ECE357, Computer Operating Systems – Problem Set #3, Problem 2

Nikola Janjušević

October 21, 2018

1 Example Usage

1.1 direct use

```
nikopj@s15:~/Documents/OS/hw3$ ./myshell.out
$ echo hello!
hello!
child process 28924:
    exit status: 0
     usr time: 0.000000s
     sys time: 0.001332
    real time: 0.001451s
$ pwd
/home/nikopj/Documents/OS/hw3
$ cd ..
$ ls -la > ls.out
child process 28930:
    exit status: 0
     usr time: 0.000000s
     sys time: 0.003688
    real time: 0.003801s
$ cat ls.out
total 28
drwxrwxr-x 6 nikopj nikopj 4096 Oct 21 21:07 .
drwxr-xr-x 7 nikopj nikopj 4096 Oct 18 21:50 ..
drwxrwxr-x 8 nikopj nikopj 4096 Oct 21 15:27 .git
drwxrwxr-x 3 nikopj nikopj 4096 Oct 17 16:04 hw2
drwxrwxr-x 2 nikopj nikopj 4096 Oct 21 21:11 hw3
-rw-rw-r-- 1 nikopj nikopj    0 Oct 21 21:14 ls.out
drwxrwxr-x 4 nikopj nikopj 4096 Oct 17 16:04 MINICAT
-rw-rw-r-- 1 nikopj nikopj   40 Oct 17 16:04 README.md
child process 28931:
    exit status: 0
     usr time: 0.001332s
     sys time: 0.000000
    real time: 0.001523s
$ cat doesnt_exist.txt
cat: doesnt_exist.txt: No such file or directory
child process 28934:
    exit status: 1
     usr time: 0.001404s
     sys time: 0.000000
    real time: 0.001602s
$ exit
nikopj@s15:~/Documents/OS/hw3$ echo $?
1
```

1.2 script interpreter

```
nikopj@s15:~/Documents/OS/hw3$ ./test.sh
hello!
bye!
child process 28966:
    exit status: 0
     usr time: 0.001555s
     sys time: 0.000000
    real time: 7.7359062s
hello!
bye!
child process 28967:
```

```
    exit status: 0
     usr time: 0.001335s
     sys time: 0.000000
    real time: 0.001482s
$ $ $ $ $ nikopj@s15:~/Documents/OS/hw3$ echo $?
123
```

1.2.1 test.sh

```
#!/home/nikopj/Documents/OS/hw3/myshell.out
# this is a comment
cat > cat.out
cat cat.out
exit 123
this should not be seen!
```

1.3 script interpreter 2

```
nikopj@s15:~/Documents/OS/hw3$ ./test2.sh < p1.txt 2> log.out
$ $ $ $ nikopj@s15:~/Documents/OS/hw3$ ls
cat2.out  cat.out  log.out  Makefile  myshell.c  myshell_debug.out  myshell.out  p1.
    txt  test1.txt  test2.sh  test2.txt  test.sh
nikopj@s15:~/Documents/OS/hw3$ cat cat2.out
OS PS3 P1
Nikola Janjusevic
1)
/bin/sh
2)
argv = ["sh","-vx","/tmp/script.sh","file1.c","file2.c"]
argc = 5
3)
/bin/ls
4)
argv = ["ls","-l","file1.c","file2.c"]
argc = 4
nikopj@s15:~/Documents/OS/hw3$ cat log.out
child process 28977:
    exit status: 0
     usr time: 0.000000s
     sys time: 0.001220
    real time: 0.001304s
```

1.3.1 test2.sh

```
#!/home/nikopj/Documents/OS/hw3/myshell.out
# this is another comment
cat > cat2.out
exit
```

Program: myshell.c

```
1  // myshell.c
2  #include <unistd.h>
3  #include <stdio.h>
4  #include <errno.h>
5  #include <string.h>
6  #include <stdlib.h>
7  // for wait3()
8  #include <sys/types.h>
9  #include <sys/time.h>
10 #include <sys/resource.h>
11 #include <sys/wait.h>
12 #include <sys/stat.h>
13 #include <fcntl.h>
14
15 #define BUFF_SIZE 2048
16 #define DEBUG 0
17
18 void eprint_tokens(const char *msg, char **tokv);
19 int io_handle(char **tokv);
20 int io_redir(const char *filename, const int rfd, int flags, mode_t mode);
21
22 int main(int argc, char **argv)
23 {
24     pid_t pid;
25     int wstatus, exstat, tcheck;
26     struct timeval tstart, tstop; time_t elps_ts; suseconds_t elps_tu;
27     char *delimiter = " \t\n";
28     char *line;
29     char *ptokes[BUFF_SIZE]; // pointers to tokens
30     struct rusage usage;
31     FILE *fstream;
32
33     if( (line = malloc(BUFF_SIZE)) == NULL )
34     {
35         perror("line malloc failure");
36         exit(-1);
37     }
38
39     // if no input file is given or it cannot be opened
40     // set the input to stdin
41     if( argv[1]==NULL || (fstream=fopen(argv[1],"r"))==NULL )
42     {
43         if( argv[1]!=NULL )
44             perror("fopen input error");
45         fstream = stdin;
46     }
47
48     while(1)
49     {
50         printf("$ ");
51
52         if( fgets(line, BUFF_SIZE, fstream) != NULL )
53         {
54             // tokenize input
55             int i;
56             ptokes[i=0] = strtok(line, delimiter);
57             while(ptokes[i]!=NULL)
58             {
59                 ptokes[++i] = strtok(NULL, delimiter);
60             }
61             if( ptokes[0]==NULL )
62                 continue;
63
```

```
64              // DEBUGGING
65              if(DEBUG)
66                  eprint_tokens("+ cmd: ", ptokes);
67
68              // parse tokens
69              if( strcmp(ptokes[0],"exit") == 0 ) // EXIT
70              {
71                  if( ptokes[1]!=NULL )
72                      exstat = strtol(ptokes[1], NULL, 10);
73                  free(line);
74                  exit(exstat);
75              }
76              else if( strncmp(ptokes[0],"#",1) == 0 ) // COMMENT
77                  continue;
78              else if( strcmp(ptokes[0],"cd")==0 && ptokes[1]!=NULL ) // CD
79              {
80                  if( chdir(ptokes[1]) == -1 )
81                      perror("cd");
82                  continue;
83              }
84              else if( strcmp(ptokes[0],"pwd")==0 && ptokes[1]==NULL ) // PWD
85              {
86                  char *cwd;
87                  if( (cwd = malloc(BUFF_SIZE)) == NULL )
88                  {
89                      perror("cwd malloc failure");
90                      continue;
91                  }
92                  else
93                  {
94                      if( (getcwd(cwd, BUFF_SIZE)) != NULL )
95                          printf("%s\n",cwd);
96                      else
97                          perror("getcwd error");
98                      free(cwd);
99                  }
100             }
101             else // FORK -> EXEC
102             {
103                 if( (tcheck = gettimeofday(&tstart,NULL)) == -1 )
104                     perror("gettimeofday error @ start");
105                 switch( pid=fork() )
106                 {
107                 case -1:
108                     perror("fork error");
109                     break;
110
111                 case 0:
112                     // close input file
113                     if( fclose(fstream)!=0 )
114                     {
115                         perror("input filestream close in child failed");
116                         free(line);
117                         exit(-1);
118                     }
119                     if(DEBUG)
120                         eprint_tokens("+ child tokes before: ", ptokes);
121                     if( io_handle(ptokes)!=0 ) // IO REDIRECTION HANDLER
122                     {
123                         fprintf(stderr,"IO redirection failure\n");
124                         free(line);
125                         exit(-1);
126                     }
127                     if(DEBUG)
128                         eprint_tokens("+ child tokes after: ", ptokes);
129                     execvp(ptokes[0], ptokes);
```

```
130                     perror("exec failed");
131                     free(line);
132                     exit(-1);
133 
134                 default:
135                     if(DEBUG)
136                         fprintf(stderr, "+ in parent, child pid is %d\n", pid);
137                     if( wait3(&wstatus, 0, &usage) == -1 )
138                         perror("wait3 error");
139                     else
140                     {
141                         if( tcheck==-1 || gettimeofday(&tstop,NULL)==-1 )
142                         {
143                             elps_ts=-1; elps_tu=0;
144                             perror("gettimeofday error @ stop, real time <- -1.0");
145                         }
146                         else
147                         {
148                             elps_ts = tstop.tv_sec - tstart.tv_sec;
149                             elps_tu = 1000000*tstop.tv_sec + tstop.tv_usec \
150                                 - 1000000*tstart.tv_sec - tstart.tv_usec;
151                         }
152                         fprintf(stderr,"child process %d:\n\texit status: %d\n"
153                             "\t usr time: %ld.%06lds \n\t sys time: %ld.%06ld\n"
154                             "\treal time: %ld.%06lds\n", \
155                             pid, exstat=WEXITSTATUS(wstatus), \
156                             usage.ru_utime.tv_sec, usage.ru_utime.tv_usec, \
157                             usage.ru_stime.tv_sec, usage.ru_stime.tv_usec, \
158                             elps_ts, elps_tu);
159                     }
160                 }
161             }
162         }
163         else if( errno !=0 )
164             perror("fgets line");
165         else printf("\n"); // continue to next line
166     }
167     free(line);
168     return 0;
169 }
170 // prints tokens pointed to by tokv to stderr
171 // msg is printed before tokens
172 void eprint_tokens(const char *msg, char **tokv)
173 {
174     int i=0;
175     fprintf(stderr, "%s", msg);
176     while(tokv[i]!=NULL)
177         fprintf(stderr, "%s ", tokv[i++]);
178     fprintf(stderr, "\n");
179 }
180 
181 // switch cases for IO redirection
182 // calls io_redir() if necessary
183 int io_handle(char **tokv)
184 {
185     int i=0;
186     int ret=0;
187     while(tokv[i]!=NULL)
188     {
189         if(tokv[i+1]!=NULL)
190         {
191             if(strcmp(tokv[i],"<")==0)
192             {
193                 if(DEBUG)
194                     fprintf(stderr,"+ rd stdin to %s\n",tokv[i+1]);
195                 ret = ret + io_redir(tokv[i+1], STDIN_FILENO, O_RDONLY, 0444);
```

```
196                 tokv[i++]=NULL;
197             }
198             else if(strcmp(tokv[i],">")==0)
199             {
200                 if(DEBUG)
201                     fprintf(stderr,"+ rd stdout to %s\n",tokv[i+1]);
202                 ret = ret + io_redir(tokv[i+1],STDOUT_FILENO,O_CREAT|O_TRUNC|
                        O_WRONLY,0666);
203                 tokv[i++]=NULL;
204             }
205             else if(strcmp(tokv[i],"2>")==0)
206             {
207                 if(DEBUG)
208                     fprintf(stderr,"+ rd stdder to %s\n",tokv[i+1]);
209                 ret = ret + io_redir(tokv[i+1],STDERR_FILENO,O_CREAT|O_TRUNC|
                        O_WRONLY,0666);
210                 tokv[i++]=NULL;
211             }
212             else if(strcmp(tokv[i],">>")==0)
213             {
214                 if(DEBUG)
215                     fprintf(stderr,"+ rd stout to %s\n",tokv[i+1]);
216                 ret = ret + io_redir(tokv[i+1],STDOUT_FILENO,O_CREAT|O_APPEND|
                        O_WRONLY,0666);
217                 tokv[i++]=NULL;
218             }
219             else if(strcmp(tokv[i],"2>>")==0)
220             {
221                 if(DEBUG)
222                     fprintf(stderr,"+ rd stdin to %s\n",tokv[i+1]);
223                 ret = ret + io_redir(tokv[i+1],STDERR_FILENO,O_CREAT|O_APPEND|
                        O_WRONLY,0666);
224                 tokv[i++]=NULL;
225             }
226         }
227         i++;
228     }
229     return ret;
230 }
231
232 // IO redirection
233 int io_redir(const char *filename, const int rfd, int flags, mode_t mode)
234 {
235     int ofd;
236     if( (ofd=open(filename, flags, mode)) < 0 )
237     {
238         fprintf(stderr,"%s open error: %s\n",filename,strerror(errno));
239         return -1;
240     }
241     else if( dup2(ofd,rfd)!=-1 )
242     {
243         close(ofd);
244         return 0;
245     }
246     else
247         fprintf(stderr,"%s to fd=%d, dup2 error: %s\n",filename,rfd,strerror(errno))
            ;
248     return -1;
249 }
```