Today we will be implementing the following portions of our Linux Shell program:

main()
prompt()

1. First in our VM’s lets make a ~/CPSC457/Shell directory and create the file 457shell.c inside it. Lets also create a Makefile to compile our program.

2. Now lets import this into a directory on our SVN repositories. First lets make a directory for all of the tutorial stuff.

   ```bash
   $ svn mkdir https://forge.cpsc.ucalgary.ca/svn/courses/c457/<your username>/Tutorials --username <your username> -m "Creating Tutorials Directory"
   
   Now lets import our Shell example into the repository.
   
   $ svn import Shell/ https://forge.cpsc.ucalgary.ca/svn/courses/c457/<your username>/Tutorials/Shell --username <your username> -m "Importing Linux Shell Example"
   
   Now lets create the directory ~/CPSC457/WorkingCopy and checkout our code from the repository into this directory. For this next part you should be at ~/CPSC457/WorkingCopy
   
   $ svn co https://forge.cpsc.ucalgary.ca/svn/courses/c457/<your username>/Tutorials/Shell --username <your username>
   
   3. Open your working copy of 457shell.c and lets define a few constants and a few empty functions that will serve as placeholders.

   ```c
   #include <stdio.h>
   #include <errno.h>

   #define PROMPT "cpsc 457 > 
   #define MAX_COMMAND_LENGTH 256
   #define MAX_ARGUMENT_COUNT 16

   #define TRUE 1
   #define FALSE 0

   static char command[MAX_COMMAND_LENGTH];
   ```
4. Lets commit our current code into the repository before we go any further.

    $ svn commit -m "Basic Code Outline"

Note: Feel free to commit any time you want to during these exercises. I won’t be prompting you to do it again except for one last reminder at the end.

5. Now lets fill in the main function. We need the main function to run a continuous cycle of prompting for user input, parsing the users input, and executing the command the user entered.
    Hint: Try thinking of while(TRUE) ... do.
    Hint 2: When executing the program for now we will stop it using $ ctrl-c

6. Now we need to have our function prompt() do what it is supposed to. That is, prompt the user for input and store the input somewhere. First lets declare some variables we will need and initialize our arrays to default values. Then we will write the PROMPT to stdout. As a thought exercise what is stdin, stdout and stderr?

    int prompt(){
        //Declare Variables we might need
        unsigned char ch=0;
        unsigned char finish = 0;
        unsigned int size = 0;
//Initialize our arrays
memset(command,0x00, MAX_COMMAND_LENGTH);
while (argument_count > 0)
    argument_list[--argument_count] = NULL;

//Write our Prompt to Stdout
if (write(1, PROMPT, sizeof(PROMPT)) != sizeof(PROMPT) )
    return -EIO;
}

You should make sure you understand what is happening in the above code. If you don't know what a function does try the man pages or google.

The following code snippet will read from stdin one byte. If it cannot read from stdin it returns an I/O error. To get user input read from stdin until the user is done entering a command. i.e. provides a new line which C recognizes as \n.
    if (read(0,&ch,1) < 0)
        return -EIO;

Hint: Once again think about a while(finish)... do loop here. Remember finish is a local variable.
Hint 2: Store what the user enters somewhere. I suggest the command array. Make sure you don't exceed the limit of the command array. i.e. Don't let it go past MAX_COMMAND_LENGTH. Using the local variable size here might help.

7. Commit your code back to the repository.

We will continue creating our Linux Shell next time by implementing parse_command(), execute_command(), execute_cd() and execute_pwd().

The code from the exercises today was written by Daniel de Castro and used with permission.