Computer Science 360 Spring 2006
Assignment 1 – Shell
Due January 31st 2006

Introduction

In this assignment you will implement a simple shell program similar to the Unix shell bash or the command prompt in Windows.

The shell will support execution of programs, the ability to change directories and background execution.

You may implement your solution in C or C++. You solution must run on the machine linux.csc.uvic.ca.

Be sure to study the man pages for the various functions suggested in this assignment. The functions are in section 2 of the man pages, so you should type (for example):

```
$ man 2 waitpid
```

Requirements

Part I – Basic execution (5 marks)

Using `fork()` and `execvp()` implement the ability for the user to execute arbitrary commands using your shell.

For example, if the user types:

```
shell> ls -l /usr/bin
```

Your shell should execute the `ls` command with the parameters `-l` and `/usr/bin`, which should result in the contents of the directory `/usr/bin` being displayed on the screen.

Part II – Changing Directories (5 marks)

Using the functions `getcwd()` and `chdir()` add functionality so that users can change directories using the command `cd`, and print the current working directory using the command `pwd`.

The `cd` command should take one argument that is the directory to change into. The special parameter `..` indicates that the current directory should move "up" one directory.

That is, if the current directory is `/home/jason/foo` and the user types `cd ..` the new directory will be `/home/jason`

The `pwd` command takes no parameters.

Part III – Background Execution (5 marks)

Many shells allow programs to be started in the background – that is, the program is run but the shell continues to accept input from the user.

You will implement a simplified version of background processing that supports a fixed number (in this case 5) of processes executing in the background.

If the user types: `bg cat foo.txt` your shell will start the command `cat` with the argument `foo.txt` in the background. That is, the program will execute and the shell will also continue to execute.

The command `bglist` will display a listing of all the commands currently executing in the background, similar to:

```
0: /home/jason/a1/foo
1: /home/jason/a1/foo
Total Background jobs: 2
```

In this case, there are 2 background jobs, both running the program `foo`.

The command `bgkill 1` will send the TERM signal to job 1 to terminate that job. See the man page for the `kill()` system call for details.

Your shell must indicate to the user when background jobs have terminated. Read the man page for the `waitpid()` system call. I suggest using the WNOHANG option.

The GNU Readline Library

We could use the simple `gets` to get input from the user, but instead we are going to use the GNU Readline Library. This library allows the user to edit their command line using the standard bash function keys. For example:

- the arrow keys move forwards and backwards
- control-A moves to the front of the line
- control-E moves to the end of the line
- control-K erases the rest of the line
- pressing the TAB key will perform filename completion

The sample code shows how to use the library. The library supports many more options (including command line history) but you are not responsible for those features.

Note: The Readline library allocates memory for the line entered by the user. It is up to the caller (you) to free the memory.

Compilation

You've been provided with a Makefile that builds the sample code. It takes care of linking in the GNU Readline library for you.

Submission

Submit a tar archive named assign1.tar of your assignment using the automated submission program at: http://www.csc.uvic.ca/~submit/index.cgi

You can create a tar archive of the current directory by typing:

```
tar cvf assign1.tar *
```

Please do not submit .o or executable files. Erase them before creating the tar archive.

Note

This assignment is to be done individually. You are encouraged to discuss the design of the solution with your classmates, but each student must implement their own assignment. The markers will submit your code to an automated plagiarism detection service.