUT FROM METERING COMMANDS

```
device meters
```

### DEVICE METERS - DISPLAYS SUMMARY OF I/O ACTIVITY FOR ALL DISK SUBSYSTEMS

- Total metering time: **0:20:13**

<table>
<thead>
<tr>
<th></th>
<th>dska</th>
<th>dskb</th>
<th>dskc</th>
<th>dskd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prior Page I/O</td>
<td>18571</td>
<td>17743</td>
<td>462</td>
<td>1273</td>
</tr>
<tr>
<td>ATB</td>
<td>65.334</td>
<td>68.383</td>
<td>2626.240</td>
<td>953.121</td>
</tr>
<tr>
<td>Other Page I/O</td>
<td>6525</td>
<td>5135</td>
<td>16</td>
<td>696</td>
</tr>
<tr>
<td>ATB</td>
<td>185.949</td>
<td>236.284</td>
<td>75832.692</td>
<td>1743.280</td>
</tr>
<tr>
<td>ATB Page I/O</td>
<td>48.347</td>
<td>53.034</td>
<td>2538.332</td>
<td>616.212</td>
</tr>
<tr>
<td>Prior VTOCE I/O</td>
<td>934</td>
<td>895</td>
<td>38</td>
<td>304</td>
</tr>
<tr>
<td>ATB</td>
<td>1299.061</td>
<td>1355.668</td>
<td>31929.554</td>
<td>3991.194</td>
</tr>
<tr>
<td>ATB I/O</td>
<td>46.612</td>
<td>51.037</td>
<td>2351.401</td>
<td>533.798</td>
</tr>
<tr>
<td>% Busy</td>
<td>76</td>
<td>76</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Avg. Page Wait</td>
<td>47.289</td>
<td>46.197</td>
<td>20.341</td>
<td>24.666</td>
</tr>
<tr>
<td>Avg. Page ^Wait</td>
<td>176.082</td>
<td>101.023</td>
<td>36.996</td>
<td>61.704</td>
</tr>
<tr>
<td>Avg. VTOCE Wait</td>
<td>41.138</td>
<td>37.610</td>
<td>38.595</td>
<td>29.090</td>
</tr>
<tr>
<td>Avg. Page I/O T</td>
<td>35.619</td>
<td>38.314</td>
<td>20.050</td>
<td>22.482</td>
</tr>
<tr>
<td>Avg. VTOCE I/O T</td>
<td>31.139</td>
<td>32.277</td>
<td>37.060</td>
<td>26.606</td>
</tr>
<tr>
<td>EDAC Corr. Errs</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Fatal Errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Not To Be Reproduced 14-13 F80A
OUTPUT FROM METERING COMMANDS

disk_meters

* DISK_METERS - DISPLAYS I/O ACTIVITY TO DISK DRIVES

ONLY ONE SUBSYSTEM SHOWN HERE TO CONSERVE SPACE

Total metering time 0:20:12

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Count</th>
<th>Waits</th>
<th>%Waits</th>
<th>Avg. Wait(ms.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>call locks</td>
<td>26005</td>
<td>217</td>
<td>0.83</td>
<td>0.259</td>
</tr>
<tr>
<td>run locks</td>
<td>112</td>
<td>0</td>
<td>0.00</td>
<td>0.000</td>
</tr>
<tr>
<td>interrupt locks</td>
<td>25998</td>
<td>239</td>
<td>0.92</td>
<td>0.208</td>
</tr>
<tr>
<td>allocations</td>
<td>26001</td>
<td>0</td>
<td>0.00</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drive</th>
<th>Reads</th>
<th>Writes</th>
<th>Seek Distance</th>
<th>ATB Reads</th>
<th>ATB Writes</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>269</td>
<td>67</td>
<td>214</td>
<td>4508</td>
<td>18102</td>
<td>3609</td>
</tr>
<tr>
<td>3</td>
<td>362</td>
<td>243</td>
<td>109</td>
<td>3350</td>
<td>4991</td>
<td>2004</td>
</tr>
<tr>
<td>4</td>
<td>309</td>
<td>131</td>
<td>184</td>
<td>3925</td>
<td>9258</td>
<td>2756</td>
</tr>
<tr>
<td>5</td>
<td>547</td>
<td>165</td>
<td>180</td>
<td>2217</td>
<td>7350</td>
<td>1703</td>
</tr>
<tr>
<td>6</td>
<td>631</td>
<td>165</td>
<td>161</td>
<td>1922</td>
<td>7350</td>
<td>1523</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>5843</td>
<td>2187</td>
<td>122</td>
<td>207</td>
<td>554</td>
<td>151</td>
</tr>
<tr>
<td>9</td>
<td>366</td>
<td>116</td>
<td>153</td>
<td>3313</td>
<td>10455</td>
<td>2516</td>
</tr>
<tr>
<td>11</td>
<td>3501</td>
<td>1431</td>
<td>200</td>
<td>346</td>
<td>847</td>
<td>245</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>7158</td>
<td>2508</td>
<td>135</td>
<td>169</td>
<td>483</td>
<td>125</td>
</tr>
</tbody>
</table>

Not To Be Reproduced 14-14 F80A
OUTPUT FROM METERING COMMANDS

**disk_queue**

*DISK_QUEUE - DISPLAYS I/O QUEUE FOR A DISK SUBSYSTEM*

*ONLY ONE SUBSYSTEM SHOWN HERE TO CONSERVE SPACE*

Connects = 2604781, 1359725, 677321, 309367, 123430, 40159, 10227, 1969.

<table>
<thead>
<tr>
<th>P</th>
<th>RW</th>
<th>VP</th>
<th>DV</th>
<th>SECTOR</th>
<th>MEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>24</td>
<td>1350330</td>
<td>27304000</td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>9</td>
<td>1020150</td>
<td>4432000</td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>16</td>
<td>1204130</td>
<td>36246000</td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>16</td>
<td>314370</td>
<td>27306000</td>
</tr>
<tr>
<td>0</td>
<td>W</td>
<td>P</td>
<td>16</td>
<td>314430</td>
<td>34166000</td>
</tr>
</tbody>
</table>

You should have a logical channel for each arm. Seek time 8 x transfer time.

As of 1971, disk queue so containing subsystem isn't necessary.
OUTPUT FROM METERING COMMANDS

\[ \text{print configuration deck} \]

PRINT_CONFIGURATION_DECK - DISPLAYS \texttt{>s11>config_deck}, WHICH CONTAINS INFORMATION ABOUT DISK LOCATIONS, THE RLV, AND PARTITIONS

ONLY THE PART OF THE CONFIG DECK RELEVANT TO VOLUME MANAGEMENT AND DISK CONFIGURATION IS SHOWN HERE

\begin{verbatim}
part bos dska 16.
part dump dska 16.
part log dska 16.
prph dska a 20. 2 451. 16.
chnl dska a 26. 2 b 24. 2 b 22. 2
prph dskb b 20. 2 0 16. 451. 16.
chnl dskb b 26. 2 a 24. 2 a 22. 2
prph dskc a 28. 2 501. 32.
chnl dskc a 30. 2 b 30. 2 b 28. 2
prph dske b 32. 2 451. 8.
chnl dske b 34. 2
prph dskf a 32. 2 501. 16.
chnl dskf a 34. 2
mpc mspa 451. a 20. 4 a 24. 4
mpc mspb 451. b 20. 4 b 24. 4
mpc mspc 607. a 28. 4
mpc mspd 607. b 28. 4
mpc mspe 451. b 32. 4
mpc mspe 607. a 32. 4
\end{verbatim}
OUTPUT FROM METERING COMMANDS

print configuration deck

- DISK CONFIGURATION CONFIG CARDS

  | ROOT
  |
  | IDENTIFIES THOSE VOLUMES IN THE ROOT LOGICAL VOLUME WHICH HAVE HC PARTITIONS, USED BY THE SUPERVISOR FOR PAGING OF SUPERVISOR SEGMENTS

  | PART
  |
  | IDENTIFIES THE LOCATIONS OF CERTAIN IMPORTANT PARTITIONS
  |
  | ONLY PARTITIONS NECESSARY FOR MULTICS OPERATIONS ARE IDENTIFIED, NOT ALT PARTITIONS
  |
  | HC PARTITIONS ARE LOCATED BY THE ROOT CARD

  | PRPH DSKn, CHNL
  |
  | IDENTIFY PHYSICAL I/O CHANNEL PATHS FOR ACCESSING DISK DRIVES

  | MPC
  |
  | IDENTIFY PHYSICAL CONNECTIONS TO MICROPROGRAMMED DISK CONTROLLERS
## Output from Metering Commands

### list_vols

list_vols - displays a table of online volumes, their location, and space utilization.

<table>
<thead>
<tr>
<th>Drive</th>
<th>Records</th>
<th>Left %</th>
<th>VTOCS Left %</th>
<th>Avg PV</th>
<th>PB/PD</th>
<th>LV Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dskc_17</td>
<td>64504</td>
<td>54730</td>
<td>85</td>
<td>13440</td>
<td>11356</td>
<td>84</td>
</tr>
<tr>
<td>dskc_18</td>
<td>64504</td>
<td>55889</td>
<td>86</td>
<td>13440</td>
<td>11352</td>
<td>84</td>
</tr>
<tr>
<td>dska_05</td>
<td>36428</td>
<td>5305</td>
<td>15</td>
<td>8400</td>
<td>2662</td>
<td>32</td>
</tr>
<tr>
<td>dska_06</td>
<td>36428</td>
<td>4523</td>
<td>12</td>
<td>8400</td>
<td>2365</td>
<td>28</td>
</tr>
<tr>
<td>dskb_19</td>
<td>36428</td>
<td>4632</td>
<td>13</td>
<td>8400</td>
<td>2813</td>
<td>33</td>
</tr>
<tr>
<td>dskb_26</td>
<td>36429</td>
<td>13690</td>
<td>38</td>
<td>8400</td>
<td>4326</td>
<td>52</td>
</tr>
<tr>
<td>dskb_27</td>
<td>36429</td>
<td>4672</td>
<td>13</td>
<td>8400</td>
<td>2333</td>
<td>28</td>
</tr>
<tr>
<td>dska_01</td>
<td>36308</td>
<td>4680</td>
<td>13</td>
<td>9000</td>
<td>450</td>
<td>5</td>
</tr>
<tr>
<td>dska_03</td>
<td>36256</td>
<td>3588</td>
<td>10</td>
<td>9200</td>
<td>1017</td>
<td>11</td>
</tr>
<tr>
<td>dska_04</td>
<td>36256</td>
<td>4500</td>
<td>12</td>
<td>9200</td>
<td>884</td>
<td>10</td>
</tr>
<tr>
<td>dska_09</td>
<td>36256</td>
<td>4816</td>
<td>13</td>
<td>9200</td>
<td>864</td>
<td>9</td>
</tr>
<tr>
<td>dskc_12</td>
<td>36256</td>
<td>4281</td>
<td>12</td>
<td>9200</td>
<td>1002</td>
<td>11</td>
</tr>
<tr>
<td>dskc_18</td>
<td>36256</td>
<td>3840</td>
<td>11</td>
<td>9200</td>
<td>539</td>
<td>6</td>
</tr>
<tr>
<td>dskc_19</td>
<td>64504</td>
<td>294</td>
<td>0</td>
<td>13440</td>
<td>5555</td>
<td>41</td>
</tr>
<tr>
<td>dskc_14</td>
<td>64504</td>
<td>269</td>
<td>0</td>
<td>13440</td>
<td>5214</td>
<td>39</td>
</tr>
<tr>
<td>dskc_01</td>
<td>64503</td>
<td>43631</td>
<td>68</td>
<td>13440</td>
<td>10032</td>
<td>75</td>
</tr>
<tr>
<td>dskc_02</td>
<td>64503</td>
<td>45502</td>
<td>71</td>
<td>13440</td>
<td>9386</td>
<td>73</td>
</tr>
<tr>
<td>dskc_03</td>
<td>64503</td>
<td>42374</td>
<td>66</td>
<td>13440</td>
<td>9839</td>
<td>73</td>
</tr>
<tr>
<td>dskc_04</td>
<td>64503</td>
<td>43591</td>
<td>68</td>
<td>13440</td>
<td>9785</td>
<td>73</td>
</tr>
<tr>
<td>dskc_09</td>
<td>64504</td>
<td>58010</td>
<td>90</td>
<td>13440</td>
<td>12394</td>
<td>92</td>
</tr>
<tr>
<td>dskc_10</td>
<td>64504</td>
<td>55786</td>
<td>88</td>
<td>13440</td>
<td>12477</td>
<td>92</td>
</tr>
<tr>
<td>dskc_21</td>
<td>64503</td>
<td>23947</td>
<td>37</td>
<td>13440</td>
<td>6446</td>
<td>48</td>
</tr>
<tr>
<td>dskc_22</td>
<td>64503</td>
<td>23744</td>
<td>37</td>
<td>13440</td>
<td>6194</td>
<td>46</td>
</tr>
<tr>
<td>dskc_29</td>
<td>64503</td>
<td>23794</td>
<td>37</td>
<td>13440</td>
<td>6185</td>
<td>46</td>
</tr>
<tr>
<td>dskc_30</td>
<td>64503</td>
<td>23114</td>
<td>37</td>
<td>13440</td>
<td>6461</td>
<td>48</td>
</tr>
<tr>
<td>dskc_07</td>
<td>64503</td>
<td>11723</td>
<td>18</td>
<td>13440</td>
<td>6149</td>
<td>46</td>
</tr>
<tr>
<td>dskc_08</td>
<td>64503</td>
<td>11665</td>
<td>18</td>
<td>13440</td>
<td>6429</td>
<td>48</td>
</tr>
<tr>
<td>dskc_23</td>
<td>64504</td>
<td>7777</td>
<td>15</td>
<td>13440</td>
<td>6094</td>
<td>45</td>
</tr>
<tr>
<td>dskc_24</td>
<td>64504</td>
<td>11805</td>
<td>18</td>
<td>13440</td>
<td>6141</td>
<td>46</td>
</tr>
<tr>
<td>dskc_25</td>
<td>64504</td>
<td>11514</td>
<td>18</td>
<td>13440</td>
<td>7407</td>
<td>55</td>
</tr>
<tr>
<td>dskc_26</td>
<td>64504</td>
<td>12958</td>
<td>20</td>
<td>13440</td>
<td>7149</td>
<td>53</td>
</tr>
<tr>
<td>dskc_06</td>
<td>37089</td>
<td>8053</td>
<td>22</td>
<td>5100</td>
<td>3797</td>
<td>74</td>
</tr>
<tr>
<td>dska_12</td>
<td>37562</td>
<td>6046</td>
<td>16</td>
<td>2735</td>
<td>957</td>
<td>35</td>
</tr>
<tr>
<td>dska_07</td>
<td>37309</td>
<td>6825</td>
<td>18</td>
<td>4000</td>
<td>2107</td>
<td>53</td>
</tr>
<tr>
<td>dska_08</td>
<td>36209</td>
<td>3827</td>
<td>11</td>
<td>7000</td>
<td>491</td>
<td>7</td>
</tr>
<tr>
<td>dska_11</td>
<td>36209</td>
<td>8597</td>
<td>24</td>
<td>7000</td>
<td>4181</td>
<td>60</td>
</tr>
<tr>
<td>dska_16</td>
<td>31263</td>
<td>2892</td>
<td>9</td>
<td>9000</td>
<td>3476</td>
<td>39</td>
</tr>
<tr>
<td>dskb_23</td>
<td>36208</td>
<td>4888</td>
<td>13</td>
<td>7000</td>
<td>4553</td>
<td>7</td>
</tr>
<tr>
<td>dskb_24</td>
<td>36209</td>
<td>3097</td>
<td>9</td>
<td>7000</td>
<td>238</td>
<td>3</td>
</tr>
<tr>
<td>dskb_25</td>
<td>36350</td>
<td>4483</td>
<td>12</td>
<td>7000</td>
<td>223</td>
<td>3</td>
</tr>
</tbody>
</table>

Not To Be Reproduced 14-18
OUTPUT FROM METERING COMMANDS

traffic control queue

- TRAFFIC_CONTROL_QUEUE - DISPLAYS THE CURRENT CONTENTS OF THE SCHEDULER QUEUES, USEFUL FOR GETTING AN IDEA OF WHAT THE USER PROCESSES ARE DOING

I FIRST PART OF OUTPUT IS ELIGIBLE QUEUE:

```
<table>
<thead>
<tr>
<th>flags</th>
<th>du</th>
<th>dpf</th>
<th>temax</th>
<th>te</th>
<th>ts</th>
<th>ti</th>
<th>tssc</th>
<th>event</th>
<th>d</th>
<th>ws</th>
<th>wc</th>
<th>process</th>
</tr>
</thead>
<tbody>
<tr>
<td>rWLE</td>
<td>148</td>
<td>6946</td>
<td>2097</td>
<td>37</td>
<td>0</td>
<td>0</td>
<td>-0.001</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>Initializer</td>
</tr>
<tr>
<td>xLED</td>
<td>15</td>
<td>1111</td>
<td>1000</td>
<td>910</td>
<td>2012</td>
<td>1897</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>6</td>
<td>Sibert</td>
</tr>
<tr>
<td>rLE</td>
<td>14</td>
<td>1370</td>
<td>500</td>
<td>89</td>
<td>0</td>
<td>0</td>
<td>-0.009</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>Diaz</td>
</tr>
<tr>
<td>rLE</td>
<td>15</td>
<td>823</td>
<td>500</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>-0.009</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>8</td>
<td>JCrow</td>
</tr>
<tr>
<td>wLE</td>
<td>13</td>
<td>634</td>
<td>500</td>
<td>422</td>
<td>0</td>
<td>0</td>
<td>0.018</td>
<td>316537</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>Gintelli</td>
</tr>
<tr>
<td>xWLED</td>
<td>6</td>
<td>108</td>
<td>1000</td>
<td>495</td>
<td>2054</td>
<td>2004</td>
<td>0.010</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>6</td>
<td>Brunelle</td>
</tr>
<tr>
<td>wLE</td>
<td>16</td>
<td>864</td>
<td>1000</td>
<td>85</td>
<td>0</td>
<td>510</td>
<td>0.016</td>
<td>504031</td>
<td>0</td>
<td>46</td>
<td>6</td>
<td>FPek</td>
</tr>
<tr>
<td>wWLE</td>
<td>468</td>
<td>2734</td>
<td>1000</td>
<td>315</td>
<td>3010</td>
<td>8000</td>
<td>0.015</td>
<td>224641</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>Spratt</td>
</tr>
<tr>
<td>wLE</td>
<td>17</td>
<td>666</td>
<td>500</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td>0.007</td>
<td>165111</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>FTowle</td>
</tr>
<tr>
<td>wLE</td>
<td>12</td>
<td>520</td>
<td>500</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0.005</td>
<td>710</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>Kress</td>
</tr>
<tr>
<td>wLE</td>
<td>69</td>
<td>1895</td>
<td>500</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
<td>177022</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0PCTL</td>
</tr>
<tr>
<td>xLED</td>
<td>12</td>
<td>372</td>
<td>500</td>
<td>85</td>
<td>0</td>
<td>0</td>
<td>0.005</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>Pandolf</td>
</tr>
<tr>
<td>rLE</td>
<td>67</td>
<td>3279</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.006</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>Lackey</td>
</tr>
</tbody>
</table>
```

Not To Be Reproduced
OUTPUT FROM METERING COMMANDS

traffic control queue

SECOND PART IS REALTIME, INTERACTIVE, AND ALL WORKCLASS QUEUES:

<table>
<thead>
<tr>
<th>WORKCLASS</th>
<th>QUEUE</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>QUEUE</td>
<td>576 ms.</td>
</tr>
<tr>
<td>3</td>
<td>QUEUE</td>
<td>242 ms.</td>
</tr>
<tr>
<td>4</td>
<td>QUEUE</td>
<td>2601 ms.</td>
</tr>
<tr>
<td>5</td>
<td>QUEUE</td>
<td>4000 ms.</td>
</tr>
<tr>
<td>6</td>
<td>QUEUE</td>
<td>-563 ms.</td>
</tr>
<tr>
<td>7</td>
<td>QUEUE</td>
<td>3962 ms.</td>
</tr>
<tr>
<td>8</td>
<td>QUEUE</td>
<td>3934 ms.</td>
</tr>
<tr>
<td>9</td>
<td>QUEUE</td>
<td>2216 ms.</td>
</tr>
<tr>
<td>10</td>
<td>QUEUE</td>
<td>4000 ms.</td>
</tr>
<tr>
<td>11</td>
<td>QUEUE</td>
<td>4000 ms.</td>
</tr>
</tbody>
</table>
OUTPUT FROM METERING COMMANDS

traffic_control_meters

0 TRAFFIC_CONTROL_METERS - DISPLAY THE STATE OF THE SCHEDULER

| OUTPUT COMES IN THREE PARTS, SHOWN OUT OF ORDER HERE: |
| QUEUE LENGTHS AND RESPONSE TIME - THESE ARE WEIGHTED AVERAGES OVER THE LAST FIFTEEN SECONDS. |
| ACTIVITIES VERSUS DEPTH - HOW DEEP THE TRAFFIC CONTROLLER HAD TO SEARCH TO FIND A SCHEDULABLE PROCESS |
| MISCELLANEOUS COUNTERS AND FREQUENCIES OF VARIOUS EVENTS |

Total metering time 0:20:34
Ave queue length 16.52
Ave eligible 13.31
Response time 0.264 sec

<table>
<thead>
<tr>
<th>DEPTH</th>
<th>%PF</th>
<th>TBPF</th>
<th>%GTW</th>
<th>TBS</th>
<th>%CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12.0</td>
<td>22.8</td>
<td>10.8</td>
<td>11.6</td>
<td>8.1</td>
</tr>
<tr>
<td>2</td>
<td>11.8</td>
<td>21.0</td>
<td>9.2</td>
<td>12.2</td>
<td>7.4</td>
</tr>
<tr>
<td>3</td>
<td>10.8</td>
<td>24.9</td>
<td>8.7</td>
<td>14.1</td>
<td>8.0</td>
</tr>
<tr>
<td>4</td>
<td>10.2</td>
<td>27.9</td>
<td>8.5</td>
<td>15.2</td>
<td>8.4</td>
</tr>
<tr>
<td>5</td>
<td>9.5</td>
<td>29.8</td>
<td>8.3</td>
<td>15.5</td>
<td>8.4</td>
</tr>
<tr>
<td>6</td>
<td>8.4</td>
<td>33.5</td>
<td>7.8</td>
<td>16.5</td>
<td>8.4</td>
</tr>
<tr>
<td>7</td>
<td>7.4</td>
<td>36.3</td>
<td>7.3</td>
<td>16.8</td>
<td>8.0</td>
</tr>
<tr>
<td>8</td>
<td>30.0</td>
<td>48.6</td>
<td>39.5</td>
<td>16.8</td>
<td>43.3</td>
</tr>
</tbody>
</table>
## OUTPUT FROM METERING COMMANDS

**traffic control meters**

<table>
<thead>
<tr>
<th>COUNTER</th>
<th>TOTAL</th>
<th>ATB</th>
<th>#/INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactions</td>
<td>7977</td>
<td>0.155 sec</td>
<td>1.525</td>
</tr>
<tr>
<td>Loadings</td>
<td>12161</td>
<td>0.102 sec</td>
<td></td>
</tr>
<tr>
<td>Blocks</td>
<td>14082</td>
<td>0.088 sec</td>
<td></td>
</tr>
<tr>
<td>Wakeups</td>
<td>36078</td>
<td>0.034 sec</td>
<td></td>
</tr>
<tr>
<td>Schedulings</td>
<td>12591</td>
<td>0.098 sec</td>
<td>1.578</td>
</tr>
<tr>
<td>Lost priority</td>
<td>1</td>
<td>1234.756 sec</td>
<td></td>
</tr>
<tr>
<td>Priority boosts</td>
<td>0</td>
<td>0.000 sec</td>
<td></td>
</tr>
<tr>
<td>I/O boosts</td>
<td>578</td>
<td>2.136 sec</td>
<td></td>
</tr>
<tr>
<td>Wait Page</td>
<td>127040</td>
<td>9.719 msec</td>
<td>15.926</td>
</tr>
<tr>
<td>Wait PTL</td>
<td>75691</td>
<td>16.313 msec</td>
<td>9.489</td>
</tr>
<tr>
<td>Wait Other</td>
<td>31912</td>
<td>38.693 msec</td>
<td>4.001</td>
</tr>
<tr>
<td>Total Waits</td>
<td>234643</td>
<td>5.262 msec</td>
<td>29.415</td>
</tr>
<tr>
<td>Notify Page</td>
<td>128954</td>
<td>9.575 msec</td>
<td></td>
</tr>
<tr>
<td>Notify PTL</td>
<td>75691</td>
<td>16.313 msec</td>
<td></td>
</tr>
<tr>
<td>Notify Other</td>
<td>25330</td>
<td>48.747 msec</td>
<td></td>
</tr>
<tr>
<td>Total Notifies</td>
<td>229975</td>
<td>5.369 msec</td>
<td></td>
</tr>
<tr>
<td>Get Processor</td>
<td>245856</td>
<td>5.022 msec</td>
<td></td>
</tr>
<tr>
<td>Pre-empts</td>
<td>94235</td>
<td>13.103 msec</td>
<td>11.813</td>
</tr>
<tr>
<td>Getwork</td>
<td>338802</td>
<td>3.644 msec</td>
<td></td>
</tr>
<tr>
<td>Retry getwork</td>
<td>4986</td>
<td>0.248 sec</td>
<td></td>
</tr>
<tr>
<td>Extra notifies</td>
<td>2949</td>
<td>0.419 sec</td>
<td></td>
</tr>
<tr>
<td>Last EN event</td>
<td>000000000071</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last NTO event</td>
<td>033022237767</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ALARM CLOCK METERS - DISPLAYS INFORMATION ABOUT THE USER ALARM TIMER FACILITY (HARDCORE INTERFACE FOR timer_manager_)

Total metering time 0:20:31

No. alarm clock sims. 2171
Simulation lag 5.245 msecs.
Max. lag 1.7340314e4 msecs.
OUTPUT FROM METERING COMMANDS

print tuning parameters

PRINT_TUNING_PARAMETERS - PRINT VALUES FOR SYSTEM CONTROL PARAMETERS. MOST CONTROL THE SCHEDULER

Current system tuning parameters:

tefirst: 0.5 seconds
telast: 1.0 seconds
timax: 8.0 seconds
priority_sched_inc: 80.0 seconds
min_eligIble: 2.
max_eligible: 20.
max_batch_elig: 0
working_set_factor: 0.5
working_set_addend: 0
deadline_mode: off
int_q_enabled: on
post_purge: on
pre empt_sample_time: 0.04 seconds
gp_at_notify: off
gp_at_pnlnotify: off
process_initial_quantum: 2.0 seconds
quit_priority: 0
sv_integration: 4.0 seconds
realtime_io_priority: on
realtime_io_deadline: 0.0 seconds
realtime_io_quantum: 0.005 seconds
notify_timeout_interval: 30.0 seconds
notify_timeout_severity: 0
write_limit: 724

THESE ARE "INTERNAL", NORMALLY NEVER CHANGED, AND ONLY PRINTED IF THE -all CONTROL ARGUMENT IS GIVEN

stack truncation: on
stack truncation always: off
stk trunc block avg factor: 0.25
trap invalid masked: off
meter ast locking: off
checksum filemap: on
**OUTPUT FROM METERING COMMANDS**

*work class meters*

WORK_CLASS_METERS - DISPLAY THE VARIOUS WORKCLASS PARAMETERS, AND SHOW WHAT RESOURCES EACH WORKCLASS IS CONSUMING.

<table>
<thead>
<tr>
<th>WC</th>
<th>%GUAR</th>
<th>%MAX</th>
<th>%TCP</th>
<th>V/ELIG</th>
<th>PW</th>
<th>IRESP</th>
<th>IQUANT</th>
<th>RESP QUANT</th>
<th>P</th>
<th>M</th>
<th>R</th>
<th>LCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>0.12</td>
<td>3</td>
<td>0.26</td>
<td>2.10</td>
<td>0.26</td>
<td>2.10</td>
<td>P</td>
<td>O</td>
<td>R</td>
<td>1</td>
<td>Wait</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.09</td>
<td>1</td>
<td>0.25</td>
<td>0.75</td>
<td>0.50</td>
<td>1.00</td>
<td>P</td>
<td>O</td>
<td>R</td>
<td>1</td>
<td>RTime</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1.44</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>System SysAdm OPR FE</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>0.49</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>SysProgr SysDev</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>0.49</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>SEngr</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>0.46</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>HEngr</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>0.25</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>MktUS MktFor MktEd</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0.36</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>DS-CC</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>0.18</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>OffAuto</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>0.62</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>Misc Mfg</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0.55</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>0.16</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>O</td>
<td></td>
<td></td>
<td>Special</td>
</tr>
</tbody>
</table>

TCPU percents (%GUAR) control non-realtime work_classes.
**OUTPUT FROM METERING COMMANDS**

```plaintext
RESPONSE_METERS - DISPLAYS RESPONSE TIME, BASED ON TERMINAL INTERACTIONS, ON A PER-WORKCLASS BASIS
```

Total metering time: 0:20:36

<table>
<thead>
<tr>
<th>WC ---Thinks---</th>
<th>---Queues---</th>
<th>VCPU Range-</th>
<th># Avg</th>
<th>Avg From To</th>
<th>Int</th>
<th>Avg VCPU</th>
<th>RT</th>
<th>Resp Fact</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>86 2.70</td>
<td>0.00 0.50</td>
<td>113</td>
<td>0.04 0.42</td>
<td>9.34 Init</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>0.15</td>
<td>0.50 1.00</td>
<td>3</td>
<td>0.55 5.51</td>
<td>10.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>3</td>
<td>2.43 14.96</td>
<td>6.15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>---- -----</td>
<td>119</td>
<td>0.12 0.91</td>
<td>7.76</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>35 11.78</td>
<td>0.00 0.50</td>
<td>34</td>
<td>0.11 0.96</td>
<td>8.74 RTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>0.21</td>
<td>0.50 1.00</td>
<td>2</td>
<td>0.83 4.06</td>
<td>4.87</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>---- -----</td>
<td>36</td>
<td>0.15 1.15</td>
<td>7.55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>593 14.90</td>
<td>0.00 0.50</td>
<td>620</td>
<td>0.05 0.49</td>
<td>10.50 System SysAdm GPR FED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>612</td>
<td>0.15</td>
<td>0.50 1.00</td>
<td>28</td>
<td>0.71 3.61</td>
<td>5.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>39</td>
<td>1.77 8.39</td>
<td>4.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>---- -----</td>
<td>687</td>
<td>0.17 1.07</td>
<td>6.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2496 6.38</td>
<td>0.00 0.50</td>
<td>2993</td>
<td>0.08 0.66</td>
<td>7.96 SysProg SysDev</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2622</td>
<td>0.17</td>
<td>0.50 1.00</td>
<td>117</td>
<td>0.68 4.22</td>
<td>6.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>66</td>
<td>2.46 13.47</td>
<td>5.49</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.00 99.99</td>
<td>10</td>
<td>56.85 99.99</td>
<td>3.33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>---- -----</td>
<td>3186</td>
<td>0.33 1.65</td>
<td>4.96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>581 15.82</td>
<td>0.00 0.50</td>
<td>663</td>
<td>0.05 0.71</td>
<td>12.94 SEngr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>590</td>
<td>0.26</td>
<td>0.50 1.00</td>
<td>13</td>
<td>0.64 4.64</td>
<td>7.30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>8</td>
<td>3.47 32.07</td>
<td>9.24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>---- -----</td>
<td>684</td>
<td>0.11 1.15</td>
<td>10.88</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>133 29.51</td>
<td>0.00 0.50</td>
<td>148</td>
<td>0.06 0.83</td>
<td>12.95 HEngr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>0.17</td>
<td>0.50 1.00</td>
<td>3</td>
<td>0.69 4.64</td>
<td>6.70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>5</td>
<td>3.15 10.24</td>
<td>3.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.00 99.99</td>
<td>2</td>
<td>26.69 48.84</td>
<td>1.83</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>---- -----</td>
<td>158</td>
<td>0.51 1.81</td>
<td>3.55</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Not To Be Reproduced 14-25 F804*
## OUTPUT FROM METERING COMMANDS

**resons meters**

<table>
<thead>
<tr>
<th>6</th>
<th>1180 11.65</th>
<th>0.00 0.50</th>
<th>977 0.05 1.13 20.91</th>
<th>MktUS MktFor MktEd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1211 0.58</td>
<td>0.50 1.00</td>
<td>24 0.64 4.95 7.74</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>16 1.69 11.88 6.29</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1017 0.10 1.39 14.38</td>
</tr>
<tr>
<td>7</td>
<td>259 14.65</td>
<td>0.00 0.50</td>
<td>292 0.08 1.53 20.17</td>
<td>DS-CC</td>
</tr>
<tr>
<td></td>
<td>287 0.98</td>
<td>0.50 1.00</td>
<td>9 0.71 9.03 12.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>17 2.04 14.76 7.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.00 99.99</td>
<td>1 12.22 99.99 9.45</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>319 0.24 2.81 11.86</td>
</tr>
<tr>
<td>8</td>
<td>73 2.69</td>
<td>0.00 0.50</td>
<td>79 0.06 0.31 5.45</td>
<td>OffAuto</td>
</tr>
<tr>
<td></td>
<td>74 0.05</td>
<td>0.50 1.00</td>
<td>2 0.60 6.40 10.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>1 3.20 6.73 2.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>82 0.11 0.54 4.94</td>
</tr>
<tr>
<td>9</td>
<td>94 41.91</td>
<td>0.00 0.50</td>
<td>80 0.11 1.23 11.07</td>
<td>Misc Mfg</td>
</tr>
<tr>
<td></td>
<td>96 0.30</td>
<td>0.50 1.00</td>
<td>11 0.69 4.74 6.91</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>13 2.99 12.82 4.30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>104 0.53 3.05 5.74</td>
</tr>
<tr>
<td>10</td>
<td>7 99.99</td>
<td>0.00 0.50</td>
<td>10 0.11 1.15 13.82</td>
<td>Other</td>
</tr>
<tr>
<td></td>
<td>8 0.21</td>
<td>1.00 10.00</td>
<td>1 3.88 39.11 10.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11 0.45 4.88 10.89</td>
</tr>
<tr>
<td>11</td>
<td>417 12.71</td>
<td>0.00 0.50</td>
<td>445 0.03 0.50 15.32</td>
<td>Special</td>
</tr>
<tr>
<td></td>
<td>420 0.16</td>
<td>0.50 1.00</td>
<td>5 0.57 3.61 6.29</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>4 1.65 11.40 6.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>454 0.05 0.63 11.93</td>
</tr>
<tr>
<td>All</td>
<td>5954 11.13</td>
<td>0.00 0.50</td>
<td>6454 0.07 0.77 11.40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6192 0.29</td>
<td>0.50 1.00</td>
<td>217 0.68 4.54 6.70</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.00 10.00</td>
<td>173 2.31 13.11 5.68</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.00 99.99</td>
<td>13 48.77 99.99 3.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6857 0.24 1.51 6.39</td>
</tr>
</tbody>
</table>

86797 calls to meter_response_time 283 invalid transitions.

Overhead = 0.09% ( 0.052 ms./call)
SYSTEM PERFORMANCE GRAPH - GIVES A ROUGH SNAPSHOT OF SYSTEM ACTIVITY, SAMPLED PERIODICALLY

up= 04/01/83 0855.6 est, sys_hours=26.3, cpu_hours=78.9

cpu=3, pages=3419, min_e=3, max_e=16, wsa=0, wsf=0.50,
tefirst=0.13, telast=-0.50, timax=32.0

Not To Be Reproduced 14-27
OUTPUT FROM METERING COMMANDS

system performance graph

SOME CHARACTERS INDICATES PERCENTAGES OF SYSTEM TIME SPENT IN VARIOUS ACTIVITIES. OTHERS INDICATE ACTUAL VALUES

PERCENTAGES

BLANK - USER PROCESSING. RING ZERO TIME BETWEEN "y" AND THE RIGHT MARGIN, USER RING BETWEEN "s" and "y"

SYSTEM SERVICES - SEGMENT FAULTS ("s"), PAGE FAULTS ("p"), TRAFFIC CONTROL ("t"), INTERRUPTS ("i")

IDLE TIME - MP IDLE ("m"), NMP IDLE ("*"), ZERO IDLE (BLANKS ON THE LEFT)

OTHER VALUES (RELATIVE TO THE LEFT MARGIN)

TRAFFIC CONTROL QUEUE LENGTHS - READY QUEUE ("q"), ELIGIBLE QUEUE ("e")

USER COUNTS - NUMBER OF USERS ("+"), LOAD UNITS ("-"

TRAFFIC CONTROL VALUES - RESPONSE TIME ("r"), QUITS PER MINUTE ("Q"), SCHEDULINGS IN TEN SECONDS ("S")

OTHER VALUES (RELATIVE TO THE RIGHT MARGIN)

I/O TRAFFIC - DISK I/O PER 100 MILLISECONDS ("D"), VTOC I/O PER 100 MILLISECONDS ("V")
OUTPUT FROM METERING COMMANDS

meter_gate

METER_GATE - SHOWS TIME SPENT CALLING THROUGH SUPERVISOR GATES (HCS_ ETC.)

Metering since 04/01/83 0855.6 est Fri.
Total non-idle time at 04/02/83 1101.9 est Sat = 39 hr. 36 min. 57 sec.

Gate meters for hcs_:

<table>
<thead>
<tr>
<th>Calls</th>
<th>Pcnt</th>
<th>Avg</th>
<th>Pwait</th>
<th>Entry Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>902768</td>
<td>4.50</td>
<td>7.11</td>
<td>0.02</td>
<td>tty_write</td>
</tr>
<tr>
<td>13857</td>
<td>2.29</td>
<td>336.01</td>
<td>2.28</td>
<td>list_dir</td>
</tr>
<tr>
<td>442912</td>
<td>1.19</td>
<td>3.85</td>
<td>0.01</td>
<td>tty_read</td>
</tr>
<tr>
<td>435510</td>
<td>1.01</td>
<td>3.30</td>
<td>0.02</td>
<td>tty_order</td>
</tr>
<tr>
<td>165154</td>
<td>0.97</td>
<td>8.35</td>
<td>0.10</td>
<td>initiate</td>
</tr>
<tr>
<td>120204</td>
<td>0.83</td>
<td>9.84</td>
<td>0.76</td>
<td>make_ptr</td>
</tr>
<tr>
<td>20620</td>
<td>0.51</td>
<td>34.95</td>
<td>0.48</td>
<td>make_seg</td>
</tr>
<tr>
<td>60796</td>
<td>0.50</td>
<td>11.69</td>
<td>0.14</td>
<td>truncate_seg</td>
</tr>
<tr>
<td>45464</td>
<td>0.45</td>
<td>14.25</td>
<td>0.16</td>
<td>status_long</td>
</tr>
<tr>
<td>337615</td>
<td>0.40</td>
<td>1.69</td>
<td>0.01</td>
<td>set_alarm_timer</td>
</tr>
<tr>
<td>532081</td>
<td>0.35</td>
<td>0.95</td>
<td>0.00</td>
<td>read_events</td>
</tr>
<tr>
<td>4085</td>
<td>0.35</td>
<td>122.90</td>
<td>3.57</td>
<td>star_dir_list</td>
</tr>
<tr>
<td>42080</td>
<td>0.33</td>
<td>11.17</td>
<td>0.02</td>
<td>list_acl</td>
</tr>
<tr>
<td>59334</td>
<td>0.32</td>
<td>7.76</td>
<td>0.01</td>
<td>status_for_backup</td>
</tr>
<tr>
<td>103029</td>
<td>0.32</td>
<td>4.44</td>
<td>0.03</td>
<td>tty_get_line</td>
</tr>
</tbody>
</table>
| 61851 | 0.32 | 7.33| 0.03  | tty_write_wholet_str#
| 48621 | 0.31 | 9.05| 0.35  | status_mins        |
| 31637 | 0.26 | 11.64| 0.04  | set_max_length_seg |
| 150703| 0.26 | 2.41| 0.03  | terminate_noname   |
| 67267 | 0.20 | 4.19| 0.03  | tty_read_echoed    |
| 14851 | 0.20 | 18.77| 0.08  | quota_get          |
| 21051 | 0.18 | 12.48| 0.72  | get_link_target    |
| 30337 | 0.18 | 8.25| 0.40  | initiate_count     |
| 6288  | 0.17 | 39.36| 0.25  | delentry_seg       |
| 247420| 0.16 | 0.95| 0.03  | wakeup             |
| 14024 | 0.16 | 15.99| 0.43  | status             |
| 476759| 0.16 | 0.47| 0.00  | level_set          |
| 13601 | 0.13 | 13.69| 0.00  | dir_quota_read     |
| 14860 | 0.13 | 12.05| 0.02  | list_dir_ac1       |
| 13496 | 0.13 | 13.22| 0.01  | get_max_length     |
| 28194 | 0.12 | 6.15| 0.41  | make_entry         |
| 23725 | 0.11 | 6.65| 0.09  | set_bc_seg         |
| 1416  | 0.11 | 108.00| 2.08  | star               |
| 13961 | 0.10 | 10.68| 0.40  | get_access_class    |
| 28696 | 0.10 | 5.03 | 0.04  | status_mins        |
| 13553 | 0.09 | 9.76| 0.02  | list_inacli_all    |

Not To Be Reproduced 14-29 F80A
OUTPUT FROM METERING COMMANDS

meter gate

<table>
<thead>
<tr>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>12881</td>
<td>0.08</td>
<td>8.50</td>
<td>0.39 get_user_effmode</td>
</tr>
<tr>
<td>46180</td>
<td>0.08</td>
<td>2.33</td>
<td>0.01 tty_read_with_mark</td>
</tr>
<tr>
<td>206</td>
<td>0.07</td>
<td>493.77</td>
<td>3.79 star_list</td>
</tr>
<tr>
<td>1109005</td>
<td>0.07</td>
<td>0.09</td>
<td>0.00 level_get</td>
</tr>
</tbody>
</table>
OUTPUT FROM METERING COMMANDS

meter gate

I HPHCS_ AND IOI_ ARE THE OTHER SUPERVISOR GATES THAT CONSUME
SIGNIFICANT RESOURCES

Metering since 04/01/83 0855.6 est Fri.
Total non-idle time at 04/02/83 1102.7 est Sat = 39 hr. 37 min. 59 sec.
Gate meters for hphcs_: total calls = 70604.
0 hr. 29 min. 51 sec. or 1.256% spent in calls through gate.

calls pcnt avg pwait entry name
869 0.57 928.77 5.40 destroy_process_finish
911 0.24 381.04 15.09 create_proc
5944 0.20 48.34 1.04 dir_quota_read
8326 0.05 8.47 0.04 tty_write_force
5946 0.05 11.31 0.00 quota_read
105 0.04 500.68 0.00 flush_core
246 0.04 208.83 0.17 flush_ast_pool
1735 0.02 16.94 0.76 star_
5595 0.02 4.12 0.00 set_backup_dump_time
3193 0.01 0.60 0.00 set_kst_attributes
1 0.01 14598.71 8.00 adk_scu

Metering since 04/01/83 0855.6 est Fri.
Total non-idle time at 04/02/83 1103.0 est Sat = 39 hr. 38 min. 22 sec.
Gate meters for ioi_: total calls = 245985.
0 hr. 12 min. 14 sec. or 0.515% spent in calls through gate.

calls pcnt avg pwait entry name
243976 0.50 2.94 0.00 connect
248 0.01 65.91 2.09 workspace
793 0.00 0.58 0.00 get_special_status
533 0.00 0.54 0.00 set_event
322 0.00 0.59 0.14 set_status
105 0.00 0.55 0.00 timeout
3 0.00 4.38 0.00 set_channel_required
5 0.00 0.65 0.00 get_detailed_status

Not To Be Reproduced 14-31 (End Of Topic)
## TOPIC XV

Evolution of Memory Addressing/Management

<table>
<thead>
<tr>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional Memory</td>
<td>15-1</td>
</tr>
<tr>
<td>Structure</td>
<td>15-1</td>
</tr>
<tr>
<td>Address Formation</td>
<td>15-1</td>
</tr>
<tr>
<td>Characteristics</td>
<td>15-1</td>
</tr>
<tr>
<td>Problems</td>
<td>15-4</td>
</tr>
<tr>
<td>Single Virtual Memory</td>
<td>15-7</td>
</tr>
<tr>
<td>Structure</td>
<td>15-7</td>
</tr>
<tr>
<td>Address Formation</td>
<td>15-7</td>
</tr>
<tr>
<td>Characteristics</td>
<td>15-7</td>
</tr>
<tr>
<td>Solved Problems</td>
<td>15-9</td>
</tr>
<tr>
<td>Problems</td>
<td>15-10</td>
</tr>
<tr>
<td>Multiple Virtual Memories</td>
<td>15-11</td>
</tr>
<tr>
<td>Structure</td>
<td>15-11</td>
</tr>
<tr>
<td>Address Formation</td>
<td>15-11</td>
</tr>
<tr>
<td>Characteristics</td>
<td>15-11</td>
</tr>
<tr>
<td>Solved Problems</td>
<td>15-13</td>
</tr>
<tr>
<td>Problems</td>
<td>15-14</td>
</tr>
<tr>
<td>Multics Virtual Memory</td>
<td>15-15</td>
</tr>
<tr>
<td>Structure</td>
<td>15-15</td>
</tr>
<tr>
<td>Address Formation</td>
<td>15-15</td>
</tr>
<tr>
<td>Characteristics</td>
<td>15-15</td>
</tr>
<tr>
<td>Solved Problems</td>
<td>15-17</td>
</tr>
<tr>
<td>Problems</td>
<td>15-19</td>
</tr>
</tbody>
</table>
CONVENTIONAL MEMORY

STRUCTURE

1-DIMENSIONAL ADDRESS SPACE USED BY THE SYSTEM
CONVENTIONAL MEMORY
CHARACTERISTICS

SPACE DIVIDED INTO REGIONS OF VARIOUS SIZES, ONE REGION PER
PROCESS/JOB/USER

REGIONS ARE:

- PROTECTED FROM ONE ANOTHER BY A BAR OR BY PROTECT KEYS

- SUBDIVIDED INTO POOLS OF STORAGE USED FOR: PROGRAMS, I/O
  BUFFERS, MEMORY ALLOCATION AREA, AUTOMATIC VARIABLES, STATIC
  VARIABLES

1-DIMENSIONAL ADDRESS SPACE WIRED DIRECTLY ONTO REAL MEMORY

PROGRAMS MUST BE LOADED INTO REAL MEMORY

- MEMORY ALLOCATION FOR REGION

- PROGRAMS AND BUFFERS ALLOCATED WITHIN REGION

- PREPARATORY ADDRESS MODIFICATION (ADDRESSES MUST BE MODIFIED TO
  REFLECT LOCATION WITHIN THE REGION)

Not To Be Reproduced 15-2
CONVENTIONAL MEMORY

CHARACTERISTICS

- LINKAGE EDITING REQUIRED

- ALL SYMBOLIC REFERENCES MUST BE RESOLVED - IMPLYING THAT ALL REFERENCED PROGRAMS MUST BE LOADED REGARDLESS OF WHETHER OR NOT THEY ARE ACTUALLY NEEDED AT RUN TIME

- EXAMPLES: GCOS, IBM OS/MFT, IBM OS/MVT
CONVENTIONAL MEMORY

PROBLEMS

- SYSTEM ADDRESS SPACE LIMITED TO SIZE OF REAL MEMORY

- USER ADDRESS SPACES (REGIONS) ARE SMALL. THIS MEANS:
  - PROGRAMS MUST BE WRITTEN TO FIT INTO SMALL REGIONS
  - SMALL REGION PROGRAMS OFTEN OPERATE LESS EFFICIENTLY IN THEIR USE OF CPU TIME THAN A SIMILAR PROGRAM DESIGNED FOR A LARGE REGION
  - BECAUSE SMALL REGION OPERATION IS PROGRAMMED IN, SUCH PROGRAMS CANNOT TAKE ADVANTAGE OF MORE MEMORY WHEN IT CAN BE MADE AVAILABLE
  - PROGRAMMER MUST WASTE INGENUITY (AND SYSTEM RESOURCES) TO MAKE PROGRAMS RUN IN A SMALL REGION:
    - WRITING OVERLAY PROGRAMS
    - DIVIDING REGION INTO OPTIONAL-SIZE POOLS
    - PROGRAMMING MECHANISMS TO EXTEND OR SWAP-OUT POOLS WHEN THEY OVERFLOW
  - PROGRAMMER TIME WASTED WHEN PROGRAMS MUST BE CONVERTED TO TAKE ADVANTAGE OF LARGER REGIONS WHEN CONFIGURATION IS INCREASED
CONVENTIONAL MEMORY

PROBLEMS

- REAL MEMORY IS USED INEFFECTIVELY
  - WITHIN A REGION, PROGRAM COMPONENTS OR DATA AREAS NOT REFERENCED STILL OCCUPY REAL MEMORY
  - UNUSED SPACE BETWEEN REGIONS WASTES REAL MEMORY (FRAGMENTATION)

- SCHEMES FOR USING REAL MEMORY MORE EFFECTIVELY ARE COSTLY
  - CPU COST OF MOVING REGIONS TO REDUCE FRAGMENTATION
  - CPU AND I/O COSTS TO SWAP OUT ENTIRE REGIONS TO SHARE REAL MEMORY AMONG MORE USERS

- NO PROTECTION OF DATA WITHIN A REGION
  - PROGRAMMING ERRORS CAN CAUSE UNWANTED WRITING INTO PROGRAM OR DATA AREAS
CONVENTIONAL MEMORY

PROBLEMS

- OVERHEAD AND INCONVENIENCE OF LOADING AND LINKAGE EDITING

- NO SHARING OF PROGRAMS AND DATA BETWEEN REGIONS

  | EACH REGION MUST CONTAIN A COPY OF SHARED DATA (INEFFICIENT USE
  | OF MEMORY)

  | MODIFICATIONS TO SHARED DATA CANNOT EASILY BE REFLECTED IN ALL
  | COPIES

- PHYSICAL INPUT/OUTPUT OPERATIONS ON DISK FILES OFTEN BECOME THE
  RESPONSIBILITY OF THE PROGRAMMER (REDUCING PRODUCTIVITY)
SINGLE VIRTUAL MEMORY
STRUCTURE

ßen A LARGE ADDRESS SPACE USED BY THE SYSTEM (EG, 4M WORDS)

ßen THE ADDRESS SPACE IS LOGICALLY DIVIDED INTO REGIONS OF VARIOUS SIZES, AS IN CONVENTIONAL MEMORY

ßen THE VIRTUAL MEMORY IS PHYSICALLY DIVIDED INTO PAGES HAVING A FIXED SIZE

ßen THE ADDRESS SPACE AND THE VIRTUAL MEMORY HAVE THE SAME SIZE

ßen THE 1-DIMENSIONAL ADDRESS SPACE IS MAPPED DIRECTLY ONTO THE PAGED, 1-DIMENSIONAL VIRTUAL MEMORY

ßen THE VIRTUAL MEMORY IS MAPPED BY A PAGING ALGORITHM ONTO A SMALLER (EG, 256K WORDS) REAL MEMORY

ßen PROGRAMS MUST BE LOADED INTO THE VIRTUAL MEMORY AS IN CONVENTIONAL MEMORY SYSTEMS
SINGLE VIRTUAL MEMORY
CHARACTERISTICS

- LINKAGE EDITING REQUIRED AS IN CONVENTIONAL MEMORY SYSTEMS

- EXAMPLES:

  IBM OS/VS-1 OR OS/VS-2 RELEASE 1
SINGLE VIRTUAL MEMORY

SOLVED PROBLEMS

1. THE SYSTEM'S ADDRESS SPACE IS MUCH LARGER THAN REAL MEMORY

   1. THE SYSTEM CAN RUN MORE AND/OR LARGER USER REGIONS

2. THE USER REGIONS CAN BE LARGER

   1. PROGRAMS CAN BE WRITTEN FOR A LARGE REGION TO TAKE ADVANTAGE OF ADDITIONAL MEMORY WHEN IT IS AVAILABLE

3. PROGRAMMER PRODUCTIVITY IMPROVES

   1. PROGRAMMERS WORRY LESS ABOUT OPTIMIZING MEMORY USAGE

   1. CONVERTING PROGRAMS TO USE LARGER MEMORY CONFIGURATIONS OFTEN UNNECESSARY - LESS COMPETITION

4. REAL MEMORY USED MORE EFFICIENTLY

   1. FEWER UNREFERENCED AREAS OF ADDRESS SPACE OCCUPY REAL MEMORY

   1. PAGING ALGORITHM SIMPLIFIES MEMORY MANAGEMENT SCHEMES
SINGLE VIRTUAL MEMORY

PROBLEMS

- Memory swapping may still be necessary to share the address space among many users.

- User regions still too small to handle every application. Some memory management still required.

- No protection of data within a region.

- Overhead and inconvenience of loading and linkage editing.

- No sharing of data between regions.

- Explicit disk I/O still required to access files.

- The advantages of solving the problems above may be outweighed by costs in hardware and software of the paging overhead.
MULTIPLE VIRTUAL MEMORIES

STRUCTURE

THE SYSTEM USES MANY LARGE (4M WORD), 1-DIMENSIONAL ADDRESS SPACES, ONE PER USER REGION

EACH ADDRESS SPACE CONTAINS THE SUPERVISOR PROGRAMS PLUS ONE LARGE USER REGION (DIVIDED INTO POOLS), PLUS PROGRAMS AND DATA SHARED AMONG ALL REGIONS

EACH ADDRESS SPACE IS MAPPED DIRECTLY ONTO ITS OWN, PAGED VIRTUAL MEMORY

EACH VIRTUAL MEMORY IS MAPPED BY A PAGING ALGORITHM ONTO THE SINGLE, SMALLER REAL MEMORY

PROGRAMS MUST BE LOADED INTO THE VIRTUAL MEMORY AS IN CONVENTIONAL MEMORY SYSTEMS

LINKAGE EDITING REQUIRED AS IN CONVENTIONAL MEMORY SYSTEMS
MULTIPLE VIRTUAL MEMORIES
CHARACTERISTICS

EXAMPLES:

IBM OS/VS-2 RELEASE 2 (MVS)
MULTIPLE VIRTUAL MEMORIES

SOLVED PROBLEMS

• The large user regions can handle all but the largest programs without special memory management. Programmer must still divide regions into pools, however, and sometimes provide pool overflow mechanisms.

• Memory swapping is now unnecessary. Previously swapped regions now occupy their own address spaces and are paged in and out.

• Programs and data (usually read-only) can be shared between regions in a limited way by occupying the same pool in every address space.

• Newly solved problems make paging overhead more worthwhile.
MULTIPLE VIRTUAL MEMORIES
PROBLEMS

NO PROTECTION OF DATA WITHIN THE MAJORITY OF A USER'S ADDRESS SPACE (REGION)

OVERHEAD AND INCONVENIENCE OF LOADING AND LINKAGE EDITING

GENERAL SHARING OF READ-WRITE DATA STILL NOT POSSIBLE

EXPlicit DISK I/O STILL REQUIRED TO ACCESS FILES
MULTICS VIRTUAL MEMORY

STRUCTURE

* THE SYSTEM USES MANY, VERY LARGE (EG, 256M WORDS), 2-DIMENSIONAL ADDRESS SPACES, ONE PER USER PROCESS (REGION)

* EACH ADDRESS SPACE IS DIVIDED INTO SEGMENTS WHICH PERFORM THE SAME FUNCTION AS POOLS IN MULTIPLE VIRTUAL MEMORY SYSTEMS

* SEGMENTS:
  1. HAVE VARYING SIZES
  2. ARE EXTENDABLE
  3. ARE FILES IN THE MULTICS STORAGE SYSTEM
  4. ARE ACCESSED AS READ-WRITE, READ-ONLY, EXECUTABLE OR CALLABLE DATA, WITH ACCESS CONTROLLED BY AN ACL, RING BRACKETS AND AN AIM CLASSIFICATION
  5. ARE SHARED AMONG ADDRESS SPACES, WITH EACH ADDRESS SPACE HAVING ITS OWN PERMISSION TO ACCESS THE SEGMENT
MULTICS VIRTUAL MEMORY

CHARACTERISTICS

* Each address space intersects in varying degrees with every other address space

* Each address space is mapped directly onto its own paged, segmented virtual memory of the same size

* Each virtual memory is mapped by a paging algorithm onto the single, smaller real memory

* No loading is required since all addresses are interpreted as offsets within segments

* No linkage editing is required since all symbolic references are resolved at run time if and when they are encountered (dynamic linking)
### MULTICS VIRTUAL MEMORY

#### SOLVED PROBLEMS

- **DATA WITHIN THE ADDRESS SPACE PROTECTED**
  - READ OR READ-EXECUTE DATA STORED IN SEPARATE SEGMENTS WHICH ARE PROTECTED FROM MODIFICATION
  - PROGRAMMING ERRORS REFERENCING OUTSIDE ARRAY BOUNDS CANNOT REFERENCE DATA IN ANOTHER SEGMENT

- **DATA CAN BE SHARED IN A GENERAL WAY BETWEEN ADDRESS SPACES**
  - EACH SEGMENT (NOT A COPY OF THE SEGMENT) CAN APPEAR IN SEVERAL ADDRESS SPACES
  - DIFFERENT PROCESSES CAN HAVE DIFFERENT ACCESS TO THE SAME SEGMENT IN THEIR ADDRESS SPACE

- **OVERHEAD AND INCONVENIENCE OF LOADING (SOFTWARE) REPLACED BY AN ADDRESS FORMATION SCHEME (HARDWARE)**
  - UNREFERENCED PROGRAMS (AND/OR PAGES OF PROGRAMS) DO NOT REQUIRE MAIN MEMORY SPACE
MULTICS VIRTUAL MEMORY

SOLVED PROBLEMS

PROGRAMS NEVER REQUIRE PREPARATORY ADDRESS MODIFICATION

EXPLICIT I/O IS NOT REQUIRED TO ACCESS FILES (SEGMENTS)

SEGMENTS CAN BE ACCESSED BY MAKING THEM KNOWN TO THE ADDRESS SPACE AND REFERENCING THE SEGMENT

THIS IS CALLED VIRTUAL FILE I/O

OVERHEAD AND INCONVENIENCE OF LINKAGE EDITING REPLACED BY DYNAMIC LINKING

UNREFERENCED PROGRAMS DO NOT REQUIRE LINKING

ADVANTAGES OF VERY LARGE ADDRESS SPACE, SHARED FILES, DATA PROTECTION, AND VIRTUAL I/O DEFINITELY OUTWEIGH THE COSTS OF PAGING AND SEGMENTATION OVERHEAD
MULTICS VIRTUAL MEMORY

PROBLEMS

• MULTICS FILE I/O IS OFTEN LESS EFFICIENT THAN SPECIAL CASED METHODS BASED ON KNOWN ACCESS PATTERNS

• THE SYSTEM'S SEGMENT SIZE DOES NOT GENERALIZE UPWARD TO HANDLE VERY LARGE DATA BASES. ALTERNATE (KLUDGY) METHODS MUST BE USED (SUCH AS MSF'S)

• THE OVERHEAD TO TOUCH A PAGE OF ONE HUNDRED DIFFERENT SEGMENTS IS CONSIDERABLY MORE THAN THE OVERHEAD TO TOUCH ONE HUNDRED PAGES OF THE SAME SEGMENT. (IE: SPARSE AND INFREQUENT ACCESSING IS EXPENSIVE)

• LACK OF EXPLICIT I/O CONTROL MAKES OVERALL SYSTEM RELIABILITY SUFFER SINCE FULL RECOVERY FROM A SYSTEM CRASH REQUIRES THE SUCCESSFUL FLUSHING OF ALL PAGES FROM MAIN MEMORY

hcs_sforce_write MAY BE USED BY THOSE APPLICATIONS REQUIRING SUCH RELIABILITY

Not To Be Reproduced 15-19
(End Of Topic)
Printout of the 59 Entries
of the
Libraries
Include.*, hard.source
Which Match the Search Names
add_type.incl.pl1, aim_template.incl.pl1, apte.incl.pl1, aste.incl.pl1, bos_dump.incl.pl1, cmp.incl.pl1, dbm.incl.pl1,
dir_ecl.incl.pl1, dir_allocation_area.incl.pl1, dir_entry.incl.pl1, dir_header.incl.pl1, dir_ht.incl.pl1,
dir_link.incl.pl1, dir_name.incl.pl1, dirlockt.incl.pl1, disk_pack.incl.pl1, dskdcl.incl.pl1, ect_structures.incl.pl1,
event_wait_list.incl.pl1, fault_vector.incl.pl1, fgbx.incl.pl1, fa_dev_types.incl.pl1, fa_types.incl.pl1,
fs_vol_label.incl.pl1, hc_lock.incl.pl1, itt_entry.incl.pl1, knt.incl.pl1, lock_array.incl.pl1, lvt.incl.pl1,
mv.incl.pl1, null_addresses.incl.pl1, ptw.168.incl.pl1, pv_holdt.incl.pl1, pvt.incl.pl1, pvte.incl.pl1, rnt.incl.pl1,
scavenger_data.incl.pl1, scs.incl.pl1, sdw.168.incl.pl1, sdw_info.incl.pl1, signaler_stack.incl.pl1, slt.incl.pl1,
site.incl.pl1, sst.incl.pl1, ssstn.incl.pl1, stack_data.incl.pl1, stack_frame.incl.pl1, stack_header.incl.pl1,
stack_seg.incl.pl1, str.incl.pl1, tcm.incl.pl1, vol_map.incl.pl1, vtoc_buffer.incl.pl1, vtoc_header.incl.pl1,
vtoc_map.incl.pl1, vtoce.incl.pl1, prds.cds, prds.cds, tc_data.cds

Printed on: 04/01/83 0037.0
Printed by: Sibert.Multics.a
Descriptor: multics_libraries_
APPENDIX A

Hardcore Include Files

add_type.incl.pl1
aim_template.incl.pl1
apte.incl.pl1
aste.incl.pl1
bos_dump.incl.pl1
cmp.incl.pl1
dbm.incl.pl1
dir allocation_area.incl.pl1
dir_entry.incl.pl1
dir_header.incl.pl1
dir ht.incl.pl1
dir_link.incl.pl1
dir_name.incl.pl1
dirlockt.incl.pl1
disk_pack.incl.pl1
dskdcl.incl.pl1
ect_structures.incl.pl1
fault_vector.incl.pl1
fgbx.incl.pl1
fs_dev_types.incl.pl1
fs_types.incl.pl1
fs_vol_label.incl.pl1
hc_lock.incl.pl1
itt_entry.incl.pl1
kst.incl.pl1
lock_array.incl.pl1
lvt.incl.pl1
mc.incl.pl1
null_addresses.incl.pl1
pds.cds
prds.cds
ptw.168.incl.pl1
pv_holdt.incl.pl1
pv.incl.pl1
pvte.incl.pl1
rnt.incl.pl1
scavenger_data.incl.pl1
scs.incl.pl1
sdw.168.incl.pl1
sdw_info.incl.pl1
signaller_stack.incl.pl1
slt.incl.pl1
site.incl.pl1
sst.incl.pl1
sstnt.incl.pl1
stack_0.data.incl.pl1
stack_frame.incl.pl1
stack_header.incl.pl1
stack_seg.incl.pl1
str.incl.pl1
tc_data.cds
tcm.incl.pl1
BEGIN INCLUDE FILE add_type.incl.pl1

/* 02/26/75 by Bernard S. Greenberg */

/* This file provides a structure for checking PTW/CME address type fields in PL/I */

dcl 1 add_type unaligned static internal,
    2 core bit (4) init (*1000"b),
    2 disk bit (4) init (*0100"b),
    2 pd bit (4) init (*0010"b),
    2 reserved bit (4) init (*0001"b),
    2 non_null bit (4) init (*1111"b); /* in core- 5/8 only in PTW */

dcl 1 badd_type unaligned based,
    2 (core, disk, pd, reserved) bit (1) unaligned;

/* Disk address */

/* Paging Device */

/* Reserved */

/* Not null address */

END INCLUDE FILE add_type.incl.pl1 */
/* BEGIN INCLUDE FILE aim_template.incl.pl1 */

/* Created 740723 by PG */
/* Modified 06/28/78 by C. D. Tavares to add rcp privilege */

/* This structure defines the components of both an access class and an access authorization as interpreted by the Access Isolation Mechanism. */

dcl 1 aim_template aligned based,
   2 categories bit (36),
   2 level fixed bin (17) unaligned,
   2 privileges unaligned,
   (3 ipc,
   3 dir,
   3 seg,
   3 soos,
   3 ring1,
   3 rcp) bit (1),
   3 pad bit (12);

/* authorization/access class template */
/* access categories */
/* sensitivity level */
/* special access privileges (in authorization only) */
/* interprocess communication privilege */
/* directory privilege */
/* segment privilege */
/* security out-of-service privilege */
/* ring 1 access privilege */
/* RCP resource access privilege */

/* END INCLUDE FILE aim_template.incl.pl1 */
```
#include "hard.source"

apte.incl.pl1

entry declaration for an active (known) process */

/* List thread */
/* Forward pointer */
/* Backward pointer */
/* Flags and miscellaneous */
/* This bit must be zero (sentinel bit) */
/* ON If process has received wakeup */
/* ON If process has received stop connect */
/* ON If process is being pre-empted by get_processor */
/* ON If process is hardcore process */
/* ON If required per-process pages are in memory and wired */
/* ON If process is eligible */
/* ON If this is an idle process */
/* ON If process has interacted recently */
/* ON If process has received pre-empt connect */
/* ON If apte.procs_required is system default */
/* ON If process is not to be unloaded */
/* ON If D8R is loaded on some CPU */
/* ON If process is receiving pre-empt */
/* OFF until process is initialized */
/* execution state */
/* total page faults for the process */
/* bit 0-17: offset of AIPE */
/* bit 18-35: sequential number */
/* virtual time since eligibility award */
/* virtual time since scheduling */
/* virtual time since interaction */
/* maximum value allowed for apte.tf */
/* relative pointer to IIT list */
/* IPS signals pending */
/* relative AITE pointers */
/* PDS (per-process) */
/* DSEG (per-process) */
/* PROS (per-processor) */
/* 7 at call to getwork (return point in pss) */
/* process to send wakeup at termination */
```
2 lock_id bit (36),
2 time_used_clock fixed bin (71),
/* File System unqieu ID associated with process */
/* Total CPU time when process last lost CPU */

/* * * * * * * * * */
2 wait_event bit (36) aligned,
2 wait_index bit (18) unaligned,
2 flags2 unaligned,
3 priority_scheduling bit (1),
3 special_wakes bit (6),
3 pad7 bit (7),
3 batch bit (1),
3 pr_tag bit (3),
2 state_change_time fixed bin (71),
2 alarm_event fixed bin (71),
2 alarm_time_thread bit (18) unaligned,
2 alarm_time_bit (54) unaligned,
/* Event ID process awaiting */
/* rel offset of WCTE */
/* ON if guaranteed eligibility */
/* Special wake up channels */
/* ON if absentee */
/* CPU tag running or last run */
/* Time apt_state last changed */
/* wakeup event for alarm clock manager */
/* thread of processes with pending alarms */
/* wakeup time for alarm */

/* * * * * * * * * */
2 term_channel fixed bin (71),
2 ws_size fixed bin,
2 temax fixed bin (35),
2 deadline fixed bin (71),
2 lock bit (18) unaligned,
2 unusable_bit (18) unaligned,
2 cpu_monitor fixed bin (35),
2 paging_measure fixed bin (71),
2 access_authorization bit (72),
2 dbr fixed bin (71),
2 virtual_cpu_time fixed bin (71),
2 ittes_sent fixed bin (18),
2 ittes_got fixed bin (18),
/* wakeup event for account overflow */
/* working set estimate for the process */
/* maximum eligibility slice (vcpu) */
/* time of next run */
/* 0 == APTE locked, unlocked = return point of last unlock */
/* locking routines destroy */
/* if not 0, send wakeup to term_processid when virtual cpu
reaches this (units = 1/1024 sec) */
/* cumulative memory units */
/* authorization of this process */
/* DBR value (constant since DSEG entry-held) */
/* cumulative virtual CPU time for the process */
/* Unprocessed ITTs sent by this process */
/* Unprocessed ITTs received by this process */

/* Cells used to drive and instrument finite-state model for response time
measurement. Maintained by meter_response_time */
2 current_response_state fixed bin (17) unaligned,
2 pad18 bit (18) unaligned,
2 number_processing fixed bin (35),
2 last_response_state_time fixed bin (71),
2 total_processing_time fixed bin (71),
/* Process state in model */
/* Number interactions */
/* Clock time at last response state change */
/* Total interaction processing time */

/* * * * * * * * * */
2 begin_interaction_vcpu fixed bin (71),
/* Virtual cpu at beginning of last interaction */

/* End of cells for finite-state model */
2 saved_temax fixed bin (35),
2 proc_required bit (8) unaligned,
2 apad (12) fixed bin (35),
/* temax at eligibility award */
/* bit mask of CPUs this process can run */

#include "**", hard.source apter.incl.pi1
F80A - Not to be reproduced
BEGIN INCLUDE FILE ...aste.incl.pl1 ...

/* Template for an AST entry. Length = 12 words. */

/* Words 0 to 7, and 11 are read by PC; they are read and modified by SC. */
Words 8, 9 and 10 are modified by PC; they should never be modified without locking the PC lock */

dcl astep ptr;

dcl 1 ast based (astep) aligned,

(2 fp bit (18), /* forward used list rel pointer */
2 bp bit (18), /* backward used list rel pointer */
* 2 infl bit (18), /* ptr to NEXT in list of ASTE's of my brothers */
2 infp bit (18), /* ptr to FIRST in list of ASTE's of my children */
2 strp bit (18), /* rel pointer to process trailer */
2 par_astep bit (18), /* rel pointer to parent aste */
2 uid bit (36), /* segment unique id */
2 msi bit (9), /* maximum segment length in 1024 word units */
2 pvtx fixed bin (8), /* physical volume table index */
2 vtocx fixed bin (17), /* vtoc entry index */
2 usedf bit (1), /* ast entry is being used if non-zero */
2 init bit (1), /* used bit - insure 1 lap */
2 gtrans bit (1), /* global transparent usage switch */
2 gms bit (1), /* global transparent modified switch */
2 hc bit (1), /* hard core segment */
2 hc_sdw bit (1), /* aste with sdw for hardcore seg if non-zero */
2 any_access_on bit (1), /* any sdw allows access, unless write_access_on */
2 write_access_on bit (1), /* any sdw allows write access */
2 inhibit_cache bit (1), /* flag not to reset above bits */
2 explicit_deact_ok bit (1), /* set if user can deactivate seg */
2 deact_error bit (1), /* set if error occurred while deactivating */
2 hc_part bit (1), /* set if pages are in a hardcore partition */
2 fm_damaged bit (1), /* set if filemap checksum was ever bad */
2 padt bit (3), /* dumper in use switch */
2 dius bit (1), /* if on prevents addition to incremental dump map */
2 nid bit (1), /* entry hold switch */
2 dmpr_pad bit (1), /* no quota switch - no checking for pages of this seg */
2 ehs bit (1), /* directory switch */
2 ngsw bit (1), /* master dir - a root for the log volume */
2 dirsw bit (1), /* volmap_seg for some volume */
2 master_dir bit (1), /* terminal quota switch - (0) for non dir pages */
2 volmap_seg bit (1), /* */
2 tqsw (0:1) bit (1), /* */
```c
2 padlc bit (10), /* Used to be astc lc */
2 dtubit (36), /* date and time segment last used */
2 dtm bit (36), /* date and time segment last modified */
2 quota (0:1) fixed bin (18) unsigned, /* sec storage quota - (0) for non dir pages */
2 used (0:1) fixed bin (18) unsigned, /* sec storage used - (0) for non dir pages */
2 csl bit (9), /* current segment length in 1024 words units */
2 fmchanged bit (1), /* turned on by page if file map changed */
2 npfs bit (1), /* file modified switch */
2 gtdp bit (1), /* no page switch */
2 dnap bit (1), /* global transparent paging device switch */
2 par_process bit (1), /* don't null out if zero page switch */
2 ddnp bit (1), /* use master quota for this entry */
2 pad2 bit (2), /* don't deposit nulled pages */
2 records bit (9), /* number of records used by the seg in sec storage */
2 np bit (9), /* number of pages in core */
2 ht fp bit (18), /* hash table forward rel pointer */
2 fmchanged1 bit (1), /* value of "fmchanged" saved by pc$get_file_map */
2 pack_ovfl bit (1), /* page fault on seg would cause pack overflow */
2 synchronized bit (1), /* Data Management synchronized segment */
2 pad3 bit (8), /* 000000000 */
2 pts1 bit (2), /* page table size index */
2 marker bit (6) unaligned; /* marker to indicate last word of ASTE */

dcl asta (0 : 8000) bit (36*12 /* sst-> sst.astsize */) based aligned;

dcl 1 aste_part aligned based (astep), /* fp and bp */
2 one bit (36) unaligned, /* part that has to be zeroed when ASTE is freed */
2 two bit (36*11 - 8) unaligned, /* pts1 and marker */
2 three bit (8) unaligned;

dcl 1 seg_aсте based (astep) aligned, /* Overlay because quota is only for dirs */
2 pad1 bit (8*36), /* page fault count: overlays quota */
2 usage fixed bin (35),
2 pad2 bit (3*36);

/* END INCLUDE FILE ... aste.incl.pl1 */
```

Include _, hard source aste.incl.pl1

*90A not to be reproduced
/* BEGIN INCLUDE FILE ... bos_dump.incl.pll ... */
/* Modified 1 September 1976 */
/* Modified 11/11/80 by J. A. Bush for the DPS8/70M CPU */
/* Modified 6/12/81 by Rich Coppola to extend the dps8 extended fault reg to
15 bits */
/* Modified 02/23/81, W. Olin Sibert, to describe old and new FDUMP style: */

dcl dumpptr ptr:
    /* pointer to following structure */

    dcl 1 dump based (dumpptr) aligned,
    2 dump_header aligned like dump_header,
        2 segs (1008),
            3 segno bit (18) unal,
            3 length bit (18) unal,
        2 amptwregs (0 : 63) bit (36),
        2 amptwptrs (0 : 63) bit (36),
        2 amsdwregs (0 : 63) bit (72),
        2 amsdwptrs (0 : 63) bit (36),
        2 cuhist (0 : 63) bit (72),
        2 cuhist (0 : 63) bit (72),
        2 duhist (0 : 63) bit (72),
        2 auhist (0 : 63) bit (72),
        2 prs (0 : 7) ptr,
        2 regs aligned like dump_registers,
            2 low_order_port bit (3),
            2 pad4 bit (36),
            2 mctime fixed bin (52),
            2 pad5 (0 : 3) bit (36),
        2 misc_registers like dump_misc_registers,
            2 ptrlen (0 : 7) bit (36),
            2 coreblocks (0 : 7),
                3 num_first bit (18) unal,
                3 num_blocks bit (18) unal,
            2 pad7 (112) fixed bin;

    dcl 1 dump_header aligned based,
        2 words_dumped fixed bin (35),
        2 valid bit (1),
            /* Standard header for FDUMP */
            /* total words in dump */
            /* if there is a 6180 dump to be had */

include.**. hard.source

bos_dump.incl.pll
Page 8

f80a - Not to be reproduced
2 time fixed bin (71),
2 errno fixed bin (18),
2 num_segs fixed bin,
2 valid_355 bit (1),
2 dumped_355s bit (4),
2 time_355 fixed bin (71),
2 version fixed bin,
2 pad0 (5) fixed bin;

dcl 1 dump_registers aligned based,
(2 x (0 : 7) bit (18),
2 a bit (36),
2 q bit (36),
2 e bit (8),
2 pad2 bit (28),
2 t bit (27),
2 pad3 bit (6),
2 ralr bit (3)) unaligned;

dcl 1 dump_misc_registers aligned based,
2 scu (0 : 7) bit (36),
2 mcm (0 : 7) bit (72),
2 intrpts bit (36),
2 bar bit (36),
2 moderreg bit (36),
2 cmoderreg bit (36),
2 faultreg bit (36),
2 ext_fault_reg bit (15) unaligned,
2 pad6 bit (21) unaligned;

dcl 1 v1_dump aligned based (dumpptr),
2 dump_header aligned like dump_header,
2 segs (688),
3 segno bit (18) unal,
3 length bit (18) unal,
2 amsdregs (0 : 15) bit (72),
2 amsdwptrs (0 : 15) bit (36),
2 amptwregs (0 : 15) bit (36),
2 amptwptrs (0 : 15) bit (36),
2 pad1 (0 : 15) bit (36),
2 ouhist (0 : 15) bit (72),
2 cuhist (0 : 15) bit (72),
2 auhist (0 : 15) bit (72),
2 duhist (0 : 15) bit (72),
2 prs (0 : 7) ptr,
2 regs aligned like dump_registers,

/* time of dump */
/* Error Report Form Number */
/* number of segments dumped */
/* = 1 if there is a dumped 355 to be had */
/* indicates which 355s were dumped */
/* time of 355 dump */
/* currently 2 */
/* pad0 to 16 words */
/* Standard (SREG) arrangement of registers in dump */
/* index registers */
/* the a register */
/* the q register */
/* the e register */
/* pad */
/* timer register */
/* pad */
/* ring alarm register */
/* from store control unit instr. */
/* memory controller masks every 64 K */
/* interrupts */
/* base address register */
/* mode register */
/* cache mode register */
/* fault register */
/* DPS8 extended fault register */
/* Old version of FDUMP (pre March, 1981) */
/* segment array */
/* segment number */
/* length of segment in sector sized blocks */
/* assoc. mem. segment descriptor word registers */
/* assoc. mem. segment descriptor word pointers */
/* assoc. mem. page table word regs */
/* assoc. mem. page table word pointers */
/* operations unit history registers */
/* control unit history registers */
/* appending unit history registers */
/* decimal unit history registers */
/* pointer registers */
/* assorted machine registers */

#include "hard.source"  bos_dump.incl.r
2 mctime fixed bin (52),
2 pad4 (0 : 5) bit (36).
2 misc_registers aligned like dump_misc_registers, /* time conditions were taken */
2 pad5 bit (36),
2 pstrlen (0 : 7) bit (36), /* Assorted registers */
2 pad6 (15) bit (36),
2 low_order_port bit (3), /* pointers and lengths for EIS */
2 coreblocks (0 : 7), /* from which clock was read */
  3 num_first bit (18) unal, /* first addr in coreblock */
  3 num_blocks bit (18) unal; /* number of blocks used */

dcl DUMP_VERSION_1 fixed bin internal static options (constant) init (1);
dcl DUMP_VERSION_2 fixed bin internal static options (constant) init (2);

/* END INCLUDE FILE ... bos_dump.incl.pl1 ... */
BEGIN INCLUDE FILE cmp.incl.plt --- October 1982 */
/* Note: This include file has an ALM counterpart NOT made with cif (for historical reasons). Keep it up to date */
dcl cmep ptr;
  dcl cmep based (cmep) aligned;
    2 cmep ptr (16) unaligned;
    2 bp bit (16) unaligned;
    2 devaddr bit (22) unaligned,
    2 pad5 bit (1) unaligned,
    2 synch_held bit (1) unaligned,
    2 lo bit (1) unaligned,
    2 pad2 bit (1) unaligned,
    2 er bit (1) unaligned,
    2 removing bit (1) unaligned,
    2 abs w bit (1) unaligned,
    2 abs usable bit (1) unaligned,
    2 notify requested bit (1) unaligned,
    2 pad3 bit (1) unaligned,
    2 pma_hedge bit (1) unaligned,
    2 contr bit (3) unaligned,
    2 pttwp bit (18) unaligned,
    2 astep bit (18) unaligned,
    2 pin_counter fixed bin (17) unaligned,
    2 synch_page_entryp bit (18) unaligned;
  dcl cmcm based (cmep) aligned;
    2 pad bit (36) unaligned,
    2 record_no bit (18) unaligned,
    2 add_type bit (4) unaligned,
    2 flags bit (14) unaligned,
    2 pad1 bit (18) unaligned;
*/ END INCLUDE FILE cmp.incl.plt */
/* BEGIN INCLUDE FILE ... dbm.incl.pl1 ... Feb 1976 */

dcl dbm_seg$ ext;
dcl dbmp ptr;

dcl 1 dbm based (dbmp) aligned,
   2 lock_data,
   3 lock bit (36),
   3 event bit (36),
   3 notify bit (1),
   2 control,
   3 init bit (1) unal,
   3 pad1 bit (35) unal,
   2 stats,
   3 sets fixed bin unal,
   3 resets fixed bin unal,
   3 allocs fixed bin unal,
   3 frees fixed bin unal,
   2 pad2 (2) bit (36),
   2 area area (255*1024 -8);

/* END INCLUDE FILE ... dbm.incl.pl1 */
/* BEGIN INCLUDE FILE ... dir_acl.incl.pl1 ... last modified Nov 1975 for nss */
/* Template for an ACL entry. Length = 8 words */
dcl acl_ep_ptr;

dcl 1 acl_entry based (aclep) aligned,
2 frp bit(18) unaligned, /* rel ptr to next entry */
2 brp bit(18) unaligned, /* rel ptr to previous entry */
2 type bit (18) unaligned, /* type = dir acl */
2 size fixed bin (17) unaligned, /* size of acl entry */
2 name unaligned,
3 pers_rp bit(18) unaligned, /* user name associated with this ACL entry */
3 proj_rp bit(18) unaligned,
3 tag char(1) unaligned,
2 mode bit (3) unaligned, /* mode for userld */
2 pad24 bit(24) unaligned,
2 ex_mode bit(36), /* extended access modes */
2 checksum bit (36), /* checksum from acl_entry.name */
2 owner bit (36); /* uuid of owning entry */
/* Template for a person or project name on ACL. Length = 14 words. */
dcl 1 access_name aligned based,
2 frp bit(18) unaligned, /* rel ptr to next name structure */
2 brp bit(18) unaligned, /* rel ptr to prev name structure */
2 type bit (18) unaligned, /* type = access name */
2 size fixed bin (17) unaligned, /* size of access name */
2 salv_flag fixed bin(17) unaligned, /* used by salvager to check for ascii names */
2 usage fixed bin(17) unaligned,
2 pad1 bit (36), /* number of ACL entries that refer to this name */
2 name char(32) aligned, /* person or project name itself */
2 checksum bit (36), /* checksum from salv_flag */
2 owner bit (36); /* uuid of containing directory */
/* END INCLUDE FILE ... dir_acl.incl.pl1 */
dcl areap ptr;

dcl 1 area based (areap) aligned,
    2 nsizes fixed bin (18),
    2 lw fixed bin (18),
    2 array (100) aligned,
    3 fptr bit (18) unaligned,
    3 size fixed bin (17) unaligned;

/* BEGIN INCLUDE FILE ... dir_allocation_area.incl.pl1 ... last, modified December 1973 */

/* Number of types. */
/* Next available word in area. */
/* Last usable word. */
/* Array of types. */
/* Free pointer for this size. */
/* Size. */

/* END INCLUDE FILE ... dir_allocation_area.incl.pl1 */
BEGIN INCLUDE FILE ... dir_entry.incl.p11 ... last modified August 1974 for nss */

/* Template for an entry. Length = 38 words */

dcl  ep ptr;

dcl 1 entry based (ep) aligned,

(2 erp bit (18), 2 ebrp bit (18)): unaligned,
2 type bit (18) unaligned,
2 size fixed bin (17) unaligned,
2 uid bit (36),
2 dtem bit (36),
2 tname bit (36),
(2 bs bit (1),
2 padO bit (17), 2 names fixed bin (17),
2 name_frp bit (18), 2 name_brp bit (18),
2 author,
3 pers rp bit (18), 3 proj rp bit (18),
3 tag char (1), 3 pad1 char (3),
2 primary name bit (504),
2 dtd bit (36),
2 pad2 bit (36),

/* the declarations below are for branch only */

2 pvid bit (36), /* physical volume id */
2 vtocx fixed bin (17), /* vtoc entry index */
2 pad3 bit (18),
2 dirsw bit (1), /* = 1 if this is a directory branch */

BEGIN INCLUDE FILE ... dir_entry.incl.p11 ... last modified August 1974 for nss */
2 oosw bit (1),
2 per_process_sw bit (1),
2 copysw bit (1),
2 safety_sw bit (1),
2 multiple_class bit (1),
2 audit_flag bit (1),
2 security_oosw bit (1),
2 encrypt_sw bit (1),
2 master_dir bit (1),
2 lpd bit (1),
2 pad4 bit (1),
2 encrypt_bound bit (14)) unaligned,
2 access_class bit (72) aligned,

(2 ring_brackets (3) bit (3),
2 ex_ring_brackets (3) bit (3),
2 acl_count fixed bin (17),
2 acl_frp bit (18),
2 acl_brp bit (18),
2 bc_author,
3 pers_rp bit (18),
3 proj_rp bit (18),
3 tag char (1),
3 pad5 bit (2),
2 bc fixed bin (24)) unaligned,
2 sons_lvid bit (36),
2 pad6 bit (36),
2 checksum bit (36),
2 owner bit (36);

/* out of service switch on + 1 */
/* indicates segment is per process */
/* = 1 make copy of segment whenever initiated */
/* if 1 then entry cannot be deleted */
/* segment has multiple security classes */
/* segment must be audited for security */
/* security out of service switch */
/* 1 if call limiter is to be enabled */
/* TRUE for master directory */
/* TRUE if this segment is never to go on the PO */
/* call limiter */
/* security attributes : level and category */
/* ring brackets on segment */
/* extended ring brackets */
/* number of entries on ACL */
/* rel ptr to start of ACL */
/* rel ptr to end of ACL */
/* user who last set the bit count */
/* name of user who set the bit count */
/* project of user who set the bit count */
/* tag of user who set the bit count */
/* bit count for segs, msf indicator for dirs */
/* logical volume id for immediate inf non dir seg */
/* checksum from dtd */
/* uid of containing directory */

#include "hard.source"
BEGIN INCLUDE FILE ... dir_header.incl.ptt */
/* Modified 8/74 for NSS */
/* Modified 8/76 to add version number and hash table rel pointer for variable hash table sizes */
/* Modified 3/82 BIN for change pclock */
/* format: style3 */

/* Template for the directory header. Length = 64 words. */

dcl dp ptr;

dcl 1 dir based (dp) aligned,

2 modify bit (36),
2 type bit (18) unaligned,
2 size fixed bin (17) unaligned,
2 dtc (3),
3 date bit (36),
3 error bit (36),
2 uid bit (36),
2 pvid bit (36),
2 sons_lvid bit (36),
2 access_class bit (72),
(2 vtocx fixed bin (17),
2 version_number fixed bin (17),
2 entryfrp bit (18),
2 pad2 bit (18),
2 entrybrp bit (18),
2 pad3 bit (18),
2 pers_frp bit (18),
2 proj_frp bit (18),
2 pers_brp bit (18),
2 proj_brp bit (18),
2 seg_count fixed bin (17),
2 dir_count fixed bin (17),
2 lcount fixed bin (17),
2 acle_total fixed bin (17),
2 arearp bit (18),

2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2 per_process_sw bit (1),
2 master_dir bit (1),
2 force_rpv bit (1),
2 rehashing bit (1),
2 pad4 bit (14),

2 lacl_count (0:7),
  3 seg fixed bin (17),
  3 dir fixed bin (17),

2 lacl (0:7),
  3 seg_frpr bit (18),
  3 seg_brpr bit (18),
  3 dir_frpr bit (18),
  3 dir_brpr bit (18),

2 htsize fixed bin (17),
2 hash_table_rp bit (18),

2 htused fixed bin (17),
2 pad6 fixed bin (17),

2 tree_depth fixed bin (17),
2 pad7 bit (18)) unaligned,

2 dts bit (36),

2 master_dir_uid bit (36),
2 change_pclock fixed bin (35),
2 pad8 (11) bit (36),
2 checksum bit (36),
2 owner bit (36);

dcl version_number_2 fixed bin int static options {constant} init (2);

/* indicates dir contains per process segments */
/* TRUE if this is a master dir */
/* TRUE if segs must be on RPV */
/* TRUE if hash table is being constructed */

/* number of initial acl entries for segs */
/* number of initial acl entries for dir */

/* pointer to initial ACLs for each ring */
/* rel ptr to start of initial ACL for segs */
/* rel ptr to end of initial ACL for segs */

/* rel ptr to start of initial for dirs */
/* rel ptr to end of initial ACL for dirs */

/* size of hash table */
/* rel ptr to start of hash table */

/* no. of used places in hash table */

/* number of levels from root of this dir */

/* date-time directory last salvaged */

/* uid of superior master dir */
/* up one each call to sumdirmod */
/* pad to make it a 64 word header */
/* checksummed from uid on */
/* uid of parent dir */
dcl htp ptr;

dcl 1 hash_table based (htp) aligned, /* htp = ptr(dp,active_hardcore_data$htrp) */
  2 modify bit (36) unal. /* type * dir hash table */
  2 type bit (18) unal. /* size of current dir hash table entry */
  2 size fixed bin (17) unal. /* rel ptr of name entry */
  2 name_rp (0:1) bit(18) unal. /* otherwise rel ptr to name */
  2 checksum bit (36) unal. /* */
  2 owner bit (36) unal; /* */

dtr ht.incl.pl
/* BEGIN INCLUDE FILE ... dir_link.incl.pl1 ... last modified August 1974 for nss */
/* Template for link. Note that it is identical to entry for first 24 words. */

dcl 1 link based (ep) aligned,

(2 efrp bit (18),
  2 ebrp bit (18),
  2 type bit (18),
  2 size fixed bin (17),
  2 uid bit (36),
  2 dtem bit (36),
  2 bs bit (1),
  2 pad0 bit (17),
  2 nnames fixed bin (17),
  2 name_frp bit (18),
  2 name_brp bit (18),
  2 author,
    3 pers rp bit (18),
    3 proj rp bit (18),
    3 tag char (1),
    3 pad1 char (3),
  2 primary_name bit (504),
  2 dtd bit (36),
  2 pad2 bit (36),

/* the declarations below are only applicable to links */

  2 pad3 bit (18),
  2 pathname_size fixed bin (17),
  2 pathname char (168 refer (pathname_size)) unaligned,
  2 checksum bit (36),
  2 owner bit (36);
/* END INCLUDE FILE ... dir_link.incl.p1 */

Note to be reproduced
BEGIN INCLUDE FILE ... dir_name.incl.pl1 ... last modified Nov 1975 for nss */

/* Template for names of branches or links. Length = 14 words. */
dcl np ptr;

dcl 1 names based aligned,
   2 fp bit(18) unaligned,
   2 bp bit(18) unaligned,
   2 type bit (18) unaligned,
   2 size fixed bin (17) unaligned,
   2 entry_rp bit(18) unaligned,
   2 ht_index fixed bin(17) unaligned,
   2 hash_thread bit (18) unal,
   2 pad3 bit (18) unal,
   2 name char(32) aligned,
   2 checksum bit (36),
   2 owner bit (36);

 END INCLUDE FILE ... dir_name.incl.pl1 */
/* BEGIN INCLUDE FILE ..... dirlockt.incl.pl1 ..... */
/* Modified BIN 1/83 cleanup to multi-read lock */
/* format: style3,tdind25 */
dcl dirlockt_seg ext; /* name of the segment containing the directory locks */
dcl dirlocktp ptr; /* pointer to the dirlock table */
dcl dirlockt based (dirlocktp) aligned; /* Table of locks for directories */
  2 lock   bit (36), /* Lock for the table itself */
  2 ind    fixed bin (35), /* Event for the above lock */
  2 notify_sw bit (1), /* Index of the last entry currently used */
  2 last   fixed bin (17), /* Highest index ever used */
  2 highest_last fixed bin (17), /* count(i) = number of times entry i was used */
  2 counter (1:59) fixed bin (35),
  2 dirlock (1:10000) aligned like dir_lock; /* entry for a directory lock */
decall dir_lock_ptr pointer;
declare dir_lock aligned based (dir_lock_ptr),
  2 pid    bit (36), /* pid of the process that locked the dir for write */
  2 ind    bit (36) aligned, /* uld of the directory - also used as event id */
  2 notify_sw bit (i) unaligned, /* ON if one or more processes are waiting for the lock */
  2 salvage_sw bit (i) unaligned, /* ON if dir was locked for salvage */
  2 pad1   bit (34) unaligned, /* POSITIVE --> write_lock */
  2 lock_count fixed bin (35); /* NEGATIVE --> number of lockers */
  /+ ZERO --> not locked */

/* ..... END dirlockt.incl.pl1 ..... */
/* BEGIN INCLUDE FILE...disk_pack.incl.pl1 */

All disk packs have the standard layout described below:

- **Record 0**: contains the label, as declared in fs.vol.label.incl.pl1.
- **Record 1 to 3**: contains the volume map, as declared in vol_map.incl.pl1.
- **Record 4 to 5**: contains the dumper bit map, as declared in dumper_bit_map.incl.pl1.
- **Record 6**: contains the vtoc map, as declared in vtoc_map.incl.pl1.
- **Record 7**: formerly contained bad track list; no longer used.
- **Records 8 to n-1**: contain the array of vtoc entries; (n is specified in the label). Each record contains 5 192-word vtoc entries. The last 64 words are unused.
- **Records n to N-1**: contain the pages of the Multics segments. (N is specified in the label).

Sundry partitions may exist within the region n to N-1, withdrawn or not as befits the meaning of the particular partition.

A conceptual declaration for a disk pack could be:

```c
#include "hard.source"

#define DEFSIZE (1024)  // For initial release compatibility
#define NHPARTSIZE (1000)  // Limited by size of VTOC Map
#define MAX_VTOCE_PER_PACK (31774)

#define LABEL_ADDR init(0), // Address of Volume Label
#define VOLMAP_ADDR init(1), // Address of first Volume Map record
#define DUMPER_BIT_MAP_ADDR init(4), // For initial release compatibility
#define VTOC_MAP_ADDR init(6), // Address of first VTOC Map Record
#define VTOC_ORIGIN init(8), // Address of first record of VTOC
#define SECTORS_PER_VTOCE init(3),
#define VTOCES_PER_RECORD init(5),
#define DEFAULT_HCPART_SIZE init(1000), // Size of Hardcore Partition
#define MAX_VTOCE_PER_PACK init(31774) // Limited by size of VTOC Map

#define INCLUDE hard.source

#include "disk_pack.incl.pl1"
```

```c
/* Address of Volume Label */
/* Address of first Volume Map record */
/* For initial release compatibility */
/* Address of first VTOC Map Record */
/* Address of first record of VTOC */
/* Size of Hardcore Partition */
/* Limited by size of VTOC Map */
```
END INCLUDE FILE...disk_pack.incl.pl1 */
dskdcl.incl.pl

/* Begin include file .... dskdcl.incl.pl */
/* Structures used by the Disk DIM */
/* format: style4,deinl,insnl,tree,ifthenstmt,indnoniterend */

dcl disk_segment ext;
dcl disksp ptr,
    diskp ptr;

dcl 1 disk_data based (disksp) aligned,
    2 subsystems fixed bin,
    2 free offsetX bit (18),
    2 status_mask bit (36),
    2 last_queue_time fixed bin (71),
    2 pad (2) fixed bin,
    2 array (32),
    
    3 offset bit (18),
    3 pad bit (18),
    3 name char (4)
) unus;

dcl 1 disktab based (diskp) aligned,
    2 lock bit (36) unus,
    2 nchan fixed bin,
    2 ndrives fixed bin,
    2 channels_online fixed bin,
    2 dev_busy bit (64),
    2 dev_queued bit (64),
    2 wq (0:1) like qht,
    2 free_q like qht,
    2 abs_mem_addr fixed bin (26) unsigned,
    2 pad fixed bin,
    2 errors fixed bin,
    2 ferrors fixed bin,
    2 edac_errors fixed bin,
    2 pg_io_count (0:1) fixed bin,
    2 vt_io_count (0:1) fixed bin,
    2 call_lock_meters like disk_lock_meters,
    2 int_lock_meters like disk_lock_meters,
    2 alloc_wait_meters like disk_lock_meters,
    2 run_lock_meters like disk_lock_meters,
    2 pg_wait (0:1) fixed bin (52),
    2 vt_wait (0:1) fixed bin (52),
    2 pg_lo (0:1) fixed bin (52),
    2 vt_lo (0:1) fixed bin (52),
    2 queue (64) like queentry,
    2 chantab (8) like chantab,

include.*, hard.source
2 devtab (0 refer (disktab.ndrives)) like devtab;

%page;
dcl qp ptr,
    cp ptr;
dcl quentry based (qp) aligned,
    (  
        2 next bit (18),  *
        2 write_sw bit (1),  *
        2 sect_sw bit (1),  *
        2 testing bit (1),  *
        2 retry bit (1),
        2 used bit (1),
        2 swap bit (1),
        2 cylinder fixed bin (11),
        2 pdl unsigned fixed bin (6),
        2 coreadd bit (24),
        2 dev unsigned fixed bin (6),
        2 sector bit (21),
        2 pdl bit (9),
        2 n_sectors fixed bin (6) unsigned,
        2 time fixed bin (36) unsigned,
    ) unal;

dcl chantab based (cp) aligned,
    (  
        2 chx fixed bin (35),
        2 iol_ctx fixed bin (35),
        2 statusp ptr,
        2 chanid char (8),
        (  
            2 padO bit (18),
            2 in_use bit (1),
            2 active bit (1),
            2 rsr bit (1),
            2 prior bit (1),
            2 iol_use bit (1),
            2 inop bit (1),
            2 broken bit (1),
            2 action_code bit (2),
            2 padI bit (9),
        ) unal,
        (  
            2 qrp bit (18),
            2 pad2 bit (3),
            2 command bit (6),
            2 sect time fixed bin (8)
        ) unal,
        2 select_data,
        (  
            3 limit bit (12),
            3 mbz bit (3),
            3 sector bit (21)
        ) unaligned,
        2 connect_time fixed bin (52),
        2 connects fixed bin.
    ) unal,

#include <hardsource>
2 detailed_status (0:17) bit (8) unal,     /* detailed status bytes */
 2 rstdcw bit (36),                    /* restore command */
 2 scdcw bit (36),                     /* select command */
 2 sddcw bit (36),                     /* select data xfer */
 2 dcdcw bit (36),                     /* command to read or write */
 2 ddcw bit (36),                      /* data xfer DCW */
 2 dscdcw bit (36),                    /* RSR command */
 2 dsddcw bit (36),                    /* RSR data xfer */
 2 rssdcw bit (36),                    /* RSS command */
 2 status bit (36) aligned;            /* saved status */

%page;

dcl 1 qht aligned based,         /* queue head/tail structure */
         (head, tail) bit (18) unal;

dcl dp ptr,                      /* pointer to device information table */
         pvtdlp ptr;

dcl 1 devtab based (dp) aligned,  /* device information table */
         (                     /* index of PVT entry for device */
            2 pvtx fixed bin (8),     /* device inoperative */
            2 inop bit (1),         /* device previously broken */
            2 was_broken bit (1),  /* device lost and gone forever */
            2 broken bit (1),
            2 abandoned bit (1),   /* other device on this spindle or 0 */
            2 pad bit (11),
            2 buddy unsigned fixed bin (6),
            2 pdi unsigned fixed bin (6)
         ) unal,
         2 queue_count fixed bin (8),/* primary device index */
         2 cylinder fixed bin (11),/* count of requests queued for device */
         2 seek_distance fixed bin (35, 18),/* current cylinder position */
         2 read_count fixed bin,  /* average seek distance */
         2 write_count fixed bin, /* count of reads */
         2 time_inop fixed bin (52);/* count of writes */

%page;

dcl 1 pvtdi based (pvtdip) aligned,/* disk DIM info in PVT entry */
         (                     /* structure index */
            2 sx fixed bin (11),/* # of usable sectors on disk cylinder */
            2 usable_sect_per_cyl fixed bin (11),/* # of unused sectors at end of cylinder */
            2 unused_sect_per_cyl fixed bin (11)
         ) unal;

dcl 1 disk_lock_meters based aligned,/* lock meters for disk DIM */
         (                     /* total number of attempts */
            2 count fixed bin,
            2 waits fixed bin,
            2 wait_time fixed bin (52);
            2 time_meters fixed bin (52);/* number of attempts which required waiting */
            2 total_time_spent_waiting */
         ) fixed bin (12) static options (constant);

include.**. hard.source dskdcl.incl.pl1

END - Not to be reproduced
```c

BEGIN INCLUDE FILE ... ect_structures.incl.plil ... Jan 1981 */

/* format: style3 */
dcl ect_ptr  ptr;               /* points to base of Event Channel Table header */
dcl ectep   ptr;               /* points to event channel table entry */

dcl 1 ect_header  aligned based (ect_ptr),  /* structure of the Event Channel Table header */
    2 ect_area  ptr;               /* pointer to area in which ect entries are allocated */
    2 ect_area_size fixed bin (19),  /* number of words in ect area */
    2 flags,  
    3 call_priority bit (1) unal,   /* = 0'b if wait chns have priority - default */
    3 unused bit (17) unal,  
    3 mask_call_count fixed bin (17) unal,  
    2 count (0:5) fixed bin,  /* number times event call chans masked */
                         /* = 2'b if call chans have priority */

    2 entry_list_ptrs (4),  /* head and tail of lists in ECT */
    3 firstp ptr,  /* head of list */
    3 lastp ptr,  /* tail of list */
    2 meters,  
    3 total_wakeups fixed bin (33),  /* total wakeups sent on all channels */
    3 total_wait_wakeups fixed bin (33),  /* wakeups sent on wait channels */
    3 total_call_wakeups fixed bin (33),  /* wakeups sent on call channels */
                /* used to generate uid portion of channel name */
                /* number invalid ITT messages received, ignored */
    2 seed fixed bin (33),  
    2 itte_tossed fixed bin (33),  /* pad to 36 words */
    2 fill   (5) fixed bin;

dcl TOTAL fixed bin static options (constant) init (0);

dcl WAIT fixed bin static options (constant) init (1);

dcl CALL fixed bin static options (constant) init (2);

dcl EV_CALL_MESSAGE fixed bin static options (constant) init (3);

dcl ITT_MESSAGE fixed bin static options (constant) init (4);

dcl EV_MESSAGE fixed bin static options (constant) init (5);

dcl 1 wait_channel  aligned based (ectep),  /* Event wait channel - type = WAIT */
    2 word_0,  
    3 unused bit (17) unal,  
    3 type fixed bin (17) unal,  /* = WAIT */
    2 next_chanp ptr unal,  /* pointer to next wait channel */
```
2 prev_chanp  ptr unal,  /* pointer to previous wait channel */
2 word_3, 3 unused2  bit (1) unal,  
3 inhibit_count  fixed bin (16) unal,  /* number of times message reception has been inhibited */
3 wakeup_count  fixed bin (18) unal unsigned,  /* number of wakeups received over this channel */
2 name  bit (72),  /* event channel name associated with this channel */
2 first_ev_msgp  ptr unal,  /* pointer to first message in queue */
2 last_ev_msgp  ptr unal,  /* pointer to last message in queue */
2 unused3  (4) fixed bin;  /* pad to 12 words */

dcl 1 call_channel  aligned based (ectep),  /* Event call channel - type = CALL */
2 word_0, 3 priority  fixed bin (17) unal,  /* indicated priority relative to other call chns */
3 type  fixed bin (17) unal,  /* = CALL */
2 next_champ  ptr unal,  /* pointer to next call channel */
2 prev_champ  ptr unal,  /* pointer to prev call channel */
2 word_3, 3 call_inhibit  bit (1) unal,  /* = "I" if call to associated proc in progress */
3 inhibit_count  fixed bin (16) unal,  /* number of times message reception has been inhibited */
3 wakeup_count  fixed bin (18) unal unsigned,  /* number of wakeups received over this channel */
2 name  bit (72),  /* event channel name associated with this channel */
2 first_ev_msgp  ptr unal,  /* pointer to first message in queue */
2 last_ev_msgp  ptr unal,  /* pointer to last message in queue */
2 data_ptr  ptr unal,  /* pointer to associated data base */
2 procedure_value  ptr unal,  /* procedure to call when message arrives */
3 procedure_ptr  ptr unal,  /* pointer to entry point */
3 environment_ptr  ptr unal,  /* pointer to stack frame */
2 unused  fixed bin;  /* pad to 12 words */

dcl 1 event_message  aligned based,  /* Event message - type = EV_MESSAGE */
2 word_0, 3 priority  fixed bin (17) unal,  /* priority of call channel */
3 type  fixed bin (17) unal,  /* = EV_MESSAGE */
2 next_ev_msgp  ptr unal,  /* pointer to next message for this channel */
2 message_data  like event_message_data aligned,  /* event message as returned from ipc_block */
2 champ  ptr unal,  /* pointer to associated event channel */
2 next_call_msgp  ptr unal,  /* pointer to next call channel message */
2 unused2  (2) fixed bin;  /* pad to 12 words */

dcl 1 itt_message  aligned based,  /* Itt message - type = ITT_MESSAGE */
2 word_0, 3 unused1  fixed bin (17) unal,  
3 type  fixed bin (17) unal,  
2 next_itt_msgp  ptr unal,  /* pointer to next itt message entry in ECT currently */
2 message_data  like event_message_data aligned,  /* pad to 12 words */
2 unused2  (4) fixed bin;  /* pad to 12 words */

dcl 1 event_channel_name  aligned based,  /* description of name of channel */
2 acte_ptr  ptr unal,  /* pointer to channel entry in ECT */
2 ring  fixed bin (3) unal unsigned,  /* ring number of ECT */
2 unique_id  fixed bin (33) unal unsigned;  /* identified unique to the process */
IncludE'
channel name aligned based.
  ecte_ptr     ptr unal,
  ring         fixed bin (3) unal unsigned,
  mbz          bit (15) unal,
  channel_index fixed bin (17) unal;

EVENT MESSAGE DATA aligned based.
  channel_id     fixed bin (71),
  message       fixed bin (71),
  sender         bit (36),
  origin,
  dev_signal    bit (18) unal,
  ring          fixed bin (17) unal;

END INCLUDE file ... ect_structures.incl.pl1 */
/* BEGIN INCLUDE FILE ... event_wait_list.incl.pl1 */

/* ipc_$block wait list structure -- Must begin on an even word boundary.

Written 9-May-79 by M. N. Davidoff.
*/

declare event_wait_list_n_channels fixed binary;
declare event_wait_list_ptr pointer;
declare 1 event_wait_list aligned based (event_wait_list_ptr), 2 n_channels fixed binary, /* number of channels in wait list */ 2 pad bit (36), 2 channel_id (event_wait_list_n_channels refer (event_wait_list.n_channels)) fixed binary (71); /* event channels to wait on */

/* END INCLUDE FILE ... event_wait_list.incl.pl1 */
BEGIN INCLUDE FILE ... fault_vector.incl.pl1 ... last modified February 1981 */

dcl fvp ptr; /* pointer to the fault and interrupt vectors */

dcl 1 fv based (fvp) aligned,
   2 lpair (0: 31), /* fault and interrupt vectors */
   3 scu bit (36), /* interrupt pairs */
   3 tra bit (36), /* SCU Instruction */
   2 fpair (0: 31), /* TRA Instruction */
   3 scu bit (36), /* fault pairs */
   3 tra bit (36), /* SCU Instruction */
   2 fpa ir (0: 31), /* TRA Instruction */
   3 scu bit (36), /* ITM pair for interrupt TRA instruction */
   3 tra bit (36), /* ITM pair for interrupt SCU instruction */
   2 I_tra_ptr (0: 31), /* ITS pairs for fault TRA instruction */
   2 I_scu_ptr (0: 31), /* ITS pairs for fault SCU instruction */

* Fault Types by fault number

 Yên (FAULT_NO_SDF init (0)), /* Shutdown */
   FAULT_NO_STR Init (1), /* Store */
   FAULT_NO_MME Init (2), /* Master Mode Entry 1 */
   FAULT_NO_FT init (3), /* Fault Tag 1 */
   FAULT_NO_TRO Init (4), /* Timer Runout */
   FAULT_NO_CMD Init (5), /* Command */
   FAULT_NO_DRL Init (6), /* Derail */
   FAULT_NO_LUF Init (7), /* Lockup */
   FAULT_NO_CON Init (8), /* Connect */
   FAULT_NO_PAR Init (9), /* Parity */
   FAULT_NO_IPR Init (10), /* Illegal Procedure */
   FAULT_NO_DNC Init (11), /* Operation Not Complete */
   FAULT_NO_SUF Init (12), /* Startup */
   FAULT_NO_OFL Init (13), /* Overflow */
   FAULT_NO_DIV Init (14), /* Divide Check */
   FAULT_NO_EXF Init (15), /* Execute */
   FAULT_NO_DFO Init (16), /* Directed Fault 0 (Segment Fault) */
   FAULT_NO_DF1 Init (17), /* Directed Fault 1 (Page Fault) */
   FAULT_NO_DF2 Init (18), /* Directed Fault 2 */
   FAULT_NO_DF3 Init (19), /* Directed Fault 3 */
   FAULT_NO_ACV Init (20), /* Access Violation */
   FAULT_NO_MME2 Init (21), /* Master Mode Entry 2 */
   FAULT_NO_MME3 Init (22), /* Master Mode Entry 3 */
   FAULT_NO_MME4 Init (23), /* Master Mode Entry 4 */
   FAULT_NO_FT2 Init (24), /* Fault Tag 2 (Linkage Fault) */
   FAULT_NO_FT3 Init (25), /* Fault Tag 3 */
   FAULT_NO_FT4 Init (26), /* Fault Numbers 26-30 unassigned */
   FAULT_NO_TRB Init (31) /* Trouble */

) fixed bin (17) int static options (constant):

include '., hard.source fault_vector.incl.pl1

F not to be reproduced
/* END INCLUDE FILE ... fault_vector.incl.pl1 */
BEGIN INCLUDE FILE ... fgbx.incl.plt */
/* last modified 5/3/77 by Noel I. Morris */
/* Modified 8/79 by R.J.C. Kissel to add FNP blast message. */
/* Modified 7/82 SIM for recognizable sentinel field */

/* The contents of this segment are data shared by Multics and BOS.

This segment occupies the 2nd, 3rd, 4th, and 5th 16-word blocks of the BOS toehold. */

dcl flagbox$ ext;
dcl fgbxp ptr;

dcl 1 fgbx based (fgbxp) aligned,
  2 flags (36) bit (1) unal,
  2 sst_segno bit (18),
  2 pad1 fixed bin,
  2 rtb,
    3 ssenb bit (1),
    3 call_bos bit (1),
    3 shut bit (1),
    3 mess bit (1),
    3 alert bit (1),
    3 pad bit (25),
  3 bos_entry fixed bin (5)) unal,

  2 sentinel char (32) aligned,
  2 sst_edw bit (72),
  2 hc_dbr bit (72),
  2 message char (64),
  2 fnp_blast char (128);

declare FLAGBOX_SENTINEL char (32) init (*"Flagbox & Toehold Valid") int static options (constant);

END INCLUDE FILE ... fgbx.incl.plt */
In: entry modified: 03/10/82 0836.8

/* Begin Include file ..... fs_dev_types.incl.pll */

/* Modified 5/19/76 by N. I. Morris */
/* Modified 12/27/78 by Michael R. Jordan to correct MSS0500 information */
/* Modified 4/79 by R.J.C. Kissel to add msu0501 information. */

dcl (maxdevt Init (7), /* maximum legal devt */
   bulkdevt Init (1), /* bulk store devt */
   msu0500devt Init (2), /* MSU0500 device type */
   msu0451devt Init (3), /* MSU0451 device type */
   msu0450devt Init (4), /* MSU0450 device type */
   msu0400devt Init (4), /* MSU0400 device type */
   dsu191devt Init (4), /* DSU191 device type */
   dsu190devt Init (5), /* DSU190 device type */
   dsu181devt Init (6), /* DSU181 device type */
   msu0501devt Init (7) /* MSU0501 device type */
) fixed bin (4) static options (constant);

dcl MODEL (10) fixed bin static options (constant) Init
(0, 500, 451, 450, 400, 402, 191, 190, 181, 501); /* Known device model numbers */

dcl MODELEX (10) fixed bin static options (constant) Init
(1, 2, 3, 4, 5, 6, 7); /* translation from model number to device type */

dcl MODELN (7) fixed bin static options (constant) Init
(0, 500, 451, 400, 190, 181, 501); /* translation from device type to model number */

dcl device_names (7) char (4) aligned static options (constant) Init
("bulk", "d500", "d451", "d400", "d190", "d181", "d501");

dcl media_removable (7) bit (1) static options (constant) Init /* ON => d'mountable pack on device */
("0"b, "0"b, "1"b, "1"b, "1"b, "1"b, "0"b);

dcl shared_spindle (7) bit (1) static options (constant) Init /* ON => 2 devices per spindle */
("0"b, "1"b, "0"b, "0"b, "0"b, "0"b, "1"b);

dcl rec_per_dev (7) fixed bin static options (constant) Init /* table of # of records on each device */
(0, 38258, 38258, 19270, 14760, 4444, 67200);

dcl cyl_per_dev (7) fixed bin static options (constant) Init /* table of # of cylinders on each device */
(0, 814, 814, 410, 410, 202, 840);

dcl rec_per_cyl (7) fixed bin static options (constant) Init /* table of # of records per cylinder on each device */
(0, 47, 47, 47, 36, 22, 80);

dcl sect_per_cyl (7) fixed bin static options (constant) Init /* table of # of sectors per cylinder on each device */
(0, 760, 760, 760, 589, 360, 1280);

dcl sect_per_rec (7) fixed bin static options (constant) Init /* table of # of sectors per record on each device */
(0, 16, 16, 16, 16, 16, 16);

/* End Include file */

This page is the included file for device types and enumeration of device models and types.
dcl tracks_per_cyl (7) fixed bin static options (constant) init /* table of # of tracks per cylinder on each device */ (0, 19, 19, 19, 19, 20, 20);

dcl sect_per_track (7) fixed bin static options (constant) init /* table of # of sectors per track on each device */ (0, 40, 40, 40, 31, 18, 64);

dcl words_per_sect (7) fixed bin static options (constant) init /* table of # of words per sector on each device */ (0, 64, 64, 64, 64, 64, 64);

dcl first_rec_num (7) fixed bin static options (constant) init /* table of # of first record on each device */ (0, 0, 0, 0, 0, 0, 0);

dcl last_rec_num (7) fixed bin (18) static options (constant) init /* table of # of last record on each device */ (0, 38257, 38116, 19128, 14651, 4399, 67199);

dcl first_sect_num (7) fixed bin (24) static options (constant) init /* table of # of first sector for each device */ (0, 0, 0, 0, 0, 0, 0);

dcl last_sect_num (7) fixed bin (24) static options (constant) init /* table of # of last sector number for each device */ (0, 618639, 616359, 309319, 239722, 71999, 1075199);

dcl first_alt_sect_num (7) fixed bin (24) static options (constant) init /* table of # of first sector of alt partition */ (0, 638400, 616360, 309320, 239723, 72000, 1075200);

dcl last_alt_sect_num (7) fixed bin (24) static options (constant) init /* table of # of last sector of alt partition */ (0, 639999, 618639, 311599, 241489, 72719, 1077759);

dcl last_physical_sect_num (7) fixed bin (24) static options (constant) init /* table of # of last sector on device (includes T&D cylinders) */ (0, 639999, 619399, 312359, 242249, 72359, 1077759);

dcl dev_time (7) float bin (27) static options (constant) init /* table of average access times for each device */ (384e0, 33187e0, 33187e0, 34722e0, 46935e0, 52631e0, 33187e0);

*/ End of include file ...... fs_dev_types.incl */
/* BEGIN INCLUDE FILE ... fs_types.incl.pl1 */

dcl ACCESS_NAME_TYPE bit (18) static options (constant) init ("000001"b3);
dcl ACLE_TYPE bit (18) static options (constant) init ("000002"b3);
dcl DIR_HEADER_TYPE bit (18) static options (constant) init ("000003"b3);
dcl DIR_TYPE bit (18) static options (constant) init ("000004"b3);
dcl LINK_TYPE bit (18) static options (constant) init ("000005"b3);
dcl NAME_TYPE bit (18) static options (constant) init ("000006"b3);
dcl SEG_TYPE bit (18) static options (constant) init ("000007"b3);
dcl HASH_TABLE_TYPE bit (18) static options (constant) init ("000008"b3);

dcl access_name_type fixed bin static options (constant) init (1);
dcl acle_type fixed bin static options (constant) init (2);
dcl dir_header_type fixed bin static options (constant) init (3);
dcl dir_type fixed bin static options (constant) init (4);
dcl link_type fixed bin static options (constant) init (5);
dcl name_type fixed bin static options (constant) init (6);
dcl seg_type fixed bin static options (constant) init (7);
dcl hash_table_type fixed bin static options (constant) init (11);

/* END INCLUDE FILE ... fs_types.incl.pl1 */
/* BEGIN INCLUDE FILE ... fs_vol_label.incl.pl1 ... last modified January 1982 for new volume map format */

/* This is the label at fixed location of each physical volume. Length 1 page */

dcl labelp ptr;

dcl label based (labelp) aligned.

/* First comes data not used by Multics.. for compatibility with GCOS */

2 gcos (5*64) fixed bin,

/* Now we have the Multics label */

2 Multics char (32) Init ("Multics Storage System Volume"), /* Identifier */
2 version fixed bin,
2 mfg_serial char (32),          /* Version */
2 pv_name char (32),             /* Manufacturer's serial number */
2 lv_name char (32),             /* Physical volume name. */
2 pvld bit (36),                 /* Name of logical volume for pack */
2 lvld bit (36),                 /* Unique ID of this pack */
2 root_pvld bit (36),            /* Unique ID of its logical vol */
2 time_registered fixed bin (71),
2 n_pv_in_lv fixed bin,          /* time imported to system */
2 vol_size fixed bin,            /* # phys volumes in logical */
2 vtoc_size fixed bin,           /* total size of volume, in records */
2 not_used bit (1) unal,         /* number of recs in fixed area + vtoc */
2 private bit (1) unal,          /* used to be multiple_class */
2 flagpad bit (34) unal,         /* TRUE if was registered as private */
2 max_access_class bit (72),     /* Maximum access class for stuff on volume */
2 min_access_class bit (72),     /* Minimum access class for stuff on volume */
2 password bit (72),             /* not yet used */
2 pad1 (16) fixed bin,           /* Maximum access class for stuff on volume */
2 time_mounted fixed bin (71),   /* Minimum access class for stuff on volume */
2 time_map_updated fixed bin (71),

/* The next two words overlay time_unmounted on pre-MR10 systems. This */
/* forces a salvage if an MR10 pack is mounted on an earlier system. */

/* size of pack error history */
2 volmap_version fixed bin,      /* time salvaged */
2 pad6 fixed bin,                /* time of last bootload */
2 time_salvaged fixed bin (71),  /* time unmounted cleanly */
2 time_of_boot fixed bin (71),   /* pvtx in that PDMAP */
2 time_unmounted fixed bin (71), /* size of pack error history */
2 last_pvtx fixed bin,           /* time last completed dump pass started */
2 pad9 (2) fixed bin,
2 err_hist_size fixed bin,
2 time_last_dmp (3) fixed bin (71),

include "hard.source"
/* what it says */

/* TRUE if the root is on this pack */
/* VTOC index of root, if it is here */
/* Status of hierarchy */

/* VTOC index of disk table on RPV */
/* UID of disk table */
/* State of esd */
/* Number of records in volume map */
/* Begin record of VTOC map */
/* Number of records in VTOC map */
/* Number of words per volume map section */
/* Begin record of VTOC */
/* Begin record of dumper bit-map */
/* Count of inconsistencies found since salvage */
/* Number of special partitions on pack */
/* Name of partition */
/* First record */
/* Number of records */

dcl Multics_ID_String char (32) init ("Multics Storage System Volume") static;

/* END INCLUDE FILE fs_vol_label.incl.pll */
/* Begin include file hc_lock.incl.pl1 BIM 2/82 */
/* Lock format suitable for use with lock$lock_fast, unlock_fast */

/* format: style3 */

declare lock_ptr

declare lock
  pid
  event
  flags
  notify_sw
  pad
  /* holder of lock */
  /* event associated with lock */
  /* certain locks use this pad, like dirs */

/* End include file hc_lock.incl.pl1 */
BEGIN INCLUDE FILE ... itt_entry.incl.pl1 ... Feb 1981 */

/* format: style3 */
dcl itte_ptr
  ptr; /* pointer to entry in ITT */

/* declaration of single entry in the ITT */
dcl 1 itt_entry
    aligned based (itte_ptr),
    2 next_itt_relp bit (18) unaligned,
    2 pad bit (18) unaligned,
    2 sender bit (36),
    2 origin,
    3 dev_signal bit (18) unaligned,
    3 ring fixed bin (17) unaligned,
    2 target_id bit (36),
    2 channel_id fixed bin (71),
    2 message fixed bin (71); /* event message */

/* end declaration */

/* thread of relative pointers */
/* id of sending process */
/* origin of event message */
/* 0 = user-event, 1 = device-signal */
/* if user-event, sender's validation ring */
/* target process' id */
/* target process' event channel */
/* event message */

END INCLUDE FILE ... itt_entry.incl.pl1 */
BEGIN INCLUDE FILE

KST header declaration

KST entry declaration

END INCLUDE FILE
BEGIN INCLUDE FILE ... lock_array.incl.pl1 */
modified BIM 2/82 to clean up */
format: style3 */
dcl  1 pds$lock_array  (0:19) external aligned like pds_entry;  
dcl  1 pds_entry    based,  
    2 lock_ptr     pointer,  
    2 event        bit (36) aligned,  
    2 flags        aligned,  
    3 dir_lock     bit (1) unaligned, /* This is a dir read lock */  
    3 one_word     bit (1) unaligned, /* lock$wait */  
    3 pad          bit (34) unaligned, /* else lock$lock$fast */  
    2 caller_ptr   pointer;  

END INCLUDE FILE ... lock_array.incl.pl1 */
/* BEGIN INCLUDE FILE lvt.incl.plt -- Written Jan. 1976 by R. Bratt */

/* This include file defines the format of the hardcore Logical Volume Table (LVT). */

dcl lvts ext;
dcl lvtep ptr;
dcl lvtep ptr;

dcl lvt aligned based (lvtp),
2 max_lvtex fixed bin (17),
2 high_water_lvtex fixed bin (17),
2 free_lvtep ptr,
2 pad l bit (36),
2 ht (0:63) ptr unaligned,
2 lvtes (1:1 refer (lvt.max_lvtex)) like lvte;

/* maximum number of LVs describable */
/* highest LVT index assigned */
/* pointer to first free lvte */
/* lvtd hash table */
/* LVT entries */

/* logical volume table entry */
/* lvtd hash thread */
/* thread of mounted PVs */
/* logical volume id */
/* access isolation mechanism stuff */
/* minimum access class allowed on LV */
/* maximum access class allowed on volume */
/* flags */
/* anyone can connect to this LV */
/* no writes on this LV */
/* pvtx for next per_process seg */

/* END INCLUDE FILE lvt.incl.plt */
BEGIN INCLUDE FILE mc.incl.pl1 Created Dec 72 for 6180 - WSS */
/* Modified 06/07/76 by Greenberg for mc.resignal */
/* Modified 07/07/76 by Morris for fault register data */
/* Modified 08/28/80 by J. A. Bush for the DPSB/70M CVPU */

/* words 0-15 pointer registers */
dcl mcp ptr;

dcl 1 mc based (mcp) aligned,
    2 prs (0:7) ptr,
    (2 regs,
     3 x (0:7) bit (18),
     3 a bit (36),
     3 q bit (36),
     3 e bit (8),
     3 pad1 bit (28),
     3 t bit (27),
     3 pad2 bit (6),
     3 ralr bit (3),
    2 scu (0:7) bit (36),
    2 mask bit (72),
    2 ips_temp bit (36),
    2 errcode fixed bin (35),
    2 fim_temp,
    3 unique_index bit (18) unal,
    3 resignal bit (1) unal,
    3 fcode bit (17) unal,
    2 fault_reg bit (36),
    2 pad2 bit (1),
    2 cpu_type fixed bin (2) unsigned,
    2 ext_fault_reg bit (15),
    2 fault_time bit (54),
    2 eis_info (0:7) bit (36)) unaligned;

dcl (apx fixed bin init (0),
    abx fixed bin init (1),
    bpx fixed bin init (2),
    bbx fixed bin init (3),
    lpx fixed bin init (4),
    lbx fixed bin init (5),
    spx fixed bin init (6),
    sbx fixed bin init (7)) internal static;

include 'hard.source' mc.incl.pl1
F80A not to be reproduced
dcl scup ptr;
dcl 1 scu based (scup) aligned.

/ *
WORD (0)
*
(2 ppr,
 3 ppr bit (3),
 3 psr bit (15),
 3 p bit (1)),

2 apu,
 3 xsf bit (1),
 3 sdwm bit (1),
 3 sd_on bit (1),
 3 ptwm bit (1),
 3 pt_on bit (1),
 3 pl_ap bit (1),
 3 dsptw bit (1),
 3 sdwp bit (1),
 3 ptw bit (1),
 3 ptw2 bit (1),
 3 fanp bit (1),
 3 fabs bit (1),

2 fault_cntr bit (3),

/ *
WORD (1)
*
2 fd,
 3 iro bit (1),
 3 oeb bit (1),
 3 e_off bit (1),
 3 orb bit (1),
 3 r_off bit (1),
 3 owb bit (1),
 3 w_off bit (1),
 3 no_ga bit (1),
 3 ocb bit (1),
 3 ocall bit (1),
 3 boc bit (1),
 3 inret bit (1),
 3 crt bit (1),
 3 ralr bit (1),
 3 am_er bit (1),
 3 oosb bit (1),
 3 paru bit (1),
 3 parl bit (1),
 3 onc_bit (1),

/* SCU DATA */

/ * PROCEDURE POINTER REGISTER */
/* procedure ring register */
/* procedure segment register */
/* procedure privileged bit */

/ * APPENDING UNIT STATUS */
/* ext seg flag - IT modification */
/* match in SDW Ass. Mem. */
/* SDW Ass. Mem. ON */
/* match in PTW Ass. Mem. */
/* PTW Ass. Mem. ON */
/* Instr Fetch or Append cycle */
/* Fetch of DSPTW */
/* Fetch of SDW non paged */
/* Fetch of SDW paged */
/* Fetch of PTW */
/* Fetch of pre-paged PTW */
/* Fetch of final address paged */
/* Fetch of final address non-paged */
/* Fetch of final address absolute */

/* number of retries of EIS instructions */

/ * FAULT DATA */
/* illegal ring order */
/* out of execute bracket */
/* no execute */
/* out of read bracket */
/* no read */
/* out of write bracket */
/* no write */
/* not a gate */
/* out of call bracket */
/* outward call */
/* bad outward call */
/* Inward return */
/* cross ring transfer */
/* ring alarm register */
/* associative memory fault */
/* out of segment bounds */
/* processor parity upper */
/* processor parity lower */
/* op not complete type 1 */
3 onc_2 bit (1),
2 port_stat,
3 isfl bit (4),
3 isac bit (3),
3 con_chan bit (3),
2 fl_num bit (5),
2 fl_flag bit (1),

/* WORD (2) */
2 tpr,
3 trr bit (3),
3 tsr bit (15),
2 pad2 bit (9),
2 cpu_no bit (3),
2 delta bit (6),

/* PORT STATUS */
/* PORT STATUS */
/* illegal action lines */
/* illegal action channel */
/* connect channel */
/* (fault/interrupt) number */
/* 1 => fault, 0 => interrupt */

/* WORD (3) */
2 word3 bit (18),
2 tsr_stat,
3 tane,
4 prn bit (3),
4 prv bit (1),
3 tsnb,
4 prn bit (3),
4 prv bit (1),
3 tanc,
4 prn bit (3),
4 prv bit (1),
2 tpr_tbr bit (6),

/* TEMPORARY POINTER REGISTER */
/* temporary ring register */
/* temporary segment register */
/* CPU number */
/* tally modification DELTA */

/* WORD (4) */
2 tlc bit (18),
2 ir,
3 zero bit (1),
3 neg bit (1),
3 carry bit (1),
3 ovfl bit (1),
3 eovf bit (1),
3 eufi bit (1),
3 oflm bit (1),
3 tro bit (1),

/* INSTRUCTION COUNTER */
/* INDICATOR REGISTERS */
/* zero indicator */
/* negative indicator */
/* carryry indicator */
/* overflow indicator */
/* eponent overflow */
/* exponent underflow */
/* overflow mask */
/* tally runout */
3 par bit (1),
3 parm bit (1),
3 bm bit (1),
3 tru bit (1),
3 mif bit (1),
3 abs bit (1),
3 pad bit (4).

/*
WORD (5)
*/

2 ca bit (18),
2 cu,
3 rf bit (1),
3 rpt bit (1),
3 rd bit (1),
3 pot bit (1),
3 pon bit (1),
3 xde bit (1),
3 xdo bit (1),
3 poa bit (1),
3 rfi bit (1),
3 its bit (1),
3 if bit (1),
2 cpu_tag bit (6)) unaligned,

/*
WORDS (6,7)
*/

2 even_inst bit (36),
2 odd_inst bit (36);

/*
ALTERNATE SCU DECLARATION
*/
dcl 1 scux based (scup) aligned,
(2 pad0 bit (36),
2 fd,
3 isn bit (1),
3 ioc bit (1),
3 la_am bit (1),
3 isp bit (1),
3 ipr bit (1),
3 nea bit (1),

/• parity error */
/* parity mask */
/* bar mode */
/* truncation mode */
/* multi-word instruction mode */
/* absolute mode */

/* COMPUTED ADDRESS */
/* CONTROL UNIT STATUS */
/* on first cycle of repeat instr */
/* repeat instruction */
/* repeat double instruction */
/* repeat link instruction */
/* IT modification */
/* return type instruction */
/* XDE from Even location */
/* XDE from Odd location */
/* operation preparation */
/* tells CPU to refetch instruction */
/* ITS modification */
/* fault occured during instruction fetch */
/* computed tag field */

/* even instruction of faulting pair */
/* odd instruction of faulting pair */

/* GROUP II FAULT DATA */
/* illegal segment number */
/* illegal op code */
/* illegal address - modifier */
/* illegal slave procedure */
/* illegal procedure */
/* non existent address */

incl        **, hard.source
mc.incl.pl1
F80A - Not to be reproduced
3 oobb bit (1),
3 pad bit (29),
2 pad2 bit (36),
2 pad3a bit (18),
2 tsr_stat (0:2),
3 prn bit (3),
3 prv bit (1),
2 pad3b bit (6)) unaligned,
2 pad45 (0:1) bit (36),
2 instr (0:1) bit (36);

/* END INCLUDE FILE mc.incl.pl1 */
null_addresses.incl.pl1

segment
entry modified: 10/13/82 1311.9

entry modified: 10/13/82 1311.9

/* BEGIN INCLUDE FILE null_addresses.incl.pl1 */

dcl (pc_move_page_table_1_null_addr init
    pc_move_page_table_2_null_addr init
    get_aste_null_addr init
    make_sdw_null_addr init
    put_aste_null_addr init
    page_bad_pd_null_addr init
    list_deposit_null_addr init
    get_file_map_null_addr init
    fill_page_table_null_addr init
    init_sst_null_addr init
    get_file_map_vt_null_addr init
    unprotected_null_addr init
    page parity_null_addr init
    page devparity_null_addr init
    get_file_map_dumper_non_null_addr init
    page_bad_null_addr init
    page_problem_null_addr init
) (*'3770070"b3),
    (*'3770100"b3),
    (*'3770110"b3),
    (*'3770120"b3),
    (*'3770130"b3),
    (*'3770150"b3),
    (*'3770160"b3),
    (*'3770170"b3),
    (*'3770200"b3),
    (*'3770210"b3),
    (*'3770220"b3),
    (*'3770230"b3),
    (*'3770240"b3),
    (*'3770250"b3)) bit (22) aligned static options (constant);

dcl create_vtoce_four_null_addr fixed bin (71) int static init (-1);/* 777777 777777 777777 777777 */

dcl (create_vtoce_null_addr init
    update_vtoce_null_addr init
    truncate_vtoce_fill_null_addr init
    truncate_vtoce_null_addr init
    pv_salv_null_addr init
    pv_scav_null_addr init
    volume_reloader_null_addr init
    volume_retriever_null_addr init
    salv_truncate_null_addr init
) (*'777777"b3),
    (*'777776"b3),
    (*'777775"b3),
    (*'777702"b3),
    (*'777704"b3),
    (*'777706"b3),
    (*'777774"b3),
    (*'777773"b3),
    (*'777705"b3)) bit (18) aligned static options (constant);

/* END INCLUDE FILE null_addresses.incl.pl1 */

FBOA - Not to be reproduced
pds.cds

segment in: >ldd>hard>source

system id: 37-20

entry modified: 03/27/83 1807.2

/* *******************************************************
 * Copyright, (C) Honeywell Information Systems Inc., 1982 *
 * ******************************************************* */

/* PDS - The Process Data Segment */

Last modified (Date and reason):
2/6/76 by S. Webber Initial coding
9/17/76 by R. Brett to add seg_fault, bounds_fault, vtoc_read, and vtoc_write meters.
11/03/76 by M. Weaver to extend stack header
04/20/77 by M. Weaver to delete rntnp and 7/77 to add name template_pds
06/07/78 by E. Donner to add ring_events (to prevent delayed ipc wakeups)
05/10/79 by B. Margulies to eliminate exmode_level
05/09/79 by Mike Grady to use shared ring 0 stacks
08/17/79 by J. A. Bush for exp under/overflow restart switches & cache parity diagnostics
02/28/80 by B. Margulies to use the include file for the default overflow
08/26/80 by J. A. Bush for the DPS8/70M CPU value.
02/23/81 by J. Bongiovanni to remove temp_mode_reg (moved to prds$mode_reg_enabled)
03/81 by E. Donner to remove next_itl and act_pointers
3/82 BIM for lock_array cleanup.
11/82 by J. Bongiovanni to make force_write_limit per-ring
2/83 by E. N. Kittlitz for hfp_exponent_enabled.
*/

/* *******************************************************
 * Copyright (c) 1972 by Massachusetts Institute of *
 * Technology and Honeywell Information Systems, Inc. *
 * ******************************************************* */

/* format: style3,ldind25 */
pds:
  proc;
/* This program creates the pds data base */
/* Automatic */
dcl 1 cdsa aligned like cds_args;
dcl code fixed bin (35);

include ., hard.source pds.cds

F80/ not to be reproduced
/* Constants */
dcl pdsname char (3) aligned static init (*pds*) options (constant);
dcl exclude_pad (1) char (32) aligned static options (constant) init (*pad*);

/* Builtins */
dcl (addr, bin, bit, hbound, mod, null, rel, size, string, unspec) builtin;

/* Entries */
dcl com_err entry options (variable);
dcl create_data_segment_ entry (ptr, fixed bin (35));
dcl get_temp_segment_ entry (char (*), ptr, fixed bin (35));
dcl release_temp_segment_ entry (char (*), ptr, fixed bin (35));
dcl hcs__chname_file entry (char (*), char (*), char (*), char (*), fixed bin (35));
dcl get_wdir_ entry () returns (char (168));

/* External Static */
dcl error_table_$segnamedup fixed bin (35) ext;
Hey John,
Our team is working on optimizing the performance of our system. We identified an issue where the page fault rate is significantly higher than expected. We suspect that it might be related to the way we are handling memory allocation. We plan to implement a new algorithm for memory management to address this issue. Let me know if you have any suggestions or if you need more information.

Best,
[Your Name]
2 stack_0_sdwp  ptr aligned,  /* ptr to stack sdwp in dseg */
2 stack_0_ptr   ptr aligned,  /* ptr to base of ring 0 stack (wired for esd) */
2 tc_argp       ptr,  /* arg ptr used by tc */
2 tc_mask       bit (72) aligned,  /* save tc mask */
2 exp_unfil_rest bit (1) aligned,  /* switch for restarting exp underflows from the fim */
2 exp_even_rest  bit (1) aligned,  /* switch for restarting exp overflows from the fim */
2 evol_value     bit (72) aligned,  /* value to load when restarting exp overflows from the fim */
2 cpar_err_data  bit (72) aligned,  /* cache parity error data (from cache) */
2 cpar_mem_data  bit (72) aligned,  /* cache parity error data (from memory) */
2 cpar_info      bit (36) aligned,  /* diagnose flag, cache level and absaddr */
2 hfp_exponent_enabled bit (1) aligned,  /* user allowed to set IR hex exp bit */
2 pad_for_trace_mod16 (14) fixed bin,
2 trace         (306) fixed bin (71),  /* system trace data */
2 timer_time_out fixed bin (52),  /* time out time for the process */
2 timer_channel  fixed bin (71),  /* event channel for time out event */
2 term_channel   fixed bin (71),  /* channel used to signal process termination */
2 term_proc      bit (36) aligned,  /* process ID of process to signal term process */
2 pl1_machine    fixed bin,  /* nonzero if we do pl1-like things */
2 validation_level fixed bin (3),  /* ACC string for condition name */
2 condition_name aligned,
3 len            fixed bin (8) unaligned,
3 char           char (31) unaligned,
2 pad_obsolete   bit (36) aligned,
2 ips_mask       (0:7) bit (35) aligned,  /* IPS masks */
2 auto_mask      (0:7) bit (36) aligned,  /* array of automatic masks for IPS signals */
2 ring_alarm_val fixed bin,  /* used in checking validation level changes */
2 lock_id        bit (36) aligned,  /* UID used in some locking */
2 mc_trace_buf   ptr unaligned,  /* packed ptr to mc_trace wired buffer */
2 pad_end_of_page_0 bit (0) unaligned,
2 pathname_am    aligned like pam,  /* pathname associative memory */
2 initial_procedure ptr,  /* first procedure executed in a new process */
2 account_id     char (32) aligned,  /* not used yet */
2 lock_array     (0:19) aligned like pds_entry,
2 access_name    aligned,  /* alternate form of process group id */
3 user           char (32) aligned,
3 project       char (32) aligned,
3 tag           char (32) aligned,
2 home_dir      char (168) aligned,  /* home directory */
2 process_dir_name char (32) aligned,  /* name of process directory */
2 wdir          (0:7) ptr,  /* pointers to per-ring working directories */
2 wdir_uid      (0:7) bit (36) aligned,  /* UID of per-ring working directories */
2 transparent   bit (36) aligned,  /* transparent usage, mod, pd switch */
2 ltt_head      bit (18) aligned,  /* top of present LTT list */
2 max_access_authorization aligned like aim_template,  /* max authorization this user can attain */
2 stacks        (0:7) ptr,  /* per-ring stack pointers */
2 kstp          ptr,  /* pointer to start of KST */
2 events_pending bit (36) aligned,  /* special wakeups-pending */
2 special_channels bit (36) aligned,  /* special channels assigned */
2 event_masks    (0:7) bit (36) aligned,  /* per-ring mask for special channels */
2 initial_ring  fixed bin (3),  /* initial ring of execution for the process */
2 interrupt_ring fixed bin (3),  /* lowest ring in which IPS interrupts are allowed */
2 highest_ring  fixed bin (3),  /* highest ring in which process can run */
2 prelinked_ring
2 unique_scu_index
2 max_lot_size
2 lot_stack_size
2 clr_stack_size
2 network_ptbl_idx
2 link_meters_bins
2 link_meters_times
2 link_meters_pgwaits
2 dmpr_copy_dirsegp
2 dmpr_pvid
2 dmpr_pvtx
2 first_call
2 mc_save_area
2 mc_save_prt
2 mc_save_limit
2 useable_lot
2 ring_events
2 force_write_limit (0:7)

2 ipc_vars
3 ep
3 retsw
3 save_entry_ret
3 truncated_stacks
3 chan
3 block_start_steps
3 atk_temp
2 ipc_block_return
2 avg_block_steps
2 block_lock_count
2 pad_for_data_mod16
2 data

/* bit(i) is ON if ring (i) is prelinked */
/* used to tag MC */
/* size of lot in stack (0 -> lot not in stack */
/* size of CLR in stack */
/* Index into NCP's process table */
/* histograms of linkage faults */
/* histogram of linkage fault times */
/* histogram of linkage faults PF's */
/* ptr to temp segment into which dirs are copied */
/* pvid of volume being dumped */
/* pvtx of volume being dumped */
/* ON until leave ring zero once */
/* rel pointer to start of saved MC area */
/* ptr to next mc save place */
/* max address where MC can be saved */
/* indicates whether lot can be referenced */
/* limit on force-writing */
/* Following must be doubleword aligned */
/* holds state of fast_hc_ipc at block */
/* ipc block return address */
/* count of locks held */
/* to mark end of PDS for MC save area */
/* Indicates whether lot can be referenced */
/* per-ring indicator that itt messages copied to ect */

/* bit (8) aligned, */
/* (36) aligned, */
/* (36) aligned, */
/* (36) aligned, */
/* (36) aligned, */
/* (36) aligned, */
/* (36) aligned, */
/* (36) aligned, */
/* (36) aligned, */
/* ON until leave ring zero once */
/* rel pointer to start of saved MC area */
/* ptr to next mc save place */
/* max address where MC can be saved */
/* indicates whether lot can be referenced */
/* limit on force-writing */
/* Following must be doubleword aligned */
/* holds state of fast_hc_ipc at block */
/* ipc block return address */
/* count of locks held */
/* to mark end of PDS for MC save area */
call get_temp_segment_("pds", pdsp, code);

/* Now begins the initialization */

pds.process_group_id = "Initializer.SysDaemon.z";

pds.access_authorization.categories = (18)"0"b;

pds.access_authorization.level = 0;

pds.access_authorization.dir = "1"b;    /* for initializer */

pds.access_authorization.seg = "1"b;

pds.access_authorization.rnc = "1"b;

pds.access_authorization.ipc = "1"b;

pds.access_authorization.soos = "1"b;

pds.max_access_authorization.categories = (18)"1"b

pds.max_access_authorization.level = 7;

pds.vtime_count = -1;

pds.process_id = (36)"1"b;

pds.lock_id = (36)"1"b;

pds.pl1_machine = 1;

pds.lps_mask (1) = (35)"1"b;

pds.force_write_limit (1) = 1;

pds.save_history_regs, pds.hregs_saved = "0"b;

pds.history_reg_data (1) = 0;

pds.mc_trace_buf = null;

pds.mc_trace_sw = "0"b;

pds.mc_trace_seg = 0;

pds.expovf1_value = unspec (Default_exponent_control_overflow_value);

pds.exp_ovfl_rest, pds.exp_undfl_rest = "0"b;

pds.stack_0_swp = null;

pds.stack_0_ptr = null;

pds.pad_for_trace_mod16 (1) = 0;

trace_ptr = addr (pds.trace);

trace.last_available = divide (hbound (pds.trace, 1) * size (page_trace_entry) - 8, 2, 17, 0);

pds.initial_procedure = null;

pds.lock_array (1).lock_ptr = null;

pds.lock_array (1).caller_ptr = null;

pds.lock_array (1).event = ""b;

pds.access_name.user = "Initializer";

pds.access_name.project = "SysDaemon";

pds.access_name.tag = "Z";

pds.home_dir = ">system_control_1";

pds.process_dir_name = ">process_dir_dir>lzzzzzbBBBBBB";

inc. o.**, hard.source pds.cds
pds.wdir (*) = null;
pds.wdir_uid (*) = "0"b;

pds.stacks (*) = null;
pds.dmpr_pid = "0"b;
pds.dmpr_ptx = 0;
pds.dmpr_copy_dirsegp = null;
pds.kstp = null;
pds.first_call = 1;
pds.initial_ring = 1;
pds.interrupt_ring = 4;
pds.highest_ring = 7;
pds.max_lot_size (*) = 1024;
pds.mc_save_area = rel (addr (pds.data));
pds.mc_save_ptr = rel (addr (pds.data));
pds.mc_save_limit = bit (bin (4096, 18), 18); /* Allow for as many as fit in 4K. */

/* Now make some checks on alignment of certain variables */
call check (addr (pds.ipc_vars), "ipc_vars", 2);
call check (addr (pds.page_fault_data), "page_fault_data", 16);
call check (addr (pds.trace), "trace", 16);
call check (addr (pds.signal_data), "signal_data", 16);
call check (addr (pds.lock_array), "lock_array", 2);
call check (addr (pds.data), "data", 16);
if bin (rel (addr (pds.pad_end_of_page_0)), 18) ~ 1024
then call com_err_ (0, pdsnname, "Wired portion must end at 1024");

/* Now set up call to create data base */
cdsa.sections (1).p = addr (pds);
cdsa.sections (1).len = size (pds);
cdsa.sections (1).struct_name = "pds";

cdsa.num_exclude_names = 1;
cdsa.exclude_array_ptr = addr (exclude_pad);

string (cdsa.swtch_names) = "0"b;
cdsa.swtch_names.have_text = "1"b;

call create_data_segment_ (addr (cdsa), code);
call release_temp_segment_ ("pds", dsp, code);
call hcs_name_file (get_wdir (), "pds", ",", "template_pds", code);
if code ~ 0
then if code = error_table__segnamedup

Include "", hard_source
F80A not to be reproduced
then call com_err_ (code, pdsname, "Unable to add name template_pds.");
check:
    proc (where, message, modulo);
    
    dcl where ptr;
    dcl message .char (*);
    dcl modulo fixed bin;
    dcl remainder fixed bin;

    remainder = mod (bin (rel where), 18), modulo;
    if remainder ^= 0
        then call com_err_(0, pdsname, "The variable "a is "d words away from being aligned on a "d-word boundary.",
            message, (modulo - remainder), modulo);

    end check;
%include cds_args;
%include sys_trace;
%include aim_template;
%include mc;

   end pds:
prds.cds segment in: >ldd>hard>source system id: 36-1 contents modified: 07/29/81 1816.2 entry modified: 03/27/83 1807.1

/* PRDS - The Processor Data Segment and Processor Stack. */
/* Last modified (Date and reason): */
// Last modified (Date and reason): 
2/6/76 by S. Webber Initial coding  
6/15/77 by M. Weaver to null signal and act pointers  
8/25/80 by J. A. Bush for the dps8/70m cpu  
2/22/81 by J. Bongiovanni for fast_connect_code  
6/27/81 by J. Bongiovanni for idle_temp  */

const char copyright[] = { /* Copyright (c) 1972 by Massachusetts Institute of Technology and Honeywell Information Systems, Inc. */
  'C', 'O', 'P', 'Y', 'R', 'I', 'G', 'H', 'T', '(', '(', 'c', ')', '1', '9', '7', '2', 'b', 'y', 'M', 'a', 's', 's', 'a', 'c', 'u', 't', 't', 's', ' ', 'I', 'n', 's', 't', 'i', 't', 'u', 't', 'e', ' ', 'o', 'f', 'T', 'e', 'c', 'h', 'n', 'o', 'l', 'o', 'g', 'y', 'a', 'n', 'd', ' ', 'H', 'o', 'n', 'g', 'w', 'e', 'l', 'l', ' ', 'I', 'n', 'f', 'o', 'r', 'm', 'a', 't', 'i', 'o', 'n', 'S', 'y', 's', 't', 'e', 'm', 's', ',', 'I', 'n', 'c', '.),*/};

prds: proc;
/* This program creates the prds data base */
/* Automatic */
dcl i fixed bin;
dcl cds alkien like cds_args;
dcl code fixed bin (35);
/* Static */
dcl prdsname char (4) aligned static init ("prds") options (constant);
dcl exclude_pad (1) char (32) aligned static options (constant) init ("pad++");
/* The following must correspond to the size of the fast connect code in fast_connect_init */
dcl FAST_CONNECT_CODE_WORDS init (162) fixed bin int static options (constant);
/* Builtins */
dcl (addr, baseptr, bin, mod, null, ptr, rel, size, string, unspec) builtin;
/* Entries */
dcl com_err_entry options (variable);
dcl create_data_segment_entry (ptr, fixed bin (35));
dcl get_temp_segment_entry (char (*), ptr, fixed bin (35));
dcl release_temp_segment_entry (char (*), ptr, fixed bin (35));
dcl prdsp ptr;

dcl 1 prds aligned based (prdsp).
  1 header aligned like stack_header,
  2 interrupt_data aligned like mc,
  2 fim_data aligned like mc,
  2 sys_trouble_data aligned like mc,
  2 ignore_data aligned like scu,
  2 liftemp fixed bin (71),
  2 last Recorded_time fixed bin (71),
  2 idle_ptr ptr,
  2 simulated_mask fixed bin (71),
  2 am_data bit (0),
  2 ptw_am_regs (4*16) fixed bin (35),
  2 ptw_am_ptrs (4*16) fixed bin (35),
  2 sdw_am_regs (4*16) fixed bin (71),
  2 sdw_am_ptrs (4*16) fixed bin (35),
  2 processor_pattern bit (8) aligned,
  2 processor_tag fixed bin (3),
  2 last_timer_setting bit (27) aligned,
  2 depth fixed bin,
  2 mode_reg bit (36) aligned,
  2 cache_luf_reg bit (36) aligned,
  2 fault_reg bit (72) aligned,
  2 apt_ptr ptr,
  2 idle_temp fixed bin (71).

/* standard stack header */
/* MC for interrupts */
/* MC for connect faults, timer runouts */
/* MC for saved sys trouble data */
/* for SCU data to be ignored at certain times */
/* temporary used by IT (surprise) */
/* used by traffic control */
/* pointer to idle process APTE for this processor */
/* simulated system controller mask register */
/* to get addr of associative memory data block */
/* page table regs (4 sets of 16 for dps8/70m) */
/* page table pointers (4 sets of 16 for dps8/70m) */
/* segment desc. regs (4 sets of 16 for dps8/70m) */
/* segment desc. pointers (4 sets of 16 for dps8/70m) */
/* 1 bit ON for this processor */
/* CPU tag from maintenance panel */
/* last timer value loaded for this CPU */
/* depth in eligible queue for running process */
/* mode register for this processor */
/* cache mode register for this CPU */
/* place to store the fault register */
/* -> apt running on this cpu */
/* used by idle process */

/* The following contains code used for handling connect faults for this processor */

2 fast_connect_code (FAST_CONNECT_CODE_WORDS) bit (36) aligned,
2 fast_connect_code_end bit (36) aligned,
2 mode_reg_enabled bit (36) aligned,
2 pad_mod_16 (3) fixed bin,
2 processor_stack aligned like stack_frame;
/* first stack frame location */
call get_temp_segment_("prds", prdsp, code);

unspec (prds) = "b;

/* Now make some checks on alignment of certain variables */
call check (addr (prds.idle_ptr), "idle_ptr", 2);
call check (addr (prds.processor_stack), "processor_stack", 16);
call check (addr (prds.ptw_am_regs), "ptw_am_regs", 16);
call check (addr (prds.sdw_am_regs), "sdw_am_regs", 32);
call check (addr (prds.fast_connect_code), "fast_connect_code", 2);

/* Now set up call to create data base */

cdsa.sections (1).p = addr (prds);

"prds":

cdsa.sections (1).len = size (prds);

cdsa.sections (1).struct_name = "prds";

cdsa.seg_name = "prds";

cdsa.num_exclude_names = 1;

cdsa.exclude_array_ptr = addr (exclude_pad);

string (cdsa.switches) = "0";

cdsa.switches.have_text = "1";

"prds", code);

call release_temp_segment_ ("prds", prdsp, code);
check: proc (where, message, modulo);

dcl where ptr;
dcl message char (*);
dcl modulo fixed bin;

    if mod (bin (rel (where), 18), modulo) ^= 0
    then call com_err_ (0, prdsname, "The variable ^a is not aligned on a ^d-word boundary.", message, modulo);

end check;
% include cds_args;
% include stack_header;
% include stack_frame;
% include mc:

    end prds;

not to be reproduced
/* BEGIN INCLUDE FILE ... ptw.168.incl.pl1 ... 02/26/81, for ADP conversion */
/* Note: This include file has an ALM counterpart made with cif. Keep it up to date */

dcl 1 168_core_ptw aligned based (ptp),
    2 frame fixed bin (14) unsigned unaligned,
    2 pad1 bit (4) unaligned,
    2 flags unaligned like 168_ptw_flags;

dcl 1 168_ptw aligned based (ptp),
    2 add bit (18) unaligned,
    2 flags like 168_ptw_flags unaligned;

dcl 1 168_special_ptw aligned based (ptp) like 168_ptw;
    2 add_type bit (4),
    2 first bit (1),
    2 er bit (1),
    2 pad1 bit (1),
    2 unusable bit (1),
    2 phu bit (1),
    2 phm1 bit (1),
    2 nypd bit (1),
    2 phm bit (1),
    2 phul bit (1),
    2 wired bit (1),
    2 os bit (1),
    2 valid bit (1),
    2 df_no bit (2)) unaligned;

/* In-core page descriptor */
/* Core frame number */
/* General declaration for out-of-core PTW */
/* Page is somewhere peculiar -- add_type = "01"b */
/* PTW for page actually on disk -- add_type = "10"b */
/* PTW for page not yet on disk -- add_type = "11"b */
/* Various software/hardware flags */
/* 0000=null, 1000=core, 0100=disk, 0010=pd, 0001=swap */
/* the page has not yet been written out */
/* error on last page I/O (also used by post-purge as temp) */
/* can't be used because hardware resets this bit */
/* page has been used bit */
/* Cumulative OR of hardware phm's */
/* must be moved to paging device */
/* page has been modified bit */
/* page has been used in the quantum */
/* page is to remain in core */
/* page is out-of-service (I/O in progress) */
/* directed fault if this is 0 (page not in core) */
/* directed fault number for page faults */

/* END INCLUDE FILE ... ptw.168.incl.pl1 */
/* BEGIN INCLUDE FILE ... pv_holdt.incl.pl1 ... */

dcl pv_holdtp ptr;
dcl 1 pv_holdt (1 : 64) based (pv_holdtp) aligned.
  2 pvtx fixed bin(17) unaligned,
  2 apterp bit(18) unaligned;

/* END INCLUDE FILE ... pv_holdt.incl.pl1 ... */
The physical volume table (PVT) is a wired-down table. It has one entry for each spindle present, be it for Storage System or *I/O* use.

```c
#include <hard.
source pvt. tncl .p11>
```

```c
dcl pvt$ ext,
pvt ptr;

dcl 1 pvt based (pvt$) aligned,
    n_entries fixed bin (17), /* number of PVT entries */
    max_n_entries fixed bin (17), /* max number of PVT entries */
    n_in_use fixed bin (17), /* number of PVT entries in use */
    rwun_pvtx fixed bin, /* rewind_unloading pvtx */
    shutdown_state fixed bin, /* state of previous shutdown */
    esd_state fixed bin, /* state of ESD, >0 iff in ESD */
    prev_shutdown_state fixed bin, /* shutdown state of previous bootload */
    prev_esd_state fixed bin, /* ESD state of previous bootload */
    time_of_bootload fixed bin (71), /* Time of bootload */
    root_lvid bit (36) aligned, /* Logical volume ID of Root Logical Volume (RLV) */
    root_pvid bit (36) aligned, /* Physical volume ID of Root Physical Volume (RPV) */
    root_pvtx fixed bin, /* VTOCE index for root (> ) */
    root_vtocx fixed bin, /* VTOCE index for disk table on RPV */
    disk_table_vtocx fixed bin, /* File System UID for disk table */
    disk_table_uid bit (36) aligned, /* Time a VTOC Map was last locked/unlocked */
    rpv_requested bit (t) aligned, /* RPVS keyword given on BOOT */
    rpv_needs_salv bit (t) aligned, /* RPV required (not requested) salvage */
    prev_vtoc_lock_wait bit (36) aligned, /* For constructing wait event: OR pvte_re1 into lower */
    prev_vtoc_idle_wait bit (36) aligned, /* For constructing wait event: OR pvte_re1 into lower */
    prev_vtoc_map_lock_wait bit (36) aligned, /* For constructing wait event: OR pvte_re1 into lower */
    n_volmap holding fixed bin (17), /* Current number of volmap locks held */
    n_vtoc_map_locks fixed bin (35), /* Current number of VTOC Map locks held */
    last_volmap_time fixed bin (71), /* Time a volmap was last locked/unlocked */
    last_vtoc_map_time fixed bin (71), /* Time a VTOC Map was last locked/unlocked */
    total_volmap_time fixed bin (71), /* Total time volmap's were locked (integral) */
    total_vtoc_map_lock_time fixed bin (71), /* Total time VTOC Maps were locked (integral) */
    n_volmap_locks fixed bin (35), /* Number times a volmap was locked */
    n_vtoc_map_locks fixed bin (35), /* Number times a vtoc_map was locked */
    volmap_lock_nowait_calls fixed bin (35), /* Number calls to lock volmap, no wait */
    volmap_lock_nowait_fails fixed bin (35), /* Number times lock failed */
```
2 volmap_lock_wait_calls fixed bin (35),
2 volmap_lock_wait_fails fixed bin (35),
2 pad (2) bit (36) aligned,
2 array fixed bin (71);

/* Number calls to lock volmap, wait */
/* Number times lock failed */
/* Array of PVTE's -- must be double-word aligned */
START OF: pvte.incl.p11  July 1982 */

dcl pvt$array aligned external;
dcl pvt$max_n_entries fixed bin external;

dcl pvt_arrayp ptr;
dcl pvtep ptr;

dcl 1 pvt_array (pvt$max_n_entries) aligned like pvte based (pvt_arrayp);

dcl 1 pvte based (pvtep) aligned.

2 pvtd bit (36), /* physical volume ID */
2 lvid bit (36), /* logical volume ID */

2 dmpr_in_use (3) bit (1) unaligned, /* physical volume dumper interlock */
2 pad2 bit (6) unaligned.

2 skip_queue_count fixed bin (18) unsigned unaligned, /* number of times this pv skipped for per-proc allocation due to saturation */
2 brother_pvtx fixed bin (8) unaligned, /* next pvte in lv chain */

2 devname char (4), /* device name */

(2 device_type fixed bin (8), /* device type */
2 logical_area_number fixed bin (8), /* disk drive number */
2 used bit (1), /* TRUE if this entry is used */
2 storage_system bit (1), /* TRUE for storage system (vs lo disk) */
2 permanent bit (1), /* TRUE if cannot be demounted */
2 testing bit (1), /* Protocol bit for read_disk$test */
2 being_mounted bit (1), /* TRUE if the physical volume is being mounted */
2 being_demounted bit (1), /* TRUE if the physical volume is being demounted */
2 check_read_incomplete bit (1), /* page control should check read incomplete */
2 device_inoperative bit (1), /* TRUE if disk_control decides dev busted */
2 rpv bit (1), /* TRUE if this is the root physical volume */
2 scav_check_address bit (1), /* TRUE is page control should check deposits/withdrawals against scaven
g table */
2 deposit_to_volmap bit (1), /* TRUE if deposits should got to volume map, not stock */
2 being_demounted2 bit (1), /* No more vtoc 1/0 during demount */
2 pad5 bit (1), /* don't put new segs on this vol */
2 hc_part_used bit (1), /* HC part set up by init_pvte */

include.*. hard.source pvte.incl.p11
2 volmap_lock_notify bit (1) unaligned, /* TRUE if notify required when volmap lock is unlocked */
2 volmap_idle_notify bit (1) unaligned, /* TRUE if notify required when volmap state is idle */
2 vtoc_map_lock_notify bit (1) unaligned, /* TRUE if notify required when vtoc map lock is unlocked */
2 n_free_vtoc fixed bin (17), /* number of free VTOC entries */
2 vtoc_size fixed bin (17), /* size of the VTOC part of the disk - in records */
2 dbmrp (2) bit (18), /* rel ptr to dbmrp bit maps for this volume */
2 nieft fixed bin (17), /* number of records left */
2 totrec fixed bin (17) unaligned, /* Total records in this map */
2 dim_info bit (36), /* Information peculiar to DIM */
2 curr_dmpr_vtocx (3) fixed bin unaligned, /* current vtoce being dumped */
2 n_vtoc fixed bin unaligned, /* number of vtoce on this volume */
2 baseadd fixed bin (18) unaligned, /* Base of paging region */
2 pad2 bit (18) unaligned,
2 volmap_seg_sdw fixed bin (71), /* SDW describing volmap_seg */
2 volmap_aste fixed bin (17), /* Packed pointer to ASTE for volmap_seg */
2 volmap_offset bit (18) unaligned, /* Offset in volmap_seg of volume map */
2 vtoc_map_offset bit (18) unaligned, /* Offset in volmap_seg of VTOC map */
2 volmap_lock bit (36) aligned, /* Lock on volume map operations */
2 vtoc_map_lock bit (36) aligned, /* Lock on VTOC map operations */
2 volmap_stock_ptr ptr unaligned, /* Packed pointer to record stock */
2 vtoc_map_stock_ptr ptr unaligned, /* Packed pointer to VTOC stock */
2 volmap_async_state fixed bin (17) unaligned, /* Asynchronous update state of Volume Map */
2 volmap_async_page fixed bin (17) unaligned, /* Page number for asynchronous update */
2 vol_trouble_count fixed bin (17) unaligned, /* Count of inconsistencies since last salvage */
2 scavenger_block_rel bit (18) unaligned; /* Offset to scavenger block, 0-> scavenging */

#include "hard.source" pvte.incl.pl1

VOLMAP_ASYNC_IDLE init (0), /* for volmap_async_state */
VOLMAP_ASYNC_READ init (1),
VOLMAP_ASYNC_WRITE init (2) fixed bin int static options (constant);

/* END OF: */ pvte.incl.pl1
/* BEGIN INCLUDE FILE RNT.INCL.P1 - WRITTEN SEPTEMBER 1974 BY R. BRATT */
/* modified July 1976 by R. Bratt; updated March 1977 by M. Weaver */
/* modified November 1977 by M. Weaver to use PL/I offsets instead of pointers */

dcl (rntp, rntep) ptr;
dcl lth fixed bin (17);
dcl based_rnt_area area based;

dcl 1 rnt aligned based (rntp),
    2 areap ptr, /* pointer to area for rnte allocations */
    2 meters,
    3 insert,
    4 trys fixed bin (17) unaligned,
    4 wins fixed bin (17) unaligned,
    3 get_segno like insert,
    3 get_refnames like insert,
    3 delete_segno like insert,
    3 delete_name like insert,
    2 rnt_area_size fixed bin,
    2 srulep ptr,
    2 name_hash_table (0:127) offset (rnt.areap -> based_rnt_area),
    2 segno_hash_table (0:127) offset (rnt.areap -> based_rnt_area);

dcl 1 rnte aligned based (rntep),
    2 name_fp offset (rnt.areap -> based_rnt_area),
    2 segno_fp offset (rnt.areap -> based_rnt_area),
    (2 segno fixed bin (17),
     2 length fixed bin (17),
     2 name char (1th refer (rnte.length)))unaligned;

/* ------------------------END RNT.INCL.P1------------------------ */
/* START OF: scavenger_data.incl.pl1 November 1982 */

dcl scavenger_data$ external;

dcl scavenger_datap ptr;
dcl sc_metersp ptr;
dcl sc_process_tablep ptr;
dcl scavenger_blockp ptr;
dcl record_blockp ptr;
dcl scavenger_optionsp ptr;

dcl sc_n_processes fixed bin;
dcl scavenger_n_records fixed bin;
dcl scavenger_n_ovfl fixed bin;

dcl 1 scavenger_data aligned based (scavenger_datap),
  2 lock aligned,
  3 lock_word bit (36) aligned,
  3 wait_event bit (36) aligned,
  3 notify_sw bit (1) aligned,
  2 process_table_ptr ptr,
  2 error_severity fixed bin,
  2 meters aligned like sc_meters,
  2 free bit (1) aligned;

dcl 1 sc_meters aligned based (sc_metersp),
  2 n_scavenge fixed bin (35),
  2 pf fixed bin (35),
  2 vcpu fixed bin (71),
  2 clock_time fixed bin (71),
  2 n_vtoses fixed bin (35),
  2 n_vtoses_damaged fixed bin (35),
  2 n_vtoses_per_proc fixed bin (35),
  2 n_vtoses_per_boot fixed bin (35),
  2 n_vtoses_freed fixed bin (35),
  2 n_vtoses_fmd fixed bin (35),
  2 n_records fixed bin (35),
  2 n_conflicts fixed bin (35),
  2 n_fmd_conflicts fixed bin (35),
  2 n_real_conflicts fixed bin (35),
  2 n_lost_records fixed bin (35);

/* Meters */
/* Number of volume scavenges */
/* Total page faults */
/* Total virtual CPU time */
/* Total clock time */
/* Number VTOCEs examined */
/* Number VTOCEs damaged by scavenge */
/* Number per-process VTOCEs freed */
/* Number per-bootload VTOCEs freed */
/* Total number VTOCEs freed */
/* Number VTOCEs with fm_damaged reset */
/* Number non-null filemap entries examined */
/* Number potential conflicts detected */
/* Number potential conflicts due to fm_damaged */
/* Number real conflicts */
/* Number lost records freed */
dcl 1 sc_process_table aligned based (sc_process_table),
 2 max_n_processes fixed bin, /* Number of table entries */
 2 n_processes fixed bin, /* Number active entries */
 2 process (sc_n_processes refer (sc_process_table.max_n_processes)) aligned,
 3 processid bit (36) aligned, /* Owner. 0=>empty */
 3 pvtep ptr unal, /* VTOCE of volume being scavenged */
 3 blockp ptr unal, /* Block w/ i scavenger_data */
 3 first_block_page fixed bin, /* Index of first page of block */
 3 n_block_pages fixed bin; /* Number of pages in block */

dcl 1 scavenger_block aligned based (scavenger_blockp),
 2 n_records fixed bin, /* Number of record addresses */
 2 n_ovfl fixed bin, /* Number of overflow blocks */
 2 ovfl_free_ix fixed bin, /* Index of first free overflow block */
 2 records (scavenger_n_records refer (scavenger_block.n_records)) aligned like record_block,
 2 overflow (scavenger_n_ovfl refer (scavenger_block.n_ovfl)) aligned like record_block;

dcl 1 record_block aligned based (record_blockp), /* One per record address */
 2 vtocx fixed bin (15) uns unal, /* Owning VTOCE index */
 2 pageno fixed bin (8) uns unal, /* Owning page number */
 2 state fixed bin (2) uns unal, /* State */
 2 lock bit (1) unal, /* Lock bit on this block */
 2 ovflx fixed bin (10) uns unal; /* Index of first overflow block on chain */

dcl 1 scavenger_data_pages
 2 page (0:255) aligned,
 3 word (0:255) aligned;

dcl 1 scavenger_options aligned based (scavenger_optionsp),
 2 print_meters bit (1) unaligned, /* ON => meter each scavenge into the log */
 2 debug bit (1) unaligned, /* ON => do special debugging things */
 2 dump bit (1) unaligned, /* ON => dump bad VTOCEs into syserr log */
 2 trap bit (1) unaligned, /* ON => trap to BOS for debug */
 2 no_optimize bit (1) unaligned; /* ON => no VTOCE read-ahead */

dcl
  STATE_UNSEEN init (0).
  STATE_FREE init (1).
  STATE_IN_USE init (2).
  STATE_CONFLICT init (3)
) fixed bin int static options (constant);

/* END OF: scavenger_data.incl.pl1 */
/* BEGIN INCLUDE FILE scs.incl.pll ... April 1982 */

/* Information about system controllers */
dcl 1 scs$controller_data (0:7) aligned ext,
   2 size fixed bin (17) unaligned,
   2 base fixed bin (17) unaligned,
   2 elma_data (4) unaligned,
   3 mask_available bit (1) unaligned,
   3 mask_assigned bit (1) unaligned,
   3 mbz bit (3) unaligned,
   3 mask_assignment fixed bin (3) unaligned,
   2 info aligned,
   3 online bit (1) unaligned,
   3 offline bit (1) unaligned,
   3 store_a_online bit (1) unaligned,
   3 store_a1_online bit (1) unaligned,
   3 store_b_online bit (1) unaligned,
   3 store_b1_online bit (1) unaligned,
   3 store_b_is_lower bit (1) unaligned,
   3 ext_interlaced bit (1) unaligned,
   3 int_interlaced bit (1) unaligned,
   2 four_word bit (1) unaligned,
   3 cyclic_priority (7) bit (1) unaligned,
   3 type bit (4) unaligned,
   3 abs_wired bit (1) unaligned,
   3 program bit (1) unaligned,
   3 mbz bit (13) unaligned,
   2 lower_store_size fixed bin (17) unaligned,
   2 upper_store_size fixed bin (17) unaligned;
   /* per-controller info */
   /* size (in 1024 word blocks) of this controller */
   /* abs address (0 mod 1024) for base of this controller */
   /* EIMA information for this controller */
   /* ON if corresponding mask exists */
   /* ON if mask assigned to a port */
   /* port to which mask is assigned */
   /* ON if controller is online */
   /* ON if controller is offline but can be added */
   /* ON if store A is online */
   /* ON if store A1 is online */
   /* ON if store B is online */
   /* ON if store B is online */
   /* ON if store B is lower */
   /* ON if this SCU is interlaced with other SCU */
   /* ON if this SCU is internally interlaced */
   /* ON if external interface is 4-word */
   /* Cyclic priority for adjacent ports */
   /* Model number for this controller */
   /* ON if controller can have abs_wired pages */
   /* PROGRAM/MANUAL switch setting */
   /* size (in 1024 word blocks) of lower store */
   /* size (in 1024 word blocks) of upper store */

/* Information about CPUs */
dcl 1 scs$processor_data (0:7) aligned ext,
   (/* Information about CPUs in the system */
   2 online bit (1),
   2 offline bit (1),
   2 release_mask bit (1),
   2 accept_mask bit (1),
   2 delete_cpu bit (1),
   2 interrupt_cpu bit (1),
   2 halted_cpu bit (1),
   2 cpu_type fixed bin (2) unsigned,
   2 mbz bit (21),
   2 expanded_port bit (1),
   2 expander_port fixed bin (2) unsigned,
   2 controller_port fixed bin (3) unsigned ) unaligned;
   /* *"b if CPU is online */
   /* *"b if CPU is offline but can be added */
   /* *"b is this CPU is to give up its mask */
   /* *"b if this CPU is to grab mask in idle loop */
   /* *"b if this CPU is to delete itself */
   /* *"b if this CPU takes hardware interrupts */
   /* *"b if this CPU has stopped itself (going to BDS) */
   /* 0 => DPS or L68, 1 => DPS8 */
   /* *"b = on expanded port */
   /* The actual expander port */
   /* Port on controller */
dcl scs$port_data (0:7) aligned external static,
  2 assigned fixed bin (4) unsigned unaligned,
  2 expander_port bit (1) unaligned,
  2 expanded_cpu (0:3) bit (1) unaligned,
  2 cpu_number fixed bin (3) unsigned unaligned,
  2 pad bit (12) unaligned;

dcl scs$cow (0:7) aligned external,
  2 pad bit (36) aligned,
  2 cow,
    3 sub_mask bit (8) unaligned,
    3 mbz1 bit (13) unaligned,
    3 expander_command bit (3) unaligned,
    3 mbz2 bit (2) unaligned,
    3 expanded_port bit (1) unaligned,
    3 expander_port fixed bin (3) unsigned unaligned,
    3 controller_port fixed bin (3) unaligned unsigned;

dcl scs$cow_ptrs (0:7) external aligned,
  2 rei_cow_ptr bit (18) unaligned,
  2 pad bit (12) unaligned,
  2 tag bit (6) unaligned;

dcl scs$reconfig_general_cow aligned external,
  2 pad bit (36) aligned,
  2 cow,
    3 sub_mask bit (8) unaligned,
    3 mbz1 bit (13) unaligned,
    3 expander_command bit (3) unaligned,
    3 mbz2 bit (2) unaligned,
    3 controller_port fixed bin (3) unsigned unsigned;
// controller port for this CPU */

dcl scs$sys_level bit (72) aligned ext;

include.**, hard.source scs.incl.pl1
dcl scs$mask_ptr (0:7) ptr unaligned ext;          /* pointers for real or simulated masks */
/* MISCELLANEOUS */
dcl 1 scs$processor_test_data aligned ext;
     (                                                   /* info used for cpu testing */
         2 active bit (1),                                /* *"b if cpu currently under test */
         2 scu_state bit (2),                              /* state of scu being used for testing (see definition below) */
         2 padi bit (15),                                  /* tag of cpu under test */
         2 cpu_tag fixed bin (5),                          /* tag of scu being used for cpu testing */
         2 scu_tag fixed bin (5),                          /* tag of active cpu that has mask assigned to above scu */
     ) unaligned;

/* scu_state = "00"b => SCU defined by scs$processor_test_data. scu_tag not yet effected */
/* scu_state = "01"b => all core removed from SCU, port mask not yet changed */
/* scu_state = "10"b => all core removed from SCU, port mask changed */
/* scu_state = "11"b => only 64k at base of SCU being used for testing, original port mask restored */
dcl scs$idle_aptep (0:7) ptr unaligned ext;        /* pointer to idle process APTE for each processor */
dcl scs$connect_lock bit (36) aligned ext;         /* lock for sending connects */
dcl scs$reconfig_lock bit (36) aligned ext;        /* Lock used during reconfiguration */
dcl scs$trouble_flags bit (8) aligned ext;         /* checkoff flags for sys_trouble stopping */
dcl scs$bos_restart_flags bit (8) aligned ext;     /* checkoff flags for restarting after sys_trouble */
dcl scs$processors fixed bin (3) ext;              /* number of running processors */
dcl scs$bos_processor_tag fixed bin (3) ext;       /* CPU tag of processor running BOS */
dcl scs$faults_initialized bit (1) aligned ext;    /* ON after faults have been enabled */
dcl scs$sys_trouble_pending bit (1) aligned ext;   /* sys_trouble event is pending in the system */
dcl scs$fast_cam_pending (0:7) bit (36) aligned ext; /* checkoff cells for cam connect */
dcl scs$interrupt_controller fixed bin (3) ext;   /* port number of low order controller */
dcl scs$processor_start_int_no fixed bin (5) ext;  /* interrupt cell for starting a processor */
dcl scs$processor_bit (8) aligned ext;             /* bits ON for online CPUs */
dcl scs$processor_start_wait bit (8) aligned ext;  /* checkoff flags for waiting for new processor */
dcl scs$trouble_dbrs (0:7) fixed bin (71);         /* DBR values at system crash time */
dcl scs$port_addressing_word (0:7) bit (3) aligned ext; /* active module port number for each controller */
dcl scs$cfg_data (0:7) fixed bin (71) aligned ext; /* RSCR-CFG data from each controller */
dcl scs$cfg_data_save fixed bin (71) aligned ext;  /* RSCR-CFG save area for ISOLTS CPU testing */
dcl scs$expanded_ports bit (1) unaligned dim (0:7) external;  /* Which ports have expanders */
dcl scs$processor_switch_data (0:4) bit (36) aligned ext; /* raw data from RSW 0 thru 4 */
dcl scs$processor_switch_template (0:4) bit (36) aligned ext; /* expected data from RSW 0 thru 4 */
dcl scs$processor_switch_compare (0:4) bit (36) aligned ext; /* discrepancies from expected data */
dcl scs$processor_switch_mask (0:4) bit (36) aligned ext; /* masks for comparing switch data */
dcl scs$controller_switch_value bit (36) aligned ext; /* Correct value for CPU data switches */
dcl scs$controller_config_size (0:7) fixed bin (14) aligned ext; /* Controller size on config card */

#include "*. hard.source" scs.incl.pl1

// Not to be reproduced
dcl scs$reconfig_locker_id char (32) aligned ext;  /* process group ID of process doing reconfiguration */

dcl scs$scas_page_table (0:31) bit (36) aligned external static;  /* PIWs for SCAS pages */

dcl scs$cycle_priority_template bit (7) aligned ext;  /* template for setting anti-hog switches */

dcl scs$set_cycle_switches bit (1) aligned ext;  /* flag to set anti-hog switches */

dcl (  
IOM_PORT init (1),  
CPU_PORT init (2),  
BULK_PORT init (3)  
) fixed bin int static options (constant);  /* values for scs$port_data assigned */

/* END INCLUDE FILE scs.incl.plt */
/* BEGIN INCLUDE FILE ... sdw.168.incl.pl1 ... Updated for ADP conversion 03/01/81 */

/* Note: This include file has an ALM counterpart made with cift. Keep it up to date */

dcl 1 168_sdw based (sdwp) aligned,

\( \begin{align*} 
(2 \text{ add bit (24)}, \\
2 \text{ rings}, \\
3 \text{ r1 bit (3)}, \\
3 \text{ r2 bit (3)}, \\
3 \text{ r3 bit (3)}, \\
2 \text{ valid bit (1)}, \\
2 \text{ df_no bit (2)}, \\
2 \text{ pad1 bit (1)}, \\
2 \text{ bound bit (14)}, \\
2 \text{ access}, \\
3 \text{ read bit (1)}, \\
3 \text{ execute bit (1)}, \\
3 \text{ write bit (1)}, \\
3 \text{ privileged bit (1)}, \\
2 \text{ unpaged bit (1)}, \\
2 \text{ not_a gate bit (1)}, \\
2 \text{ cache bit (1)}, \\
2 \text{ entry_bound bit (14) unaligned;}
\end{align*} \)

/* Level 68 Segment Descriptor Word */

/* main memory address of page table */

/* ring brackets for the segment */

/* directed fault bit (0 \to fault) */

/* directed fault number */

/* boundary field (in 16 word blocks) */

/* access bits */

/* read permission bit */

/* execute permission bit */

/* write permission bit */

/* privileged bit */

/* segment is unpaged if this is 1 */

/* if this is 0 the entry bound is checked by hardware */

/* cache enable bit */

/* entry bound */

/* END INCLUDE FILE ... sdw.168.incl.pl1 */
/* BEGIN INCLUDE FILE ... sdw_info.incl.pl1 ... 12/16/80, for ADP conversion */
/* Note: This include file has an ALM counterpart made with cif. Keep it up to date */

dcl 1 sdw_info_ptr pointer;

dcl 1 sdw_info aligned based (sdw_info_ptr), 2 address fixed bin (26), 2 size fixed bin (19), 2 access unaligned, 3 read bit (1) unaligned, 3 execute bit (1) unaligned, 3 write bit (1) unaligned, 3 privileged bit (1) unaligned, 2 pad1 bit (32) unaligned, 2 rings unaligned, 3 r1 bit (3) unaligned, 3 r2 bit (3) unaligned, 3 r3 bit (3) unaligned, 2 pad2 bit (27) unaligned, 2 flags aligned, 3 paged bit (1) unaligned, 3 faulted bit (1) unaligned, 3 cache bit (1) unaligned, 3 pad3 bit (33) unaligned, 2 gate_entry_bound fixed bin (14); 2 paged bit (1) unaligned, 3 faulted bit (1) unaligned, 3 cache bit (1) unaligned, 3 pad3 bit (33) unaligned, 2 gate_entry_bound fixed bin (14); 2 paged bit (1) unaligned, 3 faulted bit (1) unaligned, 3 cache bit (1) unaligned, 3 pad3 bit (33) unaligned, 2 gate_entry_bound fixed bin (14); /* Structure describing SDW contents */ /* Address of seg base or of page table */ /* Max length of segment (NOT offset of last word) */ /* REWP */ /* Ring brackets */ /* Number of entrypoints in gate, or zero */

/* END INCLUDE FILE ... sdw_info.incl.pl1 */
BEGIN INCLUDE FILE ... signaller_stack.incl.pl1 ... Created Feb 79 by D.Spector */

This file matches signaller_stack.incl alm and is currently used only by verify_lock */

declare 1 signaller_stack based unaligned,
2 pad (8) bit (36),  /* Make machine conditions 0 mod 16 */
2 mach_cond (48) bit (36),  /* Machine conditions */
2 mc_ptr ptr aligned,  /* Pointer to machine conditions */
2 null_ptr ptr aligned,  /* Null pointer */
2 string_descriptor bit (36),  /* Condition name descriptor */
2 ptr_descriptor bit (36),  /* M.C. ptr descriptor */
2 arglist (18) bit (36),  /* Arg list for call to signal */
2 on_unit (16) bit (36),  /* Condition name */
2 string_char (32),  /* Must be at 128 in stack frame */
2 history_registers (128) bit (36);

/* on_unit must start at 128 because trap_caller_caller sets up a stack frame
assuming this to be so. Similarly mach_cond must start at 48. */

/* END INCLUDE FILE ... signaller_stack.incl.pl1 ... */
/* BEGIN INCLUDE FILE s1t.incl.pl1 --- Last modified 2/76 SHW */

/* Declarations for Segment Loading Table header and array.
Used by Initialization and MST Checker subroutines */

dcl sltp ptr,
    names_ptr ptr,
    namep ptr,
    pathp ptr,
    aclp ptr;

declare 1 slt based (sltp) aligned,
    2 name_seg_ptr ptr,
    2 free_core_start fixed bin (24),
    2 first_sup_seg fixed bin (18),
    2 last_sup_seg fixed bin (18),
    2 first_init_seg fixed bin (18),
    2 last_init_seg fixed bin (18),
    2 free_core_size fixed bin (24),
    2 seg (0:8191) aligned,
    3 site (4) fixed bin (35);

declare 1 name_seg based (names_ptr) aligned,
    2 pad bit (18) unal,
    2 next_loc bit (18) unal,
    2 ht (0:127) bit (18) aligned;

declare 1 segnam based (namep) aligned,
    2 count fixed bin (17),
    2 names (50 refer (segnam.count)),
    3 hp bit (18) unal,
    3 ref bit (1) unal,
    3 pad bit (5) unal,
    3 segno bit (12) unal,
    3 name char (32) unal;

declare 1 path based (pathp) aligned,
    2 size fixed bin (17),
    2 name char (168 refer (path.size)) unal,
    2 aclsize fixed bin;

declare 1 aclsize based (aclp) aligned,
    2 count fixed bin,
    2 acl (50 refer (aclsize.count)),
    3 userid char (32),
    3 mode bit (36) aligned,

    /* declaration of Segment Loading Table (SLT) */
    /* pointer to base of SLT segment */
    /* pointer to base of SLT names segment */
    /* pointer to segment name list block */
    /* pointer to segment's directory path name */
    /* pointer to acl structure */

    /* auxiliary segment of SLT for storing of segment names and directory path names */

    declare 1 name_seg header +
    2 pad bit (18) unal,
    2 next available free location in name seg +
    2 ht (0:127) bit (18) aligned +

    declare 1 names hash table +
    2 count fixed bin (17),
    2 names (50 refer (segnam.count)),
    3 hp bit (18) unal,
    3 ref bit (1) unal,
    3 pad bit (5) unal,
    3 segno bit (12) unal,
    3 name char (32) unal;

    declare 1 segment number associated with this name +
    2 space for name (max 32 characters) +

    declare 1 directory path name +
    2 size fixed bin (17),
    2 name char (168 refer (path.size)) unal,
    2 aclsize fixed bin;

    declare 1 acl size +
    2 count fixed bin,
    2 acl (50 refer (aclsize.count)),
    3 userid char (32),
    3 mode bit (36) aligned,
3 pad bit (36) aligned,
3 code fixed bin;

/* END INCLUDE FILE slt.incl.pl1 */
/** BEGIN INCLUDE FILE site.incl.pll */
/* Declaration for Segment Loading Table Entry structure. */
/* Used by Initialization, MST Generation, and MST Checker subroutines */
/* last modified 5/4/76 by Noel I. Morris */
/* format: style3 */

dcl site_uns based (sitep) aligned,
( 2 names_ptr bit (18),
 2 path_ptr bit (18),
 /* rel pointer to thread of names */
 /* rel pointer to pathname (if present) */

/***** End of word 1 */
 2 access bit (4),
 2 cache bit (1),
 2 abs_seg bit (1),
 2 firmware_seg bit (1),
 2 layout_seg bit (1),
 2 pad4 bit (4),
 2 wired bit (1),
 2 paged bit (1),
 2 par_process bit (1),
 2 acl_provided bit (1),
/* ON if acl structure follows path_name on MST */

/***** End of 1st half of word 2 */
 2 pad5 bit (6),
 2 branch_required bit (1),
 2 init_seg bit (1),
 2 temp_seg bit (1),
 2 link_provided bit (1),
 2 link_sector bit (1),
 2 link_sector_wired bit (1),
 2 combine_link bit (1),
 2 pre_linked bit (1),
 2 defs bit (1),
/* segment is definitions segment if ON */

/***** End of word 2 */
 2 pad7 bit (3),
 2 cur_length fixed bin (9) uns,
 2 ringbrack (3) fixed bin (3) uns,
 2 segno fixed bin (18) uns,
/* current length of segment (in 1024 word blocks) */
/* ringbrackets */
/* text/link segment number */

/***** End of word 3 */
 2 pad8 bit (3),
 2 max_length fixed bin (9) uns,
 2 bit_count fixed bin (24) uns
) unaligned;
/* bitcount of segment */

dcl site based (sitep) aligned,
( 2 names_ptr bit (18),
 2 path_ptr bit (18),
 2 access bit (4),
 /* SDW access bit (REWP) */
 /* SDW access bit (REWP) */
/* rel pointer to thread of names */
/* rel pointer to pathname (if present) */
/* SDW access bit (REWP) */
2 cache    bit (1),    /* Segment to be allowed in cache */
2 abs_seg   bit (1),    /* segment is an abs seg if ON */
2 firmware_seg bit (1),
2 layout_seg bit (1),
2 pad2      bit (4),
2 wired     bit (1),    /* segment is wired if ON */
2 paged     bit (1),    /* segment is paged if ON */
2 per_process bit (1),    /* segment is per-process if ON */
2 pad3      bit (2),
2 acl_provided bit (1),    /* ON if acl structure follows path_name on MST */
2 pad4      bit (3),
2 branch_required bit (1),
2 init_seg  bit (1),    /* segment is init_seg if ON */
2 temp_seg  bit (1),    /* segment is temp_seg if ON */
2 link_provided bit (1),
2 link_sect  bit (1),    /* segment is linkage segment if ON */
2 link_sect_wired bit (1),    /* linkage segment is wired if ON */
2 combine_link bit (1),    /* linkage is combined if ON */
2 pre_linked bit (1),
2 def  bit (1),
2 pad5      bit (6),
2 cur_length bit (9),    /* current length of segment (in 1024 word blocks) */
2 ringbrack (3) bit (3),    /* ringbrackets */
2 segno     bit (18),    /* text/link segment number */
2 pad6      bit (3),
2 max_length bit (9),
2 bit_count bit (24),    /* maximum length for segment */
) unaligned;
/* bitcount of segment */

/* END INCLUDE FILE site.incl.p11 */
/* BEGIN INCLUDE FILE ... sst.incl.pl1 ... January 1971 */
/* Note: This include file has an ALM counterpart made with cif. Keep it up to date */

dcl sst_segs external;
dcl sstp ptr;

dcl 1 sst based (sstp) aligned,
2 space (8) fixed bin,

/* empty space to watch for bugs */

/* SST HEADER */

2 pre_page_time fixed bin (71),
2 post_purge_time fixed bin (71),
2 post_in_core fixed bin,
2 thrashing fixed bin,
2 npfs_misses fixed bin,
2 salv fixed bin,

2 ptl bit (36),
2 astl bit (36),
2 astl_event bit (36),
2 astl_notify_requested bit (1) aligned,
2 nused fixed bin,
2 ptwbase fixed bin (24),
2 tfree ptr,

2 astap ptr,
2 bulk_pvtx fixed bin (8) aligned,
2 ptt_wait_ct fixed bin,
2 astsize fixed bin,
2 cmesize fixed bin,
2 root_astep ptr,

2 pts (0: 3) fixed bin,
2 level (0: 3),
3 (ausedp, no_aste) bit (18) unaligned,

2 (atemp, atemppl) bit (18) unaligned,
2 dm_enabled bit (1) aligned,
2 (ainitp, ainitpl) bit (18) unaligned,
2 strsize fixed bin,

/* CORE MAP HEADER */

2 cmp ptr,
2 usedp bit (18),
2 wtct fixed bin,

/* pointer to first trailer on free list */
/* pointer to first used page table */
/* count of pages being written */
2 startp bit (18),
2 removep bit (18),
2 double_write fixed bin,
2 temp_w_event bit (36) aligned,
2 root_pvtx fixed bin,
2 ptw_first bit (1) aligned,
2 noLock bit (1),
2 x_fsdctp bit (18),
2 fc_skips_pinned fixed bin (35),
2 cl_skips_pinned fixed bin (35),
2 ast_ht_ptr ptr,
2 ast_ht_n_buckets fixed bin,
2 ast_ht_uid_mask bit (36) aligned,
2 meter_ast_locking fixed bin,
2 checksum_filemap fixed bin,
2 page_read_errors fixed bin,
2 page_write_errors fixed bin,
2 rws_read_errors fixed bin,
2 rws_write_errors fixed bin,
2 cycle_pv_allocation fixed bin,
2 n_trailers fixed bin,
2 synch_activations fixed bin (35),
2 synch_skips fixed bin (35),
2 lock_waits fixed bin,
2 total_locks_set fixed bin,
2 pdir_page_faults fixed bin,
2 level_1_page_faults fixed bin,
2 dir_page_faults fixed bin,
2 ring_0_page_faults fixed bin,
2 rqover fixed bin (35),
2 pc_io_waits fixed bin,
2 steps fixed bin,
2 needc fixed bin,
2 ceiling fixed bin,
2 ctwait fixed bin,
2 wired fixed bin,
2 laps fixed bin,
2 skipw fixed bin,
2 n trailers fixed bin,
2 synch_activations fixed bin (35),
2 synch_skips fixed bin (35),
2 lock_waits fixed bin,
2 total_locks_set fixed bin,
2 pdir_page_faults fixed bin,
2 level_1_page_faults fixed bin,
2 dir_page_faults fixed bin,
2 ring_0_page_faults fixed bin,
2 rqover fixed bin (35),
2 pc_io_waits fixed bin,
2 steps fixed bin,
2 needc fixed bin,
2 ceiling fixed bin,
2 ctwait fixed bin,
2 wired fixed bin,
2 laps fixed bin,
2 skipw fixed bin,
2 n trailers fixed bin,
2 synch_activations fixed bin (35),
2 synch_skips fixed bin (35),
2 lock_waits fixed bin,
2 total_locks_set fixed bin,
2 pdir_page_faults fixed bin,
2 level_1_page_faults fixed bin,
2 dir_page_faults fixed bin,
2 ring_0_page_faults fixed bin,
2 rqover fixed bin (35),
2 pc_io_waits fixed bin,
2 * 100 octal */
2 page_read_errors fixed bin,
2 page_write_errors fixed bin,
2 rws_read_errors fixed bin,
2 rws_write_errors fixed bin,
2 cycle_pv_allocation fixed bin,
2 n_trailers fixed bin,
2 synch_activations fixed bin (35),
2 synch_skips fixed bin (35),
2 lock_waits fixed bin,
2 total_locks_set fixed bin,
2 pdir_page_faults fixed bin,
2 level_1_page_faults fixed bin,
2 dir_page_faults fixed bin,
2 ring_0_page_faults fixed bin,
2 rqover fixed bin (35),
2 pc_io_waits fixed bin,
2 steps fixed bin,
2 needc fixed bin,
2 ceiling fixed bin,
2 ctwait fixed bin,
2 wired fixed bin,
2 laps fixed bin,
2 skipw fixed bin,
2 * The following (until pdmap) used to be the 'cnt' in cnt.incl.pl1 */
2 steps fixed bin,
2 needc fixed bin,
2 ceiling fixed bin,
2 ctwait fixed bin,
2 wired fixed bin,
2 laps fixed bin,
2 skipw fixed bin,
2 sktpu fixed bin,
2 sktpm fixed bin,
2 skipo fixed bin,
2 sklpspd fixed bin,
2 aused fixed bin,
2 damaged_ct fixed bin,
2 deact_count fixed bin,
2 demand_deact_attempts fixed bin,
2 demand_deactivations fixed bin,
2 reads (8) fixed bin,
2 writes (8) fixed bin,
2 short_pf_count fixed bin,
2 loop_locks fixed bin,
2 loop_lock_time fixed bin (71),
2 cpu_sf_time fixed bin (71),
2 total_sf_pf fixed bin,
2 total_sf fixed bin,
2 pre_page_size fixed bin,
2 post_list_size fixed bin,
2 post_purging fixed bin,
2 post_purge_calls fixed bin,
2 pre_page_calls fixed bin,
2 pre_page_size_fixed_bin,
2 pre_page_misses fixed bin,
2 pre_paging fixed bin,

/* 200 octal */

/* TEMPORARY WIRED PROCEDURE INFO */

2 wire_proc_data (8) fixed bin (71),

/* MAIN MEMORY USAGE INFORMATION */

2 abs_wired_count fixed bin,
2 system_type fixed bin,
2 wired Copies fixed bin,
2 recopies fixed bin,
2 first_core_block fixed bin,
2 last_core_block fixed bin,
2 fw_retries fixed bin (35),
2 pvhtp ptr unaligned,

/* AST METERS */

2 askips (0: 3) fixed bin,
2 aneedsize (0: 3) fixed bin,
2 stepsa fixed bin,
2 askipsa fixed bin,
2 asearchs fixed bin,
2 askipslevel fixed bin,

/* because of being used */
/* because of being modified */
/* because out of service */
/* number of times a block of core was skipped for active rws */
/* number of AST entries on used list */
/* count of segments that system damaged */
/* count of deactivations */
/* user requested deactivations */
/* user instigated deactivations */
/* number of reads for each did */
/* number of writes for each did */
/* count of page faults on out of service pages */
/* count of times locked PTL */
/* time spent looping on PTL */
/* cpu time spent in seg_fault */
/* total page faults while in seg_fault */
/* total number of seg_faults */
/* total pre-pagings expected */
/* total number of post-purging */
/* total number of calls to post-purge */
/* total number of calls to ppre-page */
/* total number of misses in pre-page list */
/* total number of pre-pagings */
/* count of abs-wired pages */
/* ADP_SYSTEM or LGB_SYSTEM */
/* number of times a wired page was copied */
/* number of times recopied because modified */
/* core map index for last block of core */
/* force_write retries due to ASTE move */
/* ptr to PV hold table for debugging */
/* array of skips because wrong AST size */
/* array of times needed each size */
/* count of steps taken looking for an AST entry */
/* count of skips because EHS was ON */
/* count of full searches made */
/* count of skips because pages were in core */
2 askipsinit fixed bin,
2 acost fixed bin,
2 askipslock fixed bin,
2 asklps fixed bin,
2 alaps fixed bin,
2 updates fixed bin,
2 setfaults all fixed bin,
2 total_bf fixed bin,
2 total_bf_pf fixed bin,
2 cpu_bf_time fixed bin (71),
2 asteps (0: 3) fixed bin,
2 ast_locked_at_time fixed bin (71),
2 ast_locked_total_time fixed bin (71),
2 ast_lock_wait_time fixed bin (71),
2 ast_locking_count fixed bin (35),
2 cleanup_count fixed bin,
2 cleanup_with_any_rws fixed bin,
2 cleanup_rws_count fixed bin,
2 cleanup_real_time fixed bin (71),
2 tree_count (0: 63) fixed bin,
2 pp_meters (0: 63) fixed bin,
2 pp_meters_sst fixed bin,
2 wusedp bit (18) aligned,
2 write_hunts fixed bin,
2 claim_skip_cme fixed bin,
2 claim_skip_free fixed bin,
2 claim_notmod fixed bin,
2 claim_passed_used fixed bin,
2 claim_skip_ptw fixed bin,
2 claim_write fixed bin,
2 claim_steps fixed bin,
2 rss_reads_os fixed bin,
2 pd_updates fixed bin,
2 pd_seeks_fails fixed bin,
2 pd_desperation_steps fixed bin,
2 pd_desperations fixed bin,
2 skips_nypd fixed bin,
2 pd_writehead_bit (1) aligned,
2 pd_desperations_not_mod fixed bin,
/* PRE-PAGE METERS */
/* count of times turned OFF init switch */
/* cumulative cost of deactivations */
/* count of skips because couldn't lock parent */
/* count of deactivations */
/* count of skips because DIUS was on */
/* lap counter for AST list */
/* calls to updateb */
/* setfaults done to the entire SDW */
/* setfaults done to the access field */
/* count of bound faults */
/* page faults during bound faults */
/* cpu time spent in bound fault */
/* per-size AST step counters */
/* clock reading when ast last locked */
/* total real time the ast lock was locked */
/* total real time of all waiting on ast lock */
/* number of times ast was locked */
/* calls to pc$cleanup */
/* ditto, when >0 rws's */
/* total rws's started by cleanup */
/* total real time in pc$cleanup */
/* counters for pre-page decisions */
/* counters for measuring pre-page success */
/* Relative cmp to next cme for writing */
/* Times claim_mod_core invoked */
/* Times unacceptable cme found by c_m_c */
/* Times free cme passed by c_m_c */
/* Times c_m_c passed pure page */
/* Times used page seen */
/* Times c_m_c skip_ptw fixed bin */
/* Writes queued by c_m_c */
/* Steps passed in core claiming */
/* RWS reads outstanding, in SST for debugging */
/* done time pd writes */
/* counter of times quick find_core failed */
/* steps of allocate_pd finding pdme */
/* times allocate_pd needed to force one free */
/* find_core skips for nypd pages */
/* "1"b to allocate pd at disk done time */
/* desperations on pure pages */
```plaintext
/* null addresses reinstated */
/* Pseudo-page faults on volmap_seg */
/* out-of-physical-volume page faults */
/* addresses "corrected" by post-crash pd flush */
/* ptr to damage table of pc_recover_sst */
/* addresses resurrected at RWS time */
/* addresses resurrected by double-writing */
/* page faults in seg moving */
/* Seg moves that failed */
/* Seg faults in seg moves */
/* Seg moves that completed */
/* Times claim_mod_core had to run */
/* total count of activations */
/* count of directory activations */
/* call-in updatevs */
/* call in core flush writes */
/* see evict_page.aim */
/* ptp of page being moved */
/* N/Z if page was mod */

/\* Data for metering force_write facility 08/19/78 */
 2 force_swr1tes fixed bin,
 2 force_pwrv1tes fixed bin,
 2 fw_none fixed bin,
 2 force_updatevs fixed bin,
 2 pf_pd_loop_time fixed bin (71),
 2 pf_unlock_pt1_time fixed bin (71),
 2 pf_unlock_pt1_meterings fixed bin,
 2 makeknown_activations fixed bin (35),
 2 backup_activations fixed bin (35),
 2 metering_flags aligned,
 3 activate Activated bit (1) unal,
 3 pad bit (35) unal,
 2 seg_fault_calls fixed bin (35),
/\* METERS FOR STACK TRUNCATION */
 2 stk_truncate_should_didnt,
    stk_truncate_should_did,
    stk_truncate_shouldnt_didnt,
    stk_truncate Shouldnt_didnt) fixed bin (35),
 2 stk_pages_truncated fixed bin (35),
 2 stk_pages_truncated_in_core fixed bin (35),
 2 padder (8) fixed bin,
/\* the following data is used by page multilevel */
/\* 600 octal */
```
2 pdmap ptr,
2 pdhpt ptr,
2 pd_id fixed bin (8) aligned,
2 pdsize fixed bin,
2 pdmem_no fixed bin,
2 pdusedp bit (18) unaligned,
2 pd_first fixed bin,
2 pd_addr fixed bin,
2 rwcs pdmap fixed bin,
2 pd_free fixed bin,
2 pd_using fixed bin,
2 pd_write fixed bin,
2 pd_rws_current fixed bin,
2 pd_steps fixed bin,
2 pd_skips_inactive fixed bin,
2 pd_skips_rws fixed bin,
2 pd_needed fixed bin,
2 pd_truncate_fixed bin,
2 pd_write_aborts fixed bin,
2 pd_rws_active fixed bin,
2 pd_no_free fixed bin,
2 pd_read_truncate fixed bin,
2 pd_write_truncate fixed bin,
2 pd_trace bit (16),
2 pdmap_aste fixed bin,
2 pd_trace fixed bin,
2 pd_zero_pages fixed bin,
2 trace_sw aligned,
3 pad_trace bit (32) unaligned,
3 pc_trace_pf bit (1) unaligned,
3 tty_trace bit (1) unaligned,
3 sc_trace bit (1) unaligned,
2 new_pages fixed bin,
2 rws_time_temp fixed bin (71),
2 rws_time_start fixed bin (71),
2 rws_time_done fixed bin (71),
2 pd_time_counts (4) fixed bin,
2 pd_time_values (4) fixed bin (71),
2 pd_no_free_gtpd fixed bin,
2 pd_page_fucks fixed bin,
2 pd_no_free_first fixed bin,
2 update_index fixed bin,
2 last_update fixed bin (71),
2 count_pmds fixed bin,
2 bucket_overflow fixed bin,
2 buckets (0:63) fixed bin,
2 sst_track bit (1) aligned,
2 dblk_write_fixed bin,
2 write_limit fixed bin,
2 pad4 (1) fixed bin;

/* pointer to the pd map */
/* pointer to the pd map */
/* pvt index of paging device, 0 if none */
/* the number of words in a paging device map entry */
/* the number of entries in the paging device map */
/* pointer to head of paging device used list */
/* first usable record of paging device */
/* core address of base of paging device map */
/* number of records in pd map */
/* number of free records in the paging device */
/* actual number of pd records being used */
/* number of read/write sequences queued */
/* total number of read/write sequences ever made */
/* number of times too many rws active at once */
/* total steps taken around the pd map */
/* number of entries skipped because page was in core */
/* number of entries skipped because a rws was active */
/* total number of pd records needed */
/* times a page was modified while it was being written */
/* number of pd writes aborted */
/* count of current number of active rws's */
/* number of times couldn't find a free pd record */
/* number of truncated pages during read of rws */
/* number of truncated pages during write of rws */
/* number of entries in pd hash table */
/* mask used in pd hashing algorithm */
/* pointer to temporary segment for pdmap copying */
/* count of pages truncated because all zero */
/* as above except also on paging device */
/* tracing control flags */

/* tracing for page faults, done, etc. */
/* flag used by page control primitives */
/* flag used by segment control primitives */
/* newly created pages */
/* temporary used for rws metering */
/* time spent initiating rws */
/* time spent finishing up rws */
/* time spent finishing up rws */
/* total residency time for the 4 bins */
/* number of hits in the following bins */
/* total residency time for the 4 bins */
/* total page faults from pd */
/* times pages were written to disk because gtpd ON */
/* total page faulty */
/* counter for overflows */
/* buckets for pdmap stats */
/* "l" = keep SST name table */
/* - to flush modified pages in lock$unlock */
/* Max # of outstanding writes by page control */
/* padding to 512 words (1000) */

#include "hard_source"
/* Begin include file sstnt.incl.pl1 */

/* Created 10/03/74 by Bernard Greenberg */
/* modified 08/24/79 by J. A. Bush for easier calculation of size of sstnt */

dcl sst_names $ ext;
  /* Segment containing sst name table */

dcl sstnp ptr;
  /* Pointer to sst name segment */

dcl 1 sstnt based (sstnp) aligned,
    2 valid bit (1) aligned,
    2 multics_or_bos char (4) aligned,
    2 nentries fixed bin,
    2 pad1 (5) fixed bin,

    2 ast_sizes, 
    ast_name_offsets, 
    ast_offets, 
    pad2) (0 : 3) fixed bin,

    2 names (0 : 0 refer (sstnt.nentries)) char (32) varying; /* Names of AST entries */


dcl (sstmx, ptsi_a) fixed bin (17); /* Index into name table */

dcl nm_astep ptr;
  /* astep to be used */

/* End include file sstnt.incl.pl1 */
/* BEGIN INCLUDE FILE ... stack_0_data.incl.pll */

/* Created 790509 by Mike Grady */

dcl stack_0_data$ fixed bin ext;
dcl stack_0_data_init_number_of_stacks fixed bin;
dcl sdtp ptr;

dcl 1 sdt aligned based (sdtp),
    2 lock bit (36),
    2 num_stacks fixed bin,
    2 freep bit (18),
    2 pad fixed bin,
    2 stacks (stack_0_data_init_number_of_stacks
             refer (sdt.num_stacks)) like sdte;

dcl sdtep ptr;

dcl 1 sdte aligned based (sdtep),
    2 nextp bit (18) unal,
    2 astep bit (18) unal,
    2 aptep bit (18) unal,
    2 sdw bit (72);

/* END INCLUDE FILE ... stack_0_data.incl.pll */
/* BEGIN INCLUDE FILE ... stack_frame.incl.pl1 ... */

/* Modified: 16 Dec 1977, D. Levin - to add fio_ps_ptr and pl1_ps_ptr */
/* Modified: 3 Feb 1978, P. Krupp - to add run_unit_manager bit & main_proc bit */
/* Modified: 21 March 1978, D. Levin - change fio_ps_ptr to support_ptr */

dcl sp pointer;
   /* pointer to beginning of stack frame */

stack_frame_min_length fixed bin static init(48);

dcl stack_frame based(sp) aligned,
   2 pointer_registers(0 : 7) ptr,
   2 prev_sp pointer,
   2 next_sp pointer,
   2 return_ptr pointer,
   2 entry_ptr pointer,
   2 operator_and_lp_ptr ptr,
   2 arg_ptr pointer,
   2 support_ptr ptr unaligned,
   2 private_ptr ptr unaligned,
   2 on_unit_relp1 bit(18) unaligned,
   2 on_unit_relp2 bits(18) unaligned,
   2 translator_id bit(18) unaligned,
   2 operator_return_offset bit(18) unaligned,
   2 x(0 : 7) bit(18) unaligned,
   2 a bit(36),
   2 q bit(36),
   2 e bit(36),
   2 timer bit(27) unaligned,
   2 pad bit(6) unaligned,
   2 ring_alarm_reg bit(3) unaligned;

stack_frame_flags based(sp) aligned,
   2 pad(0 : 7) bit(72),
   2 xx0 bit(22) unaligned,
   2 main_proc bit(1) unaligned,
   2 run_unit_manager bit(1) unaligned,
   2 signal bit(1) unaligned,
   2 crawl_out bit(1) unaligned,
   2 signaller bit(1) unaligned,
   2 link_trap bit(1) unaligned,
   2 support bit(1) unaligned,
   2 on_unit_relp1 bit(18) unaligned,
   2 on_unit_relp2 bits(18) unaligned,
   2 translator_id bit(18) unaligned,
   2 operator_return_offset bit(18) unaligned,
2 condition bit(1) unal, /* on if condition established in this frame */
2 xx0a bit(6) unal, /* on if this is a signal caller frame */
2 xx1 fixed bin, /* on if next frame is signalier's */
2 xx3 bit(25) unal,
2 old_crawl_out bit(1) unal,
2 old_signalier bit(1) unal,
2 xx32 bit(9) unaligned,
2 xx4(9) bit(72) aligned, /* When a V2 PL/I program calls an operator the */
2 v2_pl1_op_ret_base ptr, /* operator puts a pointer to the base of */
2 xx5 bit(72) aligned, /* the calling procedure here. (text base ptr) */
2 pl1_ps_ptr ptr; /* ptr to ps for this frame; also used by flo. */

END INCLUDE FILE ... stack_frame.inc.pl1 */
BEGIN INCLUDE FILE ... stack_header.incl.plt 3/72 Bill Silver */
modified 7/76 by M. Weaver for *system links and more system use of areas */
modified 3/77 by M. Weaver to add rnt_ptr */

dcl sb ptr; /* the main pointer to the stack header */
dcl 1 stack_header based (sb) aligned,
          /* (0) also used as arg list by outward_call_handler */
           /* (4) pointer to the lot for current ring (obsolete) */
           /* (6) pointer to area containing separate static */
           /* (8) pointer to area containing linkage sections */
           /* (10) DU number of words allowed in lot */
           /* (10) DL nonzero if main procedure invoked in run unit */
           /* (11) number of words (entries) in lot */
    2 pad1 (4) fixed bin,
    2 old_lot_ptr ptr,
    2 combined_stat_ptr ptr,
          /* (4) pointer to the lot for current ring (obsolete) */
          /* (6) pointer to area containing separate static */
          /* (8) pointer to area containing linkage sections */
          /* (10) DU number of words allowed in lot */
          /* (10) DL nonzero if main procedure invoked in run unit */
          /* (11) number of words (entries) in lot */
    2 max_lot_size fixed bin(17) unal,
    2 main_proc_invoked fixed bin (11) unal,
    2 run_unit_depth fixed bin(5) unal,
    2 cur_lot_size fixed bin(17) unal,
          /* (12) pointer to system storage area */
          /* (14) pointer to user storage area */
    2 null_ptr ptr,
    2 stack_begin_ptr ptr,
    2 stack_end_ptr ptr,
          /* (18) pointer to first stack frame on the stack */
          /* (20) pointer to next useable stack frame */
          /* (22) pointer to the lot for the current ring */
    2 signal_ptr ptr,
    2 bar_mode_sp ptr,
          /* (24) pointer to signal procedure for current ring */
          /* (26) value of sp before entering bar mode */
          /* (28) pointer to pl1_operators$operator_table */
          /* (30) pointer to standard call operator */
    2 call_op_ptr ptr,
    2 push_op_ptr ptr,
    2 return_op_ptr ptr,
          /* (34) pointer to standard return operator */
          /* (36) pointer to standard return / no pop operator */
          /* (38) pointer to standard entry operator */
    2 return_no_pop_op_ptr ptr,
    2 entry_op_ptr ptr,
          /* (40) pointer to translator operator ptrs */
          /* (42) pointer to ISDT */
          /* (44) pointer to System Condition Table */
          /* (46) pointer to unwinder for current ring */
    2 trans_op_tv_ptr ptr,
    2 sys_link_info_ptr ptr,
          /* (48) pointer to *system link name table */
    2 isot_ptr ptr,
    2 rnt_ptr ptr,
          /* (50) pointer to Reference Name Table */
    2 act_ptr ptr,
    2 assign_l!nkage_ptr ptr,
          /* (54) pointer to storage for (obsolete) hcs$assign_linkage */
    2 pad3 (8) bit (36) aligned; /* (56) for future expansion */
The following offset refers to a table within the pH operator table. */

dcl tv_offset fixed bin init(351) internal static; /* (551) octal */

The following constants are offsets within this transfer vector table. */

dcl (call_offset fixed bin init(271),
push_offset fixed bin init(273),
return_offset fixed bin init(273),
return_no_pop_offset fixed bin init(274),
entry_offset fixed bin init(275)) internal static;

The following declaration is an overlay of the whole stack header. Procedures which move the whole stack header should use this overlay. */

dcl stack_header_overlay (size(stack_header)) fixed bin based (sb);

END INCLUDE FILE ... stack_header.incl.pl1 */
/* START OF: stock_seg.incl.plt */

da1 stock_seg  ptr;

da1 record_stockp  ptr;

da1 vtoce_stockp  ptr;

da1 stock_seg  ext;

da1 n_in_record_stock  fixed bin;

da1 n_volmap_pages  fixed bin;

da1 n_in_vtoce_stock  fixed bin;

da1 1 stock_seg  aligned based (stock_seg).

  2 meters  aligned like rsmeters,

    2 record_stock_entries  fixed bin, /* Number of entries in a record stock */
    2 vtoce_stock_entries  fixed bin, /* Number of entries in a VTOCE stock */
    2 record_stock_size  fixed bin, /* Size of a record stock in words */
    2 vtoce_stock_size  fixed bin, /* Size of a VTOCE stock in words */
    2 n_stock_entries  fixed bin, /* Number of stocks of each type */
    2 record_stock_arrayp  ptr;  /* Record stock region */
    2 vtoce_stock_arrayp  ptr;  /* VTOCE stock region */

da1 1 record_stock  aligned based (record_stockp),

    2 pvtep  ptr unal, /* PVTE for this stock */

    2 n_in_stock  fixed bin (18)  uns  unal,/* Max number of addresses in stock */
    2 n_volmap_pages  fixed bin (18)  uns  unal,/* Number of pages in Volume Map */
    2 n_free_in_stock  fixed bin (18)  uns  unal,/* Number addresses currently free */
    2 n_os_in_stock  fixed bin (18)  uns  unal,/* Number addresses currently out-of-service */
    2 low_threshold  fixed bin (18)  uns  unal,/* Low threshold for withdrawing from volmap */
    2 high_threshold  fixed bin (18)  uns  unal,/* High threshold for depositing to volmap */
    2 target  fixed bin (18)  uns  unal,/* Target for stock */
    2 stock_offset  bit (18)  unal, /* Offset of stock in this structure */
    2 n_words_in_stock  fixed bin (18)  uns  unal,/* Number of words = Number of entries / 2 */
    2 search_index  fixed bin (18)  uns  unal,/* Roving pointer */
    2 old_volmap_page  (3)  aligned, /* N_OLD_VOLMAP_PAGES (clf) */
    3 last  fixed bin (18)  uns  unal,/* Roving pointer */
    3 pad  bit (18)  unal,
    2 volmap_page  (n_volmap_pages refer (record_stock.n_volmap_pages))  aligned,
2 withdraw_vstock_fails fixed bin (35); /* Number times withdraw failed */
2 deposit_vtoc_map fixed bin (35); /* Number times vtoc deposited to map */
2 withdraw_check_scav fixed bin (35); /* Number times withdraw checked an address for scavenge */
2 pad (11) fixed bin (35);

dcl N_OLD_VOLMAP_PAGES fixed bin init (3) int static options (constant);
dcl DEFAULT_N_IN_RECORD_STOCK fixed bin init (104) int static options (constant);
dcl DEFAULT_N_IN_VTOCE_STOCK fixed bin init (10) int static options (constant);

/* END OF: stock_seg.incl.pl1 */
/* BEGIN INCLUDE FILE ... str.incl.pl1 ... last modified March 1970 */

dcl str_seg$ ext,
    strp ptr;

dcl 1 str based (strp) aligned;
    /* segment or process trailer declaration */
    (2 fp bit (18),
     2 bp bit (18),
     2 segno bit (18),
     2 dstep bit (18)) unaligned;
    /* segment number */
    /* rel pointer to ring 0 dste */

dcl stra (0:8000) bit (72) based (strp) aligned;
    /* END INCLUDE FILE ... str.incl.pl1 */
<table>
<thead>
<tr>
<th>tc_data.cds</th>
<th>segment in:</th>
<th>contents modified:</th>
<th>system id:</th>
<th>entry modified:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1dd&gt;hard&gt;source</td>
<td>01/26/83 1347.1</td>
<td>37-15</td>
<td>03/27/83 1807.2</td>
</tr>
</tbody>
</table>

```/* ***********************************************************************************/
  * Copyright, (C) Honeywell Information Systems Inc., 1982 *
  ***************************************************************************/
.Throws (C) This is the Traffic Controller Database. */
  */ Last modified (Date and reason): *
  2/6/76 by S. Webber Initial coding
  6/20/79 by Mike Grady to init max_maxe
  3/4/81 by J. Bongiovanni not to set prds_length (it's done from the header
   or the TBLS Config Card)
  3/21/81 by J. Bongiovanni for max_stopped_stack_0, initialization NTO,
   response time metering
  6/27/81 by J. Bongiovanni for tuning parameter changes (-tcpu, +preempt_sample_time,
   gp_at_notify and gp_at_plnotify off by default
  1/82 BIM for stk truncation tuning parms.
  4/27/82 by J. Bongiovanni to change post_purge to OFF
  August 1982, J. Bongiovanni, for realtime_io parameters
  */
  ***********************************************************************************/
  * Copyright (C) 1972 by Massachusetts Institute of
  Technology and Honeywell Information Systems, Inc.
  ***********************************************************************************/

TC_DATA: proc;
  /* This program creates the tc_data base */
  /* Automatic */

dcl 1 cdsa aligned like cds_args;
dcl code fixed bin (35);
dcl big_time fixed bin (71);
  /* Based */

dcl 1 tc_data aligned like tcm based (tcmp);
  /* Static */

dcl exclude_pad (1) char (32) aligned static options (constant) init ("pad");
  /* Builtins */
dcl (addr, bin, null, rel, size, string, unspec) builtin;

/* Entries */

dcl com_err_entry options (variable);
dcl create_data_segment_entry (ptr, fixed bin (35));
dcl get_temp_segment_entry (char (*), ptr, fixed bin (35));
dcl release_temp_segment_entry (char (*), ptr, fixed bin (35));
call get_temp_segment_ ("tc_data", tcp, code);

/\* Check offsets assumed by BOS */
call check_offset_for_bos (addr (tc_data.apt_offset), 171, "apt_offset");
call check_offset_for_bos (addr (tc_data.apt_size), 203, "apt_size");
call check_offset_for_bos (addr (tc_data.apt_entry_size), 215, "apt_entry_size");

tc_data.apt_offset = rel (addr (tc_data.apt));
tc_data.apt_lock = -1;  /* -1 = unlocked */
tc_data.metering_lock = -1;  /* 0 = locked */
tc_data.working_set_factor = 1;
tc_data.ncpu = 0;
tc_data.ltt_size = 155;
tc_data.dtt_size = 155;
tc_data.initializer_id = (36)*1"b;
tc_data.max_eligible = 6*262144;
tc_data.max_max_eligible = 16*262144;
tc_data.max_stopped_stack_0 = 4;
tc_data.apt_entry_size = size (apte);
tc_data.pds_length = 1024;

tc_data.interactive_q.fp = rel (addr (tc_data.interactive_q));
tc_data.interactive_q.bp = rel (addr (tc_data.interactive_q));
tc_data.interactive_q.sentinel = (36)*1"b;

tc_data.max_hproc_segno = 191;  /* largest (default) hardcore segment number */
tc_data.dst_ptr = null;
tc_data.old_user = null;
tc_data.tfirst = 2000000;
tc_data.telast = 2000000;
tc_data.tmax = 8000000;
tc_data.process_initial_quantum = 2000000;
tc_data.gp_at_notify = 0;  /* off by default */
tc_data.gp_at_pfifo_notify = 0;  /* off by default */
tc_data.preempt_sample_time = 40000;  /* 40 milliseconds */
tc_data.max_timer_register = 40000;

tc_data.sort_to_elhead = 1;
tc_data.auto_tune_ws = 1;
tc_data.objects = 01b;
tc_data.stk_truncate = "1"b;
tc_data.stk_truncate_always = "0"b;

/\* See fast_hc_ipc, but the rolling average of steps/block is calculated */
/\* as NEW_AVERAGE = factor*NEW_VALUE + (1-factor)*OLD_AVERAGE */
tc_data.stk_trunc_avg_f1 = 0.9375;  /* 15/16 */
tc_data.stk_trunc_avg_f2 = 1b - tc_data.stk_trunc_avg_f1;
tc_data.lock_error_severity = 3;  /* BEEP BEEP but no crash */
tc_data.realttme_q.fp • reI (addr (tc data.realttme q»:
tc_data.realtlme_q.bp • reI (addr (tc-data.realttme-q»:
tC_data.realttme_q.senttnel - (36)"I"b;
tc_data.eltglble_q_head.fp • reI (addr (tc_data.ellgible_q_tall»:
tc_data.ellglble_q_head.bp • "O"b;
tc_data.ellgtble_q_head.senttnel • (36)"1"b:
tC_data.el Iglble_q_tail.fp • rel (addr (tc data. Idle tatl »):
tc_data.eltgible_q_tai1.bp • reI (addr (tc-data.eltglble q head»:tc_data.eltgtble_q_tatl.senttnel
(36)"I"b:
- g

tC_data.ldle_tatl.fp· "O"b:
tc_data.tdle_tall.bp" reI (addr (tc_data.eltglble_q_tatl»:
tc_data.ldle_tatl.sentlnel • (36)"I Mb:
tC_data.mtn_eltgtble • 2*262144:
tC_data.guaranteed_eltg_tnc • 250000;
tc_data.priorlty_sched_lnc • 80000000:
tC_data.tnt_q_enabled • 1:
tc_data.fnp_buffer_threshold • 30:

1* fnp tries to keep> this many free buff *1

unspec (big t I me) • "00000000000000000000 1 11111 'lit t 1111 t 111111111111 till tit 1111111 t t 11 t 11 tit t "b:
tc_data.end:of_tlme .. btg_time:
tc data.next alarm time • big ttme:
1* gets zeroed by tc_InttSpart_2 *1
tc:data.prtortty_sched_time .-btg_tlme:
tc_data,tty_polllng_ttme • big_time;
tC_data.disk_polltng_ttme • big_ttme:
tc_data.tape~0.1ltng_ttme • big_time;
tc_data.lmp~011tng_tlme • btg_tlme;
tC_data.mos_polltng_ttme • btg_ttme:
tc_data.volmap_polllng_tlme • big_time;
tc data.realtlme to deadltne • 0;
tc:data.realtlme:to:quantum u 5000:
5 milliseconds *1

,*

tC_data.tnlt_watt_ttmeout • 5000000:
1* 5 second NTO during tnltlallzatton
tc data.tnlt ttmeout severtty • 0:
1* beep
tc:data. vcpu:response_bounds_s t ze • VCPU_RESPON!iE_BOUNDS;
tc_data.vcpu_response_bounds (1) .. 500000;
1* 1/2 second
tc data.vcpu response bounds (2) • 1000000:
1* 1 second
tc:data.vcpu:response:bounds (3) .. 10000000:
1* to seconds

1* Now set

tc data.default procs required" (8) "l"b:
u~ call to creite daia base *1
cdsa.sectlons (1).p • addr (tc· data):
cdsa.sectlons (1). len • size (ic data):
cdsa. sect ions (1). struct_name
Ii tC_data " :
c:dsa.seg_name .. "tc_data";
c:dsa . num exc 1ude names
1:
C:dsa.excTude_arriy_ptr • addr (exclude_pad);

includ'

*

hard. source

F80A'.>t to be reproduced

1* all CPUs *1


string (cdsa.switches) = "0"b;
cDSA.switches.have_text  = "1"b;

call create_data_segment_ (addr (cdsa). code);

call release_temp_segment_ ("tc_data", tcmp, code);
check_offset_for_bos:
    proc (item_ptr, bos_offset, item_name);
        dcl item_ptr ptr; /* pointer to item in tc_data */
        dcl bos_offset fixed bin (18); /* location assumed by BOS */
        dcl item_name char (*); /* name of item in structure */
        if bin (rel (item_ptr)) ^= bos_offset
            then call com_err_ (0, "tc_data", "^a not at BOS-assumed offset (^d="oo)",
            item_name, bos_offset, bos_offset);
    end check_offset_for_bos;

include *., hard.source  tc_data.cds

FBC not to be reproduced
% include cds_args;
% include apte;

#include "hard.source"

tc_data.cds

F80A not to be reproduced
% include tcm;
    end tc_data;
/* BEGIN INCLUDE FILE ... tcm.incl.pll ... used to generate tc_data cds */
/* NOTE -- This include file has TWO counterparts in ALM: tc_metrics.incl alm and */
/* wcce.incl alm. They cannot be produced with cif, and must be kept up to date manually. */

dcl tcmp ptr;
dcl 1 tcm aligned based (tcmp),
  2 pad_base (3) fixed bin (18),
  2 cid2 fixed bin (18),
  2 cid3 fixed bin (18),
  2 cid4 fixed bin (18),
  2 depth_count fixed bin (18),
  2 loadings fixed bin (18),
  2 blocks fixed bin (18),
  2 wakeups fixed bin (18),
  2 waits fixed bin (18),
  2 notifies fixed bin (18),
  2 schedulings fixed bin (18),
  2 interactions fixed bin (18),
  2 avequeue fixed bin (35, 18),
  2 te_wait fixed bin (18),
  2 te_block fixed bin (18),
  2 te_i_stop fixed bin (18),
  2 te_pre_empt fixed bin (18),
  2 p_interactions fixed bin,
  2 idle fixed bin (71),
  2 mp_idle fixed bin (71),
  2 nmp_idle fixed bin (71),
  2 zero_idle fixed bin (71),
  2 last_time fixed bin (71),
  2 loop_locks fixed bin (18),
  2 loop_lock_time fixed bin (18),
  2 ave_eligible fixed bin (35, 18),
  2 sort_to_elhead fixed bin (18),
  2 processor_time fixed bin (71),
  2 response_time fixed bin (71),
  2 eligible_time fixed bin (71),
  2 response_count fixed bin,
  2 eligible_count fixed bin,
  2 quit_counts (0:5) fixed bin,
  2 loading_idle fixed bin (71),
  2 delta_vcpu fixed bin (71),
  2 post_purge_switch fixed bin,
  2 time_out_severity fixed bin,
  2 te_walt fixed bin (18),
  2 te_block fixed bin (18),
  2 te_i_stop fixed bin (18),
  2 te_pre_empt fixed bin (18),
  2 p_interactions fixed bin,
  2 idle fixed bin (71),
  2 mp_idle fixed bin (71),
  2 nmp_idle fixed bin (71),
  2 zero_idle fixed bin (71),
  2 last_time fixed bin (71),
  2 loop_locks fixed bin (18),
  2 loop_lock_time fixed bin (18),
  2 ave_eligible fixed bin (35, 18),
  2 sort_to_elhead fixed bin (18),
  2 processor_time fixed bin (71),
  2 response_time fixed bin (71),
  2 eligible_time fixed bin (71),
  2 response_count fixed bin,
  2 eligible_count fixed bin,
2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
2 lobe.polling_time fixed bin (71),
2 end_of_time fixed bin (71),
2 gp_at_notify fixed bin (18),
2 gp_at_ptlnotify fixed bin (18),
2 int_q_enabled fixed bin (18),
2 fnp_buffer_threshold fixed bin (18),

2 notify_check fixed bin,
2 quit_priority fixed bin,
3 apt_offset bit (18),
3 pad bit (18),
2 getwork_time fixed bin (71),
2 getwork_count fixed bin (18),
2 short_of_count fixed bin (18),
2 interrupt time fixed bin (71),
2 interrupt_count fixed bin (71),
2 old_user_ptr ptr,
2 newt fixed bin (18),
2 telast fixed bin (30),

/* total time spent in getwork */
/* total times through getwork */
/* number of short page faults */
/* total time spent in interrupt */
/* total number of metered interrupts */
/* fraction of core for Int're users */
/* controls whether preemption at done time */
/* total number of memory usage fixed bin */
/* value of processor_time when WC's last defined */
/* number of times priority process given high priority */
/* number of times priority process lost eligibility */
/* sum of all simalaged clock delays */
/* number of times alarm clock interrupt was simulated */
/* largest simulated alarm clock delay */

2 pdscpoyl fixed bin (18),
2 max_hproc_segment fixed bin,
2 prds_length fixed bin (18),
2 pd3_length fixed bin (18),
2 lock fixed bin (18),
2 id fixed bin (18),
2 system_shutdown fixed bin (18),
2 working_set_factor fixed bin (35, 18),
2 ncpu fixed bin (18),
2 last_eligible bit (18),
2 apt_lock fixed bin (35),
2 apt_size fixed bin (18),
2 realtime_q aligned like based_sentinel,
2 aht_size fixed bin (18),
2 itl_size fixed bin (18),
2 dst_size fixed bin (18),
2 itt_free_list bit (18),
2 used_itt fixed bin (18),
2 initializer_id bit (36) aligned,
2 n_eligible fixed bin (18),
2 max_eligible fixed bin (30),
2 wait_enable fixed bin (18),
2 apt_entry_size fixed bin (18),
2 interactive_q aligned like based_sentinel,
2 dst_ptr ptr,
2 old_user_ptr,
2 initialize_time fixed bin (71),
2 init_event fixed bin (18),
2 oldt_fixed bin (18),
2 newt fixed bin (18),
2 tefirst fixed bin (30),
2 telast fixed bin (30),

/* amount of pd3 to copy for new process */
/* largest allowed hardcore segment number */
/* length of PRDS */
/* length of PDS */
/* process id generator lock */
/* next processid to be given out */
/* working set factor */
/* number of processors currently being used */
/* last process to gain eligibility */
/* + write: 0 hidden: -1 unlocked: -(N+1) Nreaders */
/* number of APT entries */
/* processes with realtme deadlines */
/* APT hash table size */
/* number of ITT entries */
/* number of allowed DST entries */
/* pointer to ITT free list */
/* number of used ITT entries */
/* process id of initializer */
/* number of processes eligible */
/* maximum allowed number of eligible processes */
/* turned on when waiting mechanism works */
/* size of an APT entry */
/* head of interactive queue */
/* dst_ptr ptr */
/* last process to run (apt ptr ) */
/* time of initialization */
/* wait event during initialization */
/* timer reading from previous process */
/* timer setting for new process */
/* first eligible time */
/* last eligible time */
/* time in queue for lowest level */
/* thread of empty APT entries */
/* additive working set parameter */
/* head of empty list */
/* for added segdef */
/* default_procs_required */
/* non-idle virtual time */
/* credits not yet passed out */
/* offset of initializer work class table entry */
/* offset of highest wcte currently defined */
/* temp used by pxss.computer_virtual_clocks */
/* idle time_used.clock */
/* total number of credits awarded at once */
/* temp for pxss.find_next_eligible */
/* clock time when workclasses last dagined */
/* 0>= ti sorts, else deadline sorts */
/* Maximum of make */
/* Maximum stack 0's suspended by stopped procs */
/* Number stack 0's suspended by stopped procs */
/* for heals */
/* for heals */
/* used by wait/notify during initialization */
/* notify-timeout interval during initialization */
/* notify-timeout severity during initialization */
/* count of NTO recursion during initialization */
/* max cpu burst */
/* tuning parameter - max time between samples */
/* used for limiting eligibility on governed work classes*/
/* eligibility quantum first eligibility */
/* default mask of CPUs required */
/* Tuning Parameters for Stack Truncation */

2 stk_truncate bit (1) aligned,
2 stk_truncate_always bit (1) aligned,
2 stk_trunc_avg_f1 fixed bin (35, 18),
2 stk_trunc_avg_f2 fixed bin (35, 18),
2 lock_error_severity fixed bin,
2 gv_integration fixed bin (35),
2 gv_integration_set bit (1) aligned,
2 pad8 fixed bin (38),
2 volmap_polling_time fixed bin (71),
2 pad9 fixed bin (71),
2 realtime_io_priority_switch fixed bin,
2 realtime_io_deadline fixed bin (35),
2 realtime_io_quantum fixed bin (35),
2 realtime_priorities fixed bin (35),
2 relinquishes fixed bin (35),
2 abort_ips_mask bit (36) aligned.

/* 500 octal */
2 pad5 (192) fixed bin (35),
/* 1000 octal */
2 pad7 (64) fixed bin (35),
/* 1100 octal */
2 pad6 (8) fixed bin (35),
2 work_class_table aligned.
   3 wcte (0:16) aligned like wct_entry.
/* 3000 octal */
2 apt fixed bin;

dcl wctep ptr;

dcl 1 wct_entry aligned based (wctep),
2 thread unaligned,
3 fp bit (18),
3 bp bit (18),
2 flags unaligned,
3 mnbz bit (1),
3 defined bit (1),
3 io_priority bit (1),
3 governed bit (1),
3 interactive_q bit (1),
3 pad bit (31),
2 credits fixed bin (35),
2 minf fixed bin (35),
2 credits fixed bin (35),
/* idle time due to work class restrictions */

2 work_class_idle fixed bin (71),
/* idle time due to work class restrictions */

2 indx fixed bin (35),
/* syserr severity */
2 gv_inclination fixed bin (35),
/* Integration interval for governing */
2 gv_inclination_set bit (1) aligned,
2 pad8 fixed bin (38),
2 volmap_polling_time fixed bin (71),
2 pad9 fixed bin (71),
/* O -> give I/O interrupt wakeups realtime priority */
2 realtime_io_priority fixed bin (35),
/* Delta to clock for I/O realtime deadline */
2 realtime_io_deadline fixed bin (35),
/* Quantum for I/O realtime burst */
2 realtime_io_quantum fixed bin (35),
/* Count for metering */
2 realtime_priorities fixed bin (35),
/* Calls to relinquish_priority */
2 relinquishes fixed bin (35),
/* IPS mask for tc_util$check_abort */
2 abort_ips_mask bit (36) aligned.

/* 500 octal */
2 pad5 (192) fixed bin (35),
/* room for expansion compatibly */
/* 1000 octal */
2 pad7 (64) fixed bin (35),
/* 1100 octal */
2 pad6 (8) fixed bin (35),
/* array of per workclass information */
2 work_class_table aligned.
   3 wcte (0:16) aligned like wct_entry.
/* 3000 octal */
2 apt fixed bin;

dcl wctep ptr;

dcl 1 wct_entry aligned based (wctep),
2 thread unaligned,
3 fp bit (18),
3 bp bit (18),
2 flags unaligned,
3 mnbz bit (1),
3 defined bit (1),
3 io_priority bit (1),
3 governed bit (1),
3 interactive_q bit (1),
3 pad bit (31),
2 credits fixed bin (35),
2 minf fixed bin (35),
/* Work class entry */
/* Ready list */
/* Head of ready list */
/* tail of ready list */
/* Sentinel bit must not be zero. */
/* Current worthiness of group */
/* min fraction of cpu */
2 pin_weight fixed bin (35),  /* number of cycles to pin pages */
2 eligibilities fixed bin (35),  /* Count of eligibilities awarded */
2 cpu_sum fixed bin (71),  /* CPU used by members */
2 resp1 fixed bin (71),
2 resp2 fixed bin (71),
2 quantum1 fixed bin (35),
2 quantum2 fixed bin (35),
2 rcnt1 fixed bin (35),
2 rcnt2 fixed bin (35),
2 realtime fixed bin (35),
2 purging fixed bin (35),
2 maxel fixed bin (35),
2 nel fixed bin (35),
2 number_thinks fixed bin (35),  /* number times process entered "think" state */
2 number_queues fixed bin (35),  /* number times process entered "queued" state */
2 total_think_time fixed bin (71),
2 total_queue_time fixed bin (71),
2 number_processing (VCPU_RESPONSE_BOUNDS+1) fixed bin (35),  /* number times entered "processing" state */
2 total_processing_time (VCPU_RESPONSE_BOUNDS+1) fixed bin (71),
2 total_vcpu_time (VCPU_RESPONSE_BOUNDS+1) fixed bin (71),
2 maxf fixed bin (35),  /* maximum fraction of cpu time */
2 governingcredits fixed bin (35),  /* for limiting cpu resources */
2 padt (4) fixed bin (35);

dcl 1 based_sentinel aligned based,  /* format of pxss-style sentinel */
2 fp bit (18) unal,
2 bp bit (18) unal,
2 sentinel bit (36) aligned;

dcl VCPU_RESPONSE_BOUNDS fixed bin init (3) int static options (constant);

/* END INCLUDE FILE tcm.incl.pl1 */
dcl vol_mapp ptr;
dcl 1 vol_map based (vol_mapp) aligned,
  2 n_rec fixed bin(17),
  2 base_add fixed bin(17),
  2 n_free_rec fixed bin(17),
  2 bit_map_n_words fixed bin(17),
  2 pad (60) bit(36),
  2 bit_map (3*1024 - 64) bit(36);

/* number of records represented in the map */
/* record number for first bit in bit map */
/* number of free records */
/* number of words of the bit map */
/* pad to 64 words */
/* bit map - the entire vol map occupies 3 records */
vtoc_buffer.incl.pl1

/* START OF: vtoc_buffer.incl.pl1 November, 1982 */

dcl vtoc_buffer_seg$ ext:

dcl vtoc_buffer_seg$ ptr;

dcl vtoc_buf_desc$ ptr;

dcl vtoc_buf$ ptr;

dcl vtoc_buf_desc$ array$ ptr;

dcl vtoc_buf$ array$ ptr;

dcl vtoc_n_buffers fixed bin;

dcl vtoc_n_buckets fixed bin;

1 vtoc_buffer aligned based (vtoc_buffer_seg$),

2 lock,
3 processid bit (36) aligned, /* Global lock for VT0C buffers */
3 wait_event bit (36) aligned, /* Owner */
3 notify_sw bit (1) aligned, /* For lock */
3 notify_sw bit (1) aligned, /* ON => notify on unlock */
2 n_buffers fixed bin, /* Number of full VT0C buffers */
2 n_hash_buckets fixed bin, /* Number of hash table buckets */
2 hash_mask bit (36) aligned, /* Mask for hash algorithm */
2 abs_addr fixed bin (24), /* Absolute address of vtoc_buffer_seg */
2 wait_event_constant fixed bin (36) uns aligned, /* Constant to add to part index to form wait event */
2 buf_desc_offset bit (18), /* Offset of buf_desc */
2 buf_offset bit (18), /* Offset of buf */
2 hash_table_offset bit (18), /* Offset of hash_table */
2 search_index fixed bin, /* Roving pointer for buffer selection */
2 unsafe_pvtx fixed bin, /* Pseudo-Clock for scavenger-free-other-allocate race */
2 scavenger_free_p_clock fixed bin (35), /* Pseudo-Clock for scavenger-free-other-allocate race */

2 meters,
3 call_get fixed bin (36), /* Calls to get_vtoce */
3 call_put fixed bin (36), /* Calls to put_vtoce */
3 call_free fixed bin (36), /* Calls to free_vtoce */
3 call_alloc fixed bin (36), /* Calls to alloc_and_put_vtoce */
3 call_wait fixed bin (35), /* Calls to await_vtoce */
3 steps fixed bin (35), /* Steps through buffer allocation */
3 skip_os fixed bin (35), /* Skipped because out-of-service */
3 skip_hot fixed bin (35), /* Skipped because buffer hot */
3 skip_wait fixed bin (35), /* Skipped because notify_sw set */
3 disk_reads fixed bin (35), /* Number of same */
3 disk_writes fixed bin (35), /* Number of same */
3 get_buffer_calls fixed bin (35), /* Number of calls to GET_BUFFER */
3 get_buffer_hits fixed bin (35), /* Number times VT0CE in Buffer */
3 wait_calls fixed bin (35), /* Number of calls to WAIT */
3 wait_os fixed bin (35), /* Number of times had to wait */
3 scavenger_free_checks
fixed bin (35).  /* Number of times had to check pseudo-clock */
3 scavenger_free_losses
fixed bin (35),  /* Number of times race lost between scavenger freeing and other allocat
3 pad (15)  fixed bin (35),
2 hash_table  (vtoc_buf_n_buckets refer (vtoc_buffer.n_hash_buckets)) bit (18) aligned,
2 buf_desc  (vtoc_buf_n_buffers refer (vtoc_buffer.n_bufs)) aligned like vtocbuf_desc,
2 buffer  (vtoc_buf_n_buffers refer (vtoc_buffer.n_bufs)) aligned like vtocbuf;
dcl 1 vtocbuf_desc_array  (vtoc_buffer.n_bufs) aligned based (vtocbuf_desc_arrayp) like vtocbuf_desc;
dcl 1 vtocbuf_desc  aligned based (vtocbuf_descp),
   2 pvtx  fixed bin (17) unal,  /* PVTE index */
   2 vtocx  fixed bin (17) unal,  /* VTOCE Index */
   2 parts_used  bit (3) unal,  /* Mask of parts used or os */
   2 err  bit (1) unal,  /* ON => I/O error on buffer */
   2 notify_sw  bit (1) unal,  /* ON => notify required on I/O completion */
   2 write_sw  bit (1) unal,  /* ON => write I/O */
   2 os  bit (1) unal,  /* ON => I/O in progress */
   2 ioq  bit (1) unal,  /* ON => I/O has been requested */
   2 used  bit (1) unal,  /* ON => this descriptor is in use */
   2 pad  bit (5) unal,
   2 wait_index  fixed bin (17) unal,  /* Buffer index for forming wait event */
   2 ht_thread  bit (18) unal,  /* Offset of next entry in hash table */
   2 buf_rel  bit (18) unal;  /* Offset of buffer in segment */
dcl 1 vtocebuf_array  (vtoc_buffer.n_bufs) aligned based (vtocbuf_arrayp) like vtocebuf;
dcl 1 vtocebuf  aligned based (vtocbufp),
   3 parts  (3) aligned,
   3 words  (64) bit (36) aligned;
dcl N_PARTS_PER_VTOCE  fixed bin int static options (constant) init (3);
dcl VTOCE_PART_SIZE  fixed bin int static options (constant) init (64);
dcl VTOCE_BUFFER_SIZE  fixed bin int static options (constant) init (3 * 64);
dcl N_VTOCE_PER_RECORD  fixed bin int static options (constant) init (5);
dcl N_SECTOR_PER_VTOCE  fixed bin int static options (constant) init (3);
/* END OF:  vtocbuf.incl.pl1 */
/* BEGIN INCLUDE FILE ... vtoc_header.incl.pll */

dcl vtoc_headerp ptr;

dcl 1 vtoc_header based (vtoc_headerp) aligned.

2 version fixed bin (17),
2 n_vtoc fixed bin (17),
2 vtoc_last_reco fixed bin (17),
2 n_free_vtoc fixed bin (17),
2 first_free_vtoc fixed bin (17),
2 pad (3) bit (36),
2 dmpr_bit_map (2048 - 8) bit (36);

/* version number. The current version number is 1. */
/* number of vtoc entries */
/* record number of the last record of the vtoc */
/* number of free vtoc entries */
/* index of the first vtoc in the free list */
/* space for dmpr bit map */

/* END INCLUDE ... vtoc_header */
vtoc_map.incl.pl1

/* START OF: vtoc_map.incl.pl1 ... March 1982 ... */
dcl vtoc_map
    ptr;

dcl bit_map_wordp
    ptr;

dcl 1 vtoc_map
    aligned based (vtoc_map).
    2 n_vtoc
        fixed bin, /* Number of VTOCES on the device */
    2 n_free_vtoc
        fixed bin, /* Number of free VTOCES */
    2 bit_map_n_words
        fixed bin, /* Number of words in the bit map below */
    2 vtoc_last_recno
        fixed bin, /* Last record number in VTOC */
    2 pad
        (4) fixed bin,
    2 bit_map
        (0:1024 - 9) bit (36); /* This structure consumes exactly 1 page */

dcl 1 bit_map_word
    aligned based (bit_map_wordp).
    2 pad1
        bit (1) unal,
    2 bits
        bit (32) unal, /* 32 VTOCES ON => free */
    2 pad2
        bit (3) unal;

/* END OF: vtoc_map.incl.pl1 */
BEGIN INCLUDE FILE ...vtoce.incl.pl11 ... last modified September 1982 */

/* Template for a VTOC entry. Length = 192 words. (3 * 64). */
dcl vtocep ptr;

dcl 1 vtoce based (vtocep) aligned.

(2 pad_free_vtoce_chain bit (36), /* Used to be pointer to next free VTOCE */
  2 uid bit (36), /* segment’s uid - zero if vtoce is free */
  2 msl bit (9), /* maximum segment length in 1024 word units */
  2 csl bit (9), /* current segment length - in 1024 word units */
  2 records bit (9), /* number of records used by the seg in second storage */
  2 pad2 bit (9),
  2 dtu bit (36), /* date and time segment was last used */
  2 dtm bit (36), /* date and time segment was last modified */
  2 ngsw bit (1), /* no quota switch - no checking for pages of this seg */
  2 deciduous bit (1), /* true if hc_sdw */
  2 nid bit (1), /* no incremental dump switch */
  2 dnzp bit (1), /* dont null zero pages */
  2 gtpd bit (1), /* Global transparent paging device */
  2 per_process bit (1), /* Per process segment (deleted every bootload) */
  2 damaged bit (1), /* TRUE if contents damaged */
  2 fm_damaged bit (1), /* TRUE if filemap checksum bad */
  2 fm_checksum_valid bit (1), /* TRUE if the checksum has been computed */
  2 synchronized bit (1), /* TRUE if this is a data management synchronized segment */
  2 pad3 bit (8), /* directory switch */
  2 dirsw bit (1), /* master directory - a root for the logical volume */
  2 master_dir bit (1), /* not used */
  2 pad4 bit (16)) unaligned,

2 fm_checksum bit (36) aligned, /* Checksum of used portion of file map */

(2 quota (0:1) fixed bin (18) unsigned, /* sec storage quota - (0) for non dir pages */
  2 used (0:1) fixed bin (18) unsigned, /* sec storage used - (0) for non dir pages */
  2 received (0:1) fixed bin (18) unsigned, /* total amount of storage this dir has received */
  2 trp (0:1) fixed bin (71), /* time record product - (0) for non dir pages */
  2 trp_time (0:1) bit (36), /* time time_record_product was last calculated */
2 fm (0:255) bit (18),
2 pad6 (10) bit (36),
2 ncd bit (1),
2 pad7 bit (17),
2 pad8 bit (18),
2 dtd bit (36),
2 valid (3) bit (36),
2 master_dir_uid bit (36),
2 uid_path (0:15) bit (36),
2 primary_name char (32),
2 time_created bit (36),
2 par_pvid bit (36),
2 par_vtocx fixed bin (17),
2 branch_rp bit (18) unaligned,
2 cn_salv_time bit (36),
2 access_class bit (72),
2 perm_flags aligned,
3 per_bootload bit (1) unal,
3 pad9 bit (35) unal,
2 owner bit (36);
dcl vtoce_parts (3) bit (36 + 64) aligned based (vtocep);
dcl 1 seg_vtoce based (vtocep) aligned,
2 pad1 bit (7+36),
2 usage fixed bin (35),
2 pad2 bit (18+36);

/* file map - 256 entries - 18 bits per entry */
/* not used */
/* no complete dump switch */
/* date-time-dumped */
/* volume ids of last incremental, consolidated, and complete dumps */
/* superior master directory uid */
/* uid pathname of all parents starting after the root */
/* primary name of the segment */
/* time the segment was created */
/* physical volume id of the parent */
/* vtoc entry index of the parent */
/* rel pointer of the branch of this segment */
/* time branch - vtoce connection checked */
/* access class in branch */
/* ON => deleted each bootload */
/* pvid of this volume */

/* Overlay for vtoce of segments, which don't have quota */
/* page fault count: overlays quota */
APPENDIX B

MULTICS TECHNICAL PAPERS

Multics - The First Seven Years ........................................ B-1
The Multics Virtual Memory: Concepts and Design .............. B-1
A Simple Linear Model of Demand Paging Performance ........ B-1
A Hardware Architecture for Implementing Protection
  Rings ............................................................................. B-1
The Multics PL/I Compiler .................................................. B-1
Virtual Memory, Processes, and Sharing in Multics ............. B-1
Introduction and Overview of the Multics System ............... B-1
System Design of a Computer for Time Sharing
  Applications ..................................................................... B-1
Structure of the Multics Supervisor ................................... B-1
A General-Purpose File System for Secondary Storage ....... B-1
Communications and I/O Switching in a Multiplexed
  Computing System ......................................................... B-1
Some thoughts about the Social Implications of
  Accessible Computing .................................................... B-1
This page has intentionally been left blank.
<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0556.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0601.00</td>
<td>QS</td>
</tr>
<tr>
<td>0606.00</td>
<td>Q</td>
</tr>
<tr>
<td>0611.00</td>
<td>QS</td>
</tr>
<tr>
<td>0616.00</td>
<td>Q</td>
</tr>
<tr>
<td>0621.00</td>
<td>QS</td>
</tr>
<tr>
<td>0626.00</td>
<td>Q</td>
</tr>
<tr>
<td>0631.00</td>
<td>S</td>
</tr>
<tr>
<td>0636.00</td>
<td>Q</td>
</tr>
<tr>
<td>0641.00</td>
<td>S</td>
</tr>
<tr>
<td>0646.00</td>
<td>Q</td>
</tr>
<tr>
<td>0651.00</td>
<td>S</td>
</tr>
<tr>
<td>0656.00</td>
<td>Q</td>
</tr>
<tr>
<td>0701.00</td>
<td>Q</td>
</tr>
<tr>
<td>0706.00</td>
<td>S</td>
</tr>
<tr>
<td>0711.00</td>
<td>Q</td>
</tr>
<tr>
<td>0716.00</td>
<td>Q</td>
</tr>
<tr>
<td>0721.00</td>
<td>Q</td>
</tr>
<tr>
<td>0726.00</td>
<td>Q</td>
</tr>
<tr>
<td>0731.00</td>
<td>S</td>
</tr>
<tr>
<td>0736.00</td>
<td>Q</td>
</tr>
<tr>
<td>0746.00</td>
<td>Q</td>
</tr>
<tr>
<td>0751.00</td>
<td>Q</td>
</tr>
<tr>
<td>0756.00</td>
<td>Q</td>
</tr>
<tr>
<td>0806.01</td>
<td>Q</td>
</tr>
<tr>
<td>0816.01</td>
<td>Q</td>
</tr>
<tr>
<td>0826.01</td>
<td>Q</td>
</tr>
<tr>
<td>0856.13</td>
<td>Q</td>
</tr>
<tr>
<td>0916.00</td>
<td>Q</td>
</tr>
<tr>
<td>0926.01</td>
<td>Q</td>
</tr>
<tr>
<td>0936.01</td>
<td>Q</td>
</tr>
<tr>
<td>0946.12</td>
<td>Q</td>
</tr>
<tr>
<td>0956.01</td>
<td>Q</td>
</tr>
</tbody>
</table>
Multics MR10.2, load 45.0/60.0: 47 users, 24 interactive, 22 daemons.
Absentee users 1/4
avg = 11, elapsed time = 930 sec, 30 active last 15 sec.
Virtual CPU Time 29.02 41.46
Zero Idle 30.00
NMP Idle 30.00

<table>
<thead>
<tr>
<th>Page Faults</th>
<th>11.74</th>
<th>16.78</th>
<th>2717.319</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocations</td>
<td>33127</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>allocations</td>
<td>68230</td>
<td>9515</td>
<td>13.95</td>
</tr>
</tbody>
</table>

Seg Faults 5.11 7.30 18332.578
Lap Time(sec) 191.9 706.6 1973.2 84.0
Other Fault 10.29 14.71
Getwork 4.53 6.48 498.232
MP Idle 5.92 3.46
clock 4 ast
root dska 1 dskb 2
schd 400000 10 20 60 2 10
tbls sstn 32. str 64.
tcd 75. 299.
parm ttyb 14080. ttyg 1536.
part bos dska 1
part dump dska 1
part log dskb 2
mpc mspa 451. a 20 4 a 24 3
mpc mspb 451. a 30 4 a 34 2
mpc mtpa 601. a 16 1 a 17 1
mpc urpa 600. a 10 4
fnp a a 14
fnp b a 15
cpu a 7 on dps8 70. 8.
cpu b 6 on dps8 70. 8.
chnl dska a 34 2
chnl dskb a 30 4
sst 600. 400. 300. 750.
salv dcf
prph dska a 20 4 451. 20
prph dskb a 24 4 451. 20
prph tapa a 16 2 500. 10
prph prta a 10 1600. 600. 136.
prph prtb a 11 1600. 600. 136.
prph rdr a 12 301.
prph opc a 37 6601. 80. on
prph puna a 13 300.
intk warm 4
iom a 0 nsa on
mem d 512. on
mem c 512. on
mem b 512. on
mem a 2048. on
fnp c a 36
<table>
<thead>
<tr>
<th>Drive</th>
<th>Records</th>
<th>Left</th>
<th>%</th>
<th>VTOCEs Left</th>
<th>%</th>
<th>Avg PV</th>
<th>PB/PD</th>
<th>LV Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dska_05</td>
<td>36584</td>
<td>1354</td>
<td>4</td>
<td>7625</td>
<td>60</td>
<td>11 cis2 pb</td>
<td>cis</td>
<td></td>
</tr>
<tr>
<td>dska_11</td>
<td>36584</td>
<td>2455</td>
<td>7</td>
<td>7625</td>
<td>63</td>
<td>12 cis1 pb</td>
<td>cis</td>
<td></td>
</tr>
<tr>
<td>dska_12</td>
<td>36584</td>
<td>6226</td>
<td>17</td>
<td>7625</td>
<td>54</td>
<td>8 ldd1 pb</td>
<td>ldd</td>
<td></td>
</tr>
<tr>
<td>dskb_09</td>
<td>36720</td>
<td>27412</td>
<td>75</td>
<td>7650</td>
<td>99</td>
<td>160 progra pb</td>
<td>progra</td>
<td></td>
</tr>
<tr>
<td>dskb_05</td>
<td>36720</td>
<td>2521</td>
<td>7</td>
<td>7650</td>
<td>1986</td>
<td>26 public01 pb</td>
<td>pd public</td>
<td></td>
</tr>
<tr>
<td>dskb_06</td>
<td>36720</td>
<td>1954</td>
<td>5</td>
<td>7650</td>
<td>1717</td>
<td>22 public03 pb</td>
<td>pd public</td>
<td></td>
</tr>
<tr>
<td>dskb_07</td>
<td>36720</td>
<td>2373</td>
<td>6</td>
<td>7650</td>
<td>1791</td>
<td>23 public02 pb</td>
<td>pd public</td>
<td></td>
</tr>
<tr>
<td>dska_01</td>
<td>29811</td>
<td>1121</td>
<td>4</td>
<td>17400</td>
<td>12774</td>
<td>73 rpv pb</td>
<td>root</td>
<td></td>
</tr>
<tr>
<td>dska_02</td>
<td>35837</td>
<td>1968</td>
<td>5</td>
<td>7575</td>
<td>2971</td>
<td>39 root2 pb</td>
<td>root</td>
<td></td>
</tr>
<tr>
<td>dska_15</td>
<td>36584</td>
<td>3604</td>
<td>10</td>
<td>7625</td>
<td>5209</td>
<td>68 vadis_misc1 pb</td>
<td>pd vadis_misc</td>
<td></td>
</tr>
<tr>
<td>dska_15</td>
<td>36584</td>
<td>5178</td>
<td>14</td>
<td>7625</td>
<td>5478</td>
<td>72 vadis_misc2 pb</td>
<td>pd vadis_misc</td>
<td></td>
</tr>
<tr>
<td>dska_04</td>
<td>37967</td>
<td>182</td>
<td>0</td>
<td>710</td>
<td>562</td>
<td>79 vadis_pf7 pb</td>
<td>vadis_pf</td>
<td></td>
</tr>
<tr>
<td>dska_06</td>
<td>37967</td>
<td>168</td>
<td>0</td>
<td>710</td>
<td>560</td>
<td>79 vadis_pf5 pb</td>
<td>vadis_pf</td>
<td></td>
</tr>
<tr>
<td>dska_10</td>
<td>37967</td>
<td>433</td>
<td>1</td>
<td>710</td>
<td>563</td>
<td>79 vadis_pf3 pb</td>
<td>vadis_pf</td>
<td></td>
</tr>
<tr>
<td>dska_13</td>
<td>37967</td>
<td>398</td>
<td>1</td>
<td>710</td>
<td>561</td>
<td>79 vadis_pf2 pb</td>
<td>vadis_pf</td>
<td></td>
</tr>
<tr>
<td>dskb_08</td>
<td>38094</td>
<td>299</td>
<td>1</td>
<td>710</td>
<td>561</td>
<td>79 vadis_pf1 pb</td>
<td>vadis_pf</td>
<td></td>
</tr>
<tr>
<td>dskb_14</td>
<td>37967</td>
<td>60</td>
<td>0</td>
<td>710</td>
<td>561</td>
<td>79 vadis_pf6 pb</td>
<td>vadis_pf</td>
<td></td>
</tr>
<tr>
<td>dskb_16</td>
<td>37967</td>
<td>310</td>
<td>1</td>
<td>710</td>
<td>561</td>
<td>79 vadis_pf4 pb</td>
<td>vadis_pf</td>
<td></td>
</tr>
</tbody>
</table>
Current system tuning parameters:

tefirst: 1. seconds
 bậc: 2. seconds
timax: 6. seconds
priority_sched_inc: 80. seconds
min_eligible: 2.
max_eligible: 8.
max_batch_elig: 0
working_set_factor: 0.5
working_set_addend: 0
deadline_mode: off
int_q_enabled: on
post_purge: off
preempt_sample_time: 0.04 seconds
gp_at_notify: off
gp_at_ptlnotify: off
process_initial_quantum: 2. seconds
quit_priority: 0.
notify_timeout_interval: 30. seconds
notify_timeout_severity: 3
write_limit: 100
gv_integration: 8. seconds
realtime_io_priority: off
realtime_io_deadline: 0. seconds
realtime_io_quantum: 0.005 seconds
CTP should change it in conf. file. Put it in P.C.

! answer yes or no. CTP TEL: last 2 -silent

- suppresses messages on other console for init.

notify timeout severity,

proc_id:

[Diagram]

Event that did not occur \( \rightarrow \) hardcore_events, etc. info

If low overhead then adding CPU helps

5%

look at avg CPU per interaction + Buckets in meters

- Salvagers \( \rightarrow \) checks damage to stripe + runs or damage switch.

Directory Salvaging \( \rightarrow \) online salvager maybe do one later

Quota \( \rightarrow \) can be done right away.

Scavenger \( \rightarrow \) if recent inconsistency switch is on - online scavenger.

Can be delayed. Does 1 volume at a time.

15 min per volume.

Physical Volume Salvager - offline salvaging - boot RLVS or PLVS

Sweep PV - connection failures \( \rightarrow \) reverse connection failures.

runs once a month.
DBR info comes from APT
when process is eligible Page of Dseg is rem wired.
DSEG is made active AT Proc Creation semi hold switch set on for Process life
SDW allocated making Segment Known
SDW filled in AT Segment time.
ASTW for Seg is filled in from VTCW AT Segment activation.
PTW gets filled in AT page fault time.
Seg Active

ASIE Entries filled in by VTDC

ASIE's chained together.
As you come in, you are at end of list if low priority.

If we scan and find ready process, we use idle
process which are always at end of eligible queue.

Notify us a process just makes it ready, doesn't make it run
immediately.

2MW Memory

2MW.

170 => 200 10

2000 - 150 = 8 or 10

F-6 Proc per MW

Norm

8 MW X 5 = ME of 40

When a process is eligible
it looks for first 2 pages of DSOG, if they are in
memory they are marked wired.

If they are not, then a
Disk read is done. When
they are in memory they marked wired.

MAX-E
Balance of
CPV usual vs
Available Memory

MAX-MAXE
Rings to
Stacks to
Create, CTP

Can't let you
change MAXE
This solution

(MAX) MAX Eligible Parameter related to
for Ring O Stacks.

The working set of System can process
Can't exceed Main Memory, it will cause
Page Thrashing.
losing eligibility.

runs Post_Purge which uses PDS to find # page faults
To find the working set i.e. # of pages touched during eligibility.
This breaks down because of page eviction done while
waiting while you go to ineligible.

By default Post_Purge is off.

useless params

Min_eligible
Post_purge
MinE ?? used to be = # CPUs
WSE = GWS + (WS * WSP + TWS)
WS = last end

MaxIdle - Idle time from idle process
At MaxE it had eligible processes
Which means low MaxE.

QUES

First time in you go through IQ (interactive Q)
(interactive + SYSTEM)
Tmax normally = 8 sec
used up all of these Ti.

The more times you cycle in a work class, the lower you sink
abs Tmax normally = 16 lets interactive favorable over Absforce.
Do not put Absforces in separate work classes by themselves.
The will all accumulate and get more time. It is better
abs to change Tmax.

Real Time: mainly used for Prior Denous

Percentages of work classes

Real Time. doesn't use percentages, it is higher
Priority.

Process follows IQ
deadlines = clock(time1) + IRQ. to once deadline occurs,
also they start at top of queue instead of bottom.
Proc A deactivating Seg Y, known to Proc B & C.
Abs Seg. Supervisor Seg. used to plug in whatever SOW we need. Short address from trailer is plugged in Abs.Seg. Which uses offset to point to O500.

Deciduous – Not by Seg. Supervisor Segments dir entry is fabricated even though it is not in hierarchy. Fabrication is in /dev/config_deck back to avoid gaps.

Other examples

RW 0,0
50 configuration
$500

Things in p0 created by init are reverse deciduous. Because deciduous are created as supervisor made to look like hierarchy. Reverse are hierarchy made to look like supervisor

Small MAKE \rightarrow hi Pi
Mine small or large \rightarrow no effect
Figure 6-2. Page Control Data Bases
Page in main memory, not on paging device

1:1 relation between CME's and Page Frames
Work Classes

Max 90 per WClass. There is a bit so that they do not go through interactive QAMV.

Load control groups -> MGT when system decides who can login at MAXV.

Project get assigned to LCG’s which get assigned WClass.

PTC - Global Page Table Lock

When unlocked, only 1 process is notified instead of all processes waiting.
Figure 1-1. Page Control Data Bases
Page not in main memory or on paging device
1. Activate seg, UID 1166
2. Read Page 0 of 1011
3. Evict Page 2 of 1771
4. Read Page 0 of 1166
5. Activate 3456
6. Evict Page 1 of 7022
7. Read Page 5 (!) of 5677
8. Activate 0016
9. Read Page 1 of 0016
10. Activate 1621
11. Read Page 0 of 3456
12. Evict Page 0 of 1166
13. Activate 7654

\*should\* This be 1 for 2nd Page 0 being first.

\*Fault* goes to 14K pool.

Also gets rethreaded at 30 if 15 after pointer so it is next one seen.
COMMON MULTICS PERFORMANCE PROBLEMS

System Tasks Taking Too Much Resources

MTAR
Too much backup during prime time
Directory salvaging after ESDless crash
System personnel favored by workclass parameters

Parameters set wrong

Work Class Parameters
Absentees in separate workclass lasts of time

Traffic Control Parameters
telstrap too high
maxe wrong, usually too low

AST Size
Segment thrashing due to small pool size(s)

Hardware Configuration
Not enough memory
Not enough disk channels
Not enough logical channels
Not enough disk arms
Insufficient CPU power

Disk I/O
Unbalanced disk I/O

Communications
HASP lines
X.25 lines cause interrupts, not much you can do about it.
breakall on dialout lines

Misc
Bootload console looping
Initializer time at maxu users
Application causing segment thrashing
Application using too much CPU time
Bad write/notify cables makes CPU clear cache memory
check CPU will check this