CS 4352
Project 4
Due – 11/22/14 at 11:59 pm (<u>No late submission allowed</u>)

<u>The purpose of this project is to study UNIX file system calls by developing the "ls" utility</u>.

Create a new folder "proj4" in your 'cs4352' and implement the following stages. The purpose of each stage is summarized below.

Stage1: Familiarize with the UNIX file attributes and relevant system calls by conducting experiments with a given program. Implement a program to perform "ls –l file".

Stage2: Familiarize with the UNIX directory system calls. Develop a traverse procedure to visit all entries of the source directory. It should perform "ls –l directory". This directory has no sub-directories.

Stage3: Expand stage2 traversal method. Stage3 should visit entries of all descendant sub-directories as well. Effectively, it should perform "ls –l –R directory".

Detailed description of each stage is given next.

<u>Stage1</u>: Read the man page of system calls "open", stat, lstat, and "fstat". Note down the header files. Visit /usr/include/sys/stat.h and make a note of commonly used file attributes of UNIX files; See lines 75-90 and lines 413 and up of stat.h. Next read the manual page of "ls" command and make a note of all attributes printed by "ls –l".

Copy "~cs4352/tutorials/test_stat.c" to your "proj4". It shows use of fstat(). Compile and run it by passing a regular file name as the only argument. Edit this file to print the file size; it should help you to understand attribute retrieval. This program also shows use of perror( ), how to print actual error messages; see its man page. <u>Always print first the file name and then any error message.</u>

Next develop stage1.c. It should contain two procedures, main (argc, argv) and print_attr (BUF, Fname). Main() uses fstat() to acquire attributes of argv[1] in a structure, say BUF. It also extracts the last component of the path, say Fname, by scanning '/'. It then calls print_attr() passing BUF and Fname. This procedure extracts attributes from BUF and prints a line as printed by "ls –l" keeping the same format. To simplify the project, you do not have to print the date/time item, but print the remaining items in order.

You can retrieve user's uid and group gid from the BUF. These are numeric. To obtain the user name and the group name, use the library functions getpwuid() and getgrgid(). See their man pages.

<u>Stage2</u>: Read the man page of 'opendir', and 'readdir' library functions. Visit /usr/include/sys/dirent.h and note down the definition of DIR structure.

Copy stage1.c to stage2.c. Expand main() so that it tests the file type of argv[1]. If it is a directory type, it calls a procedure, say visitDir(source), passing argv[1] as source. An outline of visitDir() is as follows:

```
visitDir(path)
{
Print a blank line and a line giving path
Dir = opendir(path)
While ((entry = readdir(Dir)) not NULL)
{
  If entry → name is "." Or ".." continue;
  Full_path = malloc (strlen(path) + strlen(entry → name) + 2)
 Strcat path, "/", and entry → name in full_path.
  Use open() and fstat() and save attributes in Buf
  Print_attr (Buf, entry → name);
 Close()
Free (Full_path)
}
Closedir(Dir)
}
```

Use the code given in the man page of ‘readdir’ as the base of this procedure. Note that visitDir() should skip “.” and “..” entries, and not those that begin with dot such as “.XYZ”.

Testing: Run your program with “~cs4352/tutorials/sched” as the source. Compare your output with “ls –l”. You should have exactly the same number of entries, but possibly in a different order.

Stage3: Copy stage2.c to stage3.c. Code a procedure DepthList(path). Its logic is as follows:

```
DepthList(path)
{
vistDir(path)
Dir = opendir(path)
While ((entry = readdir(Dir)) not NULL)
{
  If entry → name is “.” Or “..” continue;
  Full_path = malloc (strlen(path) + strlen(entry → name) + 2)
 Strcat path, “/”, and entry → name in full_path.
   If (lstat() on full_path <0) {free(full_path); print error message; continue;}
  Get ftype
  If ftype is directory { DepthList(full_path);}
  Else if ftype is regular file {print_attr(BUF, filename);  continue;
  Else printf an error message in stderr (do not exit)
Free (Full_path)
}
Closedir(Dir)
}
```

You may assume that there are only two types of entries: directory, and regular file.

Testing: Go to “~cs4352/projects/proj4”; copy proj4.tar to your proj4 folder and un-tar it (tar –xvf proj4.tar). You will have a replica of my proj4 folder. It contains some test directories such as stage2_Test1 and stage3_Test2, etc. Compare your output with “ls –l –R”. The order of directories visited should be the same. The error messages, if any, should also be the same.

Stage1, stage2, and stage3 will account for 40%, 40%, and 20%.

Remove all files except stage1.c, stage2.c, and stage3.c from your proj4 folder, before submitting the project.

Extra credit – a) Print directory entries in order so that it matches with the one produced by “ls”. B) Compute and print data and time of last modification as printed by “ls”. (Extra 30%)