Simon Fraser University
School of Computing Science
CMPT 300: Assignment #4

The UNIX ls Command

Reminder: The rules of academic conduct apply as described in the course outline. Written answers are to be done alone. All coding is to be done alone. We will be using electronic software to compare all assignments handed in by the class to each other, as well as to assignments handed in for previous terms that this, or similar, assignment may have been given in.

Be sure that this assignment compiles on the Linux computers in the CSIL lab in using the gcc compiler. You can access the Linux server remotely as detailed on the course discussion forum.

What to Hand In:

1. Every source file (**UnixLs.c**) you completed for this assignment.
2. A Makefile which will compile your program creating an executable named **UnixLs**.
3. Sample output showing that your **ls** program works.

The ls command

The Unix/Linux ls command is used to list files and directories in the file system. For this assignment you will be implementing a version of ls which supports only a limited set of options.

Objectives:
1. Understand what an *inode* is, and understand its role in the Unix file system.
2. Know how to use system calls to navigate the Unix file system from a program.
3. Understand how the data in files (and directories) are organized and stored in Unix.

Coding:

- Write ***UnixLs***, a program that emulates ***ls***, in **UnixLs.c**
- Implement **only** the following options:
    1. **-i**
    2. **-l**
    3. **-R**

Testing:

- To test the program simply run the program and specify the options the same way you would for the ***ls*** command. Be sure to test every option you have implemented and all the permutations of the various options (e.g. *-iRl*, *-Ri*, *-li*, *-iR*, *-lR*, etc.) Of course, don't forget the most obvious test of all, no options. You can determine if your output is correct by observing what ***ls*** with similar options produces.

The output of your program should mimic the standard ***ls*** command with the following exceptions:

1. You should require that any options (-l -R -lR -i) to your command come before any directory names. The standard ***ls*** command allows options to appear between and after directory arguments.
2. File listing order does not have to be identical to ***ls***. You may list the files in whatever order is convenient for you rather than sorted alphabetically as ***ls*** normally does.
3. To simplify the printing of dates, you are to use the format

   mmm dd yyyy hh:mm

   regardless of the date. An example date would look like:

   Oct 2 2009 13:32

   The real ls command omits the year if the date is less than a year in the past and replaces the minutes and seconds with the year if the date is more than a year in the past.
4. When performing a long listing (-l) there is no need to print the line that begins with "total".
5. You should print the files one per line (as ***ls -1*** does, rather than attempting to format the list of files into multiple columns.

Here are a few other test cases you should try:

% ls -R ~/
% ls -l
% ls -Ri -l .. ~
% ls -liR . .. ~

Then replace "***ls***" with "***UnixLs***". The output should be very similar to the original *ls* command; however the order with which you list the entries is allowed to be different. Also note that you may have an **alias** for ls that provides some arguments all the time. To see if you have an alias for ls, type `alias ls`, and to make the alias go away (for the duration of the life of your current shell) type `unalias ls`.

Hints

- Initially concentrate on getting the listing for a directory correct and do not concentrate on recursing through directories.
- Focus on extracting the proper information and just printing it. Once you have that all done work on making the format match the ls command.
- Consult the the provided file infodemo.c to see what calls to use to get actual group and user names.
- Consult the provided tutorials in the “Assignments” section of the course webpage for information on UNIX file system structure.
- Don't forget to test your code on directories with symbolic links in them.