Operating Systems (coe628)

Lab 3

Due the Week of February 5, 2024

Objectives

- Learn how to use fork(), execXXO and wait()
- Write a simple shell.

Getting started

- Create a C Project called **lab3.**
- Your first shell will print the prompt "Your command> " and then read one line of input that consists of a single word that names a command.
- The shell then forks and the child process should execute the command.
- The parent process should wait for the child to complete **unless** the line ended with an ampersand character (&).
- Assume that if the line ends with an '&' that it is **not** preceded by a space. (For example, "ls" and "ls&" are OK but not "ls &". (This assumption will make your life easier, not harder!)

Continuing on

Next, modify the "command line parser", so that the line can consist of 1 or more words optionally followed by the ampersand character.

Making the shell a loop

Finally, put the whole thing in a loop so that commands can be executed one after the other.

And Finally: Submit your lab

To submit your lab (on a departmental computer) do:

1- Change to the lab3 directory
```
cd coe628
```
2- Zip your source code files (*.c, *.h) into a file called Lab3.zip

3- Submit the zip file with the command:
```
submit coe628 lab3 Lab3.zip
```

Hints and suggestions

- Use getchar() not scanf (..) to read stdin to "collect" the input line.
- For example, look at the basic structure for "filter" applications (programs that read from "stdin" and write to "stdout").

```
int ch;
while ( (ch = getchar ()) != EOF) {
        putchar(toupper(ch) ) ;
}
```

- Use this pattern. **But**, look for a newline ('\ n ') instead of "end-of-file" (EOF).
- As each character is read, put it in the next position of an array of chars (or dynamically allocated memory). (You may assume that the maximum length of a line is 100 characters.)
- Once the line has been read, check if the last character read was '&'. If so, set a flag and use the flag to determine if the parent should wait for its child to die.
- Make sure that you end the line with a null ('\0') character.
- When the command is a single word such as "ls", "ps", "pwd", etc. You need only make your "arg pointer" be the address of the first character in the word.
- To parse a multi-word command, you can read in the whole line. Then replace each space with a null character and make the next "arg pointer" be the address of the character following the space. (Alternatively you could do this while reading the line.)
- Parsing the input should be done before forking. (The child inherits all of the parent's variable including the array of "arg pointers".
- I suggest you use the "execvp" version of "exec".