VSH: A Tiny Shell with Binary Validation

Program Three
CS 3411 Spring 2022
Due: Friday, Mar. 4, 11:00pm

Motivation

In this project, you will get experience with the fork/exec paradigm in UNIX. You will write a simple shell called `vsh`. The shell executes a single command entered by the user. The command may have parameters. `Vsh` additionally provides some protection to the user by only allowing execution of binaries for which an MD5 hash run over the requested command matches a known valid hash for that command. The hash/command pairs are kept in a file protected through the UNIX file protection bits. This both restricts the commands that a user can execute and validates the binary that implements an allowed command.

Requirements

The requirements are in two parts: binary verification and command processing.

Binary Verification

The shell only accepts fully-qualified binary names. For example, `/bin/ls` is accepted but `ls` is not. When a user enters a binary name, the following checks are performed:

- The named binary file is read into memory and an md5sum is calculated over the file. Code that calculates the md5sum is available in `/home/campus13/jmayo/public/cs3411/projects/p3/justmd5.c`. You may incorporate this code but you are responsible to ensure it meets the coding expectations for this course.

- The md5sum is checked against the file `allowedsums` in the current working directory. Before `allowedsums` is used, it must meet the following critera: (*i*) it must be an ordinary file; (*ii*) none of the group or other protection bits may be set; and (*iii*) it must be owned by real UID of the executing process. The `getuid` system call returns the real UID of the executing process. The `lstat` system call can be used to get the file type, ownership, and protection bit values.

  Each line of `allowedsums` can be assumed to contain the following 5 fields: (1) 0 or more whitespace characters (tab and space), (2) a contiguous set of the characters a-z,A-Z, 0-9 and / that represent a file name, (3) 1 or more whitespace characters (tab and space), (4) a contiguous set of the characters lower case a-f and 0-9 that represent the hexadecimal value of the hash, and (5) one or more while space characters and a trailing newline. Note that the hash value is a text representation of the hash while the md5sum code returns a binary value. For simplicity, you may assume the file is well-formed. Your program must handle any file that follows this format, but for simplicity it is not required to detect whether a file violates this format.

  There is an example `allowedsums` file in `/home/campus13/jmayo/public/cs3411/projects/p3/allowedsums`. New hashes can be generated with the command: `cat file-name | md5sum`.

  Hidden characters can be seen using the command `cat -A file`. This could be used, for example, to see hidden characters (like tab vs. whitespace) in an `allowedsums` file.

Command Processing

- The shell only needs to handle a single command along with the command parameters. The following commands are supported:

  ```
  /usr/bin/grep -e drinks -e snacks -e gold somefile
  /bin/ls mydir
  ```

  Input and output redirection and pipes are not supported. The following commands are **not** supported.

  ```
  /bin/ls mydir | grep -e mayo
  /bin/cat < somefile
  ```

- You must use the `execvp` variant of exec to execute commands.
- The prompt for `vsh` should (initially) be set to `vsh#`. (Note that there is no '.' in the prompt.)
- You may not exec any variant of `sh` nor can you use the `system()` or `popen()` system calls.
- The command `q` should cause your shell to terminate. The shell must support one additional built-in commands: `prompt <string>`. The `prompt <string>` command makes `<string>` the new prompt. The string is a contiguous sequence of characters between ASCII value 33 and 126.

Notes

Please note the following.

- There are standard routines that will reduce the work required to parse the command line. Browse the man pages. (Start with `strtok` and look at the SEE ALSO section as needed.)
- The shell need not support background processing.
- The shell must gracefully handle failure of a command as well as an empty string for a command (e.g. the user just hits ⟨ENTER⟩ at the prompt). Specifically, **this should not cause the shell to terminate**.

Submission

Submit all code through Canvas in a tar file named `vsh.tgz`. The tar file can be created by making your project directory the current working directory and typing the command: `tar czvf vsh.tgz *`. Please do not use subdirectories within your project directory so that the grader can easily access all the submitted files. Include a makefile so that the commands `gtar zxf vsh.tgz; make` create a binary file named `vsh`. Typing `make clean` should remove all object files and the created binary `vsh`.

Collaboration

Empty hands discussions are allowed for this project.