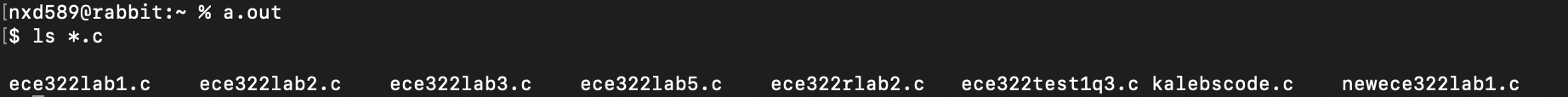
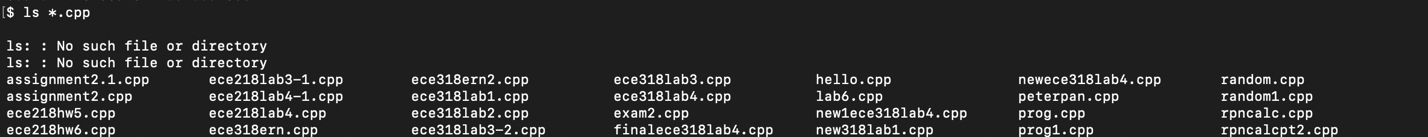
Nikeem Dunkelly-Allen, ECE322, Assignment 5

Screenshots:





Shape

Description automatically generated with medium confidence

A picture containing schematic

Description automatically generated

#include <stdio.h>

#include <sys/types.h>

#include <string.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <dirent.h>

struct alias{

char \*cue;

char \*argument;

char \*condition;

struct alias \*next;

};

struct alias \*head= NULL;

void adding(char \*cued, char \*argumentd, char\*conditiond){

struct alias \* link = (struct alias\*) malloc(sizeof(struct alias));

link->cue = cued;

link->argument = argumentd;

link->condition = conditiond;

link->next = head;

head = link;

}

//parsing arguments

void arguments(char \*buff, char \*\*argv)

{

while (\*buff != '\0') {

while (\*buff == ' ' || \*buff == '\n')

\*buff++ = '\0';

\*argv++ = buff;

while (\*buff != '\0' && \*buff != ' ' && \*buff != '\n')

buff++;

}

\*argv = '\0';

}

//running arguments

void execute(char \*\*argv)

{

pid\_t pid;

int status;

if ((pid = fork()) < 0) {

perror("fork error");

exit(1);

}

else if (pid == 0) {

if (execvp(\*argv, argv) < 0) {

perror("couldn't execute");

exit(1);

}

}

else {

while (wait(&status) != pid) {

if (WIFEXITED(status)){

short int code = WEXITSTATUS(status);

if (code != 0)

printf("Exit code not zero: %d\n", code);

}

}

}

}

//set argument

void setting(char \*\*argv){

char \* var = argv[1];

char \* val = argv[3];

setenv(var, val, 1);

}

//alias argument

void aliases(char \*\*argv){

adding(argv[1],argv[3], argv[4]);

}

//checking list of aliases

void search(char \*\*argv){

struct alias \*temp = head;

if (head == NULL){

return;

}

while(temp != NULL){

char \*ptr = argv[0];

char \*str = temp->cue;

if (strcmp(ptr, str) == 0){

bzero(argv, 64);

argv[0] = temp->argument;

argv[1] = temp->condition;

return;

}

else{

temp = temp->next;

}

}

}

//comparing the end of file name with \*.xyz glob pattern

int equal (int pattern\_len, char str1[], char\*str2){

char \* ptrrr = &str1[strlen(str1) - pattern\_len +1];

for (int j = 0; j < pattern\_len; j++){

if(strcmp(ptrrr, str2) != 0){

return 0;

}

}

return 1;

}

int main(void)

{

char buff[1024];

while (1) {

char \*argv[64];

//argument line

printf("$ ");

fgets(buff, 1024, stdin);

buff[strcspn(buff, "\n")] = 0;

printf("\n");

//parsing arguments

arguments(buff, argv);

//check aliases for argument

search(argv);

//glob patterns

if (argv[1][0] == '\*'){

int pattern\_len = strlen(argv[1]);

char argv\_c[10];

//char \* argv\_c = malloc(sizeof(\*argv));

DIR \* d = opendir(".");

if (d == NULL)

{ fprintf(stderr, "Can't open .\n");

exit(1); }

int i = 1;

//\*.xyz pater

if (argv[1][1] == '.'){

for (int n = 0; n < pattern\_len; n++){

argv\_c[n] = argv[1][n];

}

while (1){

struct dirent \* entry = readdir(d);

if (entry == NULL){

break;

}

if (strlen(entry->d\_name) < pattern\_len){

continue;

}

if (entry->d\_name[0] == '.'){

continue;

}

if (equal(pattern\_len, entry->d\_name, &argv\_c[1]) == 1){

argv[i] = entry->d\_name;

i++;

continue;

}

else{

continue;

}

i++;

}

}

else{ // \* pattern

while (1){

struct dirent \* entry = readdir(d);

if (entry == NULL)

break;

if (entry->d\_name[0] == '.'){

continue;

}

argv[i] = entry->d\_name;

i++;

}}

}

//echo

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

int i = 1;

while(1){

char \* argptr = argv[i];

if (argptr == NULL){

break;

}

if (argv[i][0] == '$'){

char \*str = getenv(&argv[i][1]);

printf("%s ", str);

i++;

continue;

//break;

}

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

i++;

}

printf("\n");

continue;

}

//exit

if (strcmp(argv[0], "exit") == 0){

exit(0);

}

//set

if (strcmp(argv[0], "set") == 0){

setting(argv);

continue;

}

//alias

if (strcmp(argv[0], "alias") == 0){

aliases(argv);

//printf("%s\n", argv[2]);

//return 1;

continue;

}

execute(argv);

}

}