Dhvanil Shah
ECE-357
Prof. Hakner
21 October 2018

Programming Assignment 3

“myshell” catching invalid commands and ignoring comments and blank lines

```
dhvanils-mbp:code dhvanil$ ./myshell
Hello this is an invalid command
Error: Unable to execute the command [Hello]. Error code 2: No such file or directory.
Command returned with the return code: 1,
consuming 0.003793 real seconds, 0.000357 user, 0.001109 system


#this line is going to be ignored
#you can also skip lines

#as shown here
```

“myshell” running its built-in commands

```
dhvanils-mbp:code dhvanil$ ./myshell
#Now running built in commands
pwd
/Users/dhvanil/Documents/OS/prog3/code
cd ../../prog2
pwd
/Users/dhvanil/Documents/OS/prog2
cd
pwd
/Users/dhvanil
exit 123
dhvanils-mbp:code dhvanil$ echo $?
123
dhvanils-mbp:code dhvanil$ ./myshell
exit
dhvanils-mbp:code dhvanil$ echo $?
0
dhvanils-mbp:code dhvanil$
```

“myshell” running test.sh (attached)

```
dhvanils-mbp:code dhvanil$ ./myshell test.sh
Hello, this is a test of myshell running .sh files!
I hope this works...
Command returned with the return code: 0,
consuming 30.019280 real seconds, 0.001110 user, 0.002510 system
Hello, this is a test of myshell running .sh files!
I hope this works...
Command returned with the return code: 0,
consuming 0.003758 real seconds, 0.001286 user, 0.001805 system
dhvanils-mbp:code dhvanil$
```

“myshell” redirecting test.sh error to “stats.txt”

```
dhvanils-mbp:code dhvanil$ ./myshell test.sh 2>stats.txt
Hello, this is the same example. Now the stats are redirected to
the file stats.txt
Hello, this is the same example. Now the stats are redirected to
the file stats.txt
dhvanils-mbp:code dhvanil$ echo $?
123
```

stats.txt

```
Command returned with the return code: 0,
consuming 43.718259 real seconds, 0.001065 user, 0.001737 system
Command returned with the return code: 0,
consuming 0.003799 real seconds, 0.001292 user, 0.001711 system
```

“myshell” running test2.sh (attached) with input redirected to input.txt

```
dhvanils-mbp:code dhvanil$ cat input.txt
Hello Word
This is my input test file

The output text file should--
look the same as this.
dhvanils-mbp:code dhvanil$ ./myshell test2.sh <input.txt
Command returned with the return code: 0,
consuming 0.003369 real seconds, 0.001100 user, 0.001602 system
dhvanils-mbp:code dhvanil$ echo $?
0
dhvanils-mbp:code dhvanil$ diff input.txt cat2.out
dhvanils-mbp:code dhvanil$
```

“myshell” running commands (error redirected to stats.txt for readability)

```
dhvanils-mbp:code dhvanil$ ./myshell 2>>stats.txt
ls
Makefile        cat2.out        myshell         myshell.o       test.sh
cat.out         input.txt       myshell.c       stats.txt       test2.sh
ls -l
total 120
-rw-r--r--  1 dhvanil  staff    221 Oct 19 17:24 Makefile
-rw-r--r--  1 dhvanil  staff     73 Oct 21 13:53 cat.out
-rw-r--r--  1 dhvanil  staff     92 Oct 21 13:48 cat2.out
-rw-r--r--  1 dhvanil  staff     92 Oct 21 13:47 input.txt
-rwxr-xr-x  1 dhvanil  staff  14044 Oct 21 13:28 myshell
-rw-r--r--  1 dhvanil  staff   6473 Oct 21 13:28 myshell.c
-rw-r--r--  1 dhvanil  staff   5852 Oct 21 13:28 myshell.o
-rw-r--r--@ 1 dhvanil  staff    466 Oct 21 13:57 stats.txt
-rwxr-xr-x  1 dhvanil  staff    808 Oct 21 10:07 test.sh
-rw-r--r--  1 dhvanil  staff    459 Oct 21 10:06 test2.sh
mkdir testdir
ls
Makefile        input.txt       myshell.o       test2.sh
cat.out         myshell         stats.txt       testdir
cat2.out        myshell.c       test.sh
cd testdir
pwd
/Users/dhvanil/Documents/OS/prog3/code/testdir
cd ../
rmdir testdir
ls
Makefile        cat2.out        myshell         myshell.o       test.sh
cat.out         input.txt       myshell.c       stats.txt       test2.sh

exit
dhvanils-mbp:code dhvanil$ echo $?
0
dhvanils-mbp:code dhvanil$
```

stats.txt

```
Command returned with the return code: 0,
consuming 0.004562 real seconds, 0.001451 user, 0.002126 system
Command returned with the return code: 0,
consuming 0.009659 real seconds, 0.002032 user, 0.003205 system
Command returned with the return code: 0,
consuming 0.003655 real seconds, 0.001158 user, 0.001577 system
Command returned with the return code: 0,
consuming 0.004895 real seconds, 0.001451 user, 0.002694 system
Command returned with the return code: 0,
consuming 0.003786 real seconds, 0.001167 user, 0.001659 system
Command returned with the return code: 0,
consuming 0.004473 real seconds, 0.001430 user, 0.002102 system
```

“myshell” performing I/O redirection

```
cat <input.txt
Hello Word
This is my input test file

The output text file should--
look the same as this.


ls -l >out.txt
cat out.txt
total 120
-rw-r--r--  1 dhvanil  staff    221 Oct 19 17:24 Makefile
-rw-r--r--  1 dhvanil  staff     73 Oct 21 13:53 cat.out
-rw-r--r--  1 dhvanil  staff     92 Oct 21 13:48 cat2.out
-rw-r--r--  1 dhvanil  staff     92 Oct 21 13:47 input.txt
-rwxr-xr-x  1 dhvanil  staff  14044 Oct 21 13:28 myshell
-rw-r--r--  1 dhvanil  staff   6473 Oct 21 13:28 myshell.c
-rw-r--r--  1 dhvanil  staff   5852 Oct 21 13:28 myshell.o
-rw-r--r--  1 dhvanil  staff      0 Oct 21 14:12 out.txt
-rw-r--r--@ 1 dhvanil  staff    106 Oct 21 14:11 stats.txt
-rwxr-xr-x  1 dhvanil  staff    808 Oct 21 10:07 test.sh
-rw-r--r--  1 dhvanil  staff    459 Oct 21 10:06 test2.sh
```

```
ls >>out.txt
cat out.txt
total 120
-rw-r--r--  1 dhvanil  staff    221 Oct 19 17:24 Makefile
-rw-r--r--  1 dhvanil  staff     73 Oct 21 13:53 cat.out
-rw-r--r--  1 dhvanil  staff     92 Oct 21 13:48 cat2.out
-rw-r--r--  1 dhvanil  staff     92 Oct 21 13:47 input.txt
-rwxr-xr-x  1 dhvanil  staff  14044 Oct 21 13:28 myshell
-rw-r--r--  1 dhvanil  staff   6473 Oct 21 13:28 myshell.c
-rw-r--r--  1 dhvanil  staff   5852 Oct 21 13:28 myshell.o
-rw-r--r--  1 dhvanil  staff      0 Oct 21 14:12 out.txt
-rw-r--r--@ 1 dhvanil  staff    106 Oct 21 14:11 stats.txt
-rwxr-xr-x  1 dhvanil  staff    808 Oct 21 10:07 test.sh
-rw-r--r--  1 dhvanil  staff    459 Oct 21 10:06 test2.sh
Makefile
cat.out
cat2.out
input.txt
myshell
myshell.c
myshell.o
out.txt
stats.txt
test.sh
test2.sh

cat -fakearg 2>error.txt
cat error.txt
cat: illegal option -- f
usage: cat [-benstuv] [file ...]

cat -fakearg2 2>>error.txt
cat error.txt
cat: illegal option -- f
usage: cat [-benstuv] [file ...]
cat: illegal option -- f
usage: cat [-benstuv] [file ...]
```

```
#!/absolute/path/to/your/shell
#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
#######################################################################

```

```
#!/absolute/path/to/your/shell
#here is 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 $?
```

```c
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/wait.h>

void throwError(char *message, char *file)
{
    if (file)
        fprintf(stderr, "%s [%s]. Error code %i: %s.\n", message, file, errno, strerror(errno));
    else
        fprintf(stderr, "%s\n", message);
}

int mycd(char *path)
{
    char *location;
    if (path == NULL)
    {
        location = getenv("HOME");
    }
    else
    {
        location = path;
    }

    if (chdir(location) < 0)
    {
        throwError("Error: Could not change to directory", path);
    }
    return 0;
}

void myexit(char *code)
{
    if (code != NULL)
        exit(atoi(code));
    exit(0);
}

int mypwd()
{
    char cwd[4096];
```

```c

    if (getcwd(cwd, sizeof(cwd)) != NULL)
    {
        printf("%s\n", cwd);
    }
    else
    {
        throwError("Error: Could not print current directory due to an error with getcwd().", NULL);
    }
    return 0;
}

int run(char **argVec, char *redirFile, int redirFD, int redirMode)
{
    int fd, status;
    pid_t process, waiting;
    struct rusage rusage;
    struct timeval start, end;

    if ((gettimeofday(&start, NULL)) < 0)
    {
        throwError("Error: Unable to start timing command execution.", NULL);
        return 1;
    }
    switch ((process = fork()))
    {
    case -1:
        throwError("Error: Unable to execute command due to an error in the fork process", NULL);
        exit(1);
        break;
    case 0:
        if (redirFD > -1)
        {
            if ((fd = open(redirFile, redirMode, 0666)) < 0)
            {
                throwError("Error: Unable to open file for redirection", redirFile);
                return 1;
            }
            if (dup2(fd, redirFD) < 0)
            {
                throwError("Error: Unable to redirect to appropriate file.", NULL);
                return 1;
            }
            if (close(fd) < 0)
            {

```

```
                throwError("Error: Unable to close redirected file descriptor.", NULL);
                return 1;
            }
        }
        if (execvp(argVec[0], argVec) == -1)
        {
            throwError("Error: Unable to execute the command", argVec[0]);
            return 1;
        }
        //we should never get here
        break;
    default:
        if ((waiting = wait3(&status, WUNTRACED, &rusage)) < 0)
        {
            throwError("Error: Unable to wait for completion of child process", argVec[0]);
            return 1;
        };
        if ((gettimeofday(&end, NULL)) < 0)
        {
            throwError("Error: Unable to start timing command execution.", NULL);
            return 1;
        }
        fprintf(stderr, "Command returned with the return code: %i,\n", WEXITSTATUS(status));
        fprintf(stderr, "consuming %ld.%06u real seconds, %ld.%06u user, %ld.%06u system\n", (end.tv_sec - start.tv_sec), (end.tv_usec - start.tv_usec), rusage.ru_utime.tv_sec, rusage.ru_utime.tv_usec, rusage.ru_stime.tv_sec, rusage.ru_stime.tv_usec);
        break;
    }
    return 0;
}

void myshell(FILE *infile)
{
    char *line = NULL, *delims = " \r\n", *arg;
    int redirFD = -1, redirMode, i = 0;
    size_t len = 0;
    ssize_t nread;
    char *redirFile = malloc(BUFSIZ * (sizeof(char)));
    if (redirFile == NULL)
        throwError("Error: Failure to dynamically allocate memory.", NULL);
    char **argVec = malloc(BUFSIZ * (sizeof(char *)));
    if (argVec == NULL)
        throwError("Error: Failure to dynamically allocate memory.", NULL);

    while ((nread = getline(&line, &len, infile)) != -1)
```

```
    {
        if (line[0] == '#' || nread <= 1)
        {
            continue;
        }
        else
        {
            arg = strtok(line, delims);
            while (arg != NULL)
            {
                if (arg[0] == '<')
                {
                    redirFD = 0;
                    redirMode = O_RDONLY;
                    strcpy(redirFile, (arg + 1));
                }
                else if (arg[0] == '>')
                {
                    if (arg[1] == '>')
                    {
                        redirMode = O_WRONLY | O_APPEND | O_CREAT;
                        strcpy(redirFile, (arg + 2));
                    }
                    else
                    {
                        redirMode = O_WRONLY | O_TRUNC | O_CREAT;
                        strcpy(redirFile, (arg + 1));
                    }
                    redirFD = 1;
                }
                else if (arg[0] == '2' && arg[1] == '>')
                {
                    if (arg[2] == '>')
                    {
                        redirMode = O_WRONLY | O_APPEND | O_CREAT;
                        strcpy(redirFile, (arg + 3));
                    }
                    else
                    {
                        redirMode = O_WRONLY | O_TRUNC | O_CREAT;
                        strcpy(redirFile, (arg + 2));
                    }
                    redirFD = 2;
                }
                else
                {
                    argVec[i++] = arg;
                }
                arg = strtok(NULL, delims);

```

```
            }
            argVec[i] = NULL;
            if (strcmp(argVec[0], "pwd") == 0)
            {
                mypwd();
            }
            else if (strcmp(argVec[0], "cd") == 0)
            {
                mycd(argVec[1]);
            }
            else if (strcmp(argVec[0], "exit") == 0)
            {
                myexit(argVec[1]);
            }
            else
            {
                if ((run(argVec, redirFile, redirFD, redirMode)) > 0)
                {
                    exit(1);
                }
            }

            redirFD = -1;
            redirMode = 0;
            i = 0;
        }
    }

    free(redirFile);
    free(argVec);
    free(line);
    return;
}

int main(int argc, char *argv[])
{
    FILE *infile;

    if (argc > 1)
    {
        if ((infile = fopen(argv[1], "r")) == NULL)
        {
            throwError("Error: Unable to open input file", argv[1]);
            return -1;
        }
        myshell(infile);
    }
    else
    {
```

```c

        myshell(stdin);
    }
    if (argc > 1 && fclose(infile) != 0)
    {
        throwError("Error: Unable to close input file", argv[1]);
        return -1;
    }
    fprintf(stderr, "EOF Characted Detected. Exiting myshell\n");
    return 0;
}
```