Step1 - Retrieve Starting Code

We used git pull to uprate the current version of our repository, and we when got into lab1 folder, the team had made the agreement that we would use dev_F as Mike's branch and dev_P as Paul's branch. We mainly work together and used dev_F branch. We compiled it using the command make and ran it with the command shell.

Step2 - Process the Command Line

First we make a temp variable as the pointer which points to the commandBuffer, and use the sum of the temp with the strlen (commandBuffer) function (counts the number of characters in a given string and returns the integer value) and minus one to remove newline from end of buffer. To call splitCommandLine, the right parameters are commandBuffer, args, MAXARG(by the lab instruction). For the skipChar function, we made a big if else statement which detect three different situations and make execution accordingly: 1. return itself when is a NULL 2. skip all the ' ' and return the first char pointer 3. skip to find the first char and return the first ' ' pointer. For the splitCommandLine function, we first initial a counter to zero and used this counter to walk through the list. While the commandBuffer is not equal to NULL, we used an if loop to skip all the space to find first char if detected Ox20 (' ') or put the first char address in the args, change the last space of char into NULL and point to the next on otherwise. Lastly, if the counter is greater than the maxargs which means the counter goes outer of the list and we check whether the args is overflowed as default.

Step 3 - Command Type

The name for typedef is cmdFuncPtr, and we passed in char * args[], int nargs as parameters. Inside struct cmdType, we made a char *cmdName to point to cmdFunction, and *args points to nargs.

Step 4 - Dispatch Table

We a callFunction which has cmdFuncPtr func, char * args[], int nargs as parameters for each command handling function. Then we made an array called commandArrayName which including lsFunc, cdFunc, pwdFunc, exitFunc and {NULL, NULL} situation, and declared them globally to make them as global array.

Step 5 - Command Loop

For doCommand function, which passed in char * args[], int nargs as parameters, we initialized counterA as the array command size and conterC to jump between different command. The first if condition is to jump out the loop if hit NULL in commandArrayName. The second if condition will print error when hit the end of array. The Last if loop will execute the function and move on if stay in same command or increment counter and jump to the next command otherwise.

Step 6 - Adding More Command

Declare the exitFunc break out the function. For the pwdFunc, we get the content of getcwd to a pointer called cwd(copy working directory). Print the working directory and then free it. For cdFunc, it goes back to home directory first. Inside if condition, first report an error when home directory is NULL. Inside else part, condition one is there is no argument, print out the home path. Condition two is that this is another argument, then change to that argument. Lastly,

chdir() returns none 0 values when it can not change. As for IsFunc, the if loop store list in namelist, return number of lists to numEnts, condition of excluding “.” Documents by using checkDotDocument to check. Else it will use strncmp to find “- “symbol, within this else if loop, we fprinted all the document under the list and checked extra flags. The last else if loop checked if input arguments more than 2 and return error. Print "unknown argument" for the second argument cannot be recognize. For checkDotDocument function, the if loop will check if is not a dot file it will return true; else it will return false. Lastly, we used the function in Linux to make the test file to test the correctness of our code.

Special features of the C language

1. Structured programming language: C is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify. Functions also provide code reusability. We defined many functions in lab1 and those functions make our coding process easier.

2. Rich Library: C provides a lot of inbuilt functions that make the development fast. We used some libraries such as <stdio.h>, <unistd.h>, <stdlib.h> and used functions they provided.

3. Pointer: C provides the feature of pointers. We can directly interact with the memory by using the pointers. We can use pointers for memory, structures, functions, array, etc. We were almost using pointers during functions all the time. Without those pointers those functions would be more difficult to construct.