Description of Program:

The main goal of this lab was to create a shell program with change directory, print working directory, list directory and exit functions that mirror the Linux program. The first step in developing this program, it required taking input from the user correctly and parsing the input. We developed a function called splitLineCommand, which takes in the user input and returns each word as a separate string, using the skipChar function, and counts the number of words in the string. To find the end of each word, strchr is used to find the first instance of a space character, which then ends the word and sets the first space, if there are many, equal to the null character (\0).

The skipChar command used in splitLineCommand, takes in a pointer to the character and the character to skip. If the character to skip is the space or anything you increase the position of the character pointer until you reach the null character, then return the pointer.

Next, the command structure was created for the shell program which has a corresponding name and function. In the program, it is named cmdStruct. This is created so that we could create a dispatch table of the commands the program could look for ie. exit, ls, cd, pwd.

The doCommand function is used to look through the user's input arguments, and compares them to the desired arguments in the dispatch table, by using the strcmp() function. If one matches, the corresponding function is run.

The four command functions were created in the order of exit, pwd, cd, and ls. The exit command uses the exit() command found in C to exit the shell program.

The pwd function takes in the pointer to the character array of user inputs, and the number of arguments found. Using getcwd() function found in C, we then set a pointer to the current working directory. We then print the entire working directory and then free the pointer.

The change directory function takes in the pointer to the character array of user inputs, and the number of arguments found. The program then uses the function call getpwuid(getuid()) to set the new password element to the users password. The program then checks if there are any arguments behind the cd call and if not it returns to the directory /home/student/. If there is a command behind cd, it attempts to set the directory to what is typed. By using the chdir() function, it returns whether or not the directory requested is a valid directory or not and if it is not valid it returns an error.

For the list function, the program takes in the same arguments as cd and pwd. It creates a pointer namelist to the directory entry using the dirent structure. If ls is called without any other arguments, the program will use scandir(), with the use of isDot as the third argument, to find the number of elements in the namelist pointer. If there are two arguments it uses strcmp() to check if the argument is "-a", and if so it uses scandir() to find the entire number of elements in the directory, including hidden ones. If the second argument is not “-a”, the program throws an error. It then prints the desired elements in the directory.

The isDot function created uses the dirent structures pointer. If that pointer points to d_name and its first character is equal to a “.” it will return a zero for true and if not it will return a 1 for a false. Used in the scandir, the true and false will determine whether or not the element will be added to the list of elements to print out.

Testing Results:

To test the code, a script was used to record running a series of commands in the Linux shell, before then testing the same commands in the shell that we made during the lab. The commands tested were cd (change current directory), ls (list entries in current directory), pwd (print current working directory), and exit (exits shell). Lines 2-4 of the script file demonstrate the use of the cd and pwd commands in the Linux shell to change the directory to ‘elec377-group-122/lab1/’ and then print the current directory. Following that, cd is called with no arguments and then pwd is called to demonstrate how this changes your directory to the basic home directory (line 7). Cd is then called with a file directory that does not exist to show the error handling (line 9). Cd was then used again to return to the ‘elec377-group-122/lab1/’ directory where ls is called with no arguments to show all non-hidden files in the current directory (line 16). Finally, ls is called with the argument ‘-a’ to show all files, including any hidden files, in the current directory (line 18).

Upon running the shell program, two basic strings are inputted to demonstrate the program’s ability to take user input and separate strings into single words/commands. This also demonstrates the program output when an invalid command is inputted and how the program can handle multiple spaces between arguments (lines 25-34). Following this, the pwd command was called to demonstrate how the current working directory is printed (line 38). The cd command is then called with no arguments and pwd is used to show that this command changes you to the home directory (lines 39-45). Cd is then used again with an argument to return to the ‘/home/student/elec377-group-122/lab1’ directory (line 46). Cd is then called with an argument that is a non-existent directory to demonstrate the error handling when the requested directory cannot be found (lines 54-58). Finally, the ls command is tested without an argument to display all non-hidden entries in the current directory and then with the ‘-a’ argument to display hidden entries as well (lines 63-80). To end the program, the exit command is used on line 81 to exit the shell.