**Device driver**

Making hardware work is tedious. To write to a hard disk, for example, requires that you write magic numbers in magic places, wait for the hard drive to say that it is ready to receive data, and then feed it the data it wants, very carefully. To write to a floppy disk is even harder, and requires that the program supervise the floppy disk drive almost constantly while it is running.

Instead of putting code in each application you write to control each device, you share the code between applications. To make sure that that code is not compromised, you protect it from users and normal programs that use it. If you do it right, you will be able to add and remove devices from your system without changing your applications at all. Furthermore, you need to be able to load your program into memory and run it, which the operating system also does. So an operating system is essentially a priviledged, general, sharable library of low-level hardware and memory and process control functions and routines.

All versions of Unix have an abstract way of reading and writing devices. By making the devices act as much as possible like regular files, the same calls (read(), write(), etc.) can be used for devices and files. Within the kernel, there are a set of functions, registered with the filesystem, which are called to handle requests to do I/O on ``device special files,'' which are those which represent devices. (See mknod(1,2) for an explanation of how to make these files.)

All devices controlled by the same device driver are given the same **major number**, and of those with the same major number, different devices are distinguished by different **minor numbers**.

int mknod(const char \**path*, mode\_t *mode*, dev\_t *dev*); [Option End]

***DESCRIPTION***

The *mknod*() function shall create a new file named by the pathname to which the argument *path* points.

The file type for *path* is OR'ed into the *mode* argument, and the application shall select one of the following symbolic constants:

|  |  |
| --- | --- |
| **Name** | **Description** |
| S\_IFIFO | FIFO-special |
| S\_IFCHR | Character-special (non-portable) |
| S\_IFDIR | Directory (non-portable) |
| S\_IFBLK | Block-special (non-portable) |
| S\_IFREG | Regular (non-portable) |

The only portable use of *mknod*() is to create a FIFO-special file.

Ex1:

#include <sys/types.h>

#include <sys/stat.h>

dev\_t dev;

int status;

...

status = mknod("/home/cnd/mod\_done", S\_IFIFO | S\_IWUSR |

S\_IRUSR | S\_IRGRP | S\_IROTH, dev);

## ismod

lsmod - list loaded modules. 

## SYNOPSIS

**lsmod** [-hV] 

## DESCRIPTION

**lsmod** shows information about all loaded modules.

The format is name, size, use count, list of referring modules. The information displayed is identical to that available from /proc/modules.

If the module controls its own unloading via a *can\_unload* routine then the user count displayed by lsmod is always -1, irrespective of the real use count. 

## OPTIONS

**-h, --help**

Display a summary of options and immediately exit.

**-V, --version**

Display the version of **lsmod** and immediately exit.