91.308 Operating Systems
Project #8: Unix File System
due April 20, 2005

- This assignment is due on Wed, April 20.
- The assignment must be done using C/C++ on a Unix operating system. The objective of the assignment includes learning about the Unix file system.
- You must submit the **course cover-sheet**, **hardcopy** of your source code, your output and a short write-up described below. The problem you must solve has been described in class and is formalized as follows:
- The program you will write will accept from 0 to an arbitrary number of command line file names and produce an output structure for each identified file as shown below:

```
FILENAME
FILE_TYPE
PERMISSIONS
OWNER_NAME
GROUP_NAME
DATE_OF_LAST_MODIFICATION
LINK_COUNT
SIZE_IN_BYTES OR DEV INFO
INODE_NUMBER

Example:

file:           alpha
type:           ordinary
perm:           rw- r-- r--
owner:          jedwards
group:          grad
last mod:       Mar 30 08:11 2003
link cnt:       2
size:           1345 (or 12, 6 type dev info)
I-number:       347
```

*******< a blank line between entries >*******

System calls needed on UNIX include:

#include <dirent.h>

getdirentries (int fd, char *buf, int nbytes, long *basep)

which reads up to nbytes of data into buf in the form:

```
unsigned long       d_ino;
unsigned short      d_reclen;
unsigned short      d_namlen;
char                d_name[MAXNAMLEN + 1];
```

see the man pages for more detail.

#include <time.h>

char *ctime (long *clock)

ctime() converts a long integer, pointed to by clock, to a 26-character string of the form:

```
Sun May 16 01:03:52 2003\n\0
```

see the man pages for more detail.

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

stat (char *path, struct stat *buf)

which fills in a data structure of the form:

```
struct stat {
    dev_t     st_dev;    /* device inode resides on */
    ino_t     st_ino;    /* this inode's number */
    mode_t    st_mode;   /* protection */
    nlink_t   st_nlink;  /* number or hard links to file */
    uid_t     st_uid;    /* user-id of owner */
    gid_t     st_gid;    /* group-id of owner */
    dev_t     st_rdev;   /* dev type, for inode that is dev */
    off_t     st_size;   /* total size of file */
    time_t    st_atime;  /* file last access time */
```

```
    time_t    st_mtime;   /* file last modify time */
    time_t    st_ctime;   /* file last status change time */
    uint_t    st_blksize; /* opt blocksize for file sys i/o */
    int       st_blocks;  /* actual number of blocks alloc'd */
…….. etc……..
    };
```

see the man pages for more detail.

The **getdirentries()** call requires that you use the **open()** system call to open a directory, and you can then use **getdirentries()** to extract filenames from the directory. Your program will have to work in two basic modes:

- if called with **no arguments** (as with ls) it must find the names of all the files in the current directory (including dotted files) and print information in the format shown above for each file object.
- if called with a **series of file names** (from the command line as with ls abc xyz etc) it must print information in the format shown above for each named object in the **argv [ ]** vector.

File types include **ordinary** (-), **directory** (d), **symbolic link** (l), **character device** (c), **and block device** (b). **You must show sample output with each of these types.** (You do not have to worry about pipe (p) and UNIX domain socket (s) types, nor do you have to print resolution names for symbolic link (l) types.)

Credits: This assignment was developed by Prof. Moloney, UML CS Dept.