Principles of Systems Programming

Program 5 – Program Management and External Commands

Throughout this course, you will be performing a set of progressive programming exercises. The goal of these exercises is to create a new shell, nsh5\_*gid*. Each program will build on the previous program, thus it is important to complete each program before moving on to the next one.

The fourth programming exercise expands program 4 by:

1. Your main function should be stored in the file “nsh5\_*gid*.c” and your executable file should be named “nsh5\_*gid*” where *gid* is replaced by your two digit group number.
2. Your screen capture of execution must demonstrate sample execution of the features of the program. Preferably, this screen capture will demonstrate all of the examples in this and previous assignments thus demonstrating regression testing.
3. Create a makefile to support: clean and build operations.
4. Place functions in separate files in support of functionality (I.e., file operations, string operations, sorting operations, memory operations, environment variable operations, alias operations, etc.).
5. Comments, indentation, and variables names must henceforth take on one of the styles documented in the textbook. Each file must include a comment block at the beginning of the file which should include a description functional nature of the file content as well as your name with the date of your last modification. A comment block should be included in each header file before each function prototype which includes a description of the functionality of the function as well as a list of the arguments along with their type, nature, and input/output status. (See section 6.2.5. Your effort in applying these styles will be reflected in the value of your comments).
6. Place your submission in a zip file. This zip file needs to include:
   1. All source files,
   2. Your report,
   3. Your evidence of compilation and execution,
   4. A packing list with file sizes, and
   5. A ReadMeFirst file which contains instruction on how to build and execute your program.
7. Implement the internal command “where” (with usage “where type name”) which will determine the location of the file “name” with the permission “type” within the PATH variable list of directories, if “name “ does not contain any “/” characters. If name does contain any “/” characters, then the program will ignore the PATH variable and look at the absolute or relative location to determine if the permission is set correctly according to “type.” Aliases must be observed. If name refers to an internal command, it should be noted as such. The value of “type” is allowed to take any of the following values: any, read, write, run, dir, or file. This command should return the first qualified path to the file which meets the “type” specified by the user. If none are found, this needs to be reported.
8. Implement external commands via the fork(), exec(), and wait() families of functions. The use of system() is prohibited. The user may execute an external command in any directory included in the internal environment variable PATH. Concatenate the detected path with the file name before issuing the external command. If the command file cannot be found in any of the directories in PATH, the program shall issue the result “command “requested command” not found.” You must to use your implementation for part 7 to solve part 8.
9. All operations of Program 1-4 must still be functional.

Examples:

The following sequence of commands must result in: /usr/bin/edit.

set PATH /bin!/usr/bin!..!. ~ Applies to all of the examples presented.

alias e edit

where run e

The following sequence of commands should result in: set is an internal command.

where run set

The following sequence of commands should result in: alias is an internal command.

alias a alias

where run a

The following sequence of commands should result in: adventure was not found.

alias fun adventure

where run fun

Assuming that the directory xyzzy is found in your parent directory and your current directory in the PATH, the following sequence of commands should result in: ../xyzzy.

where dir xyzzy

Assuming that the file plugh is found in your parent directory with read and write permissions and your current directory with execute permission in the PATH, the following sequence of commands should result in: ./plugh.

where run plugh

Complex string processing

alias ll {ls –al}

ll ~ should produce the same results as it would in bash

ps –ef ~ should produce the same results as it would in bash