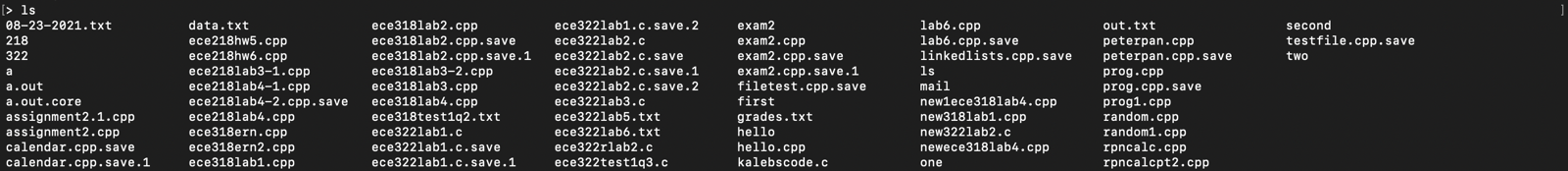
Nikeem Dunkelly-Allen, ECE322, Assignment 2



A picture containing table

Description automatically generated

Graphical user interface, text

Description automatically generated



#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <string.h>

#define MAX\_LINE\_LENGTH 1024

int main() {

char line[MAX\_LINE\_LENGTH];

char \*tokens[MAX\_LINE\_LENGTH];

int token\_count;

while (1) {

// Print a prompt

printf("> ");

// Read line from user

fgets(line, MAX\_LINE\_LENGTH, stdin);

// Split the line into its components

token\_count = 0;

tokens[token\_count] = strtok(line, " \t\n");

while (tokens[token\_count] != NULL) {

token\_count++;

tokens[token\_count] = strtok(NULL, " \t\n");

}

// If the first component is "echo" or "exit", obey it

if (strcmp(tokens[0], "echo") == 0) {

int i;

for (i = 1; i < token\_count; i++) {

printf("%s ", tokens[i]);

}

printf("\n");

} else if (strcmp(tokens[0], "exit") == 0) {

exit(0);

} else {

// Use fork and exec to make it happen

int child\_pid;

if ((child\_pid = fork()) == -1) {

printf("Error: fork failed\n");

} else if (child\_pid == 0) {

// Child process

execvp(tokens[0], tokens);

} else {

// Parent process

int status;

waitpid(child\_pid, &status, 0);

// if WIFEXITED(status) was true it returns 1

if ( WIFEXITED(status) != 1 ) {

int exit\_status = WEXITSTATUS(status);

printf("exit code %d\n", exit\_status);

}

}

}

}

return 0;

}