# Assignment 1: Writing a Shell

## Program Behaviours

### Part A: Set 1 and Set 2

Note: programs bgSleep and interactive are provided on this assignment page as attached C files.

For Set 1 functions, you need to be able to run commands that are in /bin or /usr/bin (not considering the Set 3 builtin functions yet)

Example 1

```
> ls

bgSleep  bgSleep.c
```

Example 2

```
> ls -l

total 9

-rwxr-xr-x 1 dastacey faculty 16712 Jan 31 14:56 bgSleep

-rw-r--r-- 1 dastacey faculty   684 Jan 31 14:56 bgSleep.c
```

Make sure that you can handle errors – misspelling basic commands or the name of executable files or improper arguments

Example 3

```
> ks
```

```
-bash: ks: command not found
```

Example 4

```
$ ./execNotThere
```

```
-myShell: ./execNotThere: No such file or directory
```

Example 5

```
> ls -0
```

```
ls: invalid option -- '0'
```

```
Try 'ls --help' for more information.
```

There is a program named bgSleep that takes 1 argument (number of seconds) and sleeps for that number of seconds

Example 6

```
> ./bgSleep 2
```

Backgrounded programs look like the following – they show the pid and they immediately allow another command to be entered and when the backgrounded program finishes, your shell should let everyone know:

```
> ./bgSleep 5 &

[1] 14607

> ps

  PID TTY          TIME CMD

14607 pts/53   00:00:00 bgSleep

24004 pts/53   00:00:00 bash

24336 pts/53   00:00:00 ps

[1]+  Done                    ./bgSleep 5
```

If you can print out line [1] 14607 only you will get part marks

If you can also print out when the child finishes (whatever the message) – more marks

And if you can print out the correct format for the "Done" line – more marks

For full marks your shell will have to do the following for background processes:

```
> ./bgSleep 30 &

[1] 21370

> ./bgSleep 20 &

[2] 29940

> ./bgSleep 10 &
```

```
[3] 7780
> ps
  PID TTY          TIME CMD
 7780 pts/53   00:00:00 bgSleep
15822 pts/53   00:00:00 ps
21370 pts/53   00:00:00 bgSleep
24004 pts/53   00:00:00 bash
29940 pts/53   00:00:00 bgSleep
[3]+  Done                    ./bgSleep 10
> ps
  PID TTY          TIME CMD
11117 pts/53   00:00:00 ps
21370 pts/53   00:00:00 bgSleep
24004 pts/53   00:00:00 bash
29940 pts/53   00:00:00 bgSleep
[2]+  Done                    ./bgSleep 20
> ps
  PID TTY          TIME CMD
 1946 pts/53   00:00:00 ps
```

```
21370 pts/53   00:00:00 bgSleep
24004 pts/53   00:00:00 bash
[1]+  Done                    ./bgSleep 30
> ps
  PID TTY          TIME CMD
23957 pts/53   00:00:00 ps
24004 pts/53   00:00:00 bash
```

## Set 2 Functions – Redirection and Piping

Example 1: Redirecting stdout

```
> ls -l > fileList.txt
> more fileList.txt
total 10
-rwxr-xr-x 1 dastacey faculty 16712 Jan 31 14:56 bgSleep
-rw-r--r-- 1 dastacey faculty   684 Jan 31 14:56 bgSleep.c
-rw-r--r-- 1 dastacey faculty     0 Jan 31 15:00 fileList.txt
```

Example 2: simple pipe

```
> ls -l | wc
      3      20     124
```

Example 3: redirecting stdin: the program interactive has one argument – a number of questions/inputs that will be asked for and then it asks for input from the user and prints out what it was given:

```
> more inputFile.txt

How are you today?

What is the weather like where you are?

> ./interactive 2

What is your question? > my Question 1

Question 1: my Question 1

What is your question? > My Q 2

Question 2: My Q 2

Bye!

> ./interactive 2 < inputFile.txt

What is your question? > Question 1: How are you today?

What is your question? > Question 2: What is the weather like
where you are?

Bye!

> ./interactive 2 < inputFile.txt > outputFile

> more outputFile
```

```
What is your question? > Question 1: How are you today?

What is your question? > Question 2: What is the weather like
where you are?

Bye!

> more outputFile | sort

Bye!

What is your question? > Question 1: How are you today?

What is your question? > Question 2: What is the weather like
where you are?

> ./interactive 2 < inputFile.txt | sort

Bye!

What is your question? > Question 1: How are you today?

What is your question? > Question 2: What is the weather like
where you are?

> ./interactive 2 < inputFile.txt | sort > saveFile

> more saveFile

Bye!

What is your question? > Question 1: How are you today?

What is your question? > Question 2: What is the weather like
where you are?
```

```
> ./interactive 2 < inputFile.txt | sort > outputFile

> more outputFile

Bye!

What is your question? > Question 1: How are you today?

What is your question? > Question 2: What is the weather like where you are?
```

## Set 3 - The built-in Functions

If the files

- .CIS3110_profile
- .CIS3110_history

are not in the directory where your shell program is executed from then you must create these files.

An example .CIS3110_profile file:

```
export PATH=/usr/bin:/bin:$HOME

export HOME=/home/faculty/dastacey
```

You should implement the echo built-in command so that you can find out what PATH and HOME are set to.

```
> echo $PATH

/usr/bin:/bin:/home/faculty/dastacey

> echo $HOME
```

/home/faculty/dastacey

There are 3 versions of the history built-in command that the markers will look for:

```
> history
  1  ./interactive 2 < inputFile > outputFile | wc
  2  ./interactive 2 < inputFile > outputFile | sort
  3  ./interactive 2 < inputFile | sort > outputFile
  4  rm outputFile
  5  ./interactive 2 < inputFile | sort > outputFile
  6  more outputFile
   7  ls
   8  pwd
   9  ls
  10  cp inputFile inputFile.txt
  11  ls
 12  rm inputFile
 13  echo $PATH
 14  export PATH=$PATH:$HOME
 15  echo $PATH
 16  echo $HOME

 17  history

> history 3
 16  echo $HOME

 17  history

 18  history -3

> history -c
> history

   1  history
```

Directories

The cd (change directory) command is built-in: you have to keep track of what directory you are in.  This is needed if you are going to truly implement PATH and HOME environment variables.

You should be able to do the following:

```
> cd ..

> cd ~

> cd subdir1/subdir2

> cd /usr/bin/
```

After any of these cd commands pwd should show you the correct directory (full path name).

For full marks, change your prompt from > to the current directory (e.g. /home/faculty/dastacey/CIS3110> )

```
/home/faculty/dastacey/CIS3110> pwd

/home/faculty/dastacey/CIS3110

/home/faculty/dastacey/CIS3110> cd ..

/home/faculty/dastacey> cd CIS3110/Test

/home/faculty/dastacey/CIS3110/Test> cd ~

/home/faculty/dastacey>
```