Part 1 – Shell Source Code

https://github.com/NootCode/CS2600-Final/tree/master/Part1-Shell

#include <sys/wait.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

char \*lsh\_read\_line(void){

    #ifdef LSH\_USE\_STD\_GETLINE

    char \*line = NULL;

    ssize\_t bufsize = 0;

    if(getline(&line, &bufsize, stdin) == -1){

        if(feof(stdin)){

            exit(EXIT\_SUCCESS);

        }else{

            perror("readline");

            exit(EXIT\_FAILURE);

        }

    }

    return line;

#else

#define LSH\_RL\_BUFSIZE 1024

    int bufsize = LSH\_RL\_BUFSIZE;

    int position = 0;

    char \*buffer = malloc(sizeof(char)\*bufsize);

    int c;

    if(!buffer){

        fprintf(stderr, "lsh: allocation error\n");

        exit(EXIT\_FAILURE);

    }

    while(1){

        c = getchar();

        if(c == EOF || c == '\n'){

            buffer[position] = '\0';

            return buffer;

        } else{

            buffer[position] = c;

        }

        position++;

        if(position >= bufsize){

            bufsize += LSH\_RL\_BUFSIZE;

            buffer = realloc(buffer, bufsize);

            if(!buffer){

                fprintf(stderr, "lsh: allocation error\n");

                exit(EXIT\_FAILURE);

            }

        }

    }

#endif

}

#define LSH\_TOK\_BUFSIZE 64

#define LSH\_TOK\_DELIM " \t\r\n\a"

char \*\*lsh\_split\_line(char \*line){

    int bufsize = LSH\_TOK\_BUFSIZE, position = 0;

    char \*\*tokens = malloc(bufsize \* sizeof(char\*));

    char \*token;

    if(!tokens){

        fprintf(stderr, "lsh: allocation error\n");

        exit(EXIT\_FAILURE);

    }

    token = strtok(line, LSH\_TOK\_DELIM);

    while(token != NULL){

        tokens[position] = token;

        position++;

        if(position >= bufsize){

            bufsize += LSH\_TOK\_BUFSIZE;

            tokens = realloc(tokens, bufsize \* sizeof(char\*));

             if(!tokens){

                fprintf(stderr, "lsh: allocation error\n");

                exit(EXIT\_FAILURE);

            }

        }

        token = strtok(NULL, LSH\_TOK\_DELIM);

    }

    tokens[position] = NULL;

    return tokens;

}

int lsh\_launch(char \*\*args){

    pid\_t pid, wpid;

    int status;

    pid = fork();

    if(pid == 0){

        if(execvp(args[0], args) == -1){

            perror("lsh");

        }

        exit(EXIT\_FAILURE);

    }else if (pid > 0){

        perror("lsh");

    }else{

        do{

            wpid = waitpid(pid, &status, WUNTRACED);

        }while(!WIFEXITED(status) && !WIFSIGNALED(status));

    }

    return 1;

}

int lsh\_cd(char \*\*args);

int lsh\_help(char \*\*args);

int lsh\_exit(char \*\*args);

char \*builtin\_str[] = {

    "cd", "help", "exit"

};

int(\*builtin\_func[])(char\*\*)= {

    &lsh\_cd,

    &lsh\_help,

    &lsh\_exit

};

int lsh\_num\_builtins(){

    return sizeof(builtin\_str) / sizeof(char \*);

}

int lsh\_cd(char \*\*args){

    if(args[1] == NULL){

        fprintf(stderr, "lsh: expected argument to \"cd\"\n");

    }else{

        if(chdir(args[1]) != 0){

            perror("lsh");

        }

    }

    return 1;

}

int lsh\_help(char \*\*args){

    int i;

    printf("Andre Nalbandians LSH\n");

    printf("Type Program names and Arguments and hit enter.\n");

    printf("The following are built in: \n");

    for(i = 0; i < lsh\_num\_builtins(); i++){

        printf(" %s\n", builtin\_str[i]);

    }

    printf("Use the man command for info on other programs.\n");

    return 1;

}

int lsh\_exit(char \*\*args){

    return 0;

}

int lsh\_execute(char \*\*args){

    int i;

    if(args[0] == NULL){

        return 1;

    }

    for(i = 0; i < lsh\_num\_builtins(); i++){

        if(strcmp(args[0], builtin\_str[i]) == 0){

            return (\*builtin\_func[i])(args);

        }

    }

    return lsh\_launch(args);

}

void lsh\_loop(void){

    char \*line;

    char \*\*args;

    int status;

    do{

        printf("> ");

        line = lsh\_read\_line();

        args = lsh\_split\_line(line);

        status = lsh\_execute(args);

        free(line);

        free(args);

    }while(status);

}

int main(int argc, char \*\*argv){

    lsh\_loop();

    return EXIT\_SUCCESS;

}

Part 2 – Text Editor Source Code