Sun386i man Pages Supplement

December 1988
Credits and Copyright

PostScript is a trademark of Adobe Systems, Inc.
UNIX is a registered trademark and UNIX System V is a trademark of AT&T Bell Laboratories.
Compaq is a registered trademark and COMPAQ DESKPRO 386 is a trademark of COMPAQ Computer Corporation.
DEC, PDP-11, VAX, and VT are registered trademarks of Digital Equipment Corporation.
Intel is a registered trademark of Intel Corporation.
IBM is a registered trademark of International Business Machines.
Ada is a trademark of the Joint Program Office, U.S. Department of Defense.
Ethernet is a registered trademark of Xerox Corporation.
Frame Maker is a trademark of Frame Technology Corporation.
Interleaf is a trademark of Interleaf, Inc.
Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.
Motorola is a registered trademark of Motorola, Incorporated.
NeWS, SunCore, Sun Microsystems, and the Sun logo are registered trademarks and SunOS, Sun-CDI, SunIPC, SunLink, SunView, Sun-2, Sun-3, Sun-4, and Sun386i are trademarks of Sun Microsystems, Inc.
SunINGRES is a trademark of Sun Microsystems, Inc., and is derived from INGRES, a product marketed by Relational Technology, Inc.
UNIFY is a registered trademark of Unify Corporation.
VMEbus is a trademark of Motorola, Incorporated.

Copyright © 1983 – 1988 Sun Microsystems, Inc. All Rights Reserved.

No part of this work covered by copyright hereon may be reproduced in any form or by any means – graphic, electronic, or mechanical – including photocopying, recording, taping, or information storage and retrieval systems, without the prior permission of the copyright owner.

Restricted rights legend: Use, duplication, or disclosure by U.S. government is subject to restrictions as set forth in subparagraph c.1.ii of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and in similar clauses in the FAR and NASA FAR Supplement.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California. We acknowledge the following individuals and institutions for their role in its development: The Regents of the University of California and the Electrical Engineering and Computer Sciences Department at the Berkeley Campus of the University of California and Other Contributors.
# Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>December 1988</td>
<td>First release of this manual.</td>
</tr>
</tbody>
</table>

A December 1988 First release of this manual.
Sun386i man Pages Supplement

The manual pages in this document augment and in some cases replace the manual pages in the SunOS Reference Manual (800-1751-10). The new and altered manual pages document enhancements and functionality in the Sun386i SunOS 4.0.1 release.

You can replace existing man pages and insert new pages in SunOS Reference Manual with the pages from this supplement. Replacement pages include the original page number, and new pages have no page number. Insert the new pages after these page numbers in SunOS Reference Manual:

- fdformat(1) after page 181
- fontflip(8) after page 1626
- help_open(1) after page 224
- keytables(5) after page 1402
- loadkeys(1) after page 268
- orgrc(5) after page 1410
- start_applic(8) after page 1767
- ypsync(8) – after page 1800

The man pages in this supplement are listed below according to section.

Section 1

bar(1) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
cc(1v) – updated for Sun386i SunOS release 4.0.1.
dos(1) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
fdformat(1) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
fontedit(1) – updated for Sun386i SunOS release 4.0.1.
help_open(1) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
help_viewer(1) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
input_from_defaults(1) – updated for Sun386i SunOS release 4.0.1.
ld(1) – updated for Sun386i SunOS release 4.0.1.
load(1), loadc(1) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
loadkeys(1), dumpkeys(1) – Sun386i systems only, new for Sun386i SunOS release 4.0.1
organizer(1) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
strip(1) – updated for Sun386i SunOS release 4.0.1.
textedit(1) – updated for Sun386i SunOS release 4.0.1.
uucp(1c) – updated for Sun386i SunOS release 4.0.1.

Section 3
getmntent(3) – updated for Sun386i SunOS release 4.0.1.

Section 4
kb(4m) – updated for Sun386i SunOS release 4.0.1.

Section 5
bar(5) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
help(5) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
help_viewer(5) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
keytables(5) – new for Sun386i SunOS release 4.0.1.
orgrc(5) – Sun386i systems only, new for Sun386i SunOS release 4.0.1.
vfont(5) – updated for Sun386i SunOS release 4.0.1.

Section 8
fontflip_to_68k(8), fontflip_to_i386(8) – Sun386i systems only, new for Sun386i SunOS release 4.0.1.
ipallocd(8C) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
kadb(8s) – updated for Sun386i SunOS release 4.0.1.
modload(8) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
modstat(8) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1
mount(8) – updated for Sun386i SunOS release 4.0.1.
rarpd(8c) – updated for Sun386i SunOS release 4.0.1.
rwhod(8) – updated for Sun386i SunOS release 4.0.1.
start_applic(8) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1.
unconfigure(8) – Sun386i systems only, updated for Sun386i SunOS release 4.0.1.
ypsync(8) – Sun386i systems only, new for Sun386i SunOS release 4.0.1.
NAME
bar – create tape archives, and add or extract files

SYNOPSIS
bar [-] crxtu [014578feovwbXIFmhpBisHSUGZRTINLODPVd] [barfile] [blocksize]
[exclude-file] [Volume Header ID] [from-directory to-directory] [user_id] [group_id]
[include-file] [date (ymmdhhmm)] [prompt] [volume_number] [output_filename]
filename ... [-C dir filename ...] ...

AVAILABILITY
Sun386i systems only.

DESCRIPTION
bar archives and extracts multiple files onto a single bar, file archive, called a barfile. It is quite similar to tar(1), but it has additional function modifiers, can read and write multiple volumes of tapes or diskettes, and writes and reads a format that is incompatible with tar (see bar(5)). A barfile is usually a magnetic tape, but it can be any file. bar’s actions are controlled by the first argument, the key, a string of characters containing exactly one function letter from the set rxtuc, and one or more of the optional function modifiers listed below. Other arguments to bar are file or directory names that specify which files to archive or extract. In all cases, the appearance of a directory name refers recursively to the files and subdirectories of that directory.

FUNCTION LETTERS

- c Create a new barfile and write the named files onto it.
- r Write the named files on the end of the barfile. Note: this option does not work with quarter-inch archive tapes.
- x Extract the named files from the barfile. If a named file matches a directory with contents written onto the tape, this directory is (recursively) extracted. The owner, modification time, and mode are restored (if possible). If no filename arguments are given, all files in the archive are extracted. Note: if multiple entries specifying the same file are on the tape, the last one overwrites all earlier versions.
- t List the table of contents of the barfile.
- u Add the named files to the barfile if they are not there or if they have been modified since they were last archived. Note: this option does not work with quarter-inch archive tapes.

FUNCTION MODIFIERS

014578
Select an alternate drive on which the tape is mounted. The numbers 2, 3, 6, and 9 do not specify valid drives. The default is /dev/rmt8.

f Use the next argument as the name of the barfile. If f is omitted, use the device indicated by the TAPE environment variable, if set. Otherwise, use /dev/rmt8 by default. If barfile is given as ‘-‘, bar writes to the standard output or reads from the standard input, whichever is appropriate. Thus, bar can be used as the head or tail of a filter chain. bar can also be used to copy hierarchies with the command:

example% cd fromdir; bar cf - . | (cd todir; bar xfbp -)

o Suppress information specifying owner and modes of directories which bar normally places in the archive. Such information makes former versions of bar generate an error message like:

<filename>: cannot create
when they encounter it.

v Normally bar does its work silently; the v (verbose) option displays the name of each file bar treats, preceded by the function letter. When used with the t function, v displays the barfile entries in a form similar to ‘ls –l’ . Each entry displayed is followed by the date the bar archive was created and the volume number on which the entry can be found.
w Wait for user confirmation before taking the specified action. If you use w, bar displays the action to be taken followed by the file name, and then waits for a y response to proceed. No action is taken on the named file if you type anything other than a line beginning with y.

b Use the next argument as the blocking factor for tape records. The default blocking factor is 20 blocks. The block size is determined automatically when reading tapes (key letters x and t). This determination of the blocking factor may be fooled when reading from a pipe or a socket (see the B key letter below). The maximum blocking factor is determined only by the amount of memory available to bar when it is run. Larger blocking factors result in better throughput, longer blocks on nine-track tapes, and better media utilization. Note: the blocking factor on tapes is forced to 126 and the blocking factor on diskettes is forced to 18. These are the optimal blocking factors for these devices and are necessary in reading and writing multi-volume archives.

X Use the next argument as a file containing a list of named files (or directories) to be excluded from the barfile when using the key letters 'c', 'x', or 't'. Multiple X arguments may be used, with one exclude file per argument.

l Display error messages if all links to archived files cannot be resolved. If l is not used, no error messages are printed.

F With one F argument specified, exclude all directories named SCCS from barfile. With two arguments FF, exclude all directories named SCCS, all files with .o as as their suffix, and all files named errs, core, and a.out.

m Do not extract modification times of extracted files. The modification time will be the time of extraction.

h Follow symbolic links as if they were normal files or directories. Normally, bar does not follow symbolic links. Note: symbolic links followed in this way are not archived as symbolic links; they are archived as directories or files. When these directories and files are restored, they are not restore as symbolic links, but as directories and files.

L Follow directory symbolic links as if they were normal directories. Note: these directories are archived and restored as symbolic links.

p Restore the named files to their original modes, ignoring the present umask(2). Setuid and sticky information are also extracted if you are the super-user. This option is only useful with the x key letter.

B Force bar to perform multiple reads (if necessary) so as to read exactly enough bytes to fill a block. This option exists so that bar can work across the Ethernet, since pipes and sockets return partial blocks even when more data is coming.

i Ignore directory checksum errors.

s Force the ownership of extracted files to match the user’s effective user ID and group ID.

H The string of up to 128 characters is to be used as a volume header ID. A null volume header ID is written in each volume header of the archive when this function modifier is not specified. See bar(5) for the volume header’s format. This option is only useful with the c key letter.

S Followed by two arguments: the 'from' directory and the 'to' directory. If the pathname of any extracted file begins with 'from' directory, then bar replaces 'from' directory with 'to' directory. This function is only useful with the x function letter and is useful in restoring files and directories to a different location.

U Use the next argument as the user ID in the volume header.

G Use the next argument as the group ID in the volume header.

Z Specify compression. bar will compress files when used with the c function letter and will uncompress files when used with the x function letter. bar will neither compress a compressed file, nor uncompress an uncompressed file. Uses compress(1).
When extracting files with the x function letter, issue an error message if the user ID in the volume header of the bar archive does not match that of the user extracting the files.

Read the volume header of the bar archive and print the information to stdout.

D Use the next argument (in the form 'yyymmdhhmm', where 'yy' is a year, 'mm' is a month from 01-12, 'dd' is a day from 01-31, 'hh' is an hour from 01-24, and 'mm' is a minute from 00-59) as the date in the volume header, instead of the current date. This function modifier is only useful with the c function letter.

Use the next argument as the starting volume number in the prompt for media changes. This function modifier is useful in situations where some volumes in a sequence are not written in bar format.

Use the next argument as the prompt for media change conditions. If this argument, which is a string, contains a printf(3S) conversion specification in the form of '%d', then that conversion specification will be replaced with the current volume number.

Do not overwrite bar archives with the c function letter if the user ID in the volume header of the archive does not match that of the user creating the new archive.

When using the x or t function letters, terminate the search of the media after all the files specified are extracted (for x) or listed (for t).

Use the next argument as a file containing a list of named files, one per line, to be included in the bar archive. The include file expects filenames to be followed by a semicolon and newline character.

In the case where excluded files (see X flag) also exist, excluded files take precedence over all included files. So, if a file is specified in both the include and exclude files (or on the command line), it will be excluded.

Use the next argument, which is a filename, as a second output for the bar archive.

Options

-C dir filename

In a c (create) or r (replace) operation, bar performs a chdir (see csh(1)) to that directory before interpreting filename. This allows multiple directories not related by a close common parent to be archived using short relative path names. For example, to archive files from /usr/include and from /etc, one might use:

eexample% bar c -C /usr/include -C /etc .

If you get a table of contents from the resulting barfile, you will see something like:

include/
includexa.out.h
and all the other files in /usr/include ...
and all the other files in /etc

Note: the -C option only applies to one following directory name and one following file name.

Examples

Here is a simple example using bar to create an archive of your home directory on a tape mounted on drive /dev/rmt0:

eexample% cd
example% bar cvf /dev/rmt0 .

messages

The c option means create the archive; the v option makes bar tell you what it's doing as it works; the f option means that you are specifically naming the file onto which the archive should be placed (/dev/rmt0 in this example).
BAR(l) USER COMMANDS

BAR (1)

Here is another example: /dev/rmt0:

```
ex ample% cd

example% bar cvfH /dev/rmt0 "THIS IS MY HEADER" .
```

As in the first example, the c option means create the archive; the v option makes bar tell you what it's doing as it works; the f option means that you are specifically naming the file onto which the archive should be placed (/dev/rmt0 in this example). The H option says to use the string "THIS IS MY HEADER" as the ID field in the volume header.

Now you can read the table of contents from the archive like this:

```
ex ample% bar tvf /dev/rmt0

(read user-id/group-id size mod. date filename)

rw-r--r-- 1677/40 2123 Nov 7 18:15:1985 /archive/test.c

... example%
```

You can extract files from the archive like this:

```
ex ample% bar xvf /dev/rmt0

messages

mod. date Nov 7 18:15:1985
filename )/archive/test.c
```

If there are multiple archive files on a tape, each is separated from the following one by an EOF marker. bar does not read the EOF mark on the tape after it finishes reading an archive file because bar looks for a special header to decide when it has reached the end of the archive. Now if you try to use bar to read the next archive file from the tape, bar does not know enough to skip over the EOF mark and tries to read the EOF mark as an archive instead. The result of this is an error message from bar to the effect:

```
bar: blocksize=0
```

This means that to read another archive from the tape, you must skip over the EOF marker before sbarting another bar command. You can accomplish this using the mt command, as shown in the example below. Assume that you are reading from /dev/nrmt0.

```
ex ample% bar xvf /dev/nrmt0

read first archive from tape

example% mt fsf 1

skip over the end-of-file marker

example% bar xvf /dev/nrmt0

read second archive from tape
```

Finally, here is an example using bar to transfer files across the Ethernet. First, here is how to archive files from the local machine (example) to a tape on a remote system (host):

```
ex ample% bar cvfb - 20 filenames rsh@hostdd

messages

example%
```

In the example above, we are creating a barfile with the c key letter, asking for verbose output from bar with the v option, specifying the name of the output barfile using the f option (the standard output is where the barfile appears, as indicated by the – sign), and specifying the blocksize (20) with the b option. If you want to change the blocksize, you must change the blocksize arguments both on the bar command and on the dd command.

Now, here is how to use bar to get files from a tape on the remote system back to the local system:

```
ex ample% rsh -n host dd if=/dev/rmt0 bs=20b | bar xvfBfb - 20 filenames

messages

example%
```
In the example above, we are extracting from the barfile with the x key letter, asking for verbose output from bar with the v option, telling bar it is reading from a pipe with the B option, specifying the name of the input barfile using the f option (the standard input is where the barfile appears, as indicated by the ‘-’ sign), and specifying the blocksize (20) with the b option.

FILES
/dev/rmt?
/dev/rfd0?
/dev/rar?
/dev/rst?
/tmp/bar*

ENVIRONMENT
TAPE If specified, in the environment, the value of TAPE indicates the default tape device.

NOTES
bar will handle multiple volumes gracefully. If a tape error is encountered, bar issues a message on the standard error requesting a new volume. The presence of a new volume is confirmed when bar reads a line beginning with Y or y on the standard input; a line beginning with N or n aborts the archive; with any other character bar reissues the prompt.

SEE ALSO
cpio(1), umask(2), bar(5), tar(5), dump(8), restore(8)

BUGS
Neither the r option nor the u option can be used with quarter-inch archive tapes, since these tape drives cannot backspace.

There is no way to ask for the nth occurrence of a file.

The u option can be slow.

There is no way selectively to follow symbolic links.

When extracting tapes created with the r or u options, directory modification times may not be set correctly.

Filename substitution wildcards do not work for extracting files from the archive. To get around this, use a command of the form:

```
bar xvf... /dev/rst0 'bar tf... /dev/rst0 | grep 'pattern'*
```

If you specify ‘-’ as the target file and the archive spans volumes, the request for a new volume may get lost.

Beta versions of bar archives cannot be read by later versions (4.0 and 4.0.1) of bar unless the H modifier is specified when the Beta version is created. Under Beta the H modifier causes the bar volume header to be written. The volume header is always written by post-Beta versions of bar, whether or not the H modifier is specified.
NAME
cc - C compiler

SYNOPSIS
cc [ -a ] [ -align _block ] [ -B binding ] [ -c ] [ -C ] [ -dryrun ] [ -Dname [=def] ] [ -E ]
[ float_option ] [ -fsingle ] [ -g ] [ -go ] [ -help ] [ -J ] [ -library ]
[ -Ldirectory ] [ -M ] [ -misalign ] [ -o outfile ] [ -O[level] ] [ -p ] [ -P ] [ -pg ] [ -pic ]
[ -PIC ] [ -pipe ] [ -Qoption prog opt ] [ -Qpath pathname ] [ -Qproduce sourcetype ] [ -R ]
[ -S ] [ target_arch ] [ -temp=directory ] [ -time ] [ -Uname ] [ -v ] [ -w ] sourcefile ...

SYSTEM V SYNOPSIS
/usr/bin/cc arguments

Note: arguments to /usr/bin/cc are identical to those listed above.

DESCRIPTION
cc is the C compiler. It translates programs written in the C programming language into executable
load modules, or into relocatable binary programs for subsequent loading with the ld(1) link editor.

In addition to the many options, cc accepts several types of filename arguments. For instance, files with
names ending in .c are taken to be C source programs. They are compiled, and each resulting object
program is placed in the current directory. The object file is named after its source file — the suffix .o
replacing .c in the name of the object. In the same way, files whose names end with .s are taken to be
assembly source programs. They are assembled, and produce .o files. Filenames ending in .ii are taken
to be inline expansion code template files; these are used to expand calls to selected routines in-line
when code optimization is enabled. See FILES, below for a complete list of compiler-related filename
suffixes.

Other arguments refer to assembler or loader options, object programs, or object libraries. Unless -c,
-S, -E -P or -Qproduce is specified, these programs and libraries, together with the results of any
specified compilations or assemblies, are loaded (in the order given) to produce an output file named
a.out. You can specify a name for the executable by using the -o option.

If a single C program is compiled and loaded all at once, the intermediate file is deleted.

OPTIONS
When debugging or profiling objects are compiled using the -g or -pg options, respectively, the ld
command for linking them should also contain the appropriate option.

See ld(1) for link-time options.

-a  (Available on Sun-2, Sun-3, and Sun-4 systems.) Insert code to count how many times
each basic block is executed. Invokes a run-time recording mechanism that creates a
.d file for every .c file (at normal termination). The .d file accumulates execution data
for the corresponding source file. The tcov(1) utility can then be run on the source
file to generate statistics about the program. Since this option entails some optimiza-
tion, it is incompatible with -g.

-align _block  Force the global uninitialized data symbol block to be page-aligned by increasing its
size to a whole number of pages, and placing its first byte at the beginning of a page.

-B binding  Specify whether bindings of libraries for linking are static or dynamic, indicating
whether libraries are non-shared or shared, respectively.

-c  Suppress linking with ld(1) and produce a .o file for each source file. A single object
file can be named explicitly using the -o option.

-C  Prevent the C preprocessor, cpp(1), from removing comments.

-dryrun  Show but do not execute the commands constructed by the compilation driver.

-Dname[=def]  Define a symbol name to the C preprocessor (cpp(1)). Equivalent to a #define
directive in the source. If no def is given, name is defined as '1'.

-E
Run the source file through cpp(1), the C preprocessor, only. Sends the output to the standard output, or to a file named with the -o option. Includes the cpp line numbering information. (See also, the -P option.)

float_option
Floating-point code generation option. Can be one of:

-f68881 Generate in-line code for Motorola MC68881 floating-point processor (supported only on Sun-3 systems).
-ffp Generate in-line code for Sun Floating Point Accelerator (supported only on Sun-3 systems).
-fsky Generate in-line code for Sky floating-point processor (supported only on Sun-2 systems).
-fsoft Generate software floating-point calls. Supported only on Sun-2 and Sun-3 systems, for which it is the default.
-fswitch Run-time-switched floating-point calls. The compiled object code is linked at runtime to routines that support one of the above types of floating point code. This was the default in previous releases. Only for use with programs that are floating-point intensive, and must be portable to machines with various floating-point hardware options (supported only on Sun-2 and Sun-3 systems).

(Sun-2, Sun-3 and Sun-4 systems)

Use single-precision arithmetic in computations involving only float expressions. Do not convert everything to double, which is the default. Note: floating-point parameters are still converted to double precision, and functions returning values still return double-precision values.

Although not standard C, certain programs run much faster using this option. Be aware that some significance can be lost due to lower-precision intermediate values.

-g
Produce additional symbol table information for dbx(1) and dbxtool(1) and pass the -lg flag to ld(1). When this option is given, the -O and -R options are suppressed.

-go
Produce additional symbol table information for adb(1). When this option is given, the -O and -R options are suppressed.

-help
Display helpful information about cc.

-Ipathname
Add pathname to the list of directories in which to search for #include files with relative filenames (not beginning with slash /). The preprocessor first searches for #include files in the directory containing sourcefile, then in directories named with -I options (if any), and finally, in /usr/include.

-J
Generate 32-bit offsets in switch statement labels (supported only on Sun-2 and Sun-3 systems).

-llibrary
Link with object library library (for ld(1)).

-Ldirectory
Add directory to the list of directories containing object-library routines (for linking using ld(1)).

-M
Run only the macro preprocessor on the named C programs, requesting that it generate makefile dependencies and send the result to the standard output (see make(1) for details about makefiles and dependencies).

-misalign
Generate code to allow loading and storage of misaligned data (Sun-4 systems only).

-o outputfile
Name the output file outputfile. outputfile must have the appropriate suffix for the type of file to be produced by the compilation (see FILES, below). outputfile cannot be
the same as sourcefile (the compiler will not overwrite the source file).

-0[level]  Optimize the object code. Ignored when either -g, -go, or -a is used. On Sun-2 and Sun-3 systems, -0 with the level omitted is equivalent to -O1; on Sun-4 systems, it is equivalent to -O2. On Sun386i systems, all levels are the same as 1. level is one of:

1  Do postpass assembly-level optimization only.
2  Do global optimization prior to code generation, including loop optimizations, common subexpression elimination, copy propagation, and automatic register allocation. -O2 does not optimize references to or definitions of external or indirect variables.
3  Same as -O2, but optimize uses and definitions of external variables. -O3 does not trace the effects of pointer assignments. Neither -O3 nor -O4 should be used when compiling either device drivers, or programs that modify external variables from within signal handlers.
4  Same as -O3, but trace the effects of pointer assignments.

-p  Prepare the object code to collect data for profiling with prof(1). Invokes a run-time recording mechanism that produces a mon.out file (at normal termination).

-P  Run the source file through cpp(1), the C preprocessor, only. Puts the output in a file with a .i suffix. Does not include cpp-type line number information in the output.

-pg  Prepare the object code to collect data for profiling with gprof(1). Invokes a run-time recording mechanism that produces a gmon.out file (at normal termination).

-pic  Produce position-independent code. Each reference to a global datum is generated as a dereference of a pointer in the global offset table. Each function call is generated in pc-relative addressing mode through a procedure linkage table. The size of the global offset table is limited to 64K on MC68000-family processors, or to 8K on SPARC processors.

-PIC  Like -pic, but allows the global offset table to span the range of 32-bit addresses in those rare cases where there are too many global data objects for -pic.

-pipe  Use pipes, rather than intermediate files, between compilation stages. (Very cpu-intensive.)

-Qoption prog opt  Pass the option opt to the program prog. The option must be appropriate to that program and may begin with a minus sign. prog can be one of: as, cpp, inline, or ld.

-Qpath pathname  Insert directory pathname into the compilation search path (to use alternate versions of programs invoked during compilation). This path will also be searched first for certain relocatable object files that are implicitly referenced by the compiler driver (such files as *crt*.o and bb_link.o ).

-Qproduce sourcetype  Produce source code of the type sourcetype. sourcetype can be one of:

   .c  C source (from bb_count).
   .i  Preprocessed C source from cpp(1).
   .o  Object file from as(1).
   .s  Assembler source (from ccom, inline(1) or c2).

-R  Merge data segment with text segment for as(1). Data initialized in the object file produced by this compilation is read-only, and (unless linked with ld -N) is shared between processes. Ignored when either -g or -go is used.

-S  Do not assemble the program but produce an assembly source file.

SunOS Release 4.0.1  Last change: 8 November 1988  54
**target_arch**

Compile object files for the specified processor architecture. Unless used in conjunction with one of the Sun Cross-Compilers, correct programs can be generated only for the architecture of the host on which the compilation is performed. **target_arch** can be one of:

- `sun2` Produce object files for a Sun-2 system.
- `sun3` Produce object files for a Sun-3 system.
- `sun4` Produce object files for a Sun-4 system.

**-temp=directory**

Set directory for temporary files to be `directory`.

**-time**

Report execution times for the various compilation passes.

**-Uname**

Remove any initial definition of the `cpp(1)` symbol `name`. (Inverse of the `-D` option.)

**-v**

Verbose. Print the version number of the compiler and the name of each program it executes.

**-w**

Do not print warnings.

**ENVIRONMENT**

**FLOAT_OPTION**

(Sun-2, Sun-3, Sun-4 systems only.) When no floating-point option is specified, the compiler uses the value of this environment variable (if set). Recognized values are: `f68881`, `ffpa`, `fsky`, `fswitch` and `fsoft`.

**FILES**

- `a.out` executable output file
- `file.a` library of object files
- `file.c` C source file
- `file.d` `tcov(1)` test coverage input file (Sun-2, Sun-3, Sun-4 systems only)
- `file.i` C source file after preprocessing with `cpp(1)`
- `file.il` `inline` expansion file
- `file.o` object file
- `file.s` assembler source file
- `file.S` assembler source for `cpp(1)`
- `file.tcov` output from `tcov(1)` (Sun-2, Sun-3, Sun-4 systems only)
- `/usr/lib/c2` object code optimizer
- `/usr/lib/ccom` compiler
- `/usr/lib/compile` compiler command-line processing driver
- `/usr/lib/cpp` macro preprocessor
- `/usr/lib/crt0.o` runtime startup code
- `/usr/lib/Fcrtl.o` startup code for `-fsoft` option (Sun-2, Sun-3, Sun-4 systems only)
- `/usr/lib/gcrt0.o` startup for profiling with `gprof(1)`
- `/usr/lib/libc.a` standard library, see `intro(3)`
- `/usr/lib/mcrt0.o` startup for profiling with `prof(1)` `intro(3)`
- `/usr/lib/Mcrt1.o` startup code for `-f68881` option (for Sun-3 systems)
- `/usr/lib/Scr1.o` startup code for `-fsky` option (for Sun-2 systems)
- `/usr/lib/Wcr1t0.o` startup code for `-fppa` option (for Sun-3 systems)
- `/usr/include` standard directory for `#include` files
- `/usr/lib/bb_link.o` basic block counting routine
- `/usr/lib/cg` code generator used with `/usr/lib/iropt`
- `/usr/lib/libc_p.a` profiling library, see `gprof(1)` or `prof(1)`
- `/usr/lib/inline` `inline` expander of library calls
- `/usr/lib/iropt` intermediate representation optimizer
- `/usr/lib/libm.a` math library
- `/usr/lib/libc.a` System V standard compatibility library, see `intro(3V)`
- `/usr/lib/libc_p.a` System V profiling library, see `gprof(1)` or `prof(1)`

SunOS Release 4.0.1 Last change: 8 November 1988
/tmp/*
mon.out
gmon.out

compiler temporary files
file produced for analysis by prof(1)
file produced for analysis by gprof(1)

SEE ALSO
adb(1), ar(1V), as(1), cpp(1), dbx(1), dbxtool(1), gprof(1), inline(1), ld(1), lint(1V), make(1), prof(1),
tcov(1), intro(3), intro(3V), monitor(3)

Floating Point Programmers Guide
SunOS Programming Utilities and Libraries

DIAGNOSTICS
The diagnostics produced by the C compiler are intended to be self-explanatory. Occasional obscure
messages may be produced by the preprocessor, assembler, or loader.

BUGS
The program context given in syntax error messages is taken from the input text after the C preproces-
sor has performed substitutions. Therefore, error messages involving syntax errors in or near macro
references or manifest constants may be misleading.

Compiling with optimization level 2 or greater may produce incorrect object code if tail-recursion elim-
nation is applied to functions called with fewer actual parameters (arguments) than the number of for-
mal parameters in the function’s definition. Such parameter-count mismatches can be detected using
lint(1V).
NAME
dos – SunView window for IBM PC/AT applications

SYNOPSIS
dos [ -b ] [ -s ] [ -p config ] [ -q ] [ -w ] [ -c command ]

AVAILABILITY
Sun386i systems only.

DESCRIPTION
A window created by dos looks and acts like the screen of an IBM PC/AT or compatible computer running MS-DOS 3.3, except that it has expanded features. It allows sharing of files with SunOS, copying and pasting data between windows, and piping and redirection. You may run any reasonable number of DOS windows simultaneously.

Shrinking or expanding the window will not change the contents to accommodate the new size.

USAGE
Menu
The menu available in the window by pressing the right mouse button allows various controls over the work in the window. Edit allows you to copy and paste between windows. The Show Screen menu item selects the type of screen display—either Hercules, CGA, or Monochrome (use the DOS MODE command to set the corresponding DOS display mode; see the Sun386i User’s Guide or on-line help for more information). The Mouse menu item allows you to control whether the mouse operates like a Microsoft or compatible mouse or in normal SunView fashion (see Sun386i Advanced Skills for instructions on enabling Microsoft mouse driver software). The Send to printer menu item allows you to send queued jobs to the print spooler. Sound controls the volume of sounds from the DOS window. Device allows you to select which disks and other devices will be used and which are to be considered read only. The Reboot DOS Window item is equivalent to restarting the window. This can also be accomplished by pressing the CONTROL, ALT, and DELETE keys simultaneously.

Printer Assignments
DOS uses three printer designations: LPT1, LPT2, and LPT3. The default settings are: files sent to LPT1 go to the default system printer. Files sent to LPT2 are appended to the file lpt-2 in your home directory. Epson-compatible print jobs can be sent to LPT3 to yield Epson FX-80 quality output on your default printer, as long as it is Postscript-compatible.

Drives
Drive A  The Sun386i 3-1/2" diskette drive, used for reading PC format diskettes onto the hard disk and writing data to be stored on floppy. Drive A is not accessible across a network.
Drive B  An optional 5-1/2" diskette drive. Same restrictions as Drive A.
Drive C  A virtual disk stored in the ~/pc/C: file. Files written to drive C cannot be accessed from SunOS. Drive C is generally intended for storage of applications and copy protected software but not data. To DOS, drive C is a 20-megabyte drive. You can install copy-protected software on drive C, but not on other drives.
Drives D through S  Equivalents of SunOS directories. They can be accessed from either DOS or SunOS, and can contain any number of files and other directories. You cannot install copy-protected software on drives D through S (install it on drive C instead). The SunOS directories referenced by DOS drives other than D, H, and R (described below) are user-defined (using the DOS EXTEND command).
Drive D  The current SunOS directory when the DOS window was opened. May subsequently be changed to any other directory.
Drive H  The home directory of the user who opened the window. May subsequently be changed to any directory in the user’s home directory tree.
Drive R  Initially equivalent to the root directory of SunOS.
File Sharing between SunOS and DOS

File names under DOS consist of 8 characters, a period, and a 3 character extension. When a SunOS filename does not comply with these rules, its name is modified by placing a tilde (~) in an appropriate location so that the file name conforms to DOS specifications while remaining unique. It is recommended that filenames conform to DOS requirements for files to be used in both SunOS and DOS.

Because SunOS and DOS use different conventions for carriage returns, dos2unix and unix2dos are provided to convert text files between the two formats.

Command Sharing between SunOS and DOS

The /etc/dos/unix directory contains a list of SunOS commands accessible from DOS. Other SunOS commands not in this list can be executed from DOS with the command `unix command'. SunOS commands always use SunOS filename conventions and DOS commands always use DOS filename conventions, regardless of whether either type of command is executed from SunOS or DOS. Only DOS commands can use drives A and C.

OPTIONS

- **b**  
  Boots (loads) DOS and opens a window using the AUTOEXEC.BAT and CONFIG.SYS files instead of ~/pc/quickpc. A DOS sign-on message is displayed in the window. Normally, DOS boots from settings in .quickpc unless C:AUTOEXEC.BAT, C:CONFIG.SYS, or /etc/dos/defaults/rom has a date newer than the .quickpc file (see the -s option).

- **s**  
  Boot DOS and save a new .quickpc file under the name specified on the SAVE line in ~/pc/setup.pc. Use this option after making changes to drive C's AUTOEXEC.BAT or CONFIG.SYS. Exits DOS after saving the .quickpc file.

- **p config**  
  Loads an alternate file instead of setup.pc.

- **q**  
  Forces dos to read settings from the quickpc file (as specified in setup.pc) even if C:AUTOEXEC.BAT, C:CONFIG.SYS or /etc/dos/defaults/rom have been updated since you last typed dos-s.

- **c command**  
  Executes the given DOS command in the newly created window. If you use the -c option, -c and the command that follows it must be the last items on the command line.

- **w**  
  Runs DOS text-only commands and applications in the current SunView Commands window.

ENVIRONMENT

DOS_LOCKING  
This environment variable determines which locking service is used to lock drive C for write access. If it is set to on, DOS uses the locking service on the server where the home directory is located. This locks drive C for access from any DOS window on the network. If it is set to off, DOS uses the local system's locking service. This locks drive C only for access from DOS windows running on the local system. The default is on. Some servers (for example, some VAX/Ultrix systems) do not provide an NFS locking service. For home directories stored on these servers, set the variable to off to avoid an error message when a DOS window starts up.

DOS_PRINTER  
The value of this environment variable indicates the timeout (in seconds) for printing. A value of 20 (the default) indicates that jobs will be sent to the UNIX print spooler after 20 seconds of no printing activity from DOS to that printer. A value of 0 indicates that the spooler must be flushed manually from the menu in the window.

DOSLOOKUP  
If on, this environment variable indicates that a command should be tried as a DOS command if not recognized by SunOS. If DOS supports the command, a DOS window is created and the command executed in that window. If the command does not exist, the normal SunOS error message results.
FILES

/etc/dos/unix
Files in this directory indicate which SunOS commands are accessible from DOS.

/etc/dos/defaults/.quickpc
Default .quickpc file copied into user’s home PC directory (“/pc”) the first time a DOS window is started. Not used by DOS in this location.

/etc/dos/defaults/setup.pc
Default setup.pc file copied into user’s home DOS directory (“/pc”) the first time a DOS window is started. Not used by DOS in this location.

/etc/dos/defaults/boards.pc
Stores information about IBM PC/XT/AT-compatible boards installed in your system.

/etc/dos/defaults/C:
Default drive C file copied into a user’s home PC directory the first time a DOS window is started.

~/.pc/autoexec.bat
Contains drive assignments, search paths, and other startup commands. Searched after C:AUTOEXEC.BAT and D:AUTOEXEC.BAT.

C:AUTOEXEC.BAT
Contains commands to access system printers and special drives. You should not need to change the AUTOEXEC.BAT on drive C. Put your changes in the AUTOEXEC.BAT on drive H (in your home directory).

D:AUTOEXEC.BAT
If an AUTOEXEC.BAT file exists in the current directory (represented by drive D), DOS executes it after running C:AUTOEXEC.BAT.

C:CONFIG.SYS
Specifies device drivers and other system parameters. C:CONFIG.SYS is not accessible from SunOS.

~/.pc/setup.pc
Defines printers, standard PC devices, and drive C. One or more of these files may exist, under various names that you assign.

~/.pc/.quickpc
An image of DOS as last saved with dos -s, including all DOS environment variables and drivers that were in effect at that time. DOS normally reads this file at startup.

~/.pc/C:
A user’s personal copy of drive C.

DIAGNOSTICS

Cannot save filename quick-start file.
The dos command was unable to save the specified quick-start file. Check the SAVE setting in your PC setup file (normally “/pc/setup.pc”) Also check file access permissions on the specified quick-start file.

Cannot load filename quick-start file.

dos was unable to read the specified quick-start file. Check the SAVE setting in your setup.pc file. Also check file access permissions on the specified quick-start file.

Possible software incompatibility. Unsupported 286 instruction instruction at address.

Possible software incompatibility. Unsupported 386 instruction.

Possible software incompatibility. Segment wrap.

Possible software incompatibility. Two-byte opcode not supported.
The application you are running was written specifically for 80286 or 80386 machines. Software run from a DOS window must be compatible with 8086 systems.

Copying default configuration files into your home directory.
This is the first time you have run the dos command. A “/pc” directory is being set up, and DOS-related files are being copied into it.

Another DOS window already has access to device.
Your PC configuration file (normally “/pc/setup.pc”) is requesting access to a physical device that another DOS window is using.

Port number number out of range for board board.
The port number specified in the /etc/dos/defaults/boards.pc is invalid.

IRQ value number out of range for board board.
The interrupt level specified in the /etc/dos/defaults/boards.pc is invalid.

Interrupt level number is used by DOS to support the device.
The interrupt level specified in the `/etc/dos/defaults/boards.pc` conflicts with an interrupt value currently being used by either a physical or emulated DOS device.

**I/O address range** `address - address requested for board already in use by device`.

The address range specified in the `/etc/dos/defaults/boards.pc` conflicts with range currently being used by either a physical or emulated DOS device.

**Cannot share device with a hardware interrupt.**

A shared device specified in the `/etc/dos/defaults/boards.pc` was also assigned an interrupt level in this file. Shared devices cannot be assigned interrupt levels.

**Couldn't find board in boards.pc.**

A file specified in the PC setup file (normally `/pc/setup.pc`) is not listed in the `/etc/dos/defaults/boards.pc` file. Check the `setup.pc` file, or add an entry for the board in `boards.pc`.

**ROM is newer than .quickpc. Rebooting program name.**

Save a new `.quickpc` file by issuing the command `dos -s`.

**Warning: Your personal drive C (pathname)**

is not protected against simultaneous access by more than one workstation. Ask your system administrator to upgrade `server` to use the lock manager. Until your home directory server is updated with this program, do not use `program name` when you are logged into more than one workstation. The system on the network where your drive C is stored has not protected the drive against access by DOS windows in other workstations on the network. This usually means that the server where your home directory is stored does not provide an NFS locking service. To avoid this error message, set the environment variable DOS_LOCKING to off.

SEE ALSO

dos2unix(1), unix2dos(1)

*Sun386i User’s Guide*

*Sun386i Advanced Skills*

*DOS Reference Manual*
NAME
fdformat – format diskettes for use under SunOS

SYNOPSIS
/usr/etc/fdformat [-L][-2]

AVAILABILITY
Sun386i systems only.

DESCRIPTION
fdformat is a program for formatting diskettes to use with the SunOS operating system. All new blank
diskettes must be formatted before use. fdformat formats and verifies each track on the diskette, and
terminates if it finds any bad sectors. fdformat destroys all existing data on the diskette.

By default, fdformat formats a 1.44 megabyte high density diskette. Use the -L option to format low
density diskettes.

Use the -2 option to format diskettes in the optional external 5 1/4" floppy drive.

To format a diskette for use under MS-DOS, use the MS-DOS format command in a DOS window on the
Sun386i system.

OPTIONS
-L Format a low density diskette (720 kilobyte) diskette.
-2 Format a diskette in the optional external 5 1/4" floppy drive.

FILES
/dev/rfd0c /dev/rfd10c /dev/rfd2c /dev/rfd12c

BUGS
The SunOS system currently doesn’t support bad sector mapping on diskettes. Therefore, a diskette is
unusable if fdformat finds an error (bad sector).

SEE ALSO
dos(1)
NAME
fontedit – a vfont screen-font editor

SYNOPSIS
fontedit [ generic-tool-argument ] . . . [ font_name ]

AVAILABILITY
This command is available with the SunView 1 User’s software installation option. Refer to Installing the Sun Operating System for information on how to install optional software.

DESCRIPTION
fontedit is an editor for fixed-width fonts in vfont format (or Sun386i vfont format) whose characters are no taller than 24 pixels (larger characters will not fit completely onto the screen). For a description of vfont format, see vfont(5).

OPTIONS
generic-tool-argument
fontedit accepts any generic tool argument as described in sunview(1). Otherwise, you can manipulate the tool using the Frame Menu.

COMMANDS
To edit a font, type ‘fontedit’. A font_name may be supplied on the command line or may be typed into the Control panel once the program has started. If it exists, the font_name file must be in vfont format (or Sun386i vfont format). When the program starts, it displays a single large window containing four subwindows. From top to bottom, the four subwindows are:
1) The top subwindow, a message subwindow, displays messages, prompts, and warnings.
2) The second subwindow from the top, an Control panel, allows you to set global parameters for the entire font and specify operations for editing any single character. The options are:
   (Load) Load in the font specified in the file name field. The program will warn you if you try to read over a modified font. For the Sun386i system, either vfont or Sun386i vfont format can be read.
   (Store) Store the current font onto disk with the name in file name field. For the Sun386i system, fontedit always stores the font in Sun386i vfont format. Use fontflip_to_68k to create a corresponding vfont format file.
   (Quit) Quit the program; warns you if you have modified the font.

Font name:
The name of the font. On the Sun386i system, the system appends the suffix .i386 before opening the file and attempting to use it; if it does not find the font, it attempts to open the original font name specified. By convention, Sun386i vfonts have the .i386 extension.

Max Width and Max Height:
The size, in pixels, of the largest character in the font. If you edit an existing font, these parameters are set automatically; you must set them if you are creating a new font. Changing either of these values for an existing font may alter the glyph of some characters of the font. If the glyph size of a character is larger than the new max size, then that character is clipped to the new size (its bottom and right edges are moved in). However, if a glyph’s size is smaller than the new size, the glyph is left alone.

Caps Height and X-Height:
The distance, in pixels, between the top of a capital and lowercase letter and the baseline. When an existing font is edited, the values of Caps Height and X-Height are estimated by fontedit, and may require some adjustment.

Baseline:
The number of pixels from the top (that is, the upper left corner) of the character to the baseline. For an existing font, the value of the largest baseline distance is used.

SunOS Release 4.0.1  Last change: 3 November 1988  190
For a new font, each character will have the same baseline distance. If this value is changed, then the baseline distance for all characters in the font will be the new value.

**Apply**

Apply the current values of *Max Width, Max Height, Caps Height, X-Height,* and *Baseline* to the font. That is, changes made to these values do not take effect until *Apply* is selected.

**Operation:**

This is a list of drawing and editing operations that you can perform on a character. For drawing, the left mouse button draws in black, and the middle draws in white. Operations are:

- **Single Pt**  
  Press a mouse button down and a grey cell will appear; move the mouse and the cell will follow it. Releasing the button will draw.

- **Pt Wipe**  
  Pressing a button down will draw and moving with the button down will continue drawing until the button is released.

- **Line**  
  Button down marks the end point of a line; moving with the button down rubber bands a line; releasing button draws the line.

- **Rect**  
  Like **Line** except draws a rectangle.

- **Cut**  
  Button down marks one end of rectangle, and moving rubber bands the outline of the rectangle. Button up places the contents of the rectangle into a buffer and then "cuts" (draws in white) the rectangular region from the character. The **Paste** operation (below) gets the data from the buffer.

- **Copy**  
  Like **Cut** except that the region is just copied; no change is made to the character.

- **Paste**  
  Button down displays a rectangle the size of the region in the buffer. Moving with the button down moves the rectangle. Button up pastes the contents of the buffer into the character.

The contents of the **paste** buffer cannot be transferred between tools.

In **Copy** or **Cut** mode, holding down the shift key while pressing the left or middle mouse button will perform a **Paste** action. For best results, after placing a region in the buffer, press down the shift key and hold it down, then press down the mouse button. Release the mouse key to paste the region and then release the shift key.

3) The third subwindow echoes the characters in the current font as they are typed. Note that the cursor must be in this window in order to see the characters. Your character delete key will delete the echoed characters.

4) The bottom subwindow, the editing subwindow, displays eight smaller squares at its top; these are called **edit buttons**. The top section of each of these buttons contains a line of text in the form \( nnn: c \), where \( nnn \) is the hexadecimal number of the character and \( c \) is the standard ASCII character corresponding to that number. In the lower section of the button the character of the current font, if it exists, is displayed. Clicking once over an editing button selects its character for editing.

Just below this row of buttons is a box with the characters "0 9 A Z a z" in it. This box is called a **slider**. The slider allows you to scroll around in the font and select which section of the font you want displayed in the edit buttons. The black rectangle near "a" is an indicator which shows the section of the font that is displayed in the buttons above. To move the indicator, select it by pressing the left or middle mouse button down over the indicator and then move the mouse to the left or right with the button down; the indicator will slide along with the cursor. Releasing the button selects the new section of the font. A faster method of moving about in the font is to just press down and release the mouse button above the area you want without bothering to drag the indicator. Another method of scrolling through the characters of the font is to press a key on the keyboard when the cursor is in the bottom window; that character is the first one displayed in the
edit buttons.

EDITING CHARACTERS:
To edit a character, click once over the edit button where the character is displayed. When you do this, an edit pad will appear in the bottom subwindow.

The edit pad consists of an editing area bordered by scales, a proof area, and 3 command buttons. The editing area is **Max Width** by **Max Height** when the pad opens, and displays a magnified view of the selected character. Black squares indicate foreground pixels. The editing area is surrounded by scales which show the current **Caps Height**, **X-Height** and **Baseline** in reverse video.

Just outside the scales, on the top, right side, and bottom of the pad, are three small boxes with the capital letters "R", "B", and "A" in them. These boxes are movable sliders that change the right edge, bottom edge, and x-axis advance of the character respectively. In a fixed-width font, these values are usually the same for all characters; however, in a variable-width font these controls can be used to set these properties for each character.

To the right of the pad is the proof area where the character is displayed at normal (that is, screen) resolution and three buttons. The three buttons are:

- **Undo** Clicking the left or middle mouse button undoes the last operation.
- **Store** Stores the current representation of the character in the font.
- **Quit** Closes the edit pad.

In the bottom subwindow, the right mouse button displays a menu of operations. These operations are the same as those in the control panel discussed above; you can select the current operation by either picking the operation in the control panel or by selecting the appropriate menu with the right button of the mouse. When the cursor is in the other subwindows, the right button displays the standard tool menu.

FILES

```
/usr/lib/fonts/fixedwidthfonts
```

Sun-supplied screen fonts

SEE ALSO

sunview(1), vswap(1), vfont(5) fontflip_to_68k(8) fontflip_to_i386(8)

BUGS

Results are unpredictable with variable-width fonts. The baseline should be greater than 0 or else the font cannot be read in by fontedit or by sunview(1).
NAME
help_open – causes help_viewer to open a file

SYNOPSIS
help_open [-a] filename

AVAILABILITY
Sun386i systems only.

DESCRIPTION
help_open is used to cause a running help_viewer to open a file. 'filename' is typically the name of a
help_viewer file. A call is made to help_viewer using the same RPC mechanism as is used by Spot Help.

If "filename" is relative, help_viewer looks for it relative to the default help directory (as defined in the
user’s defaults database). Otherwise, help_viewer treats "filename" as absolute.

If the RPC call to help_viewer fails, help_open attempts to spawn help_viewer, with "filename" as a
command line argument. If the -a command line option was given, then "filename" is first converted to
an absolute path name, as described in OPTIONS, below.

OPTIONS
-a Convert "filename" to absolute path; this option causes help_open to get the current working
directory and append it to the front of "filename" (thus creating an absolute pathname) before
passing "filename" on to help_viewer. This allows help_open to be used with other processes,
such as Sun Organizer (see organizer (1)), which deal in relative pathnames. The -a option has
no effect if "filename" begins with the character ‘/’.

EXAMPLES
maple% help_open help/Help_Basics
This causes help_viewer to open the file help/Help_Basics. This file is located relative to the default
help directory (as defined in the user’s defaults database). So in the case where the default help direc-
tory was set to /vol/help/language/USA-English/, this would be /vol/help/language/USA-
English/help/Help_Basics.
maple% help_open help/Help_Basics 3
Same as previous example, but opens Help_Basics to page 3.
maple% help_open /home/mtravis/somefile
Causes help_viewer to open somefile, relative to /home/mtravis/.
maple% cd /home/ahinkle
maple% help_open -a anotherfile
Causes help_viewer to open /home/ahinkle/anotherfile.

FILES
/usr/lib/help/*

SEE ALSO
organizer(1), help_viewer(1), help(5), help_viewer(5), Sun386i Developer’s Guide
NAME
help_viewer – SunView program providing help with applications and desktop

SYNOPSIS
/usr/bin/help_viewer [ options ]

AVAILABILITY
Sun386i systems only.

DESCRIPTION
help_viewer gives you quick access to documentation about SunView applications and the SunView Desktop. This help consists of intermixed text and graphics displayed in a window called the Help Viewer.

You start and control help_viewer by one of these methods:
1. Typing help_viewer at a shell prompt.
2. Clicking on the More Help button in a Spot Help window.
3. Sending instructions to the Help Viewer using the help_open command.

The documentation within help_viewer is extendable, but as shipped it includes handbooks for the DeskTop, mailtool(1), textedit(1), sunview(1), organizer(1), dos(1), coloredit(1), snap(1), and itself (help_viewer).

Developers and users can include additional handbooks by modifying /vol/help/format/Top_Level. See help_viewer(5).

The user moves between the various pages of help with the assistance of hypertext links. Links are connections between pages of text. The convention is to use underlined text to indicate the presence of a link. When the user double-clicks on a link, the text associated with the topic indicated by the link is shown in the Help Viewer. There are links in many places to make it quick and easy to go from place to place within the help_viewer database.

Many help topics contain more than one page of text, and in these cases a link to the next page and to the previous page is available at the upper-right corner of the Help Viewer; this allows the user to page through the document.

The user’s current position within the hierarchy of text is indicated by the links at the upper-left corner of the Help Viewer. The last link in the list is the level just above the user’s current position.

OPTIONS
The standard SunView options for window size, position, fonts, and other options are accepted. But note that the font setting only affects the font in the Help Viewer namestripe. Also, Help Viewer text does not wrap as a window is resized. See sunview(1) for details.

-dir dirname
   Name of help directory

filename [ #]
   Name of startup file relative to help directory (or /vol/help by default, as set in the Help category of defaultsedit). # is a page number separated from the filename by a SPACE. If # is omitted, the first page is shown.

FILES
/vol/help      automount point of miscellaneous help files
The files in /usr/lib/help are used by the help and the help_viewer facilities, and the SCCS help(1) facility.
Directories within /usr/lib/help named after SunView applications and the DeskTop contain specific information used by help_viewer. See help_viewer(5) for information about the files in these directories.
SEE ALSO

DIAGNOSTICS
    help_viewer(1) displays a pop-up error window if it cannot find the file required to show the requested help.
NAME
input_from_defaults, defaults_from_input – update the current state of the mouse and keyboard from the
defaults database, and vice versa

SYNOPSIS
input_from_defaults
defaults_from_input

AVAILABILITY
This command is available with the SunView 1 User's software installation option. Refer to Installing
the Sun Operating System for information on how to install optional software.

DESCRIPTION
input_from_defaults updates various parameters controlling mouse- and keyboard-processing on the
machine on which it is run. It should be used on systems that are running the SunView window sys­
tem. The parameters control the distribution of function keys on the keyboard, the assignment of but­
tons on the mouse, the scaling of mouse-to-cursor motion, and the effect of two filters on mouse-motion
originally provided to compensate for defective mice. The new values are taken from the defaults data­
base, starting with the file .defaults in the user's home directory.

On the Sun386i system, the value /Input/Keyboard_Type is read from the user's .defaults file. The
values specified there are taken from the list of keyboard maps in /usr/share/lib/keytables/*. If the
value is blank, then the keyboard will be mapped to the default map setting. Keyboard maps can be
created using the dumpkeys utility.

defaults_from_input is the inverse operation to input_from_defaults. It updates the user's private
defaults database (used by defaultsedit(1)) to reflect the current state of kernel input parameters listed
above.

FILES
$HOME/.defaults
/usr/lib/defaults/*.d
/usr/share/lib/keytables/*

SEE ALSO
defaultsedit(1), loadkeys(1), dumpkeys(1), keytables(5)
SunView Beginner's Guide

BUGS
input_from_defaults should be targetable to any user's .defaults file.
NAME
ld, ld.so – link editor, dynamic link editor

SYNOPSIS
ld [ -align datum ] [ -assert assertion-keyword ] [ -A name ] [ -B binding-keyword ] [ -d ]
[ -dc ] [ -dp ] [ -D hex ] [ -e entry ] [ -l entry ] [ -Ldir ] [ -M ] [ -n ] [ -N ] [ -o name ] [ -p ]
[ -r ] [ -s ] [ -S ] [ -t ] [ -T [ text ] hex ] [ -T data hex ] [ -u name ] [ -x ] [ -X ] [ -ysym ]
[ -z ] filename ...

DESCRIPTION
ld combines object programs to create an executable file or another object program suitable for further
ld processing (with the -r option). The object modules on which ld operates are specified on the command line, and can be:
• simple object files, which typically end in the .o suffix, and are referred to as dot-oh files
• ar(lV) library archives (.a), or libraries
• dynamically-bound, sharable object files (.so), are also referred to as shared libraries, which
are created from previous ld executions.

Unless an output file is specified, ld produces a file named a.out. This file contains the object files
given as input, appropriately combined to form an executable file.

OPTIONS
When linking debugging or profiling objects, include the -g or -pg option (see cc(lV)), as appropriate,
in the ld command.
Options should appear before filenames, except for abbreviated library names specified with -I options,
and some binding control options specified by -B (which can appear anywhere in the line).
-align datum
Force the global uninitialized data symbol datum (usually a FORTRAN common block) to be
page-aligned. Increase its size to a whole number of pages, and place its first byte at the start
of a page.
-assert assertion-keyword
Check an assertion about the link editing being performed. The assertion desired is specified
by the assertion-keyword string. ld is silent if the assertion holds, else it yields a diagnostic
and aborts. Valid assertion-keyword’s and their interpretations are:

  nodefinitions  If the resulting program were run now, there would be no run-time
                  undefined symbol diagnostics. This assertion is set by default.
  nosymbolic    There are no symbolic relocation items remaining to be resolved.
  pure-text     The resulting load has no relocation items remaining in its text.
-A name
Incremental loading: linking is to be done in a manner so that the resulting object may be read
into an already executing program. name is the name of a file whose symbol table is taken as a
basis on which to define additional symbols. Only newly linked material is entered into the
text and data portions of a.out, but the new symbol table will reflect all symbols defined before
and after the incremental load. This argument must appear before any other object file in the
argument list. One or both of the -T options may be used as well, and will be taken to mean
that the newly linked segment will commence at the corresponding addresses (which must be a
multiple of the page size). The default value is the old value of _end.
-Bbinding-keyword
Specify allowed binding times for the items which follow. Allowed values of binding-keyword are:

dynamic
Allow dynamic binding: do not resolve symbolic references, allow creation of run-time symbol and relocation environment. -Bdynamic is the default. When -Bdynamic is in effect, all sharable objects encountered until a succeeding -Bstatic may be added dynamically to the object being linked. Non-sharable objects are bound statically.
nosymbolic
Do not perform symbolic relocation, even if other options imply it.
static
Bind statically. Opposite of -Bdynamic. Implied when either -n or -N is specified. Influences handling of all objects following its specification on a command line until the next -Bdynamic.
symbolic
Force symbolic relocation. Normally implied if an entry point has been specified with -e, or if dynamic loading is in effect.

-d
Force common storage for uninitialized variables and other common symbols to be allocated in the current ld run, even when the -r flag is present (which would otherwise postpone this binding until the final linking phase).

-dc
Do -d, but also copy initialized data referenced by this program from shared objects.

-dp
Force an alias definition of undefined procedure entry points. Used with dynamic binding to improve sharing and the locality of run-time relocations.

-D hex
Pad the data segment with zero-valued bytes to make it hex bytes long.

-e entry
Define the entry point: the entry argument is made the name of the entry point of the loaded program. Implies -Bs symbolic.

-I[x,v]
This option is an abbreviation for the library name libx.a, where x is a string. ld searches for libraries first in any directories specified with -L options, then in the standard directories /lib, /usr/lib, and /usr/local/lib. A library is searched when its name is encountered, so the placement of a -l is significant. If a dynamically loadable object is found, and -Bdynamic is in effect at that point on the command line, then ld prepares to access the object for relocation at run-time. In such a case, the optional .v suffix can be used to indicate a specific library version.

-Ldir
Add dir to the list of directories in which to search for libraries. Directories specified with -L are searched before the standard directories, /lib, /usr/lib, and /usr/local/lib.

-M
Produce a primitive load map, listing the names of the files which will be loaded.

-n
Arrange (by giving the output file a 0410 magic number) that when the output file is executed, the text portion will be read-only with the data areas placed at the beginning of the next address boundary following the end of the text. Implies -Bstatic.

-N
Do not make the text portion read-only. (Use magic number 0407.) Implies -Bstatic.

-o name
name is made the name of the ld output file, instead of a.out.

-p
Arrange for the data segment to begin on a page boundary, even if the text is not shared (with the -N option).

-r
Generate relocation bits in the output file so that it can be the subject of another ld run. This flag also prevents final definitions from being given to common symbols, and suppresses the undefined symbol diagnostics.

-s
Strip the output, that is, remove the symbol table and relocation bits to save space (but impair the usefulness of the debuggers). This information can also be removed by strip(l).
-S   Strip the output by removing all symbols except locals and globals.
-t   Trace: display the name of each file as it is processed.
-T{ text } hex
     Start the text segment at location hex. Specifying -T is the same as using the -Ttext option.
-Tdata hex
     Start the data segment at location hex. This option is only of use to programmers wishing to write
     code for PROMs, since the resulting code cannot be executed by the system.
-u name
     Enter name as an undefined symbol. This is useful for loading wholly from a library, since
     initially the symbol table is empty and an unresolved reference is needed to force the loading
     of the first routine.
-x   Preserve only global (non-globl) symbols in the output symbol table; only enter external sym­
     bols. This option saves some space in the output file.
-X   Record local symbols, except for those whose names begin with L. This option is used by cc
to discard internally generated labels while retaining symbols local to routines.
-ysym
     Display each file in which sym appears, its type and whether the file defines or references it.
     Many such options may be given to trace many symbols. It is usually necessary to begin sym
     with an '_', as external C, FORTRAN and Pascal variables begin with underscores.
-z   Arrange for the process demand paged from the resulting executable file (0413 magic number).
     This is the default. Results in a (32-byte) header on the output file followed by text and data
     segments, each of which has a multiple of page-size bytes (being padded out with NULL char­
     acters in the file if necessary). With this format the first few BSS segment symbols may actu­
     ally end up in the data segment; this is to avoid wasting the space resulting from rounding the
     data segment size. Implies -Bdynamic.

USAGE

Command Line Processing
In general, options should appear ahead of the list of files to process. Unless otherwise specified, the
effect of an option covers all of ld operations, independent of that option's placement on the command
line. Exceptions to this rule include some of the binding control options specified by ' -B ' and the
abbreviated library-names specified by ' -I '. These may appear anywhere, and their influence is depend­
ent upon their location. Some options may be obtained from environment variables, such options are
interpreted before any on the command line (see ENVIRONMENT, below).

Object File Processing
The files specified on the command line are processed in the order listed. Information is extracted from
each file, and concatenated to form the output. The specific processing performed on a given file
depends upon whether it is a simple object file, a library archive, or a shared library.
Simple object (.o) files are concatenated to the output as they are encountered.
Library archive (.a) files are searched exactly once each, as each is encountered; only those archive
entries matching an unresolved external reference are extracted and concatenated to the output. If a
member of an archive references a symbol defined by another member of that same archive, the
member making the reference must appear before the member containing the definition.
On Sun386i, a library contains a dictionary of symbols. On other Sun systems, processing library
archives through ranlib(1) provides this dictionary. In addition, you can use lorder(1), in combination
with tsort(1) to place library members in calling order (see lorder(1) for details), or both (for fastest
symbol lookup). The first member of an archived processed by ranlib has the reserved name of
__SYMDEF, which ld takes to be the dictionary of all symbols defined by members of the archive.
Sharable objects (.so) are scanned for symbol definitions and references, but are not normally included in the output from ld, except in cases where a shared library exports initialized data structures and the -de option is in effect. However, the occurrence of each sharable object file in the ld command line is noted in the resulting executable file; this notation is utilized by an execution-time variant of ld, ld.so, for deferred and dynamic loading and binding during execution. See Execution-Time Loading, below, for details.

The -l option specifies a short name for an object file or archive used as a library. The full name of the object file is derived by adding the prefix lib and a suffix of either .a or .so[.v] to indicate an ar(1V) archive or a shared library, respectively. The specific suffix used is determined through rules discussed in Binding and Relocation Semantics, below.

ld searches for the desired object file through a list of directories specified by -L options, the environment variable LD_LIBRARY_PATH, and finally, the built-in list of standard library directories: /lib, /usr/lib, and /usr/local/lib.

**Binding and Relocation Semantics**

The manner in which ld processes a given object file is dependent in part upon the binding mode in which it is operating at the time the file is encountered. This binding mode is specified by the -B flag, which takes the keyword arguments:

- **dynamic** Allow dynamic binding, do not resolve symbolic references, and allow creation of execution-time symbol and relocation information. This is the default setting.
- **static** Force static binding, implied by options that generate non-sharable executable formats.

-Bdynamic and -Bstatic may be specified several times, and may be used to toggle each other on and off. Like -l, the influence of each depends upon its location within the command line. When -Bdynamic is in effect, -l searches may be satisfied by the first occurrence of either form of library (.so or .a), but if both are encountered, the .so form is preferred. When -Bstatic is in effect, ld refuses to use any .so libraries it encounters; it continues searching for the .a form. Furthermore, an explicit request to load a .so file is treated as an error.

After ld has processed all input files and command line options, the form of the output it produces is based on the information provided in both. ld first tries to reduce all symbolic references to relative numerical offsets within the executable it is building. To perform this symbolic reduction, ld must be able to determine that:

- all information relating to the program has been provided, in particular, no .so is to be added at execution time; and/or
- the program has an entry point, and symbolic reduction can be performed for all symbols having definitions existing in the material provided.

It should be noted that uninitialized common areas (for example, uninitialized C globals) are allocated by the link editor after it has collected all references. In particular, this allocation can not occur in a program that still requires the addition of information contained in a .so file, as the missing information may affect the allocation process. Initialized commons however, are allocated within the executable in which their definition appears.

After ld has performed all the symbolic reductions it can, it attempts to transform all relative references to absolute addresses. ld is able to perform this relative reduction only if it has been provided some absolute address, either implicitly through the specification of an entry point, or explicitly through ld command-line options. If, after performing all the reductions it can, there are no further relocations or definitions to perform, then ld has produced a completely linked executable.

**Execution-Time Loading**

In the event that one or more reductions can not be completed, the executable will require further link editing at execution time in order to be usable. Such executables contain a data structure identified with the symbol _DYNAMIC. An incompletely linked main program should be linked with a
bootstrap routine that invokes \texttt{ld.so}, which uses the information contained in the main program's \
\texttt{\_DYNAMIC} to assemble the rest of the executables constituting the entire program. A standard Sun 
compilation driver (such as \texttt{cc(1V)}) automatically includes such a module in each main executable.

When \texttt{ld.so} is given control on program startup, it finds all .so files specified when the program was 
constructed (and all .so's on which they depend), and loads them into the address space. \texttt{ld.so} then 
completes all remaining relocations, with the exception of procedure call relocations; failure to resolve 
given non-procedural relocation results in termination of the program with an appropriate diagnostic.

Procedure relocations are resolved when the referencing instruction is first executed. It should be noted 
that it is possible for undefined symbol diagnostics to be produced during program execution if a given 
target is not defined when referenced.

Although it is possible for binding errors to occur at execution-time, such an occurrence generally indicates 
something wrong in the maintenance of shared objects. \texttt{ld}'s \texttt{\_assert definitions} function (on by 
default) checks at \texttt{ld}-time whether or not an execution-time binding error would occur.

\section*{Version Handling for Shared Libraries}

To allow the independent evolution of .so's used as libraries and the programs which use them, \texttt{ld}'s 
handling of .so files found through \texttt{-l} options involves the retention and management of version control 
information. The .so files used as such shared libraries are post-fixed with a Dewey-decimal format 
string describing the version of the library contained in the file.

The first decimal component is called the library's major version number, and the second component its 
minor version number. When \texttt{ld} records a .so used as a library, it also records these two numbers in 
the database used by \texttt{ld.so} at execution time. In turn, \texttt{ld.so} uses these numbers to decide which of mul-
tiple versions of a given library is best or whether any of the available versions are acceptable. The 
rules are:

\begin{itemize}
  \item Major Versions Identical: the major version used at execution time must exactly match the 
version found at \texttt{ld}-time. Failure to find an instance of the library with a matching major 
version causes a diagnostic to be issued and the program's execution to be terminated.
  
  \item Highest Minor Version: in the presence of multiple instances of libraries that match the 
desired major version, \texttt{ld.so} uses the highest minor version it finds. However, if the highest 
minor version found at execution time is less than the version found at \texttt{ld}-time, a warning 
diagnostic is issued; program execution continues.
\end{itemize}

The semantics of version numbers are such that major version numbers should be changed whenever 
interfaces are changed. Minor versions should be changed to reflect compatible updates to libraries, and 
programs will silently favor the highest compatible version they can obtain.

\section*{Special Symbols}

A number of symbols have special meanings to \texttt{ld} and programs should not define these symbols. The 
symbols described below are those actually seen by \texttt{ld}. Note: C and several other languages prepend 
symbols they use with \texttt{\_}.

\begin{itemize}
  \item \texttt{\_etext} The first location after the text of the program.
  
  \item \texttt{\_edata} The first location after initialized data.
  
  \item \texttt{\_end} The first location after all data.
  
  \item \texttt{\_DYNAMIC} Identifies an \texttt{ld}-produced data structure. It is defined with a non-zero value in executables 
which require execution-time link editing. By convention, if defined, it is the first symbol in 
the symbol table associated with an \texttt{a.out} file.
  
  \item \texttt{\_GLOBAL\_OFFSET\_TABLE} A position-independent reference to an \texttt{ld}-constructed table of addresses. This table is con-
structed from position-independent data references occurring in objects that have been assem-
bled with the assembler's \texttt{-k} flag (invoked on behalf of C compilations performed with the
--pic flag). A related table (for which no symbol is currently defined) contains a series of transfer instructions and is created from position-independent procedure calls or, if -dp is specified to ld, a list of undefined symbols.

Symbols in object files beginning with the letter L are taken to be local symbols and unless otherwise specified are purged from ld output files.

ENVIRONMENT

LD_LIBRARY_PATH

A colon-separated list of directories in which to search for libraries specified with the -l option. Similar to the PATH environment variable. LD_LIBRARY_PATH also affects library searching during execution-time loading.

LD_OPTIONS

A default set of options to ld. LD_OPTIONS is interpreted by ld just as though its value had been placed on the command line, immediately following the name used to invoke ld, as in:

example% ld $LD_OPTIONS ... other-arguments ...

Note: Environment variable-names beginning with the characters 'LD_' are reserved for possible future enhancements to ld.

FILES

/usr/lib/lib*.a libraries
lib*.so.v shared libraries
lib*.sa.v exported, initialized shared library data
/usr/lib/ld.so execution-time ld
/usr/lib/*crt*.o default program bootstraps
a.out output file
/usr/local/lib

SEE ALSO

as(1), ar(1V), cc(1V), lorder(1), ranlib(1), strip(1), tsort(1)

BUGS

Options are being overloaded and are an inappropriate vehicle for describing to ld the wide variety of things it can do. There needs to be a link-editing language which can be used in the more complex situations.

The -r option does not properly handle programs assembled with the -k (position-independent) flag, invoked from cc with --pic or --PIC.
NAME
load, loadc - load Application SunOS or Developer’s Toolkit clusters

SYNOPSIS
load [ filename ...]
loadc [ cluster ...]

AVAILABILITY
Sun386i systems only.

DESCRIPTION
load loads the optional clusters in the Application SunOS or the Developer’s Toolkit that contain the
files specified in the filename arguments. loadc loads the optional clusters in the Application SunOS or
the Developers Toolkit specified in the cluster arguments. When you specify the special cluster name
appl with loadc, then loadc loads all the Application SunOS clusters; likewise, when you specify devel
to loadc, it loads all the Developer’s Toolkit clusters.

load and loadc require the user to specify the distribution media type (3.5" diskette or 1/4" tape) for the
system and to insert the specified 3.5" diskette or 1/4" tape. The user will be asked to confirm that the
specified media has been inserted. If the user confirmation is negative, no software will be loaded from
the specified media.

Without arguments, load and loadc display a summary of the clusters in the Application SunOS and
Developer’s Toolkit, including the load state and size of each cluster.

EXAMPLES
To load the cluster that contains the spell(1) command:

% load spell
Enter your distribution media type (1=1/4" tape, 2=3.5" diskette): 2
Insert diskette n to load the spellcheck cluster, confirm(y/n): y
Loading the spellcheck cluster ...
The spellcheck cluster has been loaded.
space used by clusters: 6021K bytes
total space remaining: 20432K bytes

To load the spellcheck cluster:

% loadc spellcheck
Enter your distribution media type (1=1/4" tape, 2=3.5" diskette): 2
Insert diskette n to load the spellcheck, confirm(y/n): y
Loading the spellcheck cluster ...
The spellcheck cluster has been loaded.
space used by clusters: 6021K bytes
total space remaining: 20432K bytes

To display a summary of the clusters in the Application SunOS and Developer’s Toolkit:

% load
Application SunOS Clusters:

<table>
<thead>
<tr>
<th>available cluster</th>
<th>size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes accounting</td>
<td>265K</td>
</tr>
<tr>
<td>no advanced_admin</td>
<td>501K</td>
</tr>
</tbody>
</table>

SunOS Release 4.0.1 Last change: 2 November 1988 267
Developer’s Toolkit Clusters:

<table>
<thead>
<tr>
<th>available cluster</th>
<th>size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no base_devel</td>
<td>6907K</td>
</tr>
</tbody>
</table>

space used by clusters: 6021K bytes
total space remaining: 20432K bytes

A cluster is available if it has been loaded using `load` or `loadc` or if it has been mounted across the network.

ENVIRONMENT

LOADMEDIA Used to specify the distribution media type for the system. It can be set to `diskette` to specify 3.5” diskette or `tape` to specify 1/4” tape. If it is set, `load` and `loadc` will not ask the user to enter the distribution media type.

FILES

```
/usr/loaded/appl    where Application SunOS clusters are loaded (or mounted)
/usr/loaded/devel  where Developer’s Toolkit clusters are loaded (or mounted)
/usr/lib/load/*    data files
```

SEE ALSO

`unload(1), cluster(1), toc(5)`

*Sun386i System Setup and Maintenance*

DIAGNOSTICS

Wrong `diskette/tape`

An incorrect diskette or tape was inserted. The user will again be asked to insert the specified media.

The file `filename` is not in any of the optional software clusters.

The specified file is not part of the Application SunOS or Developer’s Toolkit.

There is no `cluster` cluster.

The specified cluster is not part of the Application SunOS or Developers Toolkit.

The cluster `cluster` is already loaded, overwrite? (y/n):

The specified cluster appears to have been loaded already. Type `y` followed by RETURN to have the cluster loaded or `n` followed by RETURN to cancel the loading of the cluster.

Cluster `cluster` requires `nK`; there is not enough disk space.

There is not enough disk space to hold the specified cluster.

The `cluster` cluster has not been loaded.

The loading of the specified cluster has been canceled or interrupted by the user.

The Application SunOS (and/or) Developers Toolkit are mounted.

The Application SunOS or Developers Toolkit or both are mounted across the network and can not be loaded or unloaded.

The tape/diskette drive is currently in use.

You are trying to load a cluster from tape (or diskette) and another process currently has control of the tape (or diskette) drive.
NAME
loadkeys, dumpkeys – load and dump keyboard translation tables

SYNOPSIS
loadkeys [ filename ]
dumpkeys

DESCRIPTION
loadkeys
loadkeys reads the file specified by filename, or, if no file is specified and the keyboard is a Type 4 keyboard, a default file for the layout indicated by the DIP switches on the keyboard, and modifies the keyboard streams module's translation tables. The file is in the format specified by keytables(5).

If the layout code in the DIP switches on the keyboard has the hexadecimal value Ox dd, the file loaded by loadkeys by default is /usr/share/lib/keytables/layout_dd. These files specify only the entries that change between the different Type 4 keyboard layouts.

dumpkeys
dumpkeys writes, to the standard output, the current contents of the keyboard streams module's translation tables, in the format specified by keytables(5).

FILES
/usr/share/lib/keytables/layout_dd
default keytable files

SEE ALSO
kb(4M), keytables(5)
NAME
organizer – file and directory manager

SYNOPSIS
organizer

AVAILABILITY
Sun386i systems only.

DESCRIPTION
organizer is a SunView application for viewing and manipulating files and directories. It performs
many of the functions of the ls, cd, cp, rm, mv, mkdir, rmdir, backup, restore, find, and chmod
commands, and with a visual interface.

At any given time, the organizer window normally shows the files and directories in a single directory,
representing each file or directory with an appropriate illustrated icon. The illustration indicates
whether a file is a directory, contains text, is an executable program, or optionally a user-defined file
type.

When organizer is switched into Map mode, the icons are arranged to indicate the hierarchy of files
and directories. Double clicking on a directory icon shows the contents of that directory in a new
column.

Several display modes are available, and can be set for an individual organizer window or for all
organizer windows. You can select whether hidden files are shown, whether just the name, the name
and information, or name and icon are shown for each file and directory, and how the contents are
sorted.

Text files can be "edited" by double clicking on the file’s icon. The contents of the file are then shown
and can be edited in a separate text editor window. In the .orgrc file you can specify the EXECUTE,
EDIT, and PRINT applications for your own user-defined file types.

You can move down through the directory hierarchy by double clicking on a directory icon, and up by
double clicking on the parent directory name on the ancestor list in the upper left corner of the organizer
panel.

Copying, moving, and deleting require you to select one or more files. To select a file, click the left
button on it (don’t double click—this will open the file). To select additional files to be operated on,
click the middle button on each additional file. Copying and moving operations require a destination
directory. After the files are selected, change directories to the desired destination as described above,
and then "drop" the files with the Drop button on the command panel. If the copy involves overwriting
an identically named file, an alert will allow you to confirm that you want to overwrite the file. If you
copy a file and then "drop" it in the same directory, organizer will prepend copy_of_ to the filename
of the new file.

FILES
/usr/include/images/* file and directory icons
"/.orgrc

SEE ALSO
orgrc(5)
NAME
strip -- remove symbols and relocation bits from an object file

SYNOPSIS
strip filename ...

DESCRIPTION
strip removes the symbol table and relocation bits ordinarily attached to the output of the assembler and linker. This is useful to save space after a program has been debugged.

The effect of strip is the same as use of the -s option of ld(1).

SEE ALSO
ld(1), a.out(5) coff(5)

BUGS
Unstripped 2.0 binary files will not run if stripped by the 3.0 version. A message of the form:

pid xxx: killed due to swap problems in I/O error mapping page.

when attempting to run a program indicates that this is the problem.
NAME
textedit – SunView window- and mouse-based text editor

SYNOPSIS
textedit [ generic-tool-arguments ] [ -Ea on | off ] [ -adjust_is_pending_delete ] [ -Ei on | off ]
[ -auto_indent ] [ -Eo on | off ] [ -okay_to_overwrite ] [ -Er on | off ] [ -read_only ]
[ -Ec N ] [ -checkpoint count ] [ -EL lines ] [ -lower_context lines ] [ -Em pixels ]
[ -margin pixels ] [ -En N ] [ -number_of_lines lines ] [ -Es N ] [ -scratch_window lines ]
[ -ES N ] [ -multi_click_space radius ] [ -Et N ] [ -tab_width tabstop ] [ -ET N ]
[ -multi_click_timeout intrvl ] [ -Eu N ] [ -history_limit max ] [ -EU N ]
[ -upper_context lines ] filename

AVAILABILITY
This command is available with the SunView I User's software installation option. Refer to Installing the Sun Operating System for information on how to install optional software.

DESCRIPTION
textedit is a mouse-oriented text editor that runs within the SunView environment. It creates a window containing two text subwindows. The top subwindow (referred to as the scratch window) can be used to store small pieces of text. The bottom subwindow (referred to as the edit window) displays the contents of filename, if given.

The name of the file currently being edited is displayed in the left-hand portion of the frame header. The name of the current working directory is displayed in the right-hand portion.

OPTIONS
generic-tool-arguments
textedit accepts the SunView generic tool arguments listed in sunview(1).

-Ea on | off
-adjust_is_pending_delete
Choose whether or not an adjustment to a selection makes the selection pending-delete. The default is off. This option corresponds to, and overrides the adjust_is_pending_delete Text defaults entry.

-Ei on | off
-auto_indent
Choose whether or not to automatically indent newly-opened lines: The default is off. Corresponds to the auto_indent Text default.

-Eo on | off
-okay_to_overwrite
Set behavior to the Store as New File menu item. If on a Store as New File to the current file is treated as a Save Current File. If off (the standard default), Store as New File operations using the current filename result in an error message. Corresponds to Store_self_is_save.

-Er on | off
-read_only
Turn read-only mode on or off. When on, text cannot be modified.

-Ec N
-checkpoint count
Checkpoint after every count editing operations. If count is 0 (the standard default), no checkpointing takes place. Each character typed, each Paste, and each Cut counts as an editing operation. Corresponds to checkpoint_frequency.

-EL lines
-lower_context lines
Specify the minimum number of lines to keep between the caret and the bottom of the text subwindow. The default is 2. Corresponds to lower_context.
-Em pixels
-mARGIN pixels
  Set the scrollbar margin width in pixels. The default is 4. Corresponds to left_margin.

-En N
-number_of_lines lines
  Set the number of lines in the bottom subwindow. The default is 45.

-Es N
-scratch_window lines
  Set the number of lines in the scratch window. A zero value means that there is no
  scratch window. The standard default is 1. Corresponds to scratch_window.

-ES N
-multi_click_space radius
  Set the radius, in pixels, within which clicks must occur to be treated as a multi-click
  selection. The default is 3 pixels. Corresponds to multi_click_space.

-ET N
-tab_width tabstop
  Set the number of SPACE characters displayed per TAB stop. The default is 8. This
  option has no effect on the characters in the file. Corresponds to tab_width.

-ET N
-multi_click_timeout intrvl
  Set the interval, in milliseconds, within which any two clicks must occur to be treated as
  a multi-click selection. The default is 390 milliseconds. Corresponds to multi_click_timeout.

-Eu N
-history_limit max
  Set the maximum number of editing operations that can be undone or replayed. The
  default is 50. Corresponds to history_limit.

-EU N
-upper_context lines
  Set the minimum number of lines to keep between the caret and the top of the text
  subwindow. The default is 2. Corresponds to upper_context.

USAGE

For a description of how to use the facilities of the text subwindows, see the SunView Beginner's Guide.

Signal Processing

If textedit hangs, for whatever reason, you can send a SIGHUP signal to its process ID, which forces it
to write any changes (if possible):

    kill -HUP pid

The edits are written to the file textedit.pid in its working directory. If that fails, textedit successively
tries to write to a file by that name in /var/tmp, and then /tmp. In addition, whenever textedit catches
a fatal signal, such as SIGILL, it tries to write out the edits before aborting.

Defaults Options

There are several dozen user-specified defaults that affect the behavior of the text-based facilities. See
defaultsedit(1) for a complete description. Important defaults entries in the Text category are:

Edit_back_char  Set the character for erasing to the left of the caret. The standard default is
                DELETE. Note: the tty erase character-setting has no effect on textedit. Text-
                based tools refer only to the defaults database key settings.

Edit_back_word  Set the character for erasing the word to the left of the caret. The standard
default is CTRL-W.

**Edit_back_line** Set the character for erasing all characters to the left of the caret. The standard default is CTRL-U.

**Checkpoint_frequency**
If set to 0 (the standard default) no checkpointing is done. For any value greater than zero, a checkpoint is made each time the indicated number of editing operations has been performed since the last checkpoint. Each character typed, each Paste, and each Cut counts as an editing operation. The checkpoint file has a name of the form: *filename%%*, where *filename* is the name of the file being edited.

**Making a selection**
In textedit, the mouse is used to specify a selection, which is a character span to operate on. The mouse is also used to position the insertion point and to invoke a menu of commands.

The assignment of commands to the mouse buttons is:

<table>
<thead>
<tr>
<th>Mouse button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT</td>
<td>Starts a new selection and moves the insertion point to the end of the selection nearest the mouse cursor.</td>
</tr>
<tr>
<td>MIDDLE</td>
<td>Extends a selection, and moves the insertion point.</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Displays a menu of operations, explained below.</td>
</tr>
</tbody>
</table>

There are two types of selections: a primary selection is indicated by video-inversion of the span of characters, and tends to persist. A secondary selection is indicated by underlining the span of characters and only exists while one of the four function keys corresponding to the commands Cut, Find, Paste, or Copy, is depressed.

In addition, a selection can be pending-delete, as indicated by overlaying the span of characters with a light gray pattern. A selection is made pending-delete by holding the CTRL key while clicking the LEFT or MIDDLE mouse buttons. If a primary selection is pending-delete, it is only deleted when characters are inserted, either by type-in or by Paste or Copy. If a secondary selection is pending-delete, it is deleted when the function key is released, except in the case of the Find, which deselects the secondary selection.

You can make adjusted selections switch to pending-delete using the `adjust_is_pending_delete` defaults entry, or the `-Ea` option. In this case, CTRL-Middle makes the selection not pending-delete.

Commands that operate on the primary selection do so even if the primary selection is not in the window that issued the command.

**Inserting Text and Command Characters**
For the most part, typing any of the standard keys either inserts the corresponding character at the insertion point, or erases characters. However, certain key combinations are treated as commands. Some of the most useful are:

<table>
<thead>
<tr>
<th>Command</th>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut-Primary</td>
<td>Meta-X</td>
<td>Erases, and moves to the Clipboard, the primary selection.</td>
</tr>
<tr>
<td>Find-Primary</td>
<td>Meta-F</td>
<td>Searches the text for the pattern specified by the primary selection or by the Clipboard, if there is no primary selection.</td>
</tr>
<tr>
<td>Copy-to-Clipboard</td>
<td>Meta-C</td>
<td>Copies the primary selection to the Clipboard.</td>
</tr>
<tr>
<td>Paste-Clipboard</td>
<td>Meta-V</td>
<td>Inserts the Clipboard contents at the insertion point.</td>
</tr>
<tr>
<td>Copy-then-Paste</td>
<td>Meta-P</td>
<td>Copies the primary selection to the insertion point (through the Clipboard).</td>
</tr>
<tr>
<td>Go-to-EOF</td>
<td>CTRL-RETURN</td>
<td>Moves the insertion point to the end of the text, positioning the text so that the insertion point is visible.</td>
</tr>
</tbody>
</table>
Function Keys

The commands indicated by use of the function keys are:

<table>
<thead>
<tr>
<th>Command</th>
<th>Sun-2/3 Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>L1</td>
<td>Aborts the current command.</td>
</tr>
<tr>
<td>Again</td>
<td>L2</td>
<td>Repeats the previous editing sequence since a primary selection was made.</td>
</tr>
<tr>
<td>Undo</td>
<td>L4</td>
<td>Undoes a prior editing sequence.</td>
</tr>
<tr>
<td>Front</td>
<td>L5</td>
<td>Makes the window completely visible (or hides it, if it is already exposed).</td>
</tr>
<tr>
<td>Copy</td>
<td>L6</td>
<td>Copies the primary selection, either to the Clipboard or at the closest end of the secondary selection.</td>
</tr>
<tr>
<td>Open</td>
<td>L7</td>
<td>Makes the window iconic (or normal, if it is already iconic).</td>
</tr>
<tr>
<td>Paste</td>
<td>L8</td>
<td>Copies either the secondary selection or the Clipboard at the insertion point.</td>
</tr>
<tr>
<td>Find</td>
<td>L9</td>
<td>Searches for the pattern specified by, in order, the secondary selection, the primary selection, or the Clipboard.</td>
</tr>
<tr>
<td>Cut</td>
<td>L10</td>
<td>Erases, and moves to the Clipboard, either the primary or the secondary selection.</td>
</tr>
<tr>
<td>CAPSLOCK</td>
<td>F1</td>
<td>Forces all subsequently typed alphabetic characters to be upper-case. This key is a toggle; striking it a second time undoes the effect of the first strike.</td>
</tr>
</tbody>
</table>

*Find* usually searches the text forwards, towards the end. Holding down the *SHIFT* key while invoking *Find* searches backward through the text, towards the beginning. If the pattern is not found before the search encounters either extreme, it wraps around and continues from the other extreme. *Find* starts the search at the appropriate end of the primary selection, if the primary selection is in the subwindow that the search is made in; otherwise it starts at the insertion point, unless the subwindow cannot be edited, in which case it starts at the beginning of the text.

*CTRL-Find* invokes the *Find and Replace* pop-up frame.

The default assignment of function keys can be modified using *defaultsedit(1)*.

Menu Items

- **File**: A pull-right menu item for file operations.
- **Edit**: A pull-right menu item equivalent of the editing function keys. The *Edit* submenu provides *Again, Undo, Copy, Paste, and Cut* (same as function keys L2, L4, L6, L8, and L10).
- **Display**: A pull-right menu item for controlling the way text is displayed and line display format.
- **Find**: A pull-right menu item for find and delimiter matching operations.
- **Extras**: A user definable pull-right menu item. The *Extras* standard submenu is controlled by */usr/lib/text_extras_menu*. This file has the same syntax as *.rootmenu* file. See *sunview(1)*.

Only those items that are active appear as normal text in the menu; inactive items (which are inappropriate at the time) are grayed out.

User Defined Commands

The file */usr/lib/text_extras_menu* specifies filter programs that are included in the text subwindow *Extras* pull-right menu item. The file "*.textswrc* specifies filter programs that are assigned to (available) function keys. These filters are applied to the contents of the primary selection. Their output is entered at the caret.
The file /usr/lib/textswrc is a sample containing a set of useful filters. It is not read automatically.

FILES

`~/.textswrc` specifies bindings of filters to function keys

`/usr/lib/text_extras_menu` specifies bindings of filters for the extras menu pull-right items

`/usr/bin` contains useful filters, including `shift_lines` and `capitalize`.

`filename%` prior version of `filename` is available here after a `Save Current File` menu operation

`textedit.pid` edited version of `filename`; generated in response to fatal internal errors

`/tmp/Text*` editing session logs

SEE ALSO

defaultsedit(1), kill(1), sunview(1),

SunView Beginner’s Guide

DIAGNOSTICS

Cannot open file `filename`, aborting!

`filename` does not exist or cannot be read.

textedit produces the following exit status codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>normal termination</td>
</tr>
<tr>
<td>1</td>
<td>standard SunView help message was printed</td>
</tr>
<tr>
<td>2</td>
<td>help message was requested and printed</td>
</tr>
<tr>
<td>3</td>
<td>abnormal termination in response to a signal, usually due to an internal error</td>
</tr>
<tr>
<td>4</td>
<td>abnormal termination during initialization, usually due to a missing file or running out of swap space</td>
</tr>
</tbody>
</table>

BUGS

Multi-click to change the current selection does not work for Adjust Selection.

Handling of long lines is incorrect in certain scrolling situations.

There is no way to replay any editing sequence except the most recent.

`textedit newfile` fails if `newfile` does not exist.
NAME

uucp, uulog, uname - system to system copy

SYNOPSIS

uucp [ -acCdmr ] [ -esystem ] [ -nusername ] [ -ggrade ] [ -sspool ] [ -xdebug ] source-file ...

destination-file

uulog [ -ssystem ] [ -uusername ]

uname [ -l ]

AVAILABILITY

This command is available with the uucp software installation option. Refer to Installing the Sun Operating System for information on how to install optional software.

DESCRIPTION

uucp copies each source-file to the named destination-file. A filename may be a path name on your machine, or may have the form

system-name!pathname

where system-name is taken from a list of system names that uucp knows about. Shell metacharacters ?, *, and [ ] appearing in the pathname part will be expanded on the appropriate system.

Pathnames may be one of:

- a full pathname;
- a pathname preceded by "username/" where username is a username on the specified system and is replaced by that user's login directory;
- a pathname preceded by "/"; such a pathname will be replaced by the public uucp directory on the remote machine;
- anything else is prefixed by the pathname of the current directory.

If the result is an erroneous pathname for the remote system, the copy will fail. If the destination-file is a directory, the last component of the source-file name is used.

uucp preserves execute permissions across the transmission and gives 0666 read and write permissions (see chmod(2)).

uulog maintains a summary log of uucp and uux(1C) transactions in the file /var/spool/uucp/LOGFILE, by gathering information from partial log files named /var/spool/uucp/LOG.*.?. It removes the partial log files.

uname lists the uucp names of systems that can be accessed using uucp.

OPTIONS

uucp Options

- -a Avoid doing a getwd(3) to find the current directory. This is sometimes used for efficiency.
- -c Use the source file when copying out rather than copying the file to the spool directory. This is the default.
- -C Make a copy of outgoing files in the uucp spool directory, rather than copying the source file directly to the target system. This lets you remove the source file after issuing the uucp command.
- -d Make all necessary directories for the file copy.
- -f Do not make intermediate directories for the file copy.
- -m Send mail to the requester when the copy is complete.
- -r Do not start the transfer, just queue the job.
Send the **uucp** command to the system *system* to be executed there. This works only when the remote machine allows **uucp** to be executed by `/usr/lib/uucp/uuxqt`.

**-m**

Notify *username* on remote system (by mail) that a file was sent.

**-n**

*username*

Notify *username* on remote system (by mail) that a file was sent.

**-g**

*grade*

*grade* is a single letter or number; lower ASCII values transmit a job earlier during a particular conversation. The default *grade* is `n`. By way of comparison, **uux(1C)** defaults to 'A'; mail is usually sent at grade 'C'.

**-s**

*spool*

Use *spool* as the spool directory instead of the default.

**-x**

Turn on the debugging at level *debug*.

**uulog Options**

**-s**

*system*

Print information about work involving system *system*.

**-u**

*username*

Print information about work done for the specified *username*.

**uuname options**

**-I**

Display the local system-name.

**FILES**

`/var/spool/uucp` spool directory  
`/usr/lib/uucp/sys` list of known systems and descriptions  
`/usr/lib/uucp/*` other data and program files  
`/var/spool/uucp/LOGFILE`

**SEE ALSO**

`mail(1)`, **uux(1C)**, `chmod(2)`, `getwd(3)`

**WARNING**

The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by pathname; ask a responsible person on the remote system to send them to you. For the same reasons you will probably not be able to send files to arbitrary pathnames.

**BUGS**

All files received by **uucp** will be owned by the user ID **uucp**.

The **-m** option will only work sending files or receiving a single file. Receiving multiple files specified by special shell characters `?`, `*`, and `[ ]` will not activate the **-m** option.
NAME
getmntent, setmntent, addmntent, endmntent, hasmntopt — get file system descriptor file entry

SYNOPSIS
#include <stdio.h>
#include <mntent.h>

FILE *setmntent(filep, type)
char *filep;
char *type;

struct mntent *getmntent(filep)
FILE *filep;

int addmntent(filep, mnt)
FILE *filep;
struct mntent *mnt;

char *hasmntopt(mnt, opt)
struct mntent *mnt;
char *opt;

int endmntent(filep)
FILE *filep;

DESCRIPTION
These routines replace the getfsent() routines for accessing the file system description file /etc/fstab. They are also used to access the mounted file system description file /etc/mtab.

setmntent() opens a file system description file and returns a file pointer which can then be used with getmntent, addmntent, or endmntent. The type argument is the same as in fopen(3). getmntent() reads the next line from filep and returns a pointer to an object with the following structure containing the broken-out fields of a line in the filesystem description file, <mntent.h>. The fields have meanings described in fstab(5).

struct mntent {
    char *mnt_fsname; /* file system name */
    char *mnt_dir; /* file system path prefix */
    char *mnt_type; /* 4.2, nfs, swap, or xx */
    char *mnt_opts; /* ro, quota, etc. */
    int mnt_freq; /* dump frequency, in days */
    int mnt_passno; /* pass number on parallel fsck */
} ;

addmntent() adds the mntent structure mnt to the end of the open file filep. Note: filep has to be opened for writing if this is to work. hasmntopt() scans the mnt_opts field of the mntent structure mnt for a substring that matches opt. It returns the address of the substring if a match is found, 0 otherwise. endmntent() closes the file.

FILES
/etc/fstab
/etc/mtab

SEE ALSO
fopen(3S), getfsent(3), fstab(5)

DIAGNOSTICS
NULL pointer (0) returned on EOF or error.

BUGS
The returned mntent structure points to static information that is overwritten in each call.
NAME

kb – Sun keyboard STREAMS module

CONFIG

pseudo-device knumber

SYNOPSIS

#include <sys/stream.h>
#include <sys/stropts.h>
#include <sundev/vuid_event.h>
#include <sundev/kbio.h>
#include <sundev/kbd.h>

ioctl(fd, I_PUSH, "kb");

DESCRIPTION

The kb STREAMS module processes byte streams generated by Sun keyboards attached to a CPU serial
or parallel port. Definitions for altering keyboard translation, and reading events from the keyboard, are
in <sundev/kbio.h> and <sundev/kbd.h>. number specifies the maximum number of keyboards sup­
ported by the system.

kb recognizes which keys have been typed using a set of tables for each known type of keyboard.
Each translation table is an array of 128 16-bit words (unsigned shorts). If an entry in the table is less
than 0x100, it is treated as an ISO 8859/1 character. Higher values indicate special characters that
invoke more complicated actions.

Keyboard Translation Mode

The keyboard can be in one of the following translation modes:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR_NONE</td>
<td>Keyboard translation is turned off and up/down key codes are reported.</td>
</tr>
<tr>
<td>TR_ASCII</td>
<td>ISO 8859/1 codes are reported.</td>
</tr>
<tr>
<td>TR_EVENT</td>
<td>firm_events (see The SunView System Programmer's Guide — Appendix: Writing a Virtual User Input Device Driver) are reported.</td>
</tr>
<tr>
<td>TR_UNTRANS_EVENT</td>
<td>firm_events containing unencoded keystation codes are reported for all input events within the window system.</td>
</tr>
</tbody>
</table>

Keyboard Translation-Table Entries

All instances of the kb module share seven translation tables used to convert raw keystation codes to
event values. The tables are:

- Unshifted: Used when a key is depressed and no shifts are in effect.
- Shifted: Used when a key is depressed and a Shift key is being held down.
- Caps Lock: Used when a key is depressed and Caps Lock is in effect.
- Alt Graph: Used when a key is depressed and the Alt Graph key is being held down.
- Num Lock: Used when a key is depressed and Num Lock is in effect.
- Controlled: Used when a key is depressed and the Control key is being held down (regardless of whether a Shift key or the Alt Graph is being held down, or whether Caps Lock or Num Lock is in effect).
- Key Up: Used when a key is released.

Each key on the keyboard has a key station code that is a number from 0 to 127. This number is used
as an index into the translation table that is currently in effect. If the corresponding entry in that trans­
lation table is a value from 0 to 255, this value is treated as an ISO 8859/1 character, and that character
is the result of the translation.

If the entry is a value above 255, it is a special entry. Special entry values are classified according to the value of the high-order bits. The high-order value for each class is defined as a constant, as shown in the list below. The value of the low-order bits, when added to this constant, distinguishes between keys within each class:

**SHIFTKEYS 0x100**

A shift key. The value of the particular shift key is added to determine which shift mask to apply:

- **CAPSLOCK 0** Caps Lock key.
- **SHIFTLOCK 1** Shift lock key.
- **LEFTSHIFT 2** Left-hand Shift key.
- **RIGHTSHIFT 3** Right-hand Shift key.
- **LEFTCTRL 4** Left-hand (or only) Control key.
- **RIGHTCTRL 5** Right-hand Control key.
- **ALTGRAPH 9** Alt Graph key.
- **ALT 10** Alternate key on the Type 3 keyboard, or Alt key on the Type 4 keyboard.
- **NUMLOCK 11** Num Lock key.

**BUCKYBITS 0x200**

Used to toggle mode-key-up/down status without altering the value of an accompanying ISO 8859/1 character. The actual bit-position value, minus 7, is added.

- **METABIT 0** The Meta key was pressed along with the key. This is the only user-accessible bucky bit. It is ORed in as the 0x80 bit; since this bit is a legitimate bit in a character, the only way to distinguish between, for example, 0xA0 as META+0x20 and 0xA0 as an 8-bit character is to watch for META key up and META key down events and keep track of whether the META key was down.
- **SYSTEMBIT 1** The System key was pressed. This is a place holder to indicate which key is the system-abort key.

**FUNNY 0x300**

Performs various functions depending on the value of the low 4 bits:

- **NOP 0x300** Does nothing.
- **OOPS 0x301** Exists, but is undefined.
- **HOLE 0x302** There is no key in this position on the keyboard, and the position-code should not be used.
- **NOSCROLL 0x303** Alternately sends "S and "Q.
- **CTRLS 0x304** Sends "S and toggles NOSCROLL key.
- **CTRLQ 0x305** Sends "Q and toggles NOSCROLL key.
- **RESET 0x306** Keyboard reset.
- **ERROR 0x307** The keyboard driver detected an internal error.
- **IDLE 0x308** The keyboard is idle (no keys down).
- **COMPOSE 0x309** This key is the COMPOSE key; the next two keys should comprise a two-character COMPOSE key sequence.
- **NONL 0x30A** Used only in the Num Lock table; indicates that this
This key is not affected by the Num Lock state, so that the translation table to use to translate this key should be the one that would have been used had Num Lock not been in effect.

0x30B — 0x30F  Reserved for nonparameterized functions.

**FA_CLASS 0x400**

This key is a floating accent or dead key. Pressing this key causes the next key to generate an event for an accented character; for example, floating accent grave followed by the a key generates an event with the ISO 8859/1 code for the a with grave accent character. The low-order bits indicate which accent; the codes for the individual floating accents are as follows:

- FA_UMLAUT 0x400  uumlaut
- FA_CFLEX 0x401  circumflex
- FA_TILDE 0x402  tilde
- FA_CEDILLA 0x403  cedilla
- FA_ACUTE 0x404  acute accent
- FA_GRAVE 0x405  grave accent

**STRING 0x500**

The low-order bits index a table of strings. When a key with a STRING entry is depressed, the characters in the null-terminated string for that key are sent, character by character. The maximum length is defined as:

```
KTAB_STRLEN 10
```

Individual string numbers are defined as:

- HOMEARROW 0x00
- UPARROW 0x01
- DOWNARROW 0x02
- LEFTARROW 0x03
- RIGHTARROW 0x04

String numbers 0x05 — 0x0F are available for custom entries.

**FUNCKEYS 0x600**

Function keys. The next-to-lowest 4 bits indicate the group of function keys:

- LEFTFUNC 0x600
- RIGHTFUNC 0x610
- TOPFUNC 0x620
- BOTTOMFUNC 0x630

The low 4 bits indicate the function key number within the group:

- LF(n)  (LEFTFUNC+(n)-1)
- RF(n)  (RIGHTFUNC+(n)-1)
- TF(n)  (TOPFUNC+(n)-1)
- BF(n)  (BOTTOMFUNC+(n)-1)

There are 64 keys reserved for function keys. The actual positions may not be on left/right/top/bottom of the keyboard, although they usually are.

**PADKEYS 0x700**

This key is a numeric keypad key. These entries should appear only in the Num Lock translation table: when Num Lock is in effect, these events will be generated by pressing keys on the right-hand keypad. The low-order bits indicate which key; the codes for the individual keys are as follows:

- PADEQUAL 0x700  = key
- PADSJASH 0x701  / key
PADSTAR 0x702 * key
PADMINUS 0x703 - key
PADSEP 0x704 , key
PAD7 0x705 7 key
PAD8 0x706 8 key
PAD9 0x707 9 key
PADPLUS 0x708 + key
PAD4 0x709 4 key
PAD5 0x70A 5 key
PAD6 0x70B 6 key
PAD1 0x70C 1 key
PAD2 0x70D 2 key
PAD3 0x70E 3 key
PAD0 0x70F 0 key
PADDOT 0x710 Enter key
PADENTER 0x711 Enter key

In TR_ASCII mode, when a function key is pressed, the following escape sequence is sent:
<ESC>[0...9z
where <ESC> is a single escape character and 0...9 indicates the decimal representation of the
function-key value. For example, function key R1 sends the sequence:
<ESC>[208z
because the decimal value of RF(1) is 208. In TR_EVENT mode, if there is a VUID event code for the
function key in question, an event with that event code is generated; otherwise, individual events for the
characters of the escape sequence are generated.

Keyboard Compatibility Mode
kb is in compatibility mode when it starts up. In this mode, when the keyboard is in the TR_EVENT
translation mode, ISO 8859/1 characters from the upper half of the character set (that is, characters with
the 8th bit set) are presented as events with codes in the ISO_FIRST range (as defined in
<sundev/vuid_event.h>); the event code is ISO_FIRST plus the character value. This is for backwards
compatibility with older versions of the keyboard driver. If compatibility mode is turned off, ISO
8859/1 characters are presented as events with codes equal to the character code.

IOCTLS
Two ioctl set and retrieve the current translation mode of a keyboard:
KIOCTRANS The argument is a pointer to an int. The translation mode is set to the value in the
int pointed to by the argument.
KIOCGETRANS The argument is a pointer to an int. The current translation mode is stored in the int
pointed to by the argument.

ioctl for changing and retrieving entries from the keyboard translation table use the kiockeymap structure:

struct kiockeymap {
    int kio_tablemask; /* Translation table (one of: 0, CAPSMASK,
                      SHIFTMASK, CTRLMASK, UPMASK,
                      ALTGRAPHMASK, NUMLOCKMASK) */
#define KIOCABORT1 -1 /* Special mask: abort1 keystation */
#define KIOCABORT2 -2 /* Special mask: abort2 keystation */

`u_char kio_station; /* Physical keyboard key station (0-127) */
  u_short kio_entry; /* Translation table station's entry */
  char kio_string[10]; /* Value for STRING entries (null terminated) */
};

KIOCSKEY

The argument is a pointer to a `kiockeymap` structure. The translation table entry referred to by the values in that structure is changed.

`kio_tablemask` specifies which of the five translation tables contains the entry to be modified:

- `UPMASK 0x0080` Key Up translation table.
- `NUMLOCKMASK 0x0800` Num Lock translation table.
- `CTRLMASK 0x0030` Controlled translation table.
- `ALTGRAPHMASK 0x0200` Alt Graph translation table.
- `SHIFTMASK 0x000E` Shifted translation table.
- `CAPSMASK 0x0001` Caps Lock translation table.

(No shift keys pressed or locked)

`Unshifted translation table.`

`kio_station` specifies the keystation code for the entry to be modified. The value of `kio_entry` is stored in the entry in question. If `kio_entry` is between STRING and STRING+15, the string contained in `kio_string` is copied to the appropriate string table entry. This call may return EINVAL if there are invalid arguments.

There are a couple special values of `kio_tablemask` that affect the two step break to the PROM monitor sequence. The usual sequence is SETUP-a or L1-a. If `kio_tablemask` is KIOCABORT1 then the value of `kio_station` is set to be the first keystation in the sequence. If `kio_tablemask` is KIOCABORT2 then the value of `kio_station` is set to be the second keystation in the sequence.

KIOCGBKEY

The argument is a pointer to a `kiockeymap` structure. The current value of the keyboard translation table entry specified by `kio_tablemask` and `kio_station` is stored in the structure pointed to by the argument. This call may return EINVAL if there are invalid arguments.

KIOCTYPE

The argument is a pointer to an `int`. A code indicating the type of the keyboard is stored in the `int` pointed to by the argument:

- `KB_KLUNK` Micro Switch 103SD32-2
- `KB_VT100` Keytronics VT100 compatible
- `KB_SUN2` Sun-2 keyboard
- `KB_SUN3` Type 3 keyboard
- `KB_SUN4` Type 4 keyboard
- `KB_ASCII` ASCII terminal masquerading as keyboard
- `-1` is stored in the `int` pointed to by the argument if the keyboard type is unknown.

KIOCLAYOUT

The argument is a pointer to an `int`. On a Type 4 keyboard, the layout code specified by the keyboard’s DIP switches is stored in the `int` pointed to by the argument.

KIOCCMD

The argument is a pointer to an `int`. The command specified by the value of the `int` pointed to by the argument is sent to the keyboard. The commands that can be sent are:

Commands to the Sun-2, Type 3, and Type 4 keyboard:

- `KBD_CMD_RESET` Reset keyboard as if power-up.
KBD_CMD_BELL  Turn on the bell.
KBD_CMD_NOBELL  Turn off the bell

Commands to the Type 3 and Type 4 keyboard:
KBD_CMD_CLICK  Turn on the click annunciator.
KBD_CMD_NOCLICK  Turn off the click annunciator.

Inappropriate commands for particular keyboard types are ignored. Since there is no reliable way to get the state of the bell or click (because we cannot query the keyboard, and also because a process could do writes to the appropriate serial driver — thus going around this ioctl) we do not provide an equivalent ioctl to query its state.

KIOCSLED  The argument is a pointer to a char. On the Type 4 keyboard, the LEDs are set to the value specified in that char. The values for the four LEDs are:
   LED_CAPS_LOCK  Caps Lock light.
   LED_COMPOSE  Compose light.
   LED_SCROLL_LOCK  Scroll Lock light.
   LED_NUM_LOCK  Num Lock light.

KIOCGLED  The argument is a pointer to a char. The current state of the LEDs is stored in the char pointed to by the argument.

KIOCSCOMPAT  The argument is a pointer to an int. Compatibility mode is turned on if the int has a value of 1, and is turned off if the int has a value of 0.

KIOCGCOMPAT  The argument is a pointer to an int. The current state of compatibility mode is stored in the int pointed to by the argument.

KIOCGDIRECT  These ioctls are supported for compatibility with the system keyboard device /dev/kbd. KIOCSDIRECT has no effect, and KIOCGDIRECT always returns 1.

SEE ALSO
   click(1), loadkeys(1), kbd(4S), termio(4), win(4S)

The SunView System Programmer's Guide— Appendix: Writing a Virtual User Input Device Driver (describes firm_event format)
NAME
  bar – tape archive file format

AVAILABILITY
  Sun386i systems only.

DESCRIPTION
  bar(1), (the tape archive command) dumps several files into one, in a medium suitable for transporta-
  tion. This format is not compatible with the format generated by tar(1).

  A “bar tape” or file is a series of blocks. Each block is of size TBLOCK. A file on the tape is
  represented by a header block that describes the file, followed by zero or more blocks that give the con-
  tents of the file. At the end of the tape are two blocks filled with binary zeros, as an end-of-file indica-
  tor.

  The blocks are grouped for physical I/O operations. Each group of n blocks (where n is set by the b
  keyletter on the bar(1) command line — default is 20 blocks) is written with a single system call; on
  nine-track tapes, the result of this write is a single tape record. The last group is always written at the
  full size, so blocks after the two zero blocks contain random data. On reading, the specified or default
  group size is used for the first read, but if that read returns less than a full tape block, the reduced block
  size is used for further reads, unless the B keyletter is used.

  The header block looks like:
  #define TBLOCK 512
  union hblock {
    char dummy[TBLOCK];
    struct header {
      char mode[8];
      char uid[8];
      char gid[8];
      char size[12];
      char mtime[12];
      char chksum[8];
      char rdev[8];
      char linkflag;
      char bar_magic[2];
      char volume_num[4];
      char compressed;
      char date[12];
      char start_of_name;
    } dbuf;
  }

  start_of_name is a null-terminated string. date is the date of the archive. bar_magic is a special
  number indicating that this is a bar archive. rdev is the device type, for files that are devices. The
  other fields are zero-filled octal numbers in ASCII. Each field (of width w) contains w-2 digits, a
  space, and a null, except size, rdev, and mtime, which do not contain the trailing null. start_of_name is
  the name of the file, as specified on the bar command line. Files dumped because they were in a direc-
  tory that was named in the command line have the directory name as prefix and /filename as suffix.
  mode is the file mode, with the top bit masked off. uid and gid are the user and group numbers that
  own the file. size is the size of the file in bytes. Links and symbolic links, and special files, are
  dumped with this field specified as zero. mtime is the modification time of the file at the time it was
dumped. chksum is a decimal ASCII value that represents the sum of all the bytes in the header block.
When calculating the checksum, the chksum field is treated as if it were all blanks. linkflag is ASCII 0
if the file is “normal” or a special file, 1 if it is an hard link, 2 if it is a symbolic link, and 3 if it is a
special file (device or FIFO). The name linked-to, if any, is in a null-terminated string, following
start_of_name. Unused fields of the header are binary zeros (and are included in the checksum).
The first time a given i-node number is dumped, it is dumped as a regular file. The second and subsequent times, it is dumped as a link instead. Upon retrieval, if a link entry is retrieved, but not the file it was linked to, an error message is printed and the tape must be manually re-scanned to retrieve the linked-to file.

An additional header block (one that does not pertain to a particular file) is written to the first block of each volume of the archive. The volume header ID is copied to start_of_name and is a NULL string, unless one is specified with the bar(1) H function modifier. The size in the volume header reflects the number of bytes to skip to the start of the first full file (always zero on the first volume).

The encoding of the header is designed to be portable across machines.

SEE ALSO
bar(1)
NAME
help — help file format

SYNOPSIS
/usr/lib/help/

AVAILABILITY
Sun386i systems only.

DESCRIPTION
Each SunView application using the help feature has a simple ASCII file with the name application-
name.info and stored in the user’s default help directory. The default help directory for a user typically
is /vol/help/language/USA-English; this has links to various places, including the help files (that Sun
Microsystem supplies) in /usr/lib/help/language/USA-English.

This file contains the text of help messages for each SunView object within that program. Each help
message is separated in the file by a line beginning with a colon and identified by a keyword that
matches the HELP_DATA attribute of the SunView object.

The first character of each line in the file may be:

# comment line
: keyword line
any other 1-32 help text lines

If the line is a keyword line, it has the following structure:

:keyword[s]:datastring [pagenumber]<cr>

keyword is a 1-65 character keyword
--any displayable characters may be used
--several keywords may be present
--keywords are separated by 1-or-more blanks

datastring is 1-256 ASCII bytes, and describes the path of the data files. If it is
a help_viewer file, and it is a relative file name, then help_viewer
looks for the file relative to the default help directory as defined
in the user’s defaults database.

pagenumber is an optional page number within the help_viewer data file.

The help text that follows the :keyword line will be displayed in an Alert Box when help is requested
for one of the keywords by pressing the help key.

The datastring will be sent (by RPC) to the help_viewer procedure when the user selects the More
Help box in the Alert Box window.

EXAMPLE
Here is part of a typical help file, called mailtool.info.

:o abort
Abort button

Quits the Mail application (click
left on button). Tentative message
deletions do not become permanent.
Provides a menu of Abort options (click right on button).

:cancel:mailtool/Writing_and_Sending_Mail 1
Cancel button

Closes the message composition window without sending message (click left on button).

Provides a menu of Cancel options (click right on button).

Pressing the help key while the mouse cursor is over the Cancel or Abort button triggers the display of the corresponding text. The words cancel and abort in this file are the keywords. In the case of abort, there is no More Help available. For cancel, More Help is available and it is stored in the first page of the Writing_and_Sending_Mail file in the mailtool directory.

FILES
/usr/lib/help/* files for the pop-up help facility

SEE ALSO
help_viewer(1), help_viewer(5)

Sun386i Developer's Guide
NAME
help_viewer – help viewer file format

SYNOPSIS
/usr/lib/help/*/*

AVAILABILITY
Sun386i systems only.

DESCRIPTION
The help_viewer reads and displays the following types of files:
1. Specially formatted ASCII text (for tables of contents). Example:
   /usr/lib/help/language/USA-English/Top_Level
2. FrameMaker document files. Example:
   /usr/lib/help/language/USA-English/sunview/Desktop_Basics
3. Interleaf files. Example:
   /usr/lib/help/language/USA-English/help_guide/Help_Writer’s_Handbook

Each directory within /usr/lib/help/language/USA-English that corresponds to a SunView application name contains detailed information about that application. These are also FrameMaker files. The *.rf files in these directories store some of the pictures that appear in the help handbooks (other pictures are integral to the FrameMaker files). These *.rf files are stored in standard Sun compressed raster file format.

The Frame and Interleaf subdirectories of /usr/lib/help/format contain topic, contents, and index templates that can be used to create new Help Viewer handbooks.

By default, Help Viewer reads help files through /vol/help. The /vol/help/language/USA-English directory contains symbolic links to the individual help files and directories for the various applications on the system. Help files for Sun-supplied applications reside in /usr/lib/help/language/USA-English.

Developers who wish to add their own document directories to the system should create links to them from /vol/help.master/language/USA-English. Installation scripts should have the name of new handbooks added to the Top_Level file in /vol/help.master/language/USA-English.

FILES
/usr/lib/help/*/*

SEE ALSO
help(5), help_viewer(1), Sun386i Developer’s Guide
NAME
keytables – keyboard table descriptions for loadkeys and dumpkeys

DESCRIPTION
These files are used by loadkeys(1) to modify the translation tables used by the keyboard streams module kb(4M), and generated by dumpkeys(1) from those translation tables.

Any line in the file beginning with # is a comment, and is ignored. # is treated specially only at the beginning of a line.

Other lines specify the values to load into the tables for a particular keystation. The format is either:

    key number list_of_entries

or

    swap number1 with number2

or

    key number1 same as number2

or a blank line, which is ignored.

    key number list_of_entries

sets the entries for keystation number from the list given. An entry in that list is of the form

    tablename code

where tablename is the name of a particular translation table, or all. The translation tables are:

- base  entry when no shifts are active
- shift entry when "Shift" key is down
- caps  entry when Caps Lock is in effect
- ctrl  entry when "Control" is down
- altg  entry when "Alt Graph" is down
- numl  entry when Num Lock is in effect
- up    entry when a key goes up

All tables other than up refer to the action generated when a key goes down. Entries in the up table are used only for shift keys, since the shift in question goes away when the key goes up, except for keys such as "Caps Lock" or "Num Lock"; the keyboard streams module makes the key look as if it were a latching key.

A table name of all indicates that the entry for all tables should be set to the specified value, with the following exception: for entries with a value other than hole, the entry for the numl table should be set to nonl, and the entry for the up table should be set to nop.

The code specifies the effect of the key in question when the specified shift key is down. A code consists of either:

1) A character, which indicates that the key should generate the given character. The character can either be a single character, a single character preceded by ` which refers to a "control character" (for instance, "c is control-C), or a C-style character constant enclosed in single quote characters ('), which can be expressed with C-style escape sequences such as \r for RETURN or \000 for the null character. Note that the single character may be any character in an 8-bit character set, such as ISO 8859/1.

2) A string, consisting of a list of characters enclosed in double quote characters ("'). Note that the use of the double quote character means that a code of double quote must be enclosed in single quotes.

3) One of the following expressions:
shiftkeys+leftshift
   the key is to be the left-hand "Shift" key

shiftkeys+rightshift
   the key is to be the right-hand "Shift" key

shiftkeys+leftctrl
   the key is to be the left-hand "Control" key

shiftkeys+rightctrl
   the key is to be the right-hand "Control" key

shiftkeys+alt
   the key is to be the "Alt" shift key

shiftkeys+altgraph
   the key is to be the "Alt Graph" shift key

shiftkeys+capslock
   the key is to be the "Caps Lock" key

shiftkeys+shiftlock
   the key is to be the "Shift Lock" key

shiftkeys+numlock
   the key is to be the "Num Lock" key

buckybits+systembit
   the key is to be the "Stop" key in Sunview; this is normally the L1 key, or the SETUP
   key on the VT100 keyboard

buckybits+metabit
   the key is to be the "meta" key, i.e. the "Left" or "Right" key on a Sun-2 or Type 3
   keyboard or the "diamond" key on a Type 4 keyboard

compose
   the key is to be the "Compose" key

ctrlq
   on the "VT100" keyboard, the key is to transmit the control-Q character (this would
   be the entry for the "Q" key in the ctrl table)

ctrs
   on the "VT100" keyboard, the key is to transmit the control-S character (this would
   be the entry for the "S" key in the ctrl table)

noscroll
   on the "VT100" keyboard, the key is to be the "No Scroll" key

string+uparrow
   the key is to be the "up arrow" key

string+downarrow
   the key is to be the "down arrow" key

string+leftarrow
   the key is to be the "left arrow" key

string+rightarrow
   the key is to be the "right arrow" key

string+homearrow
   the key is to be the "home" key

fa_acute
   the key is to be the acute accent "floating accent" key

fa_cedilla
   the key is to be the cedilla "floating accent" key
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fa_cflex</td>
<td>the key is to be the circumflex &quot;floating accent&quot; key</td>
</tr>
<tr>
<td>fa_grave</td>
<td>the key is to be the grave accent &quot;floating accent&quot; key</td>
</tr>
<tr>
<td>fa_tilde</td>
<td>the key is to be the tilde &quot;floating accent&quot; key</td>
</tr>
<tr>
<td>fa_umlaut</td>
<td>the key is to be the umlaut &quot;floating accent&quot; key</td>
</tr>
<tr>
<td>nonl</td>
<td>this is used only in the Num Lock table; the key is not to be affected by the state of Num Lock</td>
</tr>
<tr>
<td>pad0</td>
<td>the key is to be the &quot;0&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad1</td>
<td>the key is to be the &quot;1&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad2</td>
<td>the key is to be the &quot;2&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad3</td>
<td>the key is to be the &quot;3&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad4</td>
<td>the key is to be the &quot;4&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad5</td>
<td>the key is to be the &quot;5&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad6</td>
<td>the key is to be the &quot;6&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad7</td>
<td>the key is to be the &quot;7&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad8</td>
<td>the key is to be the &quot;8&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad9</td>
<td>the key is to be the &quot;9&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>paddot</td>
<td>the key is to be the &quot;.&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>padenter</td>
<td>the key is to be the &quot;Enter&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>padplus</td>
<td>the key is to be the &quot;+&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>padminus</td>
<td>the key is to be the &quot;-&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>padstar</td>
<td>the key is to be the &quot;*&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>padslash</td>
<td>the key is to be the &quot;/&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>padequal</td>
<td>the key is to be the &quot;=&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>padsep</td>
<td>the key is to be the &quot;,&quot; (separator) key on the numeric keypad</td>
</tr>
<tr>
<td>lf(n)</td>
<td>the key is to be the left-hand function key n</td>
</tr>
<tr>
<td>rf(n)</td>
<td>the key is to be the right-hand function key n</td>
</tr>
<tr>
<td>tf(n)</td>
<td>the key is to be the top function key n</td>
</tr>
<tr>
<td>bf(n)</td>
<td>the key is to be the &quot;bottom&quot; function key n</td>
</tr>
<tr>
<td>nop</td>
<td>the key is to do nothing</td>
</tr>
<tr>
<td>error</td>
<td>this code indicates an internal error; to be used only for keystation 126, and must be used there</td>
</tr>
<tr>
<td>idle</td>
<td>this code indicates that the keyboard is idle (that is, has no keys down); to be used only for all entries other than the numl and up table entries for keystation 127, and must be used there</td>
</tr>
</tbody>
</table>
oops  this key exists, but its action is not defined; it has the same effect as nop
reset  this code indicates that the keyboard has just been reset; to be used only for the up
table entry for keystation 127, and must be used there

swap number1 with number2
exchanges the entries for keystations number1 and number2.

key number1 same as number2
sets the entries for keystation number1 to be the same as those for keystation number2. If the file does not specify entries for keystation number2, the entries currently in the translation table are used; if the file does specify entries for keystation number2, those entries are used.

EXAMPLES
The following entry sets keystation 15 to be a "hole" (that is, an entry indicating that there is no keysta­tion 15); sets keystation 30 to do nothing when Alt Graph is down, generate "!" when Shift is down, and generate "1" under all other circumstances; and sets keystation 76 to be the left-hand Control key.

key 15 all hole
key 30 base 1 shift ! caps 1 ctrl 1 altg nop
key 76 all shiftkeys+leftctrl up shiftkeys+leftctrl

The following entry exchanges the Delete and Back Space keys on the Type 4 keyboard:

swap 43 with 66
Keystation 43 is normally the Back Space key, and keystation 66 is normally the Delete key.

The following entry disables the Caps Lock key on the Type 3 and U.S. Type 4 keyboards:

key 119 all nop

The following specifies the standard translation tables for the U.S. Type 4 keyboard:

key 0 all hole
key 1 all buckybits+systembit up buckybits+systembit
key 2 all hole
key 3 all lf(2)
key 4 all hole
key 5 all tf(1)
key 6 all tf(2)
key 7 all tf(10)
key 8 all tf(3)
key 9 all tf(11)
key 10 all tf(4)
key 11 all tf(12)
key 12 all tf(5)
key 13 all shiftkeys+altgraph up shiftkeys+altgraph
key 14 all tf(6)
key 15 all hole
key 16 all tf(7)
key 17 all tf(8)
key 18 all tf(9)
key 19 all shiftkeys+alt up shiftkeys+alt
key 20 all hole
key 21 all rf(1)
key 22 all rf(2)
key 23 all rf(3)
key 24 all hole
key 25 all lf(3)
key 26  all lf(4)  
key 27  all hole  
key 28  all hole  
key 29  all "["  
key 30  base 1 shift ! caps 1 ctrl 1 altg nop  
key 31  base 2 shift @ caps 2 ctrl "@ altg nop  
key 32  base 3 shift # caps 3 ctrl 3 altg nop  
key 33  base 4 shift $ caps 4 ctrl 4 altg nop  
key 34  base 5 shift % caps 5 ctrl 5 altg nop  
key 35  base 6 shift ^ caps 6 ctrl ^^ altg nop  
key 36  base 7 shift & caps 7 ctrl 7 altg nop  
key 37  base 8 shift * caps 8 ctrl 8 altg nop  
key 38  base 9 shift ( caps 9 ctrl 9 altg nop  
key 39  base 0 shift ) caps 0 ctrl 0 altg nop  
key 40  base - shift _ caps - ctrl "_ altg nop  
key 41  base = shift + caps = ctrl = altg nop  
key 42  base ' shift ~ caps ~ ctrl ~ altg nop  
key 43  all '  
key 44  all hole  
key 45  all rf(4) numl padequal  
key 46  all rf(5) numl padslash  
key 47  all rf(6) numl padstar  
key 48  all bf(13)  
key 49  all lf(5)  
key 50  all bf(10) numl padequal  
key 51  all lf(6)  
key 52  all hole  
key 53  all ""  
key 54  base q shift Q caps Q ctrl "Q altg nop  
key 55  base w shift W caps W ctrl "W altg nop  
key 56  base e shift E caps E ctrl "E altg nop  
key 57  base r shift R caps R ctrl "R altg nop  
key 58  base t shift T caps T ctrl "T altg nop  
key 59  base y shift Y caps Y ctrl "Y altg nop  
key 60  base u shift U caps U ctrl "U altg nop  
key 61  base i shift I caps I ctrl "I altg nop  
key 62  base o shift O caps O ctrl "O altg nop  
key 63  base p shift P caps P ctrl "P altg nop  
key 64  base [ shift { caps { ctrl "[ altg nop  
key 65  base ] shift } caps } ctrl "} altg nop  
key 66  all "177"  
key 67  all compose  
key 68  all rf(7) numl pad7  
key 69  all string+uparrow numl pad8  
key 70  all rf(9) numl pad9  
key 71  all bf(15) numl padminus  
key 72  all lf(7)  
key 73  all lf(8)  
key 74  all hole  
key 75  all hole  
key 76  all shiftkeys+leftctrl up shiftkeys+leftctrl  
key 77  base a shift A caps A ctrl "A altg nop  
key 78  base s shift S caps S ctrl "S altg nop
### FILE FORMATS

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key 79</td>
<td>base d shift D caps D ctrl 'D altg nop</td>
</tr>
<tr>
<td>key 80</td>
<td>base f shift F caps F ctrl 'F altg nop</td>
</tr>
<tr>
<td>key 81</td>
<td>base g shift G caps G ctrl 'G altg nop</td>
</tr>
<tr>
<td>key 82</td>
<td>base h shift H caps H ctrl '</td>
</tr>
<tr>
<td>key 83</td>
<td>base j shift J caps J ctrl '0 altg nop</td>
</tr>
<tr>
<td>key 84</td>
<td>base k shift K caps K ctrl 'altg nop</td>
</tr>
<tr>
<td>key 85</td>
<td>base l shift L caps L ctrl 'L altg nop</td>
</tr>
<tr>
<td>key 86</td>
<td>base ; shift ; caps ; ctrl ; altg nop</td>
</tr>
<tr>
<td>key 87</td>
<td>base ' ' shift '' caps '' ctrl '' altg nop</td>
</tr>
<tr>
<td>key 88</td>
<td>base \ shift \ caps \ ctrl ^ altg nop</td>
</tr>
<tr>
<td>key 89</td>
<td>all '</td>
</tr>
<tr>
<td>key 90</td>
<td>all bf(11) numl padslash</td>
</tr>
<tr>
<td>key 91</td>
<td>all string+leftarrow numl pad5</td>
</tr>
<tr>
<td>key 92</td>
<td>all rf(11) numl pad5</td>
</tr>
<tr>
<td>key 93</td>
<td>all string+rightarrow numl pad6</td>
</tr>
<tr>
<td>key 94</td>
<td>all bf(8) numl pad0</td>
</tr>
<tr>
<td>key 95</td>
<td>all lf(9)</td>
</tr>
<tr>
<td>key 96</td>
<td>all hole</td>
</tr>
<tr>
<td>key 97</td>
<td>all lf(10)</td>
</tr>
<tr>
<td>key 98</td>
<td>all shiftkeys+numlock</td>
</tr>
<tr>
<td>key 99</td>
<td>all shiftkeys+leftshift up shiftkeys+leftshift</td>
</tr>
<tr>
<td>key 100</td>
<td>base z shift Z caps Z ctrl 'Z altg nop</td>
</tr>
<tr>
<td>key 101</td>
<td>base x shift X caps X ctrl 'X altg nop</td>
</tr>
<tr>
<td>key 102</td>
<td>base c shift C caps C ctrl 'C altg nop</td>
</tr>
<tr>
<td>key 103</td>
<td>base v shift V caps V ctrl 'V altg nop</td>
</tr>
<tr>
<td>key 104</td>
<td>base b shift B caps B ctrl 'B altg nop</td>
</tr>
<tr>
<td>key 105</td>
<td>base n shift N caps N ctrl 'N altg nop</td>
</tr>
<tr>
<td>key 106</td>
<td>base m shift M caps M ctrl '</td>
</tr>
<tr>
<td>key 107</td>
<td>base , shift &lt; caps , ctrl , altg nop</td>
</tr>
<tr>
<td>key 108</td>
<td>base . shift &gt; caps . ctrl . altg nop</td>
</tr>
<tr>
<td>key 109</td>
<td>base / shift ? caps / ctrl ^_ altg nop</td>
</tr>
<tr>
<td>key 110</td>
<td>all shiftkeys+rightshift up shiftkeys+rightshift</td>
</tr>
<tr>
<td>key 111</td>
<td>all '0</td>
</tr>
<tr>
<td>key 112</td>
<td>all rf(13) numl pad1</td>
</tr>
<tr>
<td>key 113</td>
<td>all string+downarrow numl pad2</td>
</tr>
<tr>
<td>key 114</td>
<td>all rf(15) numl pad3</td>
</tr>
<tr>
<td>key 115</td>
<td>all hole</td>
</tr>
<tr>
<td>key 116</td>
<td>all hole</td>
</tr>
<tr>
<td>key 117</td>
<td>all hole</td>
</tr>
<tr>
<td>key 118</td>
<td>all lf(16)</td>
</tr>
<tr>
<td>key 119</td>
<td>all shiftkeys+capslock</td>
</tr>
<tr>
<td>key 120</td>
<td>all buckybits+metabit up buckybits+metabit</td>
</tr>
<tr>
<td>key 121</td>
<td>base ' ' shift ' ' caps ' ' ctrl '@ altg ' '</td>
</tr>
<tr>
<td>key 122</td>
<td>all buckybits+metabit up buckybits+metabit</td>
</tr>
<tr>
<td>key 123</td>
<td>all hole</td>
</tr>
<tr>
<td>key 124</td>
<td>all hole</td>
</tr>
<tr>
<td>key 125</td>
<td>all bf(14) numl padplus</td>
</tr>
<tr>
<td>key 126</td>
<td>all error numl error up hole</td>
</tr>
<tr>
<td>key 127</td>
<td>all idle numl idle up reset</td>
</tr>
</tbody>
</table>

---

**SEE ALSO**

- loadkeys(1), kb(4M)
NAME
.orgrc – organizer configuration and initialization file

AVAILABILITY
Sun386i systems only.

DESCRIPTION
The organizer(1), a SunView application for viewing and manipulating files and directories, saves its
parameters in the .orgrc file between runs. The user can use this file to configure the organizer.
The first parameter in the file should always be the version number.

Version = 1.1

Do not change the version number gratuitously; if the organizer determines that this version is "old",
then it will save this version in ~/.orgrc.old and try to copy /usr/lib/Orgrc into ~/.orgrc.
The next two parameters assign default names for the system DOS Program and the default Text Editor.

DOS Program = dos
Text Editor = textedit

The DOS Program parameter should not be changed. However, the user can change the default text editor.
For example:

Text Editor = shelltool vi

is an option to textedit(1) The Properties section initializes or customizes certain properties. The possible
values for each item are listed below. The braces and vertical bars below indicate choices, they are
not used in the .orgrc file.

# Properties
PROPERTY Display Style = {Name and Icon | Name Only | Name and Info}
PROPERTY Roadmap = {Yes | No}
PROPERTY Show Hidden Files = {Yes | No}
PROPERTY Sort Type = {Name | File Type | Size | Date}
PROPERTY Sort Direction = {Ascending | Descending}
PROPERTY Update Interval = [5-300]

The Color Palette specifies all the color values used by the organizer's buttons and icons. These
values must be RGB triplets. It is listed below.

Begin Color Palette
Background Color = 255, 255, 255
Directory Name Color = 0, 146, 236
Directory Icon Foreground Color = 114, 45, 0
Directory Icon Background Color = 255, 227, 185
Directory Highlight Name Color = 255, 255, 255
Text Name Color = 0, 166, 143
Text Icon Foreground Color = 0, 0, 0
Text Icon Background Color = 255, 255, 255
Text Highlight Name Color = 255, 255, 255
Executable Name Color = 255, 0, 0
Executable Icon Foreground Color = 157, 162, 187
Executable Icon Background Color = 255, 255, 255
Executable Highlight Name Color = 255, 255, 255
Device Name Color = 113, 117, 135
Device Icon Foreground Color = 0, 0, 0
Device Icon Background Color = 174, 255, 159
Device Highlight Name Color = 255, 255, 255
Button Group1 Color = 255, 220, 187
Button Group2 Color = 201, 211, 232

SunOS Release 4.0.1    Last change: 4 October 1988
Button Group Color = 255, 244, 113
Button Foreground Color = 0, 0, 0
Button Background Color = 255, 255, 255
Button Shadow Color = 180, 180, 184
Button Highlight Color = 0, 0, 0
Scrollbar Color = 142, 106, 146
End Color Palette

The Color Labels section allows the labelling or “aliasing” of RGB triplets. The right side of a label assignment can contain an RGB triplet, a palette entry, or another label that has already been assigned. Here’s an example:

Begin Color Labels
  Black = Text Icon Foreground Color
  White = Background Color
  Orange = 255, 213, 127
  Dark Red = 232, 0, 0
  Red = Dark Red
  Dark Blue = 0, 75, 161
  Light Gray = 223, 223, 223
End Color Labels

The rest of the .orgrc file contains user defined filetypes. The user can specify that certain files be grouped together and treated in a similar fashion. That is, the same icon is used to display all files in a filetype, and the same command is used when a file is opened or edited. In the default .orgrc (/usr/lib/Orgrc) there are ten user defined file types. Here is an example of a user defined file type:

Begin File Type Definition
  Name = *.sh
  Background Icon = ~/images/scriptBackGround.icon
  Foreground Icon = ~/images/scriptForeGround.icon
  Name Color = Black
  Icon Background Color = Orange
  Icon Foreground Color = Black
  Highlight Name Color = White
  Execute Application = cmdtool "$(FILE)"
  Edit Application = cmdtool vi "$(FILE)"
  Print Application = pr -f "$(FILE)" | lpr
End File Type Definition

The right side of the Name field can contain any combination of csh(l) Filename Substitution characters. This field specifies the file type by way of its name. The next six fields together specify an organizer icon. This model allows a rich variety of icons. For more information, see the Sun386i Advanced Skills manual. The right side of the Execute Application specifies the command to execute when the user either opens or double clicks on a file of that type. Edit Application and Print Application specify the command to execute when the user requests that a file of that type be edited or printed.

FILES
  ~/orgrc Read at beginning of execution by the Organizer
  /usr/lib/Orgrc Default .orgrc

SEE ALSO
  Sun386iUser’sGuide, Sun386iAdvancedSkills, organizer(l)

LIMITATIONS
  The right side of Color Palette entries must be RGB triplets.
  Forward references for Color Labels are not allowed.
BUGS

The organizer saves its parameters as it exits; unfortunately, it doesn't know how to save user's comments in the file. So, comments get blown away.
NAME
vfont - font formats

SYNOPSIS
#include <vfont.h>

DESCRIPTION
The fonts used by the window system and printer/plotters have the following format. Each font is in a
file, which contains a header, an array of character description structures, and an array of bytes containing
the bit maps for the characters. The header has the following format:

struct header {
    short magic;       /* Magic number VFONT_MAGIC */
    unsigned short size; /* Total # bytes of bitmaps */
    short maxx;        /* Maximum horizontal glyph size */
    short maxy;        /* Maximum vertical glyph size */
    short xtend;       /* (unused) */
};
#define VFONT_MAGIC 0436

maxx and maxy are intended to be the maximum horizontal and vertical size of any glyph in the font, in
raster lines. (A glyph is just a printed representation of a character, in a particular size and font.) The
size is the total size of the bit maps for the characters in bytes. The xtend field is not currently used.

After the header is an array of NUM_DISPATCH structures, one for each of the possible characters in
the font. Each element of the array has the form:

struct dispatch {
    unsigned short addr; /* &(glyph) - &(start of bitmaps) */
    short nbytes;       /* # bytes of glyphs (0 if no glyph) */
    char up, down, left, right; /* Widths from baseline point */
    short width;        /* Logical width, used by troff */
};
#define NUM_DISPATCH 256

The nbytes field is nonzero for characters which actually exist. For such characters, the addr field is an
offset into the bit maps to where the character's bit map begins. The up, down, left, and right fields
are offsets from the base point of the glyph to the edges of the rectangle which the bit map represents.
(The imaginary "base point" is a point which is vertically on the "base line" of the glyph (the bottom
line of a glyph which does not have a descender) and horizontally near the left edge of the glyph; often
3 or so pixels past the left edge.) The bit map contains up+down rows of data for the character, each of
which has left+right columns (bits). Each row is rounded up to a number of bytes. The width field
represents the logical width of the glyph in bits, and shows the horizontal displacement to the base
point of the next glyph.

FILES
/usr/lib/vfont/*
/usr/lib/fonts/fixedwidthfonts/*

SEE ALSO
troff(1), vfontinfo(1), vswap(1) fontflip_to_68k(8) fontflip_to_i386(8)

BUGS
A machine-independent font format should be defined. The shorts in the above structures contain
different bit patterns depending whether the font file is for use on a VAX or a Sun. The vswap program
must be used to convert one to the other.
NAME

fontflip_to_i386 - change a vfont file
fontflip_to_68k - change a Sun386i vfont

SYNOPSIS

fontflip_to_i386 fontname [-o newfontname ]
fontflip_to_68k fontname [-o newfontname ]

AVAILABILITY

Sun386i systems only.

DESCRIPTION

fontflip_to_i386 takes as input a vfont file (Sun-3 fixedwidthfont) and creates a Sun386i vfont. This new font is a bitflipped version of its input. The new font is named oldfont.i386 unless otherwise specified.

fontflip_to_68k takes as input a Sun386i vfont file (fixedwidthfont) and creates a Sun-3 vfont file (also used by Sun-2 and Sun-4 systems). If the input font has the name font.i386, the new font will have the .i386 extension stripped off, resulting in the name font. If the input file does not have the .i386 extension, then the -o switch must be used to specify the output file.

By default, the system appends the suffix .i386 to a font name before it opens the font and attempts to use it. If the system doesn’t find the font, it then opens the font name specified, which by convention is a Sun-3 font. You can use either format of font, but system performance improves with use of the Sun386i format fonts. These two utilities allow you to convert between the two formats. Sun ships both formats in /usr/lib/fonts/fixedwidthfonts. Typically only developers will need to employ these utilities.

OPTIONS

-o filename Specify the name of the new flipped font.

FILES

/usr/lib/fonts/fixedwidthfonts

SEE ALSO

vfont(5) fontedit(1)
NAME
ipallocd – Ethernet-to-IP address allocator

SYNOPSIS
/usr/etc/rpc.ipallocd

AVAILABILITY
Sun386i systems only.

DESCRIPTION
ipallocd is a daemon that determines or temporarily allocates IP addresses within a network segment. The service is only available on the system which is home to the address authority for the network segment, currently the YP master of the hosts.byaddr map although the service is not tied to Yellow Pages. It has complete knowledge of the hosts listed in the yellow pages, and, if the system is running the name server, of any hosts listed in internet domain tables automatically accessed on that host through the standard library gethostbyaddr() call.

This protocol uses DES authentication (the Sun Secure RPC protocol) to restrict access to this function. The only clients privileged to allocate addresses are those whose net IDs are in the networks group. For machine IDs, the machine must be a YP server.

The daemon uses permanent entries in the /etc/ethers and /etc/hosts files when they exist and are usable. In other cases, such as when a system is new to the network, ipallocd will enter a temporary mapping in a local cache. Entries in the cache are removed when there have been no references to a given entry in the last hour. This cache survives system crashes so that IP addresses will remain consistent.

The daemon also provides corresponding IP address to name mapping.

If the file /etc/ipalloc.netrange exists, ipallocd refuses to allocate addresses on networks not listed in the netrange file, or for which no free address is available.

FILES
/etc/ipalloc.cache
/etc/ipalloc.netrange

SEE ALSO
pnp(3R), ipalloc(3R), ipalloc.netrange(5), ipallocd(8C), pnpboot(8C), netconfig(8C), rarpd(8C)
NAME

kadb - adb-like kernel and standalone-program debugger

SYNOPSIS

> b kadb [ -d ] [ boot-flags ]

DESCRIPTION

kadb is an interactive debugger that is similar in operation to adb(1), and runs as a standalone program under the PROM monitor. You can use kadb to debug the kernel, or to debug any standalone program.

Unlike adb, kadb runs in the same supervisor virtual address space as the program being debugged — although it maintains a separate context. The debugger runs as a coprocess that cannot be killed (no ‘:k’) or rerun (no ‘:r’). There is no signal control (no ‘:i’, ‘:t’, or ‘:$i’), although the keyboard facilities (CTRL-C, CTRL-S, and CTRL-Q) are simulated.

While the kernel is running under kadb, the abort sequence (LI-A or BREAK) drops the system into kadb for debugging — as will a system panic. When running other standalone programs under kadb, the abort sequence will pass control to the PROM monitor. kadb is then invoked from the monitor by jumping to the starting address for kadb found in /usr/include/debug/debug.h (currently this can be done for both Sun-2 and Sun-3 system machines with the monitor command ‘g fd00000’, and with the monitor command ‘g fe005000’ for Sun386i systems). kadb’s user interface is similar to adb. Note: kadb prompts with

kadb>

Most adb commands function in kadb as expected. Typing an abort sequence in response to the prompt returns you to the PROM monitor, from which you can examine control spaces that are not accessible within adb or kadb. The PROM monitor command c will return control to kadb. As with $p works when debugging kernels (by actually mapping in new user pages). The verbs ? and / are equivalent in kadb, since there is only one address space in use.

Sun386i System Operations

kadb on the Sun386i system can also be used to debug loadable modules. To do so, use the -sym option to the modload command. If you need to debug the module entry point routine, be sure to set a breakpoint on the kernel routine vd_entry. This is the routine that calls the module entry point routine. When kadb hits the breakpoint, the symbols for the module are usable and a breakpoint can be set in the module itself.

Care must be taken to remove kadb breakpoints before unloading modules. Since kadb inserts bpt instructions in the module itself, unloading and loading new modules while breakpoints are set can cause kadb to insert bpt instructions at incorrect places. This may cause the system to crash.

The symbol "vddebug" can be set to -1 to enable all kernel printf’s in the pseudo-driver (/dev/vd). These printf’s may be helpful when trying to load or unload modules.

OPTIONS

KADB is booted from the PROM monitor as a standalone program. If you omit the -d flag, kadb automatically loads and runs vmunix from the filesystem kadb was loaded from. The kadb vmunix variable can be patched to change the default program to be loaded.

-d Interactive startup. Prompts with

kadb:

for a file to be loaded. From here, you can enter a boot sequence line to load a standalone program. Boot flags entered in response to this prompt are included with those already set and passed to the program. If you type a RETURN only, kadb loads vmunix from the filesystem that kadb was loaded from.

boot-flags

You can specify boot flags as arguments when invoking kadb. Note: kadb always sets the -d (debug) boot flag, and passes it to the program being debugged.

SunOS Release 4.0.1 Last change: 5 October 1988 1653
USAGE
Refer to adb in Debugging Tools for the Sun Workstation.

Kernel Macros
As with adb, kernel macros are supported. With kadb, however, the macros are compiled into the debugger itself, rather than being read in from the filesystem. The kadb command $M lists macros known to kadb.

Setting Breakpoints
Self-relocating programs such as the SunOS kernel need to be relocated before breakpoints can be used. To set the first breakpoint for such a program, start it with ‘:s’; kadb is then entered after the program is relocated (when the system initializes its interrupt vectors). Thereafter, ‘:s’ single-steps as with adb. Otherwise, use ‘:c’ to start up the program.

Sun386i System Commands
The Sun386i system version of kadb has the following additional commands. Note, for the general syntax of adb commands, see adb(1).

:i
Read a byte (with the INB instruction) in from the port at address.
:o
Send a byte (with the OUTB instruction) containing count out through the port at address.
:p
Like :b in adb(1), but sets a breakpoint using the hardware debug register instead of the breakpoint instruction. The advantage of using :p is that when setting breakpoints with the debug register it is not necessary to have write access to the breakpoint location. Four (4) breakpoints can be set with the hardware debug registers.

$S
Switch I/O from the console to the serial port or vice versa.
Like :e in adb(1), but requires only one keystroke and no RETURN character.
Like :s in adb(1), but requires only one keystroke and no RETURN character.

Automatic Rebooting with kadb
You can set up your workstation to automatically reboot kadb by patching the vmunix variable in /boot with the string kadb. (Refer to adb in Debugging Tools for the Sun Workstation for details on how to patch executables.)

FILES
/vmunix
/boot
/kadb
/usr/include/debug/debug.h

SEE ALSO
adb(1), boot(8S), modload(8), modunload(8)
Debugging Tools for the Sun Workstation
Writing Device Drivers

BUGS
There is no floating-point support, except on Sun386i systems.
kadb cannot reliably single-step over instructions that change the status register.
When sharing the keyboard with the operating system the monitor’s input routines can leave the keyboard in a confused state. If this should happen, disconnect the keyboard momentarily and then reconnect it. This forces the keyboard to reset as well as initiating an abort sequence.
Most of the bugs listed in adb(1) also apply to kadb.
NAME
modload - load a Sun386i module

SYNOPSIS
modload filename [ -conf config_file ] [ -entry entry_point ] [ -exec exec_file ] [ -o output_file ]
[ -nolink ] [ -A vmunix_file ]

AVAILABILITY
Sun386i systems only.

DESCRIPTION
modload loads a loadable module into a running system. The input file filename is an object file (.o file).

OPTIONS
-conf config_file
Use this configuration file to configure the loadable driver being loaded. The commands in this file are the same as those that the config(8) program recognizes. There are two additional commands, blockmajor and charmajor, shown in the configuration file example below.

-entry entry_point
This is the module entry point. This is passed by modload to ld(l) when the module is linked. The default module entry point name is ‘xxxinit’.

-exec exec_file
This is the name of a shell script or executable image file that will be executed if the module is successfully loaded. It is always passed the module id and module type as the first two arguments. For loadable drivers, the third and fourth arguments are the block major and character major numbers respectively. For a loadable system call, the third argument is the system call number.

-o output_file
This is the name of the output file that is produced by the linker. If this option is omitted, then the output file name is filename without the ‘.o’.

-nolink This option can be used if modload has already been issued once and the output file already exists. One must take care that neither the kernel nor the module have changed.

-sym This option indicates that modload should invoke the linker without the -s option. The linker will then produce a symbol table for the module. These symbols are useful for developers using a debugger to debug the module.

-A vmunix_file
This is the file that is passed to the linker to resolve module references to kernel symbols. The default is /vmunix. The symbol file must be for the currently running kernel or the module is likely to crash the system.

EXAMPLE
controller     fdco at atmcmd csr 0x001000 irq 6 priority 3
controller     fdc2 at atmcmd csr 0x002000 irq 5 priority 2
disk           fd0 at fdc0 drive 0
disk           fd0 at fdc0 drive 1
disk           fd0 at fdc0 drive 2
device         fd0 at fdc2 drive 0 csr 0x003000 irq 4 priority 2
disk           fd0 at fdc2 drive 1  
blockmajor 51
charmajor 52

SEE ALSO
ld(1), modunload(8), modstat(8)
NAME
modstat – display status of Sun386i modules

SYNOPSIS
modstat [-id module_id]

AVAILABILITY
Sun386i systems only.

DESCRIPTION
modstat displays the status of the loaded modules. A sample status from modstat:

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Loadaddr</th>
<th>Size</th>
<th>B-major</th>
<th>C-major</th>
<th>Sysnum</th>
<th>Mod Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Drv</td>
<td>fd000000</td>
<td>d000</td>
<td>58</td>
<td></td>
<td></td>
<td>Tablet Driver</td>
</tr>
<tr>
<td>1</td>
<td>Sys</td>
<td>fd00d000</td>
<td>2000</td>
<td></td>
<td>180</td>
<td></td>
<td>Pageinfo</td>
</tr>
</tbody>
</table>

The Size displayed is a hexadecimal number in bytes of the sum of text + data + bss + symbol_table. The Size value includes the symbol_table only when the module was loaded with the modload -sym option.

OPTIONS
- -id module_id
   Display status of only this module.

SEE ALSO
modload(8), modunload(8)
NAME
mount, umount — mount and dismount filesystems

SYNOPSIS
/usr/etc/mount [-p ]
/usr/etc/mount -a[fnv] [-t type ]
/usr/etc/mount [-fnv ] [ -t type ] [ -o options ] filesystem directory
/usr/etc/mount [ -vfn ] [ -t type ] [ -h host ]
/usr/etc/umount [-t type ] [ -h host ]
/usr/etc/umount [-v fn ] [ -t type ] [ -o options ] filesystem directory ... 

DESCRIPTION
mount attaches a named filesystem to the filesystem hierarchy at the pathname location directory, which
must already exist. If directory has any contents prior to the mount operation, these remain hidden
until the filesystem is once again unmounted. If filesystem is of the form hostpathname, it is assumed
to be an NFS filesystem (type nfs).

umount unmounts a currently mounted filesystem, which can be specified either as a directory or a
filesystem.

mount and umount maintain a table of mounted filesystems in /etc/mtab, described in fstab(5). If
invoked without an argument, mount displays the contents of this table. If invoked with either a
filesystem or directory only, mount searches the file /etc/fstab for a matching entry, and mounts the
filesystem indicated in that entry on the indicated directory.

MOUNT OPTIONS
-p Print the list of mounted filesystems in a format suitable for use in /etc/fstab.
-a All. Attempt to mount all the filesystems described in /etc/fstab. If a type argument is
specified with -t, mount all filesystems of that type. Filesystems are not necessarily mounted
in the order shown in /etc/fstab.
-f Fake an /etc/mtab entry, but do not actually mount any filesystems.
-n Mount the filesystem without making an entry in /etc/mtab.
-v Verbose. Display a message indicating each filesystem being mounted.
-t type Specify a filesystem type. The accepted types are 4.2, and nfs; see fstab(5) for a description
of these types.
-r Mount the specified filesystem read-only, even if the entry in /etc/fstab specifies that it is to be
mounted read-write.
Physically write-protected and magnetic-tape filesystems must be mounted read-only. Otherwise
errors occur when the system attempts to update access times, even if no write operation is
attempted.
-o options
Specify filesystem options —list of comma-separated words from the list below. Some options
are valid for all filesystem types, while others apply to a specific type only.

options valid on all filesystems:
    rw | ro  Read/write or read-only.
    suid | nosuid Setuid execution allowed or disallowed.
    grpid Create files with BSD semantics for the propagation of the group ID.
Under this option, files inherit the GID of the directory in which they
are created, regardless of the directory’s set-GID bit.
Do not mount this filesystem that is currently mounted read-only. If the filesystem is not currently mounted, an error results.

If the file system is currently mounted, and if the entry in /etc/fstab specifies that it is to be mounted read-write or rw was specified along with remount, remount the file system making it read-write. If the entry in /etc/fstab specifies that it is to be mounted read-only and rw was not specified, the file system is not remounted. If the file system is not currently mounted, an error results.

The default is ‘rw, suid’.

options specific to 4.2 filesystems:

quota | noquota Usage limits are enforced, or are not enforced. The default is noquota.

options specific to nfs (NFS) filesystems:

bg | fg If the first attempt fails, retry in the background, or, in the foreground.
retry=n The number of times to retry the mount operation.
rsize=n Set the read buffer size to n bytes.
ssize=n Set the write buffer size to n bytes.
timeo=n Set the NFS timeout to n tenths of a second.
retrans=n The number of NFS retransmissions.
port=n The server IP port number.
soft | hard Return an error if the server does not respond, or continue the retry request until the server responds.
intr Allow keyboard interrupts on hard mounts.
secure Use a more secure protocol for NFS transactions.
acregmin=n Hold cached attributes for at least n seconds after file modification.
acregmax=n Hold cached attributes for no more than n seconds after file modification.
acdirmin=n Hold cached attributes for at least n seconds after directory update.
acdirmax=n Hold cached attributes for no more than n seconds after directory update.
actimeo=n Set min and max times for regular files and directories to n seconds.

Regular defaults are:

fg, retry=10000, timeo=7, retrans=3, port=NFS_PORT, hard,\nacregmin=3, acregmax=60, acdirmin=30, acdirmax=60

Defaults for rsize and ssize are set internally by the system kernel.

UMOUNT OPTIONS

-h host Unmount all filesystems listed in /etc/mtab that are remote-mounted from host.
-t type Unmount all filesystems listed in /etc/mtab that are of a given type.
-a Unmount all filesystems currently mounted (as listed in /etc/mtab).
-v Verbose. Display a message indicating each filesystem being unmounted.

NFS FILESYSTEMS

Background vs. Foreground

Filesystems mounted with the bg option indicate that mount is to retry in the background if the server’s mount daemon (mountd(8c)) does not respond. mount retries the request up to the count specified in the retry=n option. Once the filesystem is mounted, each NFS request made in the kernel waits time=n tenths of a second for a response. If no response arrives, the time-out is multiplied by 2 and the request is retransmitted. When the number of retransmissions has reached the number specified in the retrans=n option, a filesystem mounted with the soft option returns an error on the request; one
mounted with the **hard** option prints a warning message and continues to retry the request.

**Read-Write vs. Read-Only**

Filesystems that are mounted **rw** (read-write) should use the **hard** option.

**Interrupting Processes With Pending NFS Requests**

The **intr** option allows keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted filesystem.

**Secure Filesystems**

The **secure** option must be given if the server requires secure mounting for the filesystem.

**File Attributes**

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting **actimeo=n** extends flush time by **n** seconds for both regular files and directories.

**SYSTEM V COMPATIBILITY**

**System V File-Creation Semantics**

Ordinarily, when a file is created its GID is set to the effective GID of the calling process. This behavior may be overridden on a per-directory basis, by setting the set-GID bit of the parent directory; in this case, the GID is set to the GID of the parent directory (see **open(2)** and **mkdir(2)**). Files created on filesystems that are mounted with the **grpid** option will obey BSD semantics; that is, the GID is unconditionally inherited from that of the parent directory.

**EXAMPLES**

To mount a local disk:  

```
mount /dev/xy0g /usr
```

To fake an entry for nd root:  

```
mount -ft 4.2 /dev/nd0 /
```

To mount all 4.2 filesystems:  

```
mount -at 4.2
```

To mount a remote filesystem:  

```
mount -t nfs serv:/usr/src /usr/src
```

To hard mount a remote filesystem:  

```
mount -o hard serv:/usr/src /usr/src
```

To save current mount state:  

```
mount -p > /etc/fstab
```

**FILES**

```
/etc/mtab  table of mounted filesystems
/etc/fstab  table of filesystems mounted at boot
```

**SEE ALSO**

`mkdir(2), mount(2), umount(2), open(2), fstab(5), mtab(5), mountd(8C), nfsd(8)`

**BUGS**

Mounting filesystems full of garbage crashes the system.

If the directory on which a filesystem is to be mounted is a symbolic link, the filesystem is mounted on the directory to which the symbolic link refers, rather than being mounted on top of the symbolic link itself.
NAME
rarpd – DARPA Reverse Address Resolution Protocol service

SYNOPSIS
/usr/etc/rarpd if hostname

Sun386i SYNOPSIS
/usr/etc/rarpd if [ hostname ]

AVAILABILITY
This program is available on all Sun386i systems. On other Sun systems, it is available with the Networking Tools and Programs software installation option. Refer to Installing the SunOS for information on how to install optional software.

DESCRIPTION
rarpd starts a daemon that responds to Reverse Address Resolution Protocol (Reverse ARP) requests. The daemon forks a copy of itself, and requires root privileges.

The Reverse ARP protocol is used by machines at boot time to discover their (32 bit) IP address given their (48 bit) Ethernet address. In order for the request to be answered, a machine’s name-to-IP-address entry must exist in the /etc/hosts file and its name-to-Ethernet-address entry must exist in the /etc/ethers file. Furthermore, the server that runs the rarpd daemon must have entries in both files. Note that if the server machine is using the Yellow Pages service, the server’s files are ignored, and the appropriate Yellow Pages maps queried.

The first argument, if, is one of the interface parameter strings (listed in boot(8S)), in the form of “name unit”, for example ieO. The second argument, hostname, is the interface’s corresponding host name. The if, hostname pair should be the same as the arguments passed to the ifconfig (8) command. As with ifconfig, rarpd must be invoked for each interface that the server wishes to support. Therefore a gateway machine may invoke the rarpd multiple times, for example:

/usr/etc/rarpd ieO host
/usr/etc/rarpd iel host-backbone

Sun386i DESCRIPTION
On the Sun386i, rarpd is responsible for dynamic IP address allocation using Dynamic RARP. If the pnp policy is not set to restricted in the YP policies map, then Dynamic RARP requests may cause rarpd to request allocation of a temporarily unused IP address from the ipalloc daemon. This happens only when the system is not listed in the hosts and ethers YP maps as being on the particular network segment.

If the pnp policy is set to restricted then Dynamic RARP requests that can not be satisfied will receive an error response indicating that Automatic System Installation is not enabled on the network segment. In such a case, systems trying to install themselves on the network will report that manual installation by the network administrator is required.

Only Yellow Pages servers provide Dynamic RARP service. If any system incorrectly tries to provide Dynamic RARP service on the network, this will be detected and dynamic IP address allocation will be disabled. This is required, since otherwise two different authorities could be assigning IP addresses on the network and would probably allocate addresses that should not be allocated. Only one Address Authority may exist for a network segment; it must have the authoritative list of all Dynamic RARP clients.

IP address allocation using the RARP protocol, as well as the Dynamic RARP protocol, may be enabled by setting the ip_address_allocation policy (in the YP policies map) to the value rarp_and_drarp. If this is done, then all RARP clients must be listed in the YP databases used by rarpd. If this is not done, some clients may be returned incorrect addresses when one is dynamically assigned. The Dynamic RARP protocol may be completely disabled by setting this policy value to none. This is strongly discouraged.
FILES
/etc/ethers
/etc/hosts

SEE ALSO
boot(8S), ifconfig(8C) ipallocl(8C), ethers(5), hosts(5), ipallocl(8C), netconfig(8C), pnpboot(8), policies(5)

NAME
rwhod - system status server

SYNOPSIS
/usr/etc/in.rwhod

AVAILABILITY
Due to its potential impact on network performance, this service is commented out of the /etc/rc.local
system initialization script. It is provided only for 4.3 BSD compatibility.

This program is available with the Networking Tools and Programs software installation option. Refer
to Installing the Sun Operating System for information on how to install optional software.

DESCRIPTION
rwhod is the server which maintains the database used by the rwho(1C) and ruptime(1C) programs.
Its operation is predicated on the ability to broadcast messages on a network.

rwhod operates as both a producer and consumer of status information. As a producer of information it
periodically queries the state of the system and constructs status messages which are broadcast on a net­
work. As a consumer of information, it listens for other rwhod servers' status messages, validating
them, then recording them in a collection of files located in the directory /var/spool/rwho.

The rwho server transmits and receives messages at the port indicated in the "rwho" service
specification, see services(5). The messages sent and received, are of the form:

```c
struct outmp {
    char out_line[8]; /* tty name */
    char out_name[8]; /* user id */
    long out_time; /* time on */
};

struct whod {
    char wd_vers;
    char wd_type;
    char wd_fill[2];
    int wd_sendtime;
    int wd_recvtime;
    char wd_hostname[32];
    int wd_loadavg[3];
    int wd_boottime;
    struct whoent { /*
        struct outmp we_utmp;
        int we_idle;
    } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are as calcu­
lated by the w(1) program, and represent load averages over the 5, 10, and 15 minute intervals prior to
a server's transmission. The host name included is that returned by the gethostname(2) system call.
The array at the end of the message contains information about the users logged in to the sending
machine. This information includes the contents of the utmp(5) entry for each non-idle terminal line
and a value indicating the time since a character was last received on the terminal line.

Messages received by the rwho server are discarded unless they originated at a rwho server's port. In
addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the
message is discarded. Valid messages received by rwhod are placed in files named whod.hostname in
the directory /var/spool/rwho. These files contain only the most recent message, in the format
described above.
Status messages are generated approximately once every 60 seconds. rwhod performs an nlist(3) on /vmunix every 10 minutes to guard against the possibility that this file is not the system image currently operating.

FILES
/var/spool/rwho

DIAGNOSTICS
- Status and diagnostic messages are logged to the appropriate system log using the syslogd(8) facility.

SEE ALSO
rwho(1C), ruptime(1C), w(1), gethostname(2), nlist(3), utmp(5), syslogd(8)

BUGS
This service takes up progressively more network bandwidth as the number of hosts on the local net increases. For large networks, the cost becomes prohibitive. RPC-based services such as rup(1C) and rusers(1C) provide a similar function with greater efficiency.

rwhod should relay status information between networks. People often interpret the server dying as a machine going down.
NAME
start_applic – generic application startup procedures

SYNOPSIS
/usr/etc/start_applic

AVAILABILITY
Sun386i systems only.

DESCRIPTION
start_applic is a short generic shell script that can be copied or symbolically linked into either
/vol/local/bin/application or /usr/local/bin/application. When invoked as application, an application
installed as described below will be correctly invoked on systems of any supported processor architecture.
Installing start_applic (or a customized version of it) in one of these locations ensures that no
user's or system’s environment needs to be modified just to run the application. Applications are stored
in a single tree, not shared with any other applications. This tree may be available on different systems
in different places; if the application needs to reference its distribution tree, this should be determined
from the $application_ROOT environment variable.

The application startup script arranges that the $PATH and $application_ROOT environment variables
are set correctly while the application is running. If the application’s distribution tree (placed into
/vol/application or /usr/local/application) doesn’t have an executable binary with the name of the application
(e.g. /vol/application/bin.arch/application) then start_applic can not be used, and a customized
application startup script must be used instead. Such scripts must also allow users to invoke the application
from systems of any architecture, without requiring them to customize their own environments.

Heterogeneous Networked Installations
Applications available on the network are available through /vol/application and exported either to all
systems or just to selected ones, as licensing restrictions allow. The export point is
/export/vol/application, which is a symbolic link to the actual installation point, typically the
/files/vol/application directory. All subdirectories not explicitly tagged with a processor architecture are
shared among all processor architectures; thus while the .. ./bin.sun386 and .. ./lib.sun386 subdirectories
contain respectively binaries and libraries executable only on systems of the sun386 architecture, the
.. ./bin directory contains executables that run on any architecture (typically using an interpreter such as
/bin/sh), and the .. ./etc directory only contains sharable configuration files.

Homogeneous Single Machine Installations
Applications available only on a specific machine and its boot clients of the same architecture are
installed into /usr/local/application. This directory supports only a single architecture; this means that
/usr/local/application/bin contains binaries executable only on the local architecture, and
/usr/local/application/lib contains libraries executable only on the local architecture. Any sharable files
may be grouped in /usr/local/application/share.

If an application is to be installed onto a boot server with the intent of serving it to boot clients with
architectures other than the server’s native one, it will appear on all of those systems in
/usr/local/application as described above. However, the installation point (on the server) for application
binaries of architecture arch is /export/local/arch/application. (When the architecture is the server
architecture, this case is identical with the one above.)

Other Installations
Note that these are two contrasting models of software installation. The heterogeneous model assumes
general availability of the software, and solves the “which binaries to use” problem with no adminis-
trative overhead. The homogeneous model assumes very limited availability of software, requires
administrative procedures to ensure that /usr/local only contains binaries of the local architecture, and
doesn’t really account for networked installations. It is easier to add support for additional architectures
using a heterogeneous network model of software installation from the beginning.
Smaller applications (of only one or two files) may be installed into the appropriate /vol/local/bin.arch directory, or possibly into /export/local/arch/bin. These directories are in user’s default paths, so the application does not need to be registered using start_applic.

FILES
/files<n>/vol/application
/export/vol/application
/vol/application
/vol/application/bin.arch/application
/usr/local/application
/export/local/arch/application

SEE ALSO
automount(8), auto.vol(5), exportfs(8), exports(5)
NAME
unconfigure – reset the network configuration for a Sun386i system

SYNOPSIS
/usr/etc/unconfigure [-y]

AVAILABILITY
Sun386i systems only.

DESCRIPTION
unconfigure restores most of the system configuration and status files to the state they were in when delivered by Sun Microsystems, Inc. It also deletes all user accounts (including home directories), Yellow Pages information, and any diskless client configurations that were set up.

After running unconfigure, a system halts. Rebooting it to multi-user mode at this point will start automatic system installation.

unconfigure is intended for use in the following situations:
• As one of the final steps in Software Manufacturing.
• In systems being set up with temporary configurations, holding no user accounts or diskless clients. These will occur during demonstrations and evaluation trials.
• To allow systems that had been used as standalones to be upgraded to join a network in a role other than as a master server. (See instructions later.)

unconfigure is potentially a dangerous utility; it does not work unless invoked by the super-user. As a warning, unless the -y option is passed, it will require confirmation that all user files and system software configuration information is to be deleted.

This utility is not recommended for routine use of any sort.

Resetting Temporary Configurations
If users need to set up and tear down configurations, unconfigure can be used to restore the system to an essentially as-manufactured state. The main concern here is that user accounts will be deleted, so this should not be done casually.

To reset a temporary configuration, just become the super-user and invoke unconfigure.

Upgrading Standalones to Network Clients
Systems that are going to be networked should be networked from the very first, if at all possible. This eliminates whole classes of compatibility problems, such as pathnames and (in particular) user account ID (UID) clashes.

Automatic system installation directly supports upgrading a single standalone system to a YP master, and joining any number of unused systems (or systems upon which unconfigure has been run) into a network.

However, in the situation where standalone systems that have been used extensively are to be joined to a network, unconfigure can be used in conjunction with automatic system installation by a knowledgeable super-user to change a system’s configuration from standalone to network client. This particular procedure is not recommended for use by inexperienced administrators. Inexperienced administrators should use the directions found in the Sun386i SNAP Administration book instead.

The following procedure is needed only when user accounts or other data need to be preserved; it is intended to ensure that every UID and GID is changed so as not to clash with those in use on the network. It must be applied to each system that is being upgraded from a standalone to a network client.

The procedure is as follows:
1. Identify all accounts and files that you’ll want to save. If there are none, just run unconfigure and install the system on the network. Do not follow the remaining steps.
2. Copy /etc/yppasswd to /etc/yppasswd.bak.
3. Rename all the files (including home directories) so that they aren’t deleted. (See FILES below.) These should only be found in `/files/home` and perhaps in `/files/vol/local`.

4. Run `unconfigure` and install the system on the network.

5. For each account listed in `/etc/yppasswd.bak` that you want to save, follow this procedure:
   a. Create a new account on the network; if the UID and GID are the same as in `/etc/yppasswd.bak` on the standalone, then skip the next step. However, be sure that you do not make two different accounts with the same UID. (Instructions for manually creating user accounts may be found in the *Sun386i Advanced Administration* book.)
   b. Use the `chown -R` command to change the ownership of the home directories.
   c. You may need to rename the files you just chowned above, for example to ensure that they are the user’s home directory. This may involve updating the `auto.home(5)` and `auto.vol(5)` YP maps, as well.
   d. For the files that are to be exported, put symbolic links pointing to them from the `/export` tree, and include the pathnames of these symbolic links (e.g. `/export/home/groupname/username`) into `/etc/exports`

6. Delete `/etc/yppasswd.bak`.

**FILES**

`unconfigure` deletes the following files, if they are present, replacing some of them with the distribution version if one is supposed to exist:

```
lfB lfB lfB lfB /etc/rootkey /etc/ethers /etc/localtime /etc/publ ickey
/etc/auto.home /etc/exports /etc/net.conf /etc/sendmail.cf
/etc/auto.vol /etc/fstab /etc/netmasks /etc/syslog.conf
/etc/bootservers /etc/ypgroup /etc/networks /etc/systems
/etc/bootparams /etc/ypgroup /etc/hosts /etc/yppasswd /single/ifconfig
/var/sysex/* /etc/passwd /etc/group /etc/printcap /etc/ttytab
```

and all files in `/var/yp` except those distributed with the operating system.

`unconfigure` truncates all files in `/var/adm`. All user home directories symbolically linked to in `/export/home` are deleted, except those for the default user account `users`, which is shipped with the operating system. All diskless client configuration information symbolically linked to in `/export/root`, `/export/swap`, and `/export/dump` is deleted.

**SEE ALSO**

`find(1)`, `passwd(5)`, `group(5)`, `adduser(8)`, `chgrp(1)`, `chown(8)`

**BUGS**

More of the system configuration files should be reset.
NAME
ypsnc – collect most up-to-date YP maps

SYNOPSIS
/usr/etc/yp/ypsync [-r] [-u]

AVAILABILITY
Sun386i systems only.

DESCRIPTION
The ypsnc command is used to gather current Yellow Pages (YP) maps to the local YP server. When invoked with no arguments, it will poll all the YP servers listed in the ypservers YP map for the maps they serve, and the order of those maps. If there are any new maps that the local server does not have, or if there are maps that are more current than the local server’s copy, it invokes ypxfr to transfer those maps to the local server.

The ypsnc command eliminates the need for cron jobs to ensure that YP map updates are eventually transmitted to all YP servers, and supports different YP maps having different masters. It is invoked periodically by ypserv(8).

When invoked with the -r flag, ypsnc will recreate the local /var/yp directory and databases if needed. This facility is used when upgrading servers, since they can automatically retrieve YP maps without needing manual intervention. The YP master of the ypservers map can also designate new servers, which would automatically pick up their new maps on reboot.

When invoked with the -u flag, ypsnc will update the list of YP servers on the master of the ypservers YP map to include the local system if it doesn’t already, and then get copies of all the YP databases. A user invoking ypsnc -u may not be root, and must have the networks privilege in the YP group map.

FILES
/var/yp/YP.domainname

SEE ALSO
ypupdate(3c), ypserv(8), ypxfr(8)