IIT Bhubaneswar
School of Electrical Sciences
Operating Systems Lab (0-0-3)
Spring 2017
Lab Schedule: Thu (8:00-10:55)

Instructor: Debi Prosad Dogra (dpdogra@iitbbs.ac.in)
Teaching Assistant: Santosh K. K. (sk47@iitbbs.ac.in)

Laboratory 2 (**Submission Deadline: 12/02/2017 midnight**)

Points: 3**0 (including the practice)**

Objectives: The object of this assignment is to gain experience with some advanced programming techniques including file descriptors, signals and pipes. To do this, you will be writing your own command shell - much like csh, bsh or the DOS command shell.

Practice during Laboratory Hours:
Understand various commands and functions: getcwd(), df, which, redirect signs (<, >), pipes (|) jobs, cd, exec, history, kill, etc.
Hint for assignment: Carefully follow the pseudo-code of a basic shell interface. You can use this as the base logic to implement your shell.

```
int  main (int argc, char **argv)
{
        while (1)
        {
                int childPid;
                char * cmdLine;
                printPrompt(); //This function will print the prompt.
                cmdLine= readCommandLine(); //To read the command line given by the user.
                cmd = parseCommand(cmdLine); // Understand and parse the command
                record_command_history();// Similar feature is available in GNU
                // Do the job now
                if ( isBuiltInCommand(cmd))
                {
                        executeBuiltInCommand(cmd);
                }
                else
                {
                        childPid = fork();
                        if (childPid == 0)
                        {
                                executeCommand(cmd); //calls execvp
                        }
                        else
                        {
                                if (isBackgroundJob(cmd))
```

```
                        {
                                Record_in_list_of_background_jobs();
                        }
                        else
                        {
                                waitpid(childPid);
                        }
                }
        }
    }
}
```

Practice and Take Home Assignment for Submission:

Design a shell that supports the following features:

- The prompt you print should indicate your current working directory. It may also indicate other things like machine name or username or any other information you would like. Try getcwd(char * buf, size_t size).
- You should allow the user to specify commands either by relative or absolute pathnames. To read in the command line, you may want to consider the readline() function from the GNU **readline** library as it supports user editing of the command line.
- You should be able to redirect STDIN and STDOUT for the new processes by using < and >. For example, **xyz < infile > outfile** would create a new process to run **xyz** and assign STDIN for the new process to **infile** and STDOUT for the new process to **outfile**. In many real shells, it gets much more complicated than this (e.g. >> to append, > to overwrite, >& redirect STDERR and STDOUT, etc.)!
- You should be able to place commands in the background with **&** at the end of the command line. You do not need to support moving processes between the foreground and the background (ex. bg and fg). You also do not need to support putting built-in commands in the background.
- You should maintain a history of commands previously issued. The number of previous commands recorded can be a compile time constant, say 10. This is a FIFO list, you should start numbering them with 1 and then when you exhaust your buffer you should discard the lowest number, but keep incrementing the number of the next item. For example, if storing 10 commands, when the 11th is issued, you would be recording commands 2 through 11.
- A user should be able to repeat a previously issued command by typing ! **number**, where number indicates which command to repeat. !-1 would mean to repeat the last command. !1 would mean repeat the command numbered 1 in the list of command returned by history.
- A built-in command is one for which no new process is created, but instead the functionality is built directly into the shell itself. You should support the following built-in commands: **jobs**, **cd**, **history**, **exit** and **kill**. **Jobs** provide a numbered list of processes currently executing in the background. **cd** should change the working directory. **history** should print the list of previously executed commands. The list of commands should include be

numbered such that the numbers can be used with **!** to indicate a command to repeat. **exit** should terminate your shell process, **kill %num** should terminate the process numbered, **num** in the list of background processes returned by jobs (by sending it a **SIGKILL** signal).

- If the user chooses to **exit** while there are background processes, notify the user that these background processes exist, do not exit and return to the command prompt. The user must **kill** the background processes before exiting.
- You may assume that each item in the command string is separated on either side by at least on space (e.g. prog > outfile rather than prog>outfile).

Submission Guidelines: Source files, Readme.txt, and other necessary files should be zipped together in a file named "**RollNo_Lab3.zip**" and send it to the TA with the following subject line: "**OS_lab3_Assignment**".

Additional Reading: Read through **Chapter 2** in the **xv6** textbook. Start getting conversant with xv6 OS.