CSE 3033
OPERATING SYSTEMS
Programming Assignment # 2
DUE DATE: 31/12/2020 - 23:59PM

Muhammet Kürşat Açıkgöz - 150116020

Ahmet Elburuz Gürbüz - 150116024

Ahmet Önkol - 150117018

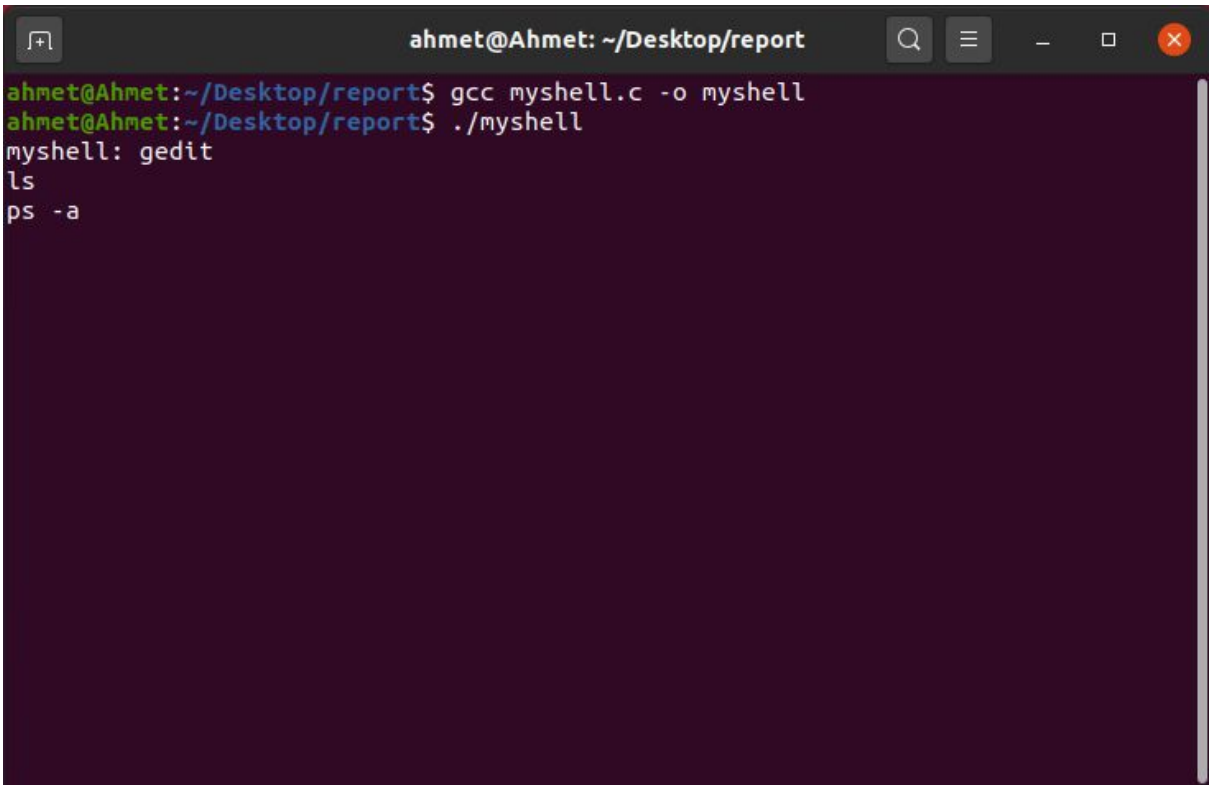
In this project, we are expected to create our own shell in C programming language containing special and built-in commands.

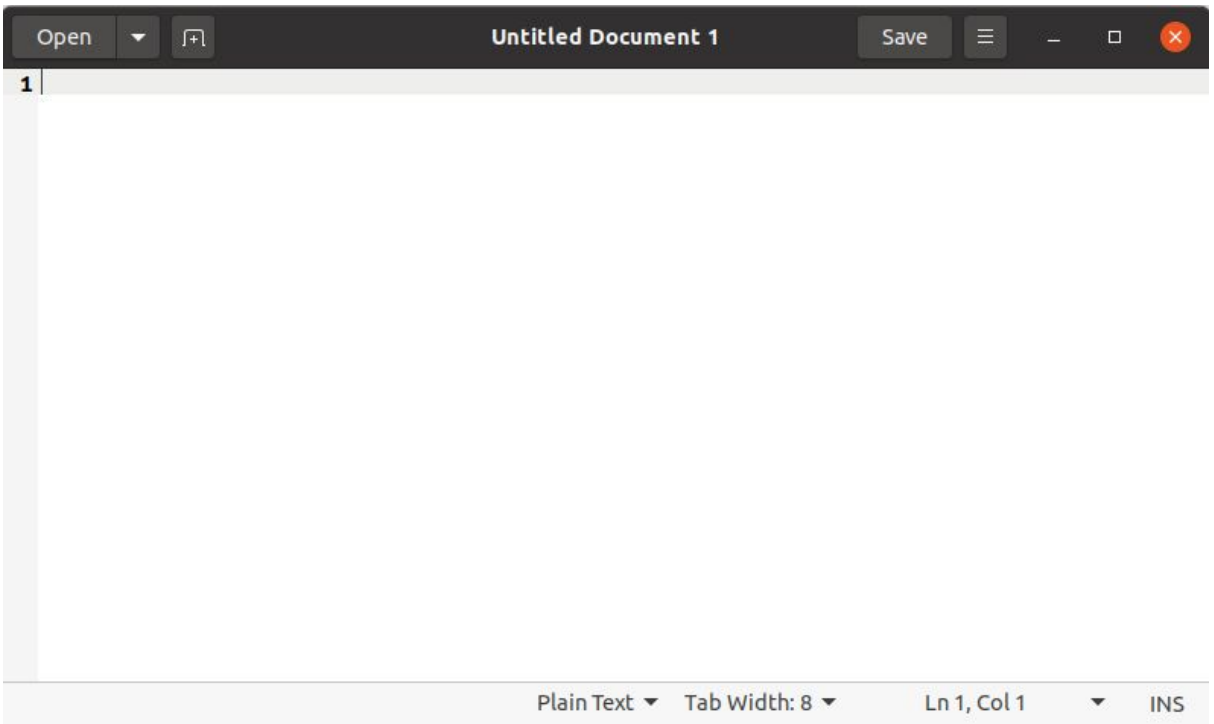
Part A

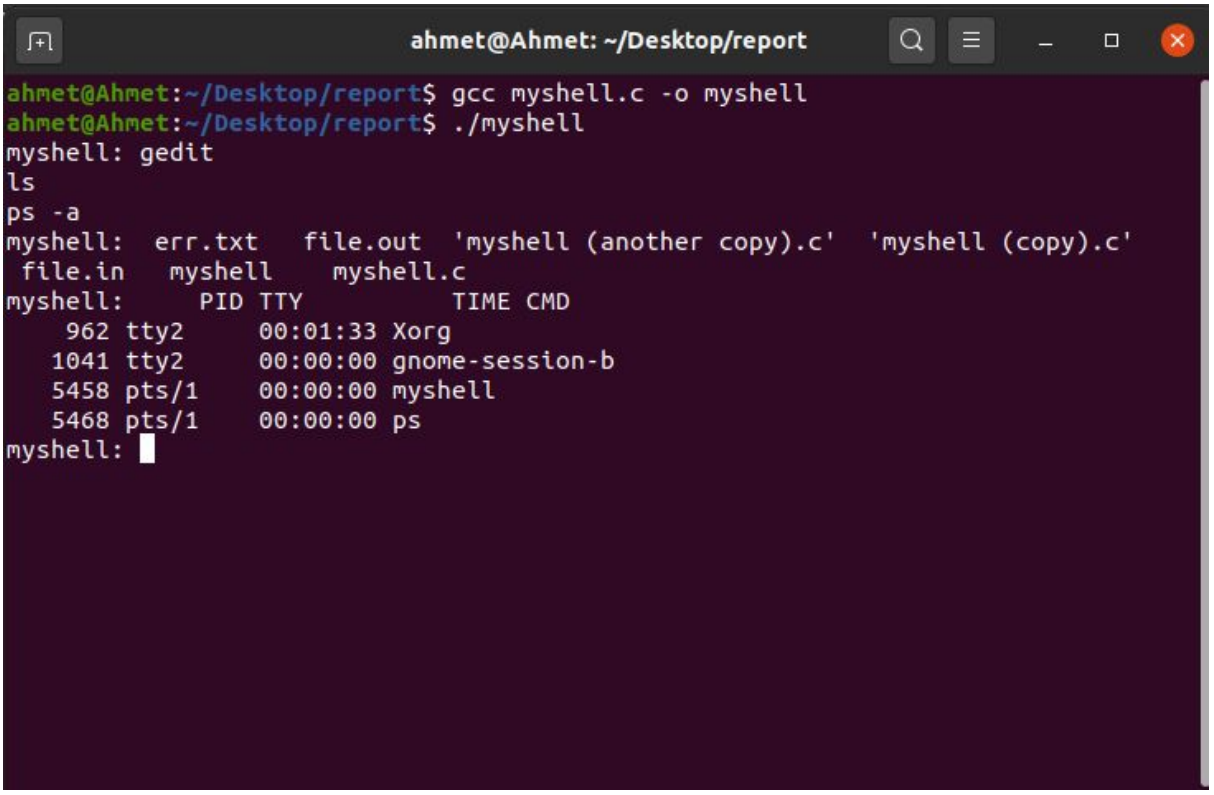
In the first part, if a command contains ‘&’ at the end of the args, the process will be executed in the background. So that we are able to perform other processes while executing previous processes simultaneously. With that way, the user can view the effects of the foreground process along with the previous process.

On the other hand, if the user doesn’t provide ‘&’ at the end of the main process, it means everything after the main process will be added to the queue and be executed from the list of background processes after execution of the main process is done.

myshell: gedit :




Open
Untitled Document 1
Save
1
Plain Text
Tab Width: 8
Ln 1, Col 1
INS


ahmet@Ahmet: ~/Desktop/report
ahmet@Ahmet:~/Desktop/report$ gcc myshell.c -o myshell
ahmet@Ahmet:~/Desktop/report$ ./myshell
myshell: gedit
ls
ps -a
myshell: err.txt file.out 'myshell (another copy).c' 'myshell (copy).c'
file.in myshell myshell.c
myshell: PID TTY TIME CMD
962 tty2 00:01:33 Xorg
1041 tty2 00:00:00 gnome-session-b
5458 pts/1 00:00:00 myshell
5468 pts/1 00:00:00 ps
myshell:

myshell: gedit & :

```
ahmet@Ahmet: ~/Desktop/report
ahmet@Ahmet:~/Desktop/report$ gcc myshell.c -o myshell
ahmet@Ahmet:~/Desktop/report$ ./myshell
myshell: gedit &

The process with the p_id = 5516 and the name = "gedit" is running
myshell: ls
 err.txt   file.out  'myshell (another copy).c'  'myshell (copy).c'
 file.in   myshell    myshell.c

The process with the p_id = 5516 and the name = "gedit" is running
myshell: ps -a
    PID TTY          TIME CMD
    962 tty2     00:01:37 Xorg
   1041 tty2     00:00:00 gnome-session-b
   5515 pts/1    00:00:00 myshell
   5516 pts/1    00:00:00 gedit
   5523 pts/1    00:00:00 ps

The process with the p_id = 5516 and the name = "gedit" is running
myshell:
```

```
Open    Untitled Document 1    Save
1
Plain Text    Tab Width: 8    Ln 1, Col 1    INS
```

Part B

In the second part, we are expected to design ps_all,ctrl+z,search, bookmark, and exit special commands. To achieve this goal we designed structures and linked lists and organized them accordingly. So overall we will have a full control on what processes should be executed next and which ones are terminated.

Ps_all command will list every background instruction whether it is terminated or not and informs users accordingly. Since we are storing every background process in a linked list with their name and ids by using structs, we can access and list them with their properties.

ctrl + z mechanism will stop the currently running foreground process and child processes which are forked. We basically use SIGTSTP (for signal - terminal stop) signal with kill command and required conditional checks since SIGTSTP is sent through the driver by a user typing on a keyboard, usually Control-z.

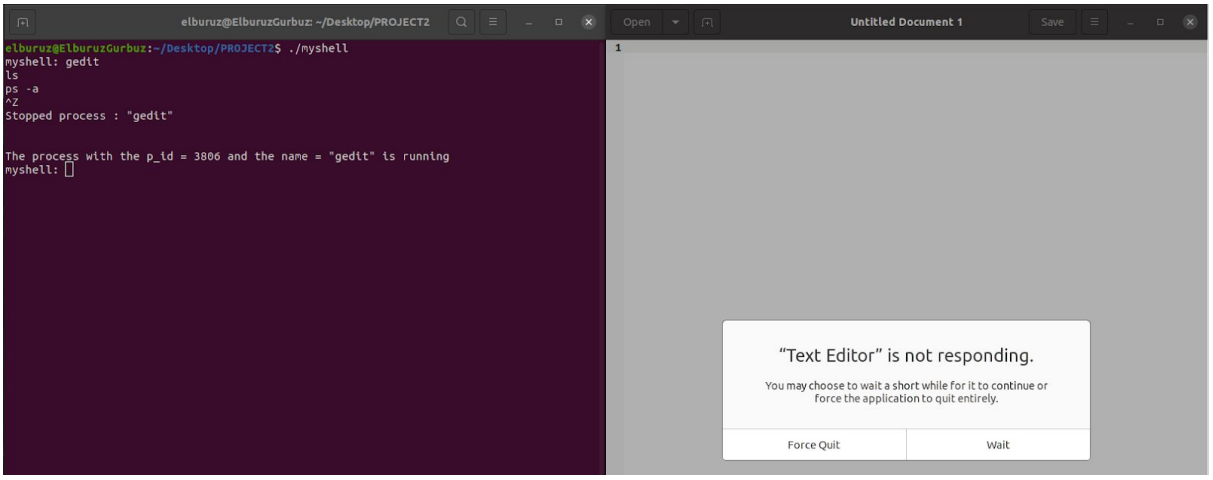
For ***searching*** in directories and current file, we basically call the same comparison function for both recursive and nonrecursive cases. We loop through every line of the current file and check if a match occurs in a given directory as long as the file is readable. If we find a match we will print the corresponding file name and the line number that match occurs.For recursive case, main idea is the same. The difference is we are also looking at subdirectories as well and do the same process again.

We received needed information and complete execution for ***bookmarks*** since we are using linked lists. We have a header pointer which will point to a list of bookmarks and in addition to that we have a current pointer which will be used to add new bookmarks to the bookmark linked list. After we are done with implementation of bookmarks linked list we can look for and execute accordingly.

Finally for ***exit*** command, we are expected to terminate our shell process. Requirements are set as given in project document such like if the user chooses to exit while there are background processes, we are notifying the user that there are background processes still running and do not terminate the process unless the user terminates all background processes.

ctrl + z :

```
elburuz@ElburuzGurbuz:~/Desktop/PROJECT2$ ./myshell
myshell: gedit
ls
ps -a
^Z
Stopped process : "gedit"
```




```
The process with the p_id = 3806 and the name = "gedit" is terminated
myshell:
```

ps_all :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    REFACTOR PREVIEW

ahmet@Ahmet:~/Desktop/report$ gcc myshell.c -o myshell
ahmet@Ahmet:~/Desktop/report$ ./myshell
myshell: gedit &

The process with the p_id = 5084 and the name = "gedit" is running
myshell: ps_all

The process with the p_id = 5084 and the name = "gedit" is running

The process with the p_id = 5084 and the name = "gedit" is running
myshell:
```

After closing newly opened document :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    REFACTOR PREVIEW

ahmet@Ahmet:~/Desktop/report$ gcc myshell.c -o myshell
ahmet@Ahmet:~/Desktop/report$ ./myshell
myshell: gedit &

The process with the p_id = 5084 and the name = "gedit" is running
myshell: ps_all

The process with the p_id = 5084 and the name = "gedit" is running

The process with the p_id = 5084 and the name = "gedit" is running
myshell: ps_all

The process with the p_id = 5084 and the name = "gedit" is terminated
myshell:
```

search:

```
elburuz@ElburuzGurbuz:~/Desktop/PROJECT2$ ./myshell
myshell: search
Input is not valid.
myshell: search -r
Input is not valid.
myshell: search running

Searching in file: test1.C
        396:            }else printf("There is no running background process\n"); // if there is no background process left

        942: // Stop the currently running foreground process , and it's childs when ctrl + Z is pressed

        952:            kill(fg_process_pid, SIGTSTP); // stop the running process


Searching in file: test2.h
        396:            }else printf("There is no running background process\n"); // if there is no background process left

        942: // Stop the currently running foreground process , and it's childs when ctrl + Z is pressed

        952:            kill(fg_process_pid, SIGTSTP); // stop the running process


Searching in file: myshell.c
        396:            }else printf("There is no running background process\n"); // if there is no background process left

        942: // Stop the currently running foreground process , and it's childs when ctrl + Z is pressed

        952:            kill(fg_process_pid, SIGTSTP); // stop the running process
```

```
myshell: search -r running

./test1.C ->
        396:            }else printf("There is no running background process\n"); // if there is no background process left

        942: // Stop the currently running foreground process , and it's childs when ctrl + Z is pressed

        952:            kill(fg_process_pid, SIGTSTP); // stop the running process


./test2.h ->
        396:            }else printf("There is no running background process\n"); // if there is no background process left

        942: // Stop the currently running foreground process , and it's childs when ctrl + Z is pressed

        952:            kill(fg_process_pid, SIGTSTP); // stop the running process


./myshell.c ->
        396:            }else printf("There is no running background process\n"); // if there is no background process left

        942: // Stop the currently running foreground process , and it's childs when ctrl + Z is pressed

        952:            kill(fg_process_pid, SIGTSTP); // stop the running process


./test4.c ->
        396:            }else printf("There is no running background process\n"); // if there is no background process left

        942: // Stop the currently running foreground process , and it's childs when ctrl + Z is pressed

        952:            kill(fg_process_pid, SIGTSTP); // stop the running process


./test3.H ->
        396:            }else printf("There is no running background process\n"); // if there is no background process left

        942: // Stop the currently running foreground process , and it's childs when ctrl + Z is pressed

        952:            kill(fg_process_pid, SIGTSTP); // stop the running process
```

bookmark:

```
elburuz@ElburuzGurbuz:~/Desktop/PROJECT2$ ./myshell
myshell: bookmark
Input is invalid
myshell: bookmark "ls -l"
myshell: bookmark "ps -a"
myshell: bookmark -l
0       "ls -l"
1       "ps -a"
myshell: bookmark -i 1
    PID TTY          TIME CMD
   1256 tty2     00:00:06 Xorg
   1265 tty2     00:00:00 gnome-session-b
   2371 pts/0    00:00:00 myshell
   2402 pts/0    00:00:00 ps
myshell: bookmark -d 0
myshell: bookmark -l
0       "ps -a"
myshell: bookmark -d 2
Not found bookmark with 2
myshell: bookmark -i 2
Bookmark not found with 2
myshell:
```

exit :

```
elburuz@ElburuzGurbuz: ~/Desktop/PROJECT2
elburuz@ElburuzGurbuz:~/Desktop/PROJECT2$ ./myshell
myshell: bookmark "ls -l"
myshell: exit
There is no running background process
elburuz@ElburuzGurbuz:~/Desktop/PROJECT2$ ./myshell
myshell: gedit &

The process with the p_id = 3970 and the name = "gedit" is running
myshell: exit

The process with the p_id = 3970 and the name = "gedit" is running
1 processes are not terminated. Terminate them before exiting the shell.

The process with the p_id = 3970 and the name = "gedit" is running
myshell:
```

File is not terminated immediately:

```
Open    Untitled Document 1    Save
1
```

Part C

In the third part, we are expected to design input output redirection and work on them. Initially we are storing args that the user enters and work accordingly. Since we are going to open and close several files, we decided to use flags and deal with reading, writing, opening and closing issues with the help of built-in libraries of C.

myshell: myprog [args] > file.out :

```
ahmet@Ahmet: ~/Desktop/report
ahmet@Ahmet:~/Desktop/report$ gcc myshell.c -o myshell
ahmet@Ahmet:~/Desktop/report$ ./myshell
myshell: ls
 err.txt   file.out  'myshell (another copy).c'  'myshell (copy).c'
 file.in   myshell    myshell.c
myshell: ls > file.out
myshell: cat file.out
err.txt
file.in
file.out
myshell
myshell (another copy).c
myshell.c
myshell (copy).c
```

```
Open   file.out  ~/Desktop/report   Save
1 err.txt
2 file.in
3 file.out
4 myshell
5 myshell (another copy).c
6 myshell.c
7 myshell (copy).c
Plain Text   Tab Width: 8   Ln 1, Col 1   INS
```

myshell: myprog [args] < file.in (below)

```
myshell: sort < file.in
a
b
c
d
f
s
x
z
myshell: cat file.in
a
d
c
s
f
x
z
b
myshell:
```

myshell: myprog [args] 2> file.out (below)

```
ahmet@Ahmet: ~/Desktop/report
ahmet@Ahmet:~/Desktop/report$ gcc myshell.c -o myshell
ahmet@Ahmet:~/Desktop/report$ ./myshell
myshell: ls 2> err.txt
 err.txt   file.out  'myshell (another copy).c'  'myshell (copy).c'
 file.in   myshell    myshell.c
myshell: ps -a 2> err.txt
    PID TTY          TIME CMD
    962 tty2     00:02:02 Xorg
   1041 tty2     00:00:00 gnome-session-b
   5896 pts/1    00:00:00 myshell
   5966 pts/1    00:00:00 ps
myshell: cat err.txt
myshell:
```

```
Open   err.txt
       ~/Desktop/report   Save
1
Plain Text   Tab Width: 8   Ln 1, Col 1   INS
```

myshell: myprog [args] >> file.out (below)

```
ahmet@Ahmet: ~/Desktop/report
ahmet@Ahmet:~/Desktop/report$ gcc myshell.c -o myshell
ahmet@Ahmet:~/Desktop/report$ ./myshell
myshell: ps -a > file.out
myshell: cat file.out
    PID TTY          TIME CMD
    962 tty2     00:02:08 Xorg
   1041 tty2     00:00:00 gnome-session-b
   6046 pts/1    00:00:00 myshell
   6054 pts/1    00:00:00 ps
myshell: ls
 err.txt   file.out  'myshell (another copy).c'  'myshell (copy).c'
 file.in   myshell    myshell.c
myshell: ls >> file.out
myshell: cat file.out
    PID TTY          TIME CMD
    962 tty2     00:02:08 Xorg
   1041 tty2     00:00:00 gnome-session-b
   6046 pts/1    00:00:00 myshell
   6054 pts/1    00:00:00 ps
err.txt
file.in
file.out
myshell
myshell (another copy).c
myshell.c
myshell (copy).c
myshell:
```

```
Open   file.out ~/Desktop/report   Save
1     PID TTY          TIME CMD
2     962 tty2     00:02:08 Xorg
3    1041 tty2     00:00:00 gnome-session-b
4    6046 pts/1    00:00:00 myshell
5    6054 pts/1    00:00:00 ps
6 err.txt
7 file.in
8 file.out
9 myshell
10 myshell (another copy).c
11 myshell.c
12 myshell (copy).c
Plain Text   Tab Width: 8   Ln 1, Col 1   INS
```

myshell: myprog [args] < file.in > file.out (below)

```
ahmet@Ahmet: ~/Desktop/report
ahmet@Ahmet:~/Desktop/report$ gcc myshell.c -o myshell
ahmet@Ahmet:~/Desktop/report$ ./myshell
myshell: cat file.out
    PID TTY          TIME CMD
    962 tty2     00:02:08 Xorg
   1041 tty2     00:00:00 gnome-session-b
   6046 pts/1    00:00:00 myshell
   6054 pts/1    00:00:00 ps
err.txt
file.in
file.out
myshell
myshell (another copy).c
myshell.c
myshell (copy).c
myshell: sort < file.in >> file.out
myshell: cat file.out
    PID TTY          TIME CMD
    962 tty2     00:02:08 Xorg
   1041 tty2     00:00:00 gnome-session-b
   6046 pts/1    00:00:00 myshell
   6054 pts/1    00:00:00 ps
err.txt
file.in
file.out
myshell
myshell (another copy).c
myshell.c
myshell (copy).c
a
b
c
d
f
s
x
z
myshell:
```

```
Open    file.out ~/Desktop/report    Save
1     PID TTY          TIME CMD
2     962 tty2     00:02:08 Xorg
3    1041 tty2     00:00:00 gnome-session-b
4    6046 pts/1    00:00:00 myshell
5    6054 pts/1    00:00:00 ps
6 err.txt
7 file.in
8 file.out
9 myshell
10 myshell (another copy).c
11 myshell.c
12 myshell (copy).c
13 a
14 b
15 c
16 d
17 f
18 s
19 x
20 z
Plain Text   Tab Width: 8   Ln 1, Col 1   INS
```