LAB 12

UNIX Files and Directories

Reading Directories. Unix files and directories can be accessed by anyone who has access permission. However, only the kernel can write to a directory. In this lab we will learn how to access entries in a directory using the following system calls:

To open a directory,

```
DIR *opendir(const char *pathname); // returns a pointer if OK, NULL on error
```

To read an entry in a directory,

```
struct dirent *readdir(DIR *dp);      // returns a pointer if OK, NULL at end of
                                      // directory or error
```

To reset directory,

```
void rewinddir(DIR *dp);
```

To close directory,

```
int closedir(DIR *dp);                // returns 0 if OK, -1 on error
```

The `dirent` structure is defined in the file `dirent.h`. It contains at least the following two members:

```
struct dirent {
    ino_t d_ino;                  // i-node number
    char  d_name[NAME_MAX + 1];   // file name
}
```

The DIR structure is an internal structure used by these four functions to maintain information about the directory being used. It severs a purpose similar to the FILE structure that is maintained by the standard I/O library. The pointer to a DIR structure that is returned by `opendir`() is then used with the other three functions. `opendir`() initializes things so that the first `readdir`() reads the first entry in the directory. The ordering of entries within the directory is implementation dependent. It is usually not alphabetical.

Reading File Metadata. All information about a file can be retrieved by calling `stat`() system call. This function takes as input a file name and fills in the members of stat structure as shown below:

```
struct stat {
  dev_t         st_dev;          // Device ID of device containing file.
  ino_t         st_ino;          // File serial number.
  mode_t        st_mode;         // Mode of file
  nlink_t       st_nlink;        // Number of hard links to the file.
  uid_t         st_uid;          // User ID of file.
  gid_t         st_gid;          // Group ID of file.
  dev_t         st_rdev;         // Device ID (if file is character or block special).
  off_t         st_size;         // For regular files, the file size in bytes.
  time_t        st_atime;        // Time of last access.
  time_t        st_mtime;        // Time of last data modification.
  time_t        st_ctime;        // Time of last status change.
  blksize_t     st_blksize       // Block size for file system
  blkcnt_t      st_blocks        // Number of blocks allocated for this object.
}
```

See manpage of `stat`() function for details. There are also macros defined to check file type using `st_mode` field. For example, `S_ISREG`() macro will determine whether the file is a regular file. You can also check its access permission bits using appropriate macros.

Exercise. Implement the **ls** command with **-l** option (**myls.c**). On each line your program should print the file type, permissions, the number of hard links, owner name, group name, size in bytes, the modification time, and the file name. You do not have to output total number of blocks allocated for the directory as ls -l does. Use `getpwuid`() to obtain the owner name and `getgrgid`() the group name. To convert a calendar time in stat structure, use `ctime`() by passing a pointer to the time field.