

Due Date: April 17th, 2022, Sunday, 11:59 pm

*****No extension will be provided**

Objective:

Implement a simple shell or command line interpreter.

What is the SHELL ???

The Shell or Command Line Interpreter is the fundamental user interface provided by an operating system. In other words, the shell is just another program. For example, the Bash shell is an executable named `bash` that is usually located in the `/bin` directory. So, `/bin/bash`.

On most Linux systems a program called `bash` (which stands for Bourne Again SHell, an enhanced version of the original Unix shell program, `sh`, written by Steve Bourne) acts as the shell program. Besides `bash`, there are other shell programs that can be installed in a Linux system. These include: `ksh`, `tcsh` and `zsh`. (REF: http://linuxcommand.org/lc3_lts0010.php)

Problem Description:

You will write a simple shell in C called `uab_sh`. The requirements are as follows;

- 1) Your shell executable file should be named `uab_sh` and your shell source code should be mainly in `shell.c` (you are free to add additional source code files, make sure that your Makefile works/compiles and generates an executable named `uab_sh` in the same top level directory as the Makefile) **[30 points]**
- 2) The shell should run continuously, and display a prompt when waiting for input. When the shell is invoked by executing the executable `uab_sh`, it should provide a prompt `uab_sh >` and waits for any input from the keyboard. **[10 points]**
- 3) The `uab_sh` shell should read a line from `stdin` one at a time. This line should be parsed out into a command and all its arguments. In other words, tokenize it. You may assume that the only supported delimiter is the whitespace character. **[10 points]**
- 4) After parsing and lexing the command, your shell should execute it. The shell should create a new process corresponding to the input entered using `fork/exec`. Additionally, you can assume that the newline character denotes the end of the input. If a newline character is entered without any other text or followed by white space, the shell should just display the prompt on the next line. The new process created should be able to read keyboard input and display output to the terminal. You can assume that these programs are executed only in the foreground, *i.e.*, the shell program will wait for the child process to complete and when the child process completes successfully it will display the command prompt. **[10 points]**
- 5) A command can either be a reference to an executable OR a built-in shell command. Executing commands that are not shell built-ins is done by invoking `fork()` and then invoking `exec()`. You may NOT use the `system()` function, as it just invokes the `/bin/sh` shell to do all the work.

CS 332/532 Systems Programming
Bonus Homework

Here is a sample interaction with the shell (assuming *hello* and *fibonacci* are in one of the directories defined by the PATH environment variable):

```
uab_sh > hello
```

```
Hello World!
```

```
uab_sh > fibonacci 10
```

```
The first 10 values: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

```
uab_sh > fibonacci
```

```
How many elements you want to display: 7
```

```
The first 7 value: 0, 1, 1, 2, 3, 5, 8
```

*** Hello World and Fibonacci sequence calculator must be implemented *** [20 points]

6) Implement the following built-in commands: [20 points]

- a. **list** – list all the files in the current directory. The default directory is the directory where the shell program was invoked
- b. **cd <directory>** – change the current directory to the <directory>. The default directory would be the directory where the shell program was invoked
- c. **help** – display the internal commands and a short description on how to use them
- d. **quit** – quit the shell program

7) [CS532 Students Only] implement the following built-in command:

- a. **history** – display all the previous command entered into the shell program. The shell should save all commands entered in a log file called *uab_sh.log* and this file must be written in the directory where the shell program was invoked. The log command could just display the contents of this file. [10 points]

Additional Requirements & Submission Guideline:

- a) All code must be written in C.
- b) Your grade will be replaced with your lowest hw grade. This is an optional homework
- c) Use best software engineering practices to design and implement this homework.
- d) Use functions and organize these functions in multiple files if necessary.
- e) Use a **Makefile** to compile and build the multiple files.
- f) Start by implementing basic functionality such as parsing the input and then gradually add features.
- g) Document your code and include instructions for compiling and executing the program in the **README.md** file.
- h) Test your program and describe how you tested this program in the README.md file.

CS 332/532 Systems Programming

Bonus Homework

- i) Use the following template to create the README.md file (this is just a sample, feel free to include any other information that you think is useful):
 - a. Project title
 - b. Project Description
 - c. Author
 - d. Acknowledgment
 - e. Getting Started
 - i. Prerequisites/dependencies
 - ii. Instructions for building the software
 - f. Running the test
 - i. How to run test cases
 - ii. Sample test cases
 - g. Screenshots/Sample Session
 - h. Contact Information

You can find details on how to use markup to format git documents here:

<https://help.github.com/articles/basic-writing-and-formatting-syntax/>