**Project # 1: Build a minimal shell (minShell): due Friday April 3rd.**

We have seen that a Unix shell (like BASH) is a sophisticated command-interpreter and scripting language that provides a powerful user interface to the operating system. In the most basic sense, a shell is a program that executes other programs.

**In this project you will build a minimalist shell that supports the following requirements:**

1. Display a command prompt for user input (e.g. minShell$)
2. Read user input, parse, and run commands
3. Your shell should support the following features:
   1. Setting/getting shell variables: support shell built-in command: *set/show*
      1. *minShell$ set path**=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin*
      2. *minShell$ show path*

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin

* + 1. *minShell$ set name=”Mike C”*

*minShell$ show name*

Mike C

* + 1. *minShell$ set num=1000*

*show num*

1000

* 1. Run (execute) external commands based on a user specified minShell path variable (*not the actual system path variable*)
     1. *minShell$ set path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin*
     2. *minShell$ ls -l/bin /bin*
* searches the specified *path* variable (above) for the *ls* program, and if found it executes the command with the arguments specified e.g. *-l /bin*
* Note: You will have to parse commands entered by the user to determine what command and arguments to properly run in your shell.
* Your shell will need to make use of at least two system calls
  + fork – creates a new (child) process by duplicating the parent process (the minShell process) -- use “man fork” for the man page
  + execv - loads (overlays) an existing process with a new process image -- use “man execv”. *Note: there is a family of exec functions, but we will use execv for this project*

That’s it! When your shell supports requirements: 1. 2. 3a and 3b, then submit the code and the output as separate documents in Blackboard under the **Project 1** assignment. Please upload your code with the name “MinShellCode”, and the output with name “MinShellOutput”.

***Note: It is up to you to generate your output/screenshots so that it clearly demonstrates you’ve met requirements 3a. and 3b.*** If you don’t clearly demonstrate you meet 3a. and 3b. you may not get full credit.

**Background and thoughts and Starting**

The basic processing logic for a minimalist shell might be summarized by the following steps:

1. Display prompt to the user (e.g. minShell$)
2. Wait for a command from the user
3. parse the command
4. execute the command
5. return to step 1

To begin building your minShell you might start by building a program in C or C++ that run (executes) other programs. To do that, consider the steps (1-5) outlined above:

1. Write a while loop that will display a prompt, and then waits/reads input from the user.
   1. Reading input from the user means the user types a command line and presses enter/return
2. Parse the command line
   1. Parsing the command means to extracting the parts and determine what actions(s) to take. i.e. set/get shell variables or run/execute an external command. For example:
      1. *minShell$ set path=”/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin”*
      2. *minShell$ ls -l*
3. Executing the parsed command requires the creation of a new (child) process. Creating a new process is performed with the *fork* system call, followed by an *exec* system call to load the process image specified in the command
   1. When a shell executes a command, it first creates a copy of itself (the child process by calling *fork*). It then calls *exec i*n the child process to load/run the command by loading specified program image into (overlaying) child process with the specified program image (***See Figure 1 below for an illustration of the process).***
      1. Please read the following Lecture from USNA to begin getting a grasp on the concept: <https://www.usna.edu/Users/cs/aviv/classes/ic221/s16/lec/14/lec.html>

* Note: I intend to give a significant amount of helper code. That will be posted soon, and a notification will be sent via piazza

This is the basic idea:

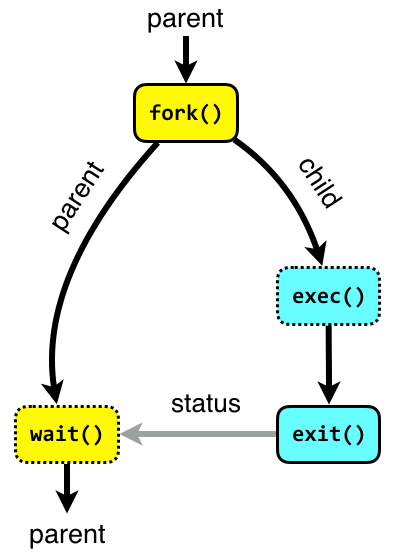


Figure 1: Command Execution

Image source: <http://www.it.uu.se/education/course/homepage/os/vt18/module-2/process-management/>