DEVICE FILES API

Device files are used to interface physical devices with application programs.
Specifically, when a process reads or writes to a device file, the kernel uses the major and minor device numbers of a file to select a device driver function to carry out the actual data transfer.
Device files may be character-based or block-based.
UNIX systems define the ***mknod*** API **to create device** files.

```
#include <sys/stat.h>
#include <unistd.h>
int mknod ( const char* path_name, mode t mode, int device_id );
```

1. The path_name argument is the path name of a directory to be created.
2. The mode argument specifies the access permission for the owner, group and others to be assigned to the file.
3. The *device_id* contains the major and minor device numbers and is constructed in most UNIX systems as follows: The lowest byte of *a device_id* is set to a minor device number and the next byte is set to the major device number. For example, to create a block device file called SCSI5 with major and minor numbers of 15 and 3, respectively, and access rights of read-write-execute for everyone, the *mknod* system call is:
   mknod("SCSI5", S_IFBLK | S_IRWXU | S_IRWXG | S_IRWXO, (15<<8) 13);
4. The major and minor device numbers are extended to fourteen and eighteen bits, respectively.
5. In UNIX, if a calling process has no controlling terminal and it opens a character device file, the kernel will set this device file as the controlling terminal of the process. How-ever, if the O_NOCTTY flag is set in the *open* call, such action will be suppressed.

6. The O_NONBLOCK flag specifies that the *open* call and any subsequent *read* or *write* calls to a device file should be nonblocking to the process.

The following *test mknod.C* program illustrates use of the *mknod, open, read, write,* and *close* APIs on a block device file.

```
#include <iostream.h>
#include <stdio.h>
 #include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
int main( int argc, char* argv[ ] ) {
if(argc!=4){
        cout << "usage: " << argv[0] << " <file> <major no> <minor no>\n";
        return 0;
}
int major = atoi( argv[2]), minor = atoi( argv[3] );
     (void) mknod( argv[1], S_IFCHR | S_IRWXU | S_IRWXG | S_IRWXO, ( major <<8) | minor );
int rc=1, fd = open(argv[1], O_RDWR | O_NONBLOCK | O_NOCTTY );
char buf[256];
while ( rc && fd != -1 )
 if (( rc = read( fd, buf, sizeof( buf )) ) < 0 )
  perror( "read" );
       else if ( rc) cout << buf << endl;
close(fd);
}
```

Source : http://elearningatria.files.wordpress.com/2013/10/cse-iv-unix-and-shell-programming-10cs44-notes.pdf