# OS LAB

## Building a Shell

We will first build a simple shell, much like the bash shell of Linux. A shell takes in user input, *forks* one or more child processes using the fork system call, calls *exec* from these children to execute user commands, and reaps the dead children using the wait system call. Learn about the *fork* system call and all variants of the *wait* and *exec* system calls before you begin this lab.

Begin your code by writing a shell that executes simple Linux commands like *ls*, *cat*, *echo* and *sleep*. These commands are readily available as executables on Linux, and your shell must simply invoke them. Your simple shell must use the string "*MTL*458 >" as the command prompt. If the user command is one of the Linux built-in commands, you must exec the corresponding Linux executable, and return for user input after execution completes. The shell must continue execution in this manner in an infinite loop, until the user hits Ctrl+C to terminate the main shell process. Any errors returned during the execution of these commands must be displayed in the shell.

The output of one process seamlessly is used as input to the next using *pipe*. For example, your shell should be able to do *grep -o foo file | wc -l*.

However, not all commands are built into Linux, and the shell must write code to implement some commands as well,e.g.,the command *cd directoryname* and the command *history*.Once you complete the execution of the built-in commands, proceed to implement support for *cd* and *history* in your shell. The semantics of the commands must be similar to what you find in the bash shell. For example, *cd directoryname* must cause the shell process to change its working directory. You will find the *chdir* system call useful to implement the cd command.

Note that for all commands you implement in this lab, an incorrect number of arguments or incorrect command format should print an error in the shell. After such errors are printed by the shell, the shell should not crash. It must simply move on and prompt the user for the next command.