

Sample Run

testme.sh

```
myshell — -bash — 78x23
[GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$ ./testme.sh ]
abc
def
ghi
Executing command cat
Command returned with return code 0
Real time: 6.-156296 sec
User time: 0.001206 sec
System time: 0.001466 sec
abc
def
ghi
Executing command cat
Command returned with return code 0
Real time: 0.003507 sec
User time: 0.001170 sec
System time: 0.001070 sec
[GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$ echo $? ]
123
GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$
```

testme2.sh

```
myshell — -bash — 78x23
[GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$ ./testme2.sh <input.txt ]
Executing command cat
Command returned with return code 0
Real time: 0.004144 sec
User time: 0.001078 sec
System time: 0.000822 sec
[GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$ echo $? ]
0
[GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$ cat input.txt ]
hello world my friendGUOWEIZHEdeMacBook-Pro:myshell guoweizhe$ cat cat2.out ]
hello world my friendGUOWEIZHEdeMacBook-Pro:myshell guoweizhe$
```

Commands running from myshell

```
myshell — -bash — 91x50

[GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$ ./myshell ]
pwd
Current working directory is /Users/guoweizhe/Documents/ECE-357/myshell/myshell/myshell
ls
main.c          myshell          test1.txt        test2.txt        test3.txt
Executing command ls
Command returned with return code 0
Real time: 0.007948 sec
User time: 0.001350 sec
System time: 0.002193 sec
ls -l >test4.txt
Executing command ls
Command returned with return code 0
Real time: 0.007464 sec
User time: 0.001695 sec
System time: 0.001153 sec
cat test4.txt
total 72
-rw-r--r--@ 1 guoweizhe  staff   6287 Oct 21 19:24 main.c
-rwxr-xr-x  1 guoweizhe  staff  14036 Oct 21 19:25 myshell
-rw-r--r--  1 guoweizhe  staff    35 Oct 21 17:44 test1.txt
-rw-r--r--  1 guoweizhe  staff    61 Oct 21 13:07 test2.txt
-rw-r--r--  1 guoweizhe  staff   309 Oct 21 18:41 test3.txt
-rw-r--r--  1 guoweizhe  staff     0 Oct 21 19:27 test4.txt
Executing command cat
Command returned with return code 0
Real time: 0.005067 sec
User time: 0.001206 sec
System time: 0.000864 sec
cd /Users/guoweizhe/Documents/ECE-357/minicat
ls
Assignment1.docx      hello.txt           minicat.c
Assignment1.pdf       minicat             minicat.xcodeproj
Executing command ls
Command returned with return code 0
Real time: 0.005915 sec
User time: 0.001430 sec
System time: 0.001357 sec
hello >hello.txt
Error while executing command hello: No such file or directory
Executing command hello
Command returned with return code 1
Real time: 0.001289 sec
User time: 0.000233 sec
System time: 0.000680 sec
exit 123
[GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$ echo $? ]
123
GUOWEIZHEdeMacBook-Pro:myshell guoweizhe$
```

```

//
//  main.c
//  myshell
//
//  Created by GUOWEIZHE on 10/20/18.
//  Copyright © 2018 GUOWEIZHE. All rights reserved.
//

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <time.h>
#define DELIM " \n"

void parse_read(FILE *input);
char *readline(FILE *input);
int exec_line(char **arg, char **redir);
int run_line(char **arg, char **redir);
int redirect(char *path, int init_fd, int options, mode_t mode);

int main(int argc, char **argv){
    FILE *infile;
    if(argc > 1){
        if((infile = fopen(argv[1], "r")) == NULL){
            fprintf(stderr, "Error while opening input file %s: %s\n", argv[1],
                strerror(errno));
            return -1;
        }
        else{
            parse_read(infile);
        }
    }
    else if(argc == 1){
        infile = stdin;
        parse_read(infile);
    }
    return 0;
}

void parse_read(FILE *input){
    char *line;
    char **line_arg;
    char **line_dir;
    int status;
    do{
        //printf("> ");
        line = readline(input);

```

```

int i = 0, j = 0;
char *token;
char *temp = malloc(BUFSIZ);
char **arg = malloc(BUFSIZ);
char **redir = malloc(BUFSIZ);
if(temp == NULL || arg == NULL || redir == NULL){
    fprintf(stderr, "Error while allocating memory using malloc: %s\n",
        strerror(errno));
    exit(EXIT_FAILURE);
}

strcpy(temp, line);
token = strtok(temp, DELIM);
while (token != NULL) {
    // check whether current token contains I/O redirection info
    if((token[0] == '>') || (token[0] == '<') || ((token[0] == '2') &&
        token[1] == '>')){
        redir[i] = token;
        i++;
    }
    else{
        arg[j] = token;
        j++;
    }
    token = strtok(NULL, DELIM);
}
redir[i] = NULL;
arg[j] = NULL;

line_arg = arg;
line_dir = redir;

status = exec_line(line_arg, line_dir);

free(temp);
free(arg);
free(redir);
}while(status);
}

char *readline(FILE *input){
    size_t len = 0;
    ssize_t nread;
    char *line = NULL;
    if((nread = getline(&line, &len, input)) == -1){
        fprintf(stderr, "Error while reading commands using getline: %s\n",
            strerror(errno));
        free(line);
        exit(EXIT_FAILURE);
    }
    return line;
}

```

```

int exec_line(char **arg, char **redir){
    if((arg[0] == NULL) || (strchr(arg[0], '#')!=NULL)){
        return 1;
    }
    else if((strcmp(arg[0], "cd")) == 0){
        if(arg[1] == NULL) {
            fprintf(stderr, "Error: Incorrect usage of cd command\n");
        }
        else {
            if(chdir(arg[1]) != 0){
                fprintf(stderr, "Error while doing cd command: %s\n",
                    strerror(errno));
            }
        }
        return 1;
    }
    else if((strcmp(arg[0], "pwd")) == 0){
        char cwd[BUFSIZ];
        if((getcwd(cwd, sizeof(cwd))) == NULL) {
            fprintf(stderr, "Error while doing pwd command: %s\n", strerror(errno));
        } else {
            printf("Current working directory is %s\n", cwd);
        }
        return 1;
    }
    else if((strcmp(arg[0], "exit")) == 0){
        if(arg[1] == NULL) {
            exit(0);
        } else {
            exit(atoi(arg[1]));
        }
    }
    return run_line(arg, redir);
}

```

```

int run_line(char **arg, char **redir){
    pid_t pid, wpid;
    int status;
    struct rusage stats;
    struct timeval start, end;
    gettimeofday(&start, NULL);
    pid = fork();
    if(pid == 0){
        // in children
        char *dir;

        for(int i = 0; redir[i] != NULL; i++){
            dir = redir[i];
            if(strstr(dir, "2>>")){
                dir = dir + 3;
                if (redirect(dir, 2, O_RDWR|O_APPEND|O_CREAT, 0666))
                    exit(1);
            }
            else if(strstr(dir, ">>")){

```

```

        dir = dir + 2;
        if(redirect(dir, 1, O_RDWR|O_APPEND|O_CREAT, 0666))
            exit(1);
    }
    else if(strstr(dir, "2>")){
        dir = dir + 2;
        if(redirect(dir, 2, O_RDWR|O_TRUNC|O_CREAT, 0666))
            exit(1);
    }
    else if(strstr(dir, "<")){
        dir = dir + 1;
        if(redirect(dir, 0, O_RDONLY, 0666))
            exit(1);
    }
    else if(strstr(dir, ">")){
        dir = dir + 1;
        if(redirect(dir, 1, O_RDWR|O_TRUNC|O_CREAT, 0666))
            exit(1);
    }
}

if(execvp(*arg, arg) == -1){
    fprintf(stderr, "Error while executing command %s: %s\n", arg[0],
        strerror(errno));
    exit(EXIT_FAILURE);
}
exit(EXIT_FAILURE);
}
else if(pid < 0){
    fprintf(stderr, "Error while forking the process %s: %s\n", arg[0], strerror
        (errno));
    return -1;
}
else{
    wpid = wait3(&status, WUNTRACED, &stats);
    gettimeofday(&end, NULL);
    fprintf(stderr, "Executing command %s\n", arg[0]);
    fprintf(stderr, "Command returned with return code %d\n",
        WEXITSTATUS(status));
    fprintf(stderr, "Real time: %ld.%06d sec\n", (end.tv_sec-start.tv_sec),
        (end.tv_usec-start.tv_usec));
    fprintf(stderr, "User time: %ld.%06d sec\n", stats.ru_utime.tv_sec,
        stats.ru_utime.tv_usec);
    fprintf(stderr, "System time: %ld.%06d sec\n", stats.ru_stime.tv_sec,
        stats.ru_stime.tv_usec);
}
return 1;
}

```

```

int redirect(char *path, int init_fd, int options, mode_t mode) {
    int red_fd;
    if((red_fd=open(path, options, mode))<0){
        fprintf(stderr, "Error while opening file %s for redirection: %s\n", path,
            strerror(errno));
    }
}

```

```
        return 1;
    }
    if(dup2(red_fd, init_fd)<0){
        fprintf(stderr, "Error while duplicating file %s for redirection: %s\n",
            path, strerror(errno));
        return 1;
    }
    if(close(red_fd)<0){
        fprintf(stderr, "Error while closing file %s for redirection: %s\n", path,
            strerror(errno));
        return 1;
    }
    return 0;
}
```

```
#!/Users/guoweizhe/Documents/ECE-357/myshell/myshell/myshell/myshell
```

```
#This is an example of a shell script that your shell must execute correctly
#notice that lines starting with a # sign are ignored as comments!
#let's say this here file is called testme.sh.  you created it with say
#vi testme.sh ; chmod +x testme.sh
#you invoked it with
#./testme.sh
cat >cat.out
#at this point, type some lines at the keyboard, then create an EOF (Ctrl-D)
#your shell invoked the system cat command with output redirected to cat.out
cat cat.out
#you better see the lines that you just typed!
exit 123
#after your shell script exits, type echo $? from the UNIX system shell
#the value should be 123.  Since your shell just exited, the following
#bogus command should never be seen
```



```
#!/Users/guoweizhe/Documents/ECE-357/myshell/myshell/myshell/myshell
```

```
# another example, say it is called test2.sh
```

```
#you invoked it with
```

```
#./test2.sh <input.txt
```

```
cat >cat2.out
```

```
#since you invoked the shell script (via the system shell such as bash)
```

```
#with stdin redirected, your shell runs cat which gets stdin from input.txt
```

```
exit
```

```
#the above exit had no specified return value, so your shell exited with 0
```

```
#because the last child spawned, cat, would have returned 0
```

```
#again, test this with echo $?
```