## Programming Assignment 2

### Objective

The objective of this assignment consists of designing a C/C++ program to serve as a shell interface that accepts user commands and then executes each command in a separate process. This project can be completed on any Linux, UNIX, or Mac OS X system.

### Assignment: Using the Fork System Call

Write a simple shell interface program that gives the user a prompt, after which the next command is entered. The example below illustrates the prompt osh> and the user’s next command: cat prog.c. (This command displays the file prog.c on the terminal using the UNIX cat command.)

osh> cat prog.c

#### Simple Shell Program Implementation

The simple shell program (s***shell.c***) is a simple text-based program that takes user input from the parent then sends that input to the child process for execution. The separate child process is created using the fork() system call, and the user’s command is executed using one of the system calls in the exec()family. A C/C++ program that provides the general operations of a command-line shell is supplied below. The main() function presents the prompt osh-> and outlines the steps to be taken after input from the user has been read. The main() function continually loops as long as should\_run equals 1; when the user enters exit at the prompt, your program will set should\_run to 0 and terminate.

The objective is to modify the main() function in the code provided below so that a child process is forked and executes the command specified by the user. This will require parsing what the user has entered into separate tokens and storing the tokens in an array of character strings. For example, if the user enters the command ps -ael at the osh > prompt, the values stored in the args array are:

args[0] = "ps"

args[1] = "-ael"

args[2] = NULL

This args array will be passed to the execvp() function, which has the following prototype:

execvp(char \*command, char \*params[]);

Here, command represents the command to be performed and params stores the parameters to this command. For this project, the execvp() function should be invoked as execvp(args[0], args). Be sure to check whether the user included an & to determine whether or not the parent process is to wait for the child to exit.

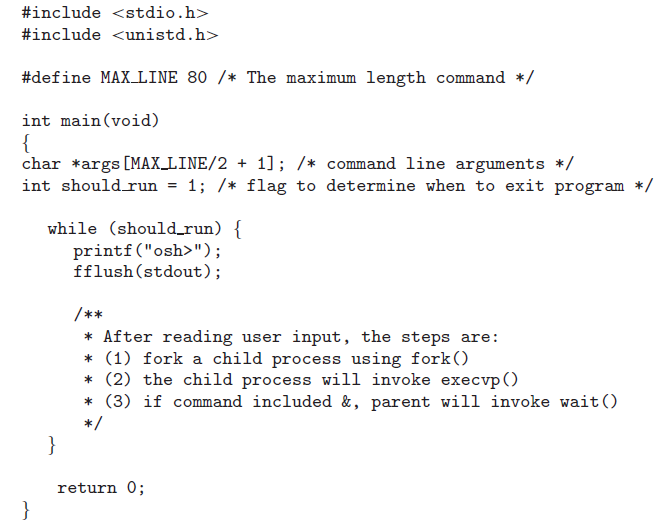


Figure 1: Outline of Simple Shell

#### Error Handling

Perform the necessary error checking to ensure that a valid shell command was entered.

### Grading

The program will be graded on the basic functionality, error handling and how well the implementation description was followed. Be sure to name your program **sshell.c** (no extra characters, capitals) Note that documentation and style are worth 10% of the assignment's grade!

### Submission

The program source code should be posted to Canvas as well as a README file with sample output.