Operating Systems (coe628) Lab 3

Week of January 30, 2017

Duration 1 week

Objectives

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

Getting started

1. Create a C Project called **`lab3`**.
2. 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.
3. The shell then forks and the child process should execute the command.
4. The parent process should wait for the child to complete **unless** the line ended with an ampersand character (&).
5. 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 do:

1. Change to the lab3 directory.

2. `submit628`

That's all folks....

Hints and suggestions

- Use `getchar()` not `scanf(...)` to read stdin to "collect" the input line.
- For example, look at the basic structure we saw in Week 1 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 lenght 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 nul ('\0') character.
- When the command is a single word such as "ls", "ps", "pwd", etc. You need only make your "arg pointer" be the adddress 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 nul 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 variables including the array of "arg pointers".
- I suggest you use the "execvp" version of "exec".
- You can run a working version of lab3 on the Department's linux machines with the command `~/courses/coe612/bin/lab3`. Note that this command prints debugging information to *stderr*. Looking at the debugging output may give you additional hints.
- To see how lab3 shoudl work without the debugging information, use `~/courses/coe612/bin/lab3 2> /dev/null`

- Good luck!