CS39002: Operating Systems Lab
Spring 2012

Assignment 1
Due: January 12, 2012, 1 pm

1. Write a C program that takes a file name as a command line parameter and sorts a set of integers stored in the file (use any sorting method). You can assume that the file will always be there in the current directory and that it will always contain a set of integers (maximum no. of integers is 1000). The sorted output is written to the display and the input file is left unchanged. Compile the C file into an executable named "*sort1*". Name the C file *sort1.c*.

   Now write a C program (xsort.c) that implements a command called "xsort" that you will invoke from the shell prompt. The syntax of the command is "xsort <filename>". When you type the command, the command opens a new xterm window, and then sorts the integers stored in the file <filename> using the program "sort1". Look up the man pages for *xterm*, *fork* and the different variations of *exec** calls (such as *execv*, *execve*, *execlp* etc.) to do this assignment.

   Submit the C files *sort1.c* and *xsort.c*.

2. In this assignment we will start writing a command interpreter (Shell). We will complete it in Assignment 2.

   The shell will give a prompt for the user to type in a command (from a set of commands), take the command, execute it, and then give the prompt back for the next command (i.e., actually give the functionality of a shell). Your program for the first assignment should do the following:

   - Give a prompt "myshell>" for the user to type in a command
   - Implement the following builtin commands:
     - **cd <dir>** : changes the directory to "dir"
     - **pwd** : prints the current directory
     - **mkdir <dir>** : creates a directory called "dir"
     - **rmdir <dir>** : removes the directory called "dir"
     - **ls :** lists the files in the current directory. It should support both ls without any option and with the option “-l”
     - **exit** : exits the shell

   The commands are the same as the corresponding Linux commands by the same name. Do "man" to see the descriptions. You can use the standard system calls **chdir**, **getcwd**, **mkdir, rmdir, readdir** etc. to implement the calls (standard C library functions are available for these; look them up).

   These commands are called *builtin* commands since your shell program will have a function corresponding to each of these commands to execute

them; no new process will be created to execute them. (Note that all these commands are not builtin commands in the bash shell, but we will make them so in our shell).

- Any other command typed at the prompt should be executed as if it is the name of an executable file. For example, typing "a.out" should execute the file *a.out*. The file can be in the current directory or in any of the directories specified by the PATH environment variable (use getenv to get the value of PATH). The file should be executed after creating a new process and then exec'ing the file onto it. The parent process should waits for the file to finish execution and then go on to read the next command from the user.

To run your shell, write another C program that will create a child process and call an appropriate form of exec to run the program above from the linux shell. The parent process simply waits for the child to finish (execute the "exit" command), after which it also exits.

Name the C file for the shell *shell.c*. Name the C program above that runs your shell *run.c*. Submit both the C files.

Please tar all the C files for both Problems 1 and 2 in a single tar file and follw instructions on the submission site to submit.