

## **Project: Shell**

**Purpose:** The purpose of this project is to familiarize you with the mechanics of process control through the implementation of a shell user interface. This includes the relationship between child and parent processes, the steps needed to create a new process, shell variables, and an introduction to user-input parsing and verification.

**Task 1.1:** The shell (command line) is just a program that continually asks for user input, perhaps does something on the user's behalf, resets itself, and again asks for user input. Design and implement a basic shell interface that supports the execution of other programs and a series of built-in functions. The shell should be robust (e.g., it should not crash under any circumstance beyond machine failure).

**Basic:** The prompt should look like this:

prompt\$

**Advanced:** The prompt should look like this:

machinename@username:~\$

Where machinename and username should change depending on the machine and user.

**Task 1.2:** Shell programs/commands.

Basic: Implement the basic functionality of the following programs: rm, cat, clear, cowsay.

**Intermediate:** Provide a few options and/or arguments for at least two programs. Additional points for creativity (e.g. implementing something that does not exist in bash, or differently than it is done in bash).

**Advanced:** Allow piping or at least redirecting output to a text file.

Task 1.3: System calls.

**Basic:** Within the C-programming example of your choice, implement the following system calls: fork(), wait(), and exec().

Intermediate: Within the C-programming example of your choice, implement kill(), execv().

**Additionally:** Carefully explore and then implement the forkbomb.



- **Task 1.4:** Add some colors to your shell and name.
- **Task 1.5:** Provide a concise and descriptive answer to the following questions.
- Q1: What does the mmap() function do?
- Q2: What happens during a context switch? Do we want the OS to perform many or few context switches? Explain.