Operating Systems Lab (CS 470):

Lab 1: Create a mini shell using C/C++ programming language to be executed in Linux/UNIX.

Overview

The OS command interpreter is the program used by the user to interact with the operating system in order to launch and control different programs and tools. On UNIX/Linux like systems the command interpreter is usually called *shell*. It is a user-level program that gives people a command-line interface to launch, suspend, and kill (terminate) other programs. The `sh`, `ksh`, `csh`, `tcsh`, and `bash` are all examples of such UNIX/Linux shells used by current operating systems.

The example below illustrates the use of the cwushell, the shell that you will create. The example shows a prompt `cwushell>` and the user's next command: `cat Prog.c`. This command displays the file *Prog.c* on the terminal using the UNIX `cat` command.

```
cwushell> cat Prog.c
```

Every shell is structured as a loop that includes the following:

1. print a prompt
2. read a line of input from the user
3. parse the line into the program/command name, and an array of parameters
4. once the command (i.e. the launched program) finishes, the shell repeats the loop by jumping to 1.

Although most of the commands people type on the prompt are the name of other UNIX/Linux programs (such as `ls` or `cat`), shells recognize some special commands (called internal commands) which are not program names. For example, the `exit` command terminates the shell, and the `cd` command changes the current working directory. Shells directly make system calls to execute these commands, instead of forking child processes to handle them.

Instructions

Write a mini shell program (in C/C++) called `cwushell` under Linux/MacOS. The shell has the following features:

- It recognizes the following internal commands:

    1. `exit [n]` -- terminates the shell, either by calling the `exit`() standard library routine or causing a return from the shell's `main`(). If an argument (n) is given, it should be the exit value of the shell's execution. Otherwise, the exit value should be the value returned by the last executed command (or 0 if no commands were executed.)
    2. `systeminformation –switch` will print on the screen different system related information based on the switch. The different switches to be implemented are:

        1) -p – the number of processors available in the system, 2) -m – the maximum number of open files per process, 3) -o – will print the processor type

3. `memoryinformation -switch --` will print on the screen different memory related information based on the switch. The different switches to be implemented are:

   1) -s – will print the total swap space in bytes, 2) -u –will print the total memory in bytes and 3) -S – will print the total shared memory in bytes available in the system.

4. all other existing shell commands (internal and external commands e.g. ls, cat, pwd, etc.) should also be executed by the cwushell.

Important Note: Beside the commands mentioned earlier, the shell should be able to remember the last *n* commands, even if we exit/stop the shell and we restart it. With the up and down key we should be able to "navigate" trough the older commands. For details check the usage of the up and down keys in a Linux/UNIX shell. The parameter n is to be provided as a command line argument when launching the cwushell. If no such parameter is provided, the default value is to be set to 5.

Notes

- Error handling should be considered (see erroneous commands, non-existing switches, etc.).
- For the cwushell a help file should be provided, which should be invoked from the shell using the manual command.
- For each command a specific help (user manual) should be invoked if called without a parameter or using –help or --h switches. This will help the user to see how to use the different commands in the cwushell. See for details the Unix/Linux manual pages. The help should be structured/formatted in the same (or at least similar) way as for the man pages.
- The shell has to receive commands until explicitly the exit command is invoked. Please study the behavior or current shells such as bash/tcsh/ksh.

Rubric

Task	**Points**
error handling, help mechanism (see manual)	1 + 1
systeminformation	3
memoryinformation	3
exit, prompt	0.5 + 0.5
command line history	3