ELEC377 Lab 2 – Documentation

Aiden Peters and Laeticia Niu

The program for Lab 2 implements a shell that uses built-in commands such as “exit”, “pwd”, “ls” and “cd”. The main function uses a read eval print loop for the shell that takes in inputs until an argument is NULL.

The skipChar function skips over a given char in a string. Both the string and char are function parameters, with the string being pointed at by a pointer. The function has a while loop which loops through the string pointer and returns a pointer to the first position in the string that isn’t the char being skipped.

The function splitCommandLine uses the skipChar function to split the string into different words and then returns the number of words (arguments) in the string. The function goes through the string and stores pointers for the beginning of each individual word, which is done using “strchr”. “fprintf” is used to print an error in the case that the number of arguments exceeds the max number of variables (maxargs).

Next, the doProgram function loops through each element of the path array. We use strlen to find the length of the current path element and then malloc to allocate space for the path, / and the command. The library stat is used

Then we have a typedef that is used as a function pointer, followed by function declarations/prototypes for exitFunc, lsFunc, cdFunc, pwdFunc and a dispatch table for those 4 functions. Before we go more in depth into those four functions, we have the doInternalCommand function. This function has a loop that compares the first field of each element of the array to the string that is given by the first argument (args[0]). We use strcmp to determine which function to call from the dispatch table.

The exitFun function is quite simple, as it calls the already existing exit function from C libraries. The lsFunc function starts off by checking if there are one or two arguments, using if statements with numargs. In the case that there are two arguments, the if statement also checks if the second argument is “-a”. If that statement is true, all the files in the current directory are printed out. The contents of the directory are listed out using “scandir”, while “dirent” structure is used as a pointer to the directory entries. The field “d_name” gives the name of the first entity in the array. If there is only one argument, we use the function checkDot that returns false if a file name starts with a dot and true otherwise. This is used to filter all the hidden files. If the number of arguments is not 1 or 2, an error message is printed. The cdFunc also checks if there are one or two arguments. If there is one argument, the directory is changed to the home directory, using the “getpwuid” function to get a pointer to the password file entry of the user. If there are two arguments, the directory is changed using chdir. Finally, the pwdFunc uses the C library function getcwd and stores it in a pointer, whose space is freed after the password as been printed.

To complete the lab, we used some special features of the C language such as the function pointers, which were predefined functions that we used in order to obtain certain pieces of information. This includes: getcwd, getpwuid, getuid, fgets, scandir