

**CTIX™ OPERATING SYSTEM MANUAL**

**Version C  
Volume 2**

Convergent Technologies is a registered trademark of  
Convergent Technologies, Inc.

Convergent, CTIX, S/80, S/280, S/480, S/640, and S/4040 are trademarks of  
Convergent Technologies, Inc.

CTIX is derived from UNIX System V by Convergent Technologies under license from  
AT&T. UNIX and RFS are trademarks of AT&T.

Material excerpted from the UNIX System V, Release 3.2 *System Administrator's/User's  
Reference Manual* and *Programmer's Reference Manual* is Copyright 1989 by AT&T  
Technologies. Reprinted by permission.

This software and documentation is based in part on the Fourth Berkeley Software  
Distribution under license from the Regents of the University of California.

This manual was prepared on a Convergent Technologies S/640 Computer System and  
was printed on an Apple LaserWriter II Laser Printer.

Second Edition (November 1989) 09-02263-01  
Update Notice 1 (November 1990) 09-02578

Copyright © 1990 by Convergent Technologies, Inc.,  
San Jose, CA. Printed in USA.

All rights reserved. No part of this document may be reproduced, transmitted, stored in a  
retrieval system, or translated into any language without the prior written consent of  
Convergent Technologies, Inc.

Convergent Technologies makes no representations or warranties with respect to the  
contents hereof and specifically disclaims any implied warranties of merchantability or  
fitness for any particular purpose. Further, Convergent Technologies reserves the right  
to revise this publication and to make changes from time to time in its content without  
being obligated to notify any person of such revision or changes.

# TABLE OF CONTENTS: VOLUME 2

How to Use This Manual ..... ix

## 1. Commands and Application Programs: M-Z

m4	macro processor
machid	mc68k, miti, mini, mega, unixpc, i386, i286,
mail	send mail to users or read mail
mailx	interactive message processing system
make	maintain, update, and regenerate groups of programs
makekey	generate encryption key
mcs	manipulate the object file comment section
mesg	permit or deny messages
mkdbsym	load symbols in kernel debugger
mkdir	make directories
mkfs	construct a file system
mkhosts	make node name commands
mkifile	make an ifile from an object file
mklost+found	make a lost+found directory for fsck
mknod	build special file
mkshlib	create a shared library
mktpy	install or relocate a PT or GT local printer
mm	print/check documents formatted with the MM macros
mmt	typeset documents, view graphs, and slides
more	text perusal
mount	mount and unmount file systems and remote resources
mountall	mount, unmount multiple file systems
mountd	NFS mount request server
mvdire	move a directory
named	Internet domain name server
nawk	pattern scanning and processing language
ncheck	generate path names from i-numbers
netstat	show network status
newaliases	rebuild the data base for the mail aliases file
newform	change the format of a text file
newgrp	log in to a new group
news	print news items
nfsd	NFS daemons
nfsstat	Network File System statistics
nice	run a command at low priority
nl	line numbering filter
nlsadmin	network listener service administration
nm	print name list of common object file
nmountall	mount, unmount Network File System resources
nohup	run a command immune to hangups and quits
nroff	format text
nsquery	Remote File Sharing name server query
occ	old C compiler
od	octal dump

pack . . . . . compress and expand files  
 passmgmt . . . . . password files management  
 passwd . . . . . change login password  
 paste . . . . . merge same lines of several files or subsequent lines of one file  
 path . . . . . locate executable file for command  
 pg . . . . . file perusal filter for CRTs  
 ping . . . . . send ICMP ECHO\_REQUEST packets to network hosts  
 portmap . . . . . DARPA port to RPC program number mapper  
 pr . . . . . print files  
 prof . . . . . display profile data  
 profiler . . . . . operating system profiler  
 prs . . . . . print an SCCS file  
 ps . . . . . report process status  
 ptx . . . . . permuted index  
 pwck . . . . . password/group file checkers  
 pwconv . . . . . install and update /etc/shadow  
 pwd . . . . . working directory name  
 pwunconv . . . . . install and update /etc/shadow  
 qinstall . . . . . install/verify software using mkfs(1) proto file database  
 qlist . . . . . print out file lists from proto file; set links based on  
 ratfor . . . . . rational FORTRAN dialect  
 rc0 . . . . . run commands performed to stop the operating system  
 rc2 . . . . . run commands performed for multi-user environment  
 rcmd . . . . . remote shell command execution  
 rcp . . . . . remote file copy  
 reboot . . . . . reboot the system  
 regcmp . . . . . regular expression compile  
 renice . . . . . alter priority of running process by changing nice  
 rexecd . . . . . remote execution server  
 rfadmin . . . . . Remote File Sharing domain administration  
 rfpasswd . . . . . change Remote File Sharing host password  
 rfstart . . . . . start Remote File Sharing  
 rfstop . . . . . stop the Remote File Sharing environment  
 rfudadmin . . . . . Remote File Sharing notification shell script  
 rfudaemon . . . . . Remote File Sharing daemon process  
 riopcfg . . . . . configure system for Remote I/O processor  
 riopqry . . . . . query Remote I/O processor for online data  
 rlogin . . . . . remote login  
 rlogind . . . . . remote login server  
 rm . . . . . remove files or directories  
 rmdel . . . . . remove a delta from an SCCS file  
 rmntstat . . . . . display mounted resource information  
 rmnttry . . . . . attempt to mount remote resources  
 rmount . . . . . retry remote resource mounts  
 rmountall . . . . . mount, unmount Remote File Sharing resources  
 route . . . . . manually manipulate the routing tables  
 routed . . . . . network routing daemon  
 rpcinfo . . . . . report RPC information  
 rshd . . . . . remote shell server  
 rsterm . . . . . manually start and stop terminal input and output  
 rtpenable . . . . . real-time priorities enabled/disabled  
 rumount . . . . . cancel queued remote request

runacct . . . . . run daily accounting  
ruptime . . . . . display status of nodes on local network  
rwho . . . . . who is logged in on local network  
rwhod . . . . . host status server  
sact . . . . . print current SCCS file editing activity  
sadb . . . . . disk access profiler  
sag . . . . . system activity graph  
sar . . . . . system activity reporter  
sar . . . . . system activity report package  
sccsdiff . . . . . compare two versions of an SCCS file  
script . . . . . make typescript of terminal session  
scsimap . . . . . set mappings for SCSI devices  
sdb . . . . . symbolic debugger  
sdiff . . . . . side-by-side difference program  
sed . . . . . stream editor  
sendmail . . . . . mail routing program  
serstat . . . . . display serial port error statistics  
setmnt . . . . . establish mount table  
setuname . . . . . set name of system  
sh . . . . . shell, the standard/restricted command programming language  
shl . . . . . shell layer manager  
showmount . . . . . show all remote mounts  
shutdown . . . . . shut down system, change system state  
size . . . . . print section sizes in bytes of common object files  
slattach . . . . . attach and detach serial lines as network interfaces  
sleep . . . . . suspend execution for an interval  
slink . . . . . streams linker, load socket configuration  
slipd . . . . . switched Serial Line Internet Protocol control facility  
sno . . . . . SNOBOL interpreter  
sort . . . . . sort and/or merge files  
spell . . . . . find spelling errors  
spline . . . . . interpolate smooth curve  
split . . . . . split a file into pieces  
starter . . . . . information about the operating system for beginning users  
stat . . . . . statistical network useful with graphical commands  
strclean . . . . . STREAMS error logger cleanup program  
strerr . . . . . STREAMS error logger daemon  
strings . . . . . extract the ASCII text strings in a file  
strip . . . . . strip symbol and line number information  
stty . . . . . set the options for a terminal  
su . . . . . become super-user or another user  
sum . . . . . print checksum and block count of a file  
swap . . . . . swap administrative interface  
sync . . . . . update the super block  
sysdef . . . . . output system definition  
tabs . . . . . set tabs on a terminal  
talk . . . . . talk to another user  
talkd . . . . . remote user communication server  
tapeset . . . . . set drive parameters for tape controllers  
tar . . . . . tape file archiver  
tdl . . . . . RS-232 terminal download  
tee . . . . . pipe fitting

telnet . . . . . user interface to TELNET protocol  
telnetd . . . . . DARPA TELNET protocol server  
test . . . . . condition evaluation command  
tftp . . . . . user interface to the DARPA TFTP protocol  
tftpd . . . . . DARPA Trivial File Transfer Protocol server  
tic . . . . . terminfo compiler  
time . . . . . time a command  
timex . . . . . time a command; report process data and system activity  
tio . . . . . tape i/o filter  
toc . . . . . graphical table of contents routines  
touch . . . . . update access and modification times of a file  
tplot . . . . . graphics filters  
tput . . . . . initialize a terminal or query terminfo database  
tr . . . . . translate characters  
troff . . . . . typeset text  
true . . . . . provide truth values  
tset . . . . . set terminal, terminal interface, and terminal environment  
tsioctl . . . . . facilitate usage of a tape drive  
tsort . . . . . topological sort  
tty . . . . . get the name of the terminal  
uadmin . . . . . administrative control  
uconf . . . . . configure the operating system  
ul . . . . . do underlining  
umask . . . . . set file-creation mode mask  
unadv . . . . . unadvertise a Remote File Sharing resource  
uname . . . . . print name of current CTIX system  
unget . . . . . undo a previous get of an SCCS file  
uniq . . . . . report repeated lines in a file  
units . . . . . conversion program  
update . . . . . provide disk synchronization  
usage . . . . . retrieve a command description and usage examples  
uucheck . . . . . check the uucp directories and permissions file  
uucico . . . . . file transport program for the uucp system  
uucleanup . . . . . uucp spool directory clean-up  
uucp . . . . . CTIX-to-CTIX system copy  
uucpd . . . . . network uucp servers  
uugetty . . . . . set terminal type, modes, speed, and line discipline  
uusched . . . . . the scheduler for the UUCP system  
uustat . . . . . uucp status inquiry and job control  
uuto . . . . . public CTIX-to-CTIX system file copy  
uux . . . . . CTIX-to-CTIX system command execution  
uuxqt . . . . . execute remote command requests  
val . . . . . validate SCCS file  
vc . . . . . version control  
vi . . . . . screen-oriented (visual) display editor based on ex  
volcopy . . . . . make literal copy of file system  
wait . . . . . await completion of process  
wall . . . . . write to all users  
wc . . . . . word count  
what . . . . . identify SCCS files  
who . . . . . who is on the system  
whodo . . . . . who is doing what

wm . . . . . window management  
write . . . . . write to another user  
xargs . . . . . construct argument list(s) and execute command  
xstr . . . . . extract and share strings in C programs  
yacc . . . . . yet another compiler-compiler

—

—

—



**NAME**

gettimeofday, settimeofday - get/set date and time

**SYNOPSIS**

```
#include <sys/time.h>

int gettimeofday(tp, tzp)
struct timeval *tp;
struct timezone *tzp;

int settimeofday(tp, tzp)
struct timeval *tp;
struct timezone *tzp;
```

**DESCRIPTION**

The current Greenwich time and the current time zone are obtained with the *gettimeofday* call, and set with the *settimeofday* call. The time is expressed in seconds and microseconds since midnight (0 hour), January 1, 1970 (GMT). The resolution of the system clock is hardware dependent, and the time may be updated continuously or in “ticks.” If *tzp* is a NULL pointer, the time zone information will not be returned or set.

The structures pointed to by *tp* and *tzp* are defined in *<sys/time.h>* as follows:

```
struct timeval {
    long   tv_sec;      /* seconds since Jan. 1, 1970 */
    long   tv_usec;    /* and microseconds (always 0 on CTIX) */
};

struct timezone {
    int    tz_minuteswest; /* of Greenwich */
    int    tz_dsttime; /* type of dst correction to apply */
};
```

The *timezone* structure indicates the local time zone (measured in minutes of time westward from Greenwich), and a flag that, if nonzero, indicates that Daylight Savings Time applies locally during the appropriate part of the year.

Only the superuser can set the time of day or time zone.

Note that you must link the sockets library to your program. Use **-lsocket** in the compile command line.

**SEE ALSO**

date(1), adjtime(2), ctime(3C).

**RETURN VALUE**

A 0 return value indicates that the call succeeded. A -1 return value indicates an error occurred, and in this case, an error code is stored into the global variable *errno*.

**ERRORS**

The following error codes may be set in *errno*:

[EFAULT]        An argument address referenced invalid memory.

[EPERM]         A user other than the superuser attempted to set the time.

## NAME

notify, unnotify, evwait, evnowait - manage notifications

## SYNOPSIS

```
#include <notify.h>

int notify(type, arg, tag)
  ushort type;
  char *arg;
  char *tag;

int unnotify(type, arg)
  ushort type;
  char *arg;

ushort evwait(tag, datum)
  char **tag;
  char **datum;

ushort evnowait(tag, datum)
  char **tag;
  char **datum;
```

## DESCRIPTION

The *notify* system call interface allows a user process to record a number of events that it is interested in, and then waits for any one of them. Like *select(2)*, it does synchronous I/O multiplexing, but *notify* waits for a wider range of events and thus has greater functionality than *select*.

The *notify* call requests a notification or set of notifications.

The *unnotify* call retracts an earlier request (or set of requests) for notification.

The *evwait* call waits for a notification to be posted to the calling process.

The *evnowait* call returns the first notification if one exists, returning immediately otherwise.

Notifications are posted FIFO (first-in, first-out) in the user process, each *evwait* returning the first notification or blocking until one is posted. When a *notify* call is given, the user must supply the *type* of notification, a *tag*, and an *argument*. The *tag* is an arbitrary number the size of a (**char \***), which is returned by any *evwait* call triggered by that notification request. The *argument* is type-specific and is described below.

The return values of *evwait* and *evnowait* are the *type* of the notification.

It is an error for *notify* to be called with a *type* and *arg* matching a currently active notification.

The *notify* calls support the following *types*:

#### **N\_FDREAD**

Queues a notification if the file descriptor *arg* is readable at the time of the *notify* call, and subsequently whenever there is data to be read. A notification is also queued at the end-of-file (EOF) or when the number of writers on a pipe goes to zero. The datum returned from an *evwait* is a count of the number of bytes available to be read. At EOF, the datum is -1, and the request is deleted. This type is implemented for sockets, pipes, ttys, and streams.

#### **N\_FDWRITE**

Queues a notification if the file descriptor *arg* is writable at the time of the *notify* call, and subsequently when the file goes from a non-writable to a writable state (that is, output is not blocked). *datum* is the number of characters writable. This type is implemented for sockets, pipes, and streams.

#### **N\_SIGNAL**

Queues a notification on receipt of a signal. This is used in conjunction with regular signal catching [see *signal(2)*]. When signal notification is in effect, all caught signals queue notifications instead of causing pseudo-interrupts. If multiple instances of a caught signal occur before the process has received the notification, the returned type is **N\_LOSTSIG** rather than **N\_SIGNAL**. Ignored or defaulted signals are handled normally. Signals are not reset upon notification.

Note that only one call to *notify*

```
notify(N_SIGNAL,ignored,tag)
```

is required to enable notification of all signals that have a signal catching function (use a null function). *evwait* and *evnowait* return the *tag* and *datum*. *datum* is a bitwise OR of all queued signals; that is, low-numbered signals are represented as low-order bits (signal *n* sets  $2^{n-1}$ ).

#### **N\_UMSGREAD,N\_UMSGWRITE**

Queues a notification if the message queue described by *arg* is or becomes readable or writable, respectively. The datum returned is the number of messages received or the number of characters that can be sent, respectively. When the message queue is removed, *datum* is -1, and the request is deleted.

**N\_INDIR**

If *type* is **N\_INDIR**, *arg* is actually a pointer to an array of the following structure (defined in `/usr/include/notify.h`):

```

struct n_request {
    ushort type;
    char *arg;
    char *tag;
}

```

The array should be terminated with an entry having *type* **N\_INDIR**. The entire set of notifications is either placed or removed. **N\_INDIR** is never returned by *evwait* or *evnowait*.

**N\_QUERY**

*Type* **N\_QUERY** is valid only as an argument to the *notify* call. *arg* is a pointer to an array of **struct n\_indir**, and *tag* is a pointer to an **int** containing the number of elements in the array.

On return, the array contains the current active notifications in a form suitable for passing to *notify* or *unnotify* (that is, terminated by **N\_INDIR**), and the **int** pointed to by *tag* contains the number of active notifications (even if there was not enough space to copy them all back).

**N\_SEMOP**

Queues a notification if the semaphore described by the **struct n\_semop** pointed to by *arg* would not block, is released, or is removed. *datum* is **semval** unless the semaphore has been removed, in which case it is -1.

```

struct n_semop {
    int semid; /* semaphore ID */
    short sem_num; /* semaphore number */
    short sem_op; /* semaphore operation */
}

```

**SEE ALSO**

*fcntl*(2), *msgop*(2), *pipe*(2), *read*(2), *select*(2), *signal*(2), *socket*(2), *wait*(2), *termio*(7).

## DIAGNOSTICS

All calls return -1 on error, setting *errno* to one of the following:

[EINVAL]	Invalid type was given
[EINVAL]	Caller never did a <i>notify</i> ( <i>unnotify</i> , <i>evwait</i> , <i>evnowait</i> )
[EINVAL]	File is not of a valid type (N_FDREAD, N_FDWRITE)
[EBADF]	File is not open (N_FDREAD, N_FDWRITE)
[EBADF]	Invalid message queue descriptor (N_UMSG)
[ENOSPC]	No space available to allocate notification queue header
[ENOSPC]	No space available to allocate table entry for this notification
[ENOSPC]	Too many active notification requests for given space (N_QUERY)
[EFAULT]	An address fault was generated by a user-supplied pointer

## EXAMPLE

```
#include "sys/types.h"
#include <sys/notify.h>
#include <stdio.h>
#include <signal.h>

int sig_catch();

main()
{
    int tag, datum, i;
    char buf[BUFSIZ];
    ushort rv, evwait();

    setbuf(stdout, NULL);
    if (notify(N_FDREAD, 0, 't') < 0)
        perror("notify for N_FDREAD of stdin failed"), exit(1);

    if (notify(N_SIGNAL, 2, 's') < 0)
        perror("notify failed"), exit(1);

    for (i=0; i<20; i++)
        signal(i, sig_catch);
```

```

for(;;) {
    /* Wait for an event */
    rv = evwait(&tag, &datum);

    /* Tell the user about it */
    printf("0v: %d tag: %d datum: %d0, rv, tag, datum);

    switch (tag) {
    case 's':
        break;
    case 't':
        /* Read the input */
        gets(buf);
        printf("read '%s'0, buf);
        if (*buf == 'q')
            exit(0);
        break;
    }
}
}
sig_catch()
{
}

```

**WARNING**

The *notify* system call interface is not portable, has little likelihood of becoming so, and may disappear in future releases of CTIX. It is therefore recommended that you use the *poll*(2) system call, and that existing software using *notify* be changed to use *poll*.

—

—

—



**NAME**

shmop - shared memory operations

**SYNOPSIS**

```
#include <sys/types.h>
```

```
#include <sys/ipc.h>
```

```
#include <sys/shm.h>
```

```
char *shmat (shmid, shmaddr, shmflg)
```

```
int shmid;
```

```
char *shmaddr;
```

```
int shmflg;
```

```
int shmdt (shmaddr)
```

```
char *shmaddr;
```

**DESCRIPTION**

*shmat* attaches the shared memory segment associated with the shared memory identifier specified by *shmid* to the data segment of the calling process. The segment is attached at the address specified by one of the following criteria:

- If *shmaddr* is equal to zero, the segment is attached at the first available address as selected by the system.
- If *shmaddr* is not equal to zero and (*shmflg* & SHM\_RND) is “true”, the segment is attached at the address given by [*shmaddr* - (*shmaddr* modulus SHMLBA)].
- If *shmaddr* is not equal to zero and (*shmflg* & SHM\_RND) is “false”, the segment is attached at the address given by *shmaddr*.

*shmdt* detaches from the calling process’s data segment the shared memory segment located at the address specified by *shmaddr*.

The segment is attached for reading if (*shmflg* & SHM\_RDONLY) is “true” {READ}, otherwise it is attached for reading and writing {READ/WRITE}.

*shmat* will fail and not attach the shared memory segment if one or more of the following are true:

- |          |  |
|----------|--|
| [EINVAL] | <i>shmid</i> is not a valid shared memory identifier.                                  |
| [EACCES] | Operation permission is denied to the calling process [see <i>intro(2)</i> ].          |
| [ENOMEM] | The available data space is not large enough to accommodate the shared memory segment. |

- [EINVAL] *shmaddr* is not equal to zero, and the value of [*shmaddr* - (*shmaddr* modulus SHMLBA)] is an illegal address.
- [EINVAL] *shmaddr* is not equal to zero, (*shmflg* & SHM\_RND) is “false”, and the value of *shmaddr* is an illegal address.
- [EMFILE] The number of shared memory segments attached to the calling process would exceed the system-imposed limit.
- [EINVAL] *shmdt* will fail and not detach the shared memory segment if *shmaddr* is not the data segment start address of a shared memory segment.

**SEE ALSO**

exec(2), exit(2), fork(2), intro(2), shmctl(2), shmget(2).

**DIAGNOSTICS**

Upon successful completion, the return value is as follows:

- *shmat* returns the data segment start address of the attached shared memory segment.
- *shmdt* returns a value of 0.

Otherwise, a value of -1 is returned and *errno* is set to indicate the error.

**NOTES**

The user must explicitly remove shared memory segments after the last reference to them has been removed.

**NAME**

syslocal - special system requests

**SYNOPSIS**

```
#include <syslocal.h>
int syslocal (cmd [ , arg ] ... )
int cmd;
```

**DESCRIPTION**

The *syslocal* routine executes certain special system calls. The specific call is indicated by the first argument. See the `<sys/syslocal.h>` file for complete documentation of parameters.

**System Type**

```
int syslocal(SYSL_SYSTEM)
```

Returns SYSL\_MITI for S/Series.

**Family Member**

```
int syslocal(SYSL_FAMILYMEMBER)
```

Returns a value identifying the specific system: SYSLFMITI1 for S/120, S/221-2, or S/320; SYSLFMITI2 for S/480 or S/640; SYSLFS80 for S/80; SYSLFS280 for S/280; and SYSLFS4000 for S/4040.

**Superblock Resynchronization**

```
int syslocal(SYSL_RESYNC, devnum)
```

```
short devnum
```

Rereads contents of superblock from disk. *devnum* specifies the file system: The high-order byte contains the major device number of the character special device; the low-order byte contains the minor device number. Only the superuser can reread the contents of the superblock from disk.

**Maximum Number of Users**

```
int syslocal(SYSL_MAXUSERS)
```

Returns maximum number of users configured for this system.

**Kernel Addresses**

```
syslocal(SYSL_KADDR, arg)
```

Returns certain addresses of kernel data structures. This allows certain programs (*ps*, *killall*) to run properly, even if `/unix` is not the currently running operating system.

*arg* is one of the following:

<b>SLA_V</b>	Returns address of var structure (sys/var.h).
<b>SLA_PROC</b>	Returns address of proc structure (sys/proc.h).
<b>SLA_ERR</b>	Returns address of err structure (sys/err.h).
<b>SCA_TIME</b>	Returns address of int time.
<b>SLA_CDT</b>	Returns address of crash dump table (CDT) = (sys/hardware.h).
<b>SLA_GDUTAB</b>	Returns address of gdutab (sys/iobuf.h).
<b>SLA_USRSTK</b>	Returns highest address of user stack.
<b>SLA_USIGN</b>	Returns signature of running UNIX (may be compared with <code>/unix</code> to see if they are identical).
<b>SLA_MEM</b>	Returns number of bytes of physical memory.
<b>SLA_BDEVCNT</b>	Returns number of slots in struct bdevsw (sys/conf.h).
<b>SLA_CDEVCNT</b>	Returns number of slots in struct cdevsw (sys/conf.h).
<b>SLA_PRELD</b>	Returns the address of the preloaded driver table.

### Object Module Type

**syslocal(SYSL\_0413MAGIC)**

Returns 1 if the kernel can support the `-F` option of `ld()`.

### Read Real-Time Clock

**syslocal(SYSL\_RDRTC, arg)**

Reads current state of real-time (battery supported) clock. *arg* is a pointer to struct `rtc` (sys/rtc.h)

### Write Real-Time Clock

**syslocal(SYSL\_WTRTC, arg)**

Writes new state of real-time clock. *arg* is a pointer to a struct `rtc` (sys/rtc.h). EIO is returned if any of the values are illegal. Only the superuser can write the real-time clock.

### Reboot System

**syslocal(SYSL\_REBOOT)**

Forces a software reset. Only the superuser may reset. Obsolete: retained for compatibility. Use `uadmin(2)` instead.

**Allocate or Bind a Loadable Driver****syslocal(SYSL\_ALLOCDRV, option, arg)****syslocal(SYSL\_BINDDRV, option, arg)**

These two functions implement the loadable driver functions of CTIX. They both require superuser privilege.

Loading drivers consists of two phases: allocation of virtual space, device numbers, and device IDs; and binding. Fully relocating a driver into memory, allocating physical space, plugging the device switch tables, calling initialization routines, and unloading require the same two phases in reverse. For information on the arguments, see `/usr/include/sys/drv.h`.

**Determine Processor Type****syslocal(SYSL\_PROCESSOR)**

Returns a value that can be used to determine what kind of processor (68010, 68020, or 68040) is running and whether floating-point hardware (68881/68040) is available.

**Enable Fixed Priority Range****syslocal(SYSL\_RTNICE, flag)**

Enables/disables the fixed priority range [see *nice(2)*]. *flag* is 1 for enable, 2 for disable. Only the superuser can execute this call, which affects every process.

**S/80, S/280, S/480, and S/640 Hardware Configuration****syslocal(SYSL\_MITICFIG)**

Returns a bit mask of the hardware that is present. Values can be found in `syslocal.h`. A more convenient way to get this information is by using *hinv(1M)*.

**S/4000 Hardware Configuration****syslocal(SYSL\_SXCFIG, arg1, arg2)**

Returns a bit mask of the hardware that is present. Values can be found in `syslocal.h`. A more convenient way to get this information is by using *hinv(1M)*.

*arg1* is one of the following:

SLA_SER	Returns a bit mask of the serial boards present.
SLA_IEN	Returns a bit mask of the Ethernet boards present.
SLA_SCSI	Returns a bit mask of the SCSI boards present.
SLA_SERNPORT	Returns number of serial ports for board denoted by <i>arg2</i> .

**SLA\_IENPORT** Returns number of Ethernet ports for board denoted by *arg2*.

**SLA\_SCSINPORT** Returns number of SCSI ports for board denoted by *arg2*.

**SEE ALSO**

*fsck(1M)*, *nice(2)*.

**DIAGNOSTICS**

Note that *syslocal* fails if one of the following is true:

[EINVAL] *cmd* or any suboption is illegal.

[EFAULT] An *arg* points outside the process's space.