CSCI 3751 Fundamentals of Unix
Programming Assignment #1 (50 points)

Objectives:

- Gain hands-on experience with the ***vi*** editor and experiment it for this lab assignment. For this assignment, use ***vi*** editor to edit your program as much as possible.
- Introduction to 'C' programming concepts, including:
  - Utilization of pointers, dynamic memory allocation with `malloc()`, and memory deallocation with `free()`.
  - Exploration of string manipulation functions such as `strchr()`, `strcpy()`, and `strtok()`.
  - Working with arrays of character pointers (`char *[]`).
- Acquire a basic understanding of the `make` tool and the creation and usage of makefiles.
- Learn about the Unix shell process model through the use of `fork()`, `exec()`, and `wait()` system calls.
- Utilize the GNU Debugger (`gdb`) as needed for debugging your programs

References:

- Advanced Programming in the UNIX Environment (APUE) textbook.

Requirements:

1. Create a dedicated subdirectory within your **$HOME** directory for this lab assignment.

2. Using the vi editor, manually type in the entire code from Figure 1.10 of the APUE textbook into a file named ***myShell2.c*** in the PA1 directory.

   - For this assignment, prioritize the use of the ***vi*** editor.

   - Configure ***vi*** to your preference, setting the tab stop between 2 to 4 spaces. (e.g., set tabstop=4)

   - Enable "show match" (`:set sm`) and other convenient settings in your **~/.vimrc** file. (**_vimrc** in MS Windows versions)

   - Compare your typed code with the original **shell2.c** untarred from HW 1 using the **diff** tool. (`diff ./myShell2.c $APUE_HOME/intro/shell2.c`)

3. Copy the **Makefile** from **apue.3e/intro/** into your assignment directory. This should be the only direct file copy from the source directory. Modify this **Makefile** to compile your ***myShell2.c*** program along with the necessary header files and libraries from the **apue.3e** directory.

4. Modify ***my*Shell2.c** to meet the following specifications:

    - Replace **execlp()** with **execvp()** for executing child processes, enabling the handling of command options and arguments.

    - Enhance the shell program to support command options and arguments, as the original program does not.

        - The original "`shell2.c`" program in Fig 1-10 of the APUE textbook page 19 (Section 1.9 Signals) supports command name only and it doesn't handle any options or arguments so '`ls -l`' command would give you "couldn't execute: …" error message. But your '`myShell2.c`" would be a bit smarter - it would support most of options and arguments, not just command name alone.

    - Ensure any dynamically allocated memory is properly freed upon program termination.

Expected Output Examples:

```
[namsu@csci-gnode-03 lab1]$ ./myShell2
% ls
   Makefile  myShell2  myShell2.c  shell2.org.c
% ls -lrt
total 28
-rw-r--r--. 1 namsu namsu   268 Feb  9 18:48 Makefile
-rwxrwxr-x. 1 namsu namsu 14120 Feb 10 00:19 myShell2
-rw-r--r--. 1 namsu namsu  2109 Feb 10 00:42 myShell2.c
% ls -l myShell2.c myShell2
-rwxrwxr-x. 1 namsu namsu 14064 Feb 12 08:38 myShell2
-rw-r--r--. 1 namsu namsu  2121 Feb 12 08:38 myShell2.c
% ls -lr ../
   total 168
   -rw-rw-r--.  1 namsu namsu 92830 Feb  7 21:23
   src.3e.tar.gz
   drwxrwxr-x.  3 namsu namsu  4096 Feb 14 01:11 lab1
   drwxrwxr-x.  2 namsu namsu  4096 Feb 14 07:44 hw2
% ls -lrt > aaa
   ls: cannot access >: No such file or directory
   ls: cannot access aaa: No such file or directory
 % ls -l she*
   ls: cannot access she*: No such file or directory
% find . -name she* -type f
```

```
./shell2.c
./shell2

% ps -ue
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT
START   TIME
namsu     23265  0.0  0.0 124120  2300 pts/1    Ss   08:10
0:00 -bash LANG=en_U
```

Hints

- For command option handling, define an array of C strings (**cmd_strs**). Consider the appropriate data structure for an array of C strings.
- Commands must be parsed into an array of C strings. For example, the command "`find . -name she* -type f`" becomes an array where each space-separated segment is an individual string.

cmd_strs[]	
0	find
1	.
2	-name
3	she*
4	-type
5	f

- Use `strtok()` function to parse the string with a proper set of delimiters. You may google '`strtok(3)`' or talk to `chatGPT` for the examples of `strtok()` function calls.
- Allocate memory for the `cmd_strs[]` array dynamically with `malloc()` and ensure all dynamically allocated memory is freed before program exit.
- The `strchr()` function can be used to find specific characters within a string for advanced parsing or handling.

Deliverable:

1. Source code files.
2. The modified **Makefile**.
3. An output file demonstrating at least the test cases provided in the instructions.
4. A reflective write-up on your experience with the `vi` editor, including the effort invested in learning and using it effectively.

Extra Credit (10 points):

- Discuss why file name expansions (*****, **?**, etc.), pipes (|), and redirections (>, <, >>) fail in the provided examples, while **.**, **..**, and file expansions in commands like **find** work.
- Enhance "myShell2" (creating "myShell2++") to more closely mimic the bash shell by using **bash -c**. This includes handling text within double quotes as a single argument to the **execvp()** system call.
- This extension requires understanding how bash interprets and executes commands, particularly with respect to special characters and quotes.

Example for Extra Credit:

```
% bash -c "ls -l > aaa"
```

In the example above,
‘bash’ is the first element,
‘-c’ is the second element,
‘ls -l > aaa’ is the third element in the array passed to `execvp()` system call

```
% ls
     aaa Makefile myShell2++ shell2.c

% cat aaa
     total 52
     -rw-rw-r--. 1 namsu namsu     0 Feb 14 08:03 aaa
     -rw-r--r--. 1 namsu namsu   268 Feb  9 18:48 Makefile
     -rwxrwxr-x. 1 namsu namsu 14168 Feb 14 08:00 myShell2++
```

Example)

```
  % bash -c "ls -l > aaa"

  In the example above,
     ‘bash’ is the first element,
     ‘-c’ is the second element,
     ‘ls -l > aaa’ is the third element in the array passed to
     execvp() system call

  % ls
  aaa Makefile myShell2++ shell2.c
  % cat aaa
  total 52
  -rw-rw-r--. 1 namsu namsu     0 Feb 14 08:03 aaa
  -rw-r--r--. 1 namsu namsu   268 Feb  9 18:48 Makefile
    -rwxrwxr-x. 1 namsu namsu 14168 Feb 14 08:00 myShell2++
```

[…]