ASSIGNMENT 4
SHELL IMPLEMENTATION

Objectives:

1. Understand the working of command line interface in Unix-like environment.
2. Understand the process forking mechanism.

You need to take the input from the command line in an infinite loop till an “exit” is entered and the corresponding output should be printed to stdout.

Required functionalities:

1. Execute all the commands (ls, clear, vi etc)
2. Shell built-ins (cd, pwd, export)
3. Print environment variables and text using echo
4. I/O redirection (<, >)
5. Pipes “|” (multiple)
6. Background and foreground functionality : &, fg
7. Support for history and '!' operator
8. Handle Interrupt Signal: On pressing "Ctrl+C", the command that is running currently should be terminated, your program should not terminate.

Note: PG1 VLSI students need to implement only 1, 2, 5 and 7, 8.

Important functions and system calls:

- int chdir(const char *path)
- int execvp(const char *file, char *const argv[])
- void exit(int status)
- pid_t fork(void)
- char *getcwd(char *buf, size_t size)
- char getenv(const char *name)
- void perror(const char *string)
- int setenv(const char *name, const char *value, int overwrite)
- sig_t signal(int sig, sig_t func)
- pid_t wait(int *status)
- pid_t waitpid(pid_t wpid, int *status, int options)
- sighandler_t signal(int signum, sighandler_t handler);
- int dup2(int oldfd, int newfd);
- pid_t setsid(void)

Examples:

1. Commands and Shell Built-ins

```
bash prompt:~$ ./a.out
My_Shell:/home/user$ ls
shell.cpp main.cpp a.out

My_Shell:/home/user$ fdresdsad
My_Shell: fdresdsad: No command found

My_Shell:/home/user$ cd OS
My_Shell:/home/user/OS$

My_Shell:/home/user/OS$ cd ..
My_Shell:/home/user/$ cd /home/user/work/temp
My_Shell:/home/user/work/temp$

My_Shell:/home/user/OS$ echo $PWD
/home/user/OS
```

2. I/O redirection and pipes

```
My_Shell:/home/user/OS$ echo Hello, this is a line > out
My_Shell:/home/user/OS$ cat out
Hello, this is a line

My_Shell:/home/user/OS$ cat out| wc
        1        5        22

My_Shell:/home/user/OS$  ls -l | grep "out" | wc | wc | grep "1" | wc
        1        3        24

My_Shell:/home/user/OS$  ls -R my_folder1/ | grep "abc" >out
```

3. Running commands in background & foreground

```
My_Shell:/home/user/OS$: sleep 10 &
[1] 4504                              #Running in background with pid 4504
My_Shell:/home/user/OS$: fg #4504
sleep 10                              #This process is running in foreground now

My_Shell:/home/user/OS$: wc &
[1] 4550                              #Running in background with pid 4504
My_Shell:/home/user/OS$: fg #4550
wc                                    #Running wc in foreground now
hello
world                                 #Press Ctrl+D now
        2        2        12
```

Some details regarding implementation of bg/fg -

- For background process use following format:
  sleep 10 & . Running this command, should return you the process id(the same way as it works in shell).
- For foreground process, use fg #PID. For running a process in foreground, you need to check if the process is still running and if that process is running, use waitpid system call for it to complete.
- You can observe this in shell also, by trying the above commands.
- For implementing a process to run in background, you can use system call "setsid()".

4. History

```
My_Shell:/home/user/OS$ history
  . . .
  43  man bash
  44  man fc
  45  man bash
  46  fc -l -10
  47  history
  48  ls -a
  49  vim .bash_history
  50  history
  51  man history
  52  history 10
  53  history

My_Shell:/home/user/OS$ history 5
```

```
   50  history
   51  man history
   52  history 10
   53  history
   54  history 5

My_Shell:/home/user/OS$ history | grep cd
   33  cd Pictures/
   37  cd ..
   39  cd Desktop/
   61  cd /usr/bin/
   68  cd
   83  cd /etc/
   86  cd resolvconf/
   90  cd resolv.conf.d/

My_Shell:/home/user/OS$ !51              # displays man page of history for our session

My_Shell:/home/user/OS$ vim file.cpp
My_Shell:/home/user/OS$ echo “hello”
My_Shell:/home/user/OS$  !v              #Should execute “vim file.cpp”
My_Shell:/home/user/OS$  !!              #Should execute “vim file.cpp”

My_Shell:/home/user/OS$ ls /usr/share/doc/manpages
My_Shell:/home/user/OS$ echo hello
My_Shell:/home/user/OS$ !-2              # lists the contents again
```

4. Exit

```
My_Shell:/home/user/OS$ exit
Bye...
bash prompt:~$
```

Deadline: 10:00 PM, 13th Oct 2015
Upload Instructions:
Upload Format: .tar.gz

1. Create a folder named your roll number.
2. Create a "README" file containing the details of functionality implemented and place your '.c' or '.cpp' files in the folder.
3. Create a tar.gz named "Rollno_Assignment4.tar.gz" and upload it.

Example:

```
20150xxxx/
        --file1.cpp
        --file2.cpp
        --file3.cpp
        ....
        --filen.cpp
        --makefile (optional, only if you are using one)
        --README
```

Create 20150xxxx_Assignment4.tar.gz

NOTE

1. Use of system() will fetch you ZERO marks.
2. Standard Template Library of C++ (STL) is allowed for this assignment.
3. You CANNOT use any data structure or external text file to store the intermediate result in pipes.
4. Strictly follow specified Upload Format.
5. If you have any confusion regarding upload format kindly clarify on courses portal.
6. Don't include any executable file (a.out) or any swap files (prog.cpp~) .
7. ***Make "man" your best friend.***

--