**Taaha Hussain Khan**

**L1F21BSCS0917**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <fcntl.h>

#define MAX\_INPUT\_SIZE 1024

#define MAX\_ARGS 64

void execute\_command(char \*command, int is\_background) {

char \*args[MAX\_ARGS];

int i = 0;

char \*token = strtok(command, " ");

while (token != NULL && i < MAX\_ARGS - 1) {

args[i++] = token;

token = strtok(NULL, " ");

}

args[i] = NULL;

if (is\_background) {

if (fork() == 0) {

setpgid(0, 0);

execvp(args[0], args);

perror("execvp");

exit(EXIT\_FAILURE);

}

} else {

if (fork() == 0) {

execvp(args[0], args);

perror("execvp");

exit(EXIT\_FAILURE);

} else {

wait(NULL);

}

}

}

void execute\_piped\_commands(char \*command) {

char \*commands[MAX\_ARGS];

int i = 0;

char \*token = strtok(command, "|");

while (token != NULL && i < MAX\_ARGS - 1) {

commands[i++] = token;

token = strtok(NULL, "|");

}

commands[i] = NULL;

int pipe\_fd[2];

int input\_fd = 0;

for (i = 0; commands[i] != NULL; i++) {

pipe(pipe\_fd);

if (fork() == 0) {

dup2(input\_fd, STDIN\_FILENO);

close(input\_fd);

if (commands[i + 1] != NULL) {

dup2(pipe\_fd[1], STDOUT\_FILENO);

}

close(pipe\_fd[0]);

execvp(strtok(commands[i], " "), &commands[i]);

perror("execvp");

exit(EXIT\_FAILURE);

} else {

wait(NULL);

close(pipe\_fd[1]);

input\_fd = pipe\_fd[0];

}

}

}

int main() {

char input[MAX\_INPUT\_SIZE];

while (1) {

printf("> ");

fgets(input, sizeof(input), stdin);

size\_t len = strlen(input);

if (len > 0 && input[len - 1] == '\n') {

input[len - 1] = '\0';

}

int is\_background = 0;

if (input[len - 1] == '&') {

is\_background = 1;

input[len - 1] = '\0';

}

char \*input\_redirect = strstr(input, "<");

char \*output\_redirect = strstr(input, ">");

if (input\_redirect != NULL) {

\*input\_redirect = '\0';

int fd = open(input\_redirect + 1, O\_RDONLY);

dup2(fd, STDIN\_FILENO);

close(fd);

}

if (output\_redirect != NULL) {

\*output\_redirect = '\0';

int fd = open(output\_redirect + 1, O\_WRONLY | O\_TRUNC | O\_CREAT, 0644);

dup2(fd, STDOUT\_FILENO);

close(fd);

}

char \*pipe\_symbol = strchr(input, '|');

if (pipe\_symbol != NULL) {

execute\_piped\_commands(input);

} else {

// No piping, just execute the command

execute\_command(input, is\_background);

}

}

return 0;

}