Name: Salah Abdo

Student ID: 6179614

Course: Computer Science

Module: 207SE Operating Systems, Security and Networks

Submission data: 10th April

Portfolio 2

Salah Abdo
Student ID: 6179614

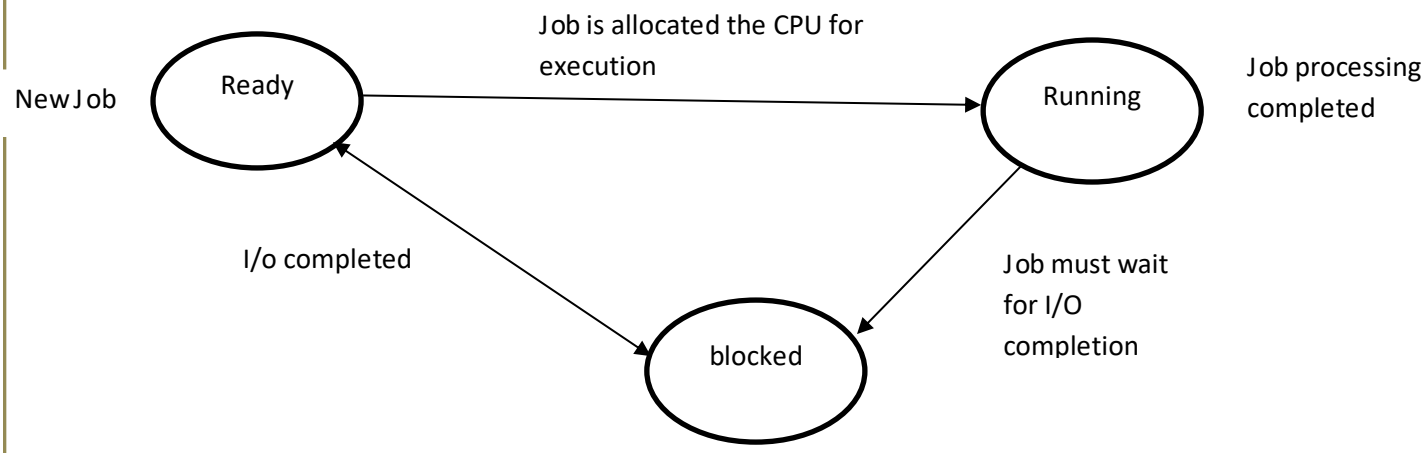
Contents

Salah Abdo
Student ID: 6179614

Lab Activity 12 – Jobs

a) Comparison between multiprogramming and multitasking

Multiprogramming is a way of scheduling process, this maximize the CPU usage by switching process that are waiting for input/output, this makes sure that the CPU is never idle. In multiprogramming, there are one or more programs which are loaded in the main memory which are ready to execute. But, only one program at a time can get to the CPU, which will execute the instructions while all the other programs are waiting. The idea of multiprogramming is to maximize the use of CPU time (gabrieletolomei.wordpress n.d). Suppose the currently running a process is performing an input output task then the operating system might interrupt that specific process and give the control control to one of the other in-main-memory programs that are ready to execute (gabrieletolomei.wordpress n.d). Multitasking is a reasonable extension of multiprogramming which involves quickly switching processed in the ready state to give the impression that they are running at the same time. The difference between multiprogramming and multitasking is that multitasking is more general sensed which means having multiple running at the same time. The term is used in modern OS when multiple tasks have similar processing resource (gabrieletolomei.wordpress n.d). At any time the CPU is executing a task only while other task are still waiting for their turn, this impression of parallelism is achieve when the CPU is given to another task (gabrieletolomei.wordpress n.d).



b) What is a scheduler

A scheduler is a software that allows an enterprise to schedule and track computer batch tasks (techopedia n.d). The way a scheduler works is by starting and haling jobs automatically, this is done by manipulating a prepared job control language algorithm. Schedulers that we see currently usually offer a GUI and a single point of control for all tasks in the distributed PC network (techopedia n.d).

c) References

techopedia (n.d) *scheduler.* [online] Available from <https://www.techopedia.com/definition/25078/scheduler> [09/04/2017]

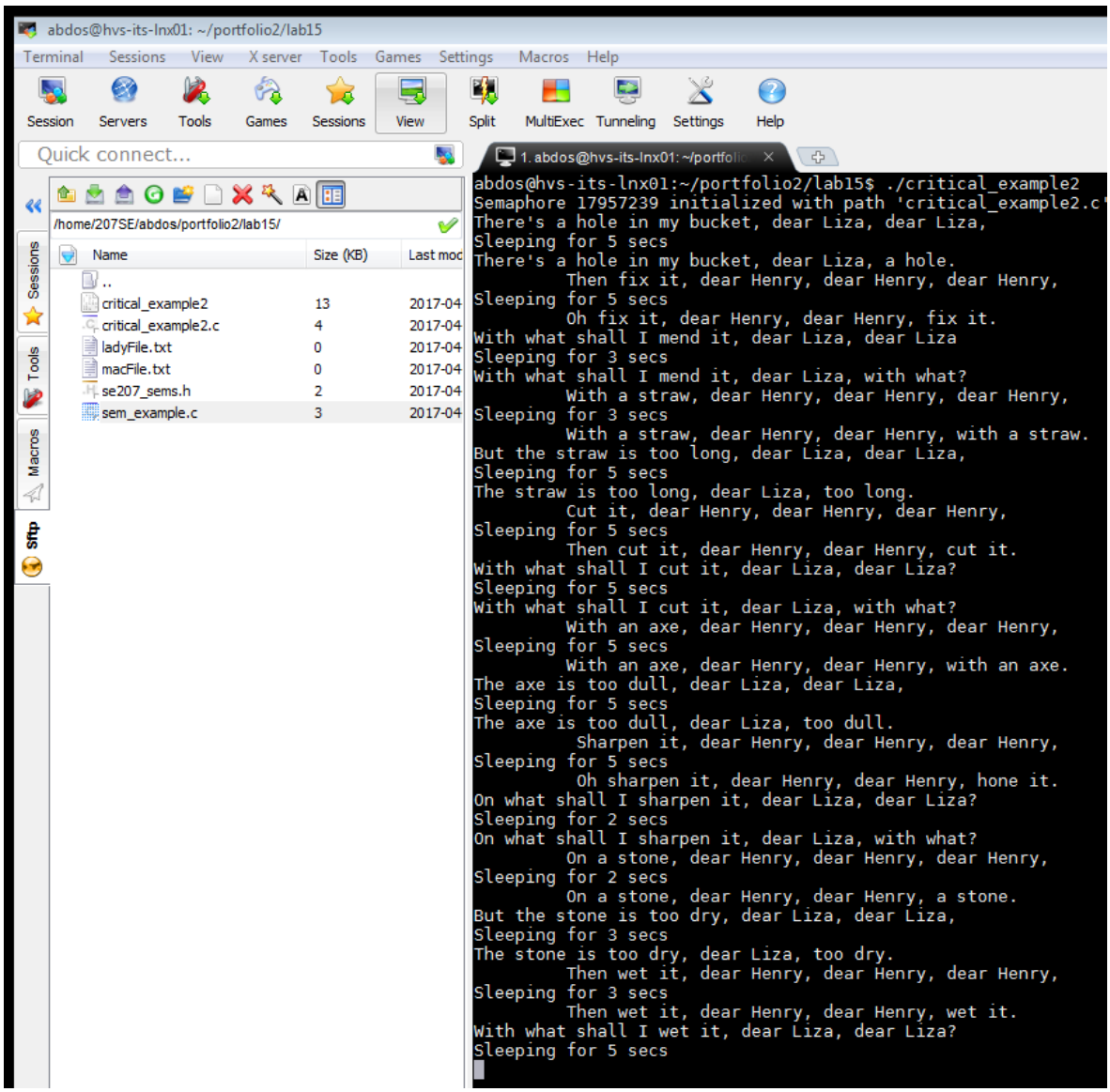
gabrieletolomei.wordpress (n.d) *multiprogramming,Multiprocessing, Multitasking and multithreading* [online] Available from <https://gabrieletolomei.wordpress.com/miscellanea/operating-systems/multiprogramming-multiprocessing-multitasking-multithreading/ > [09/04/2017]

Salah Abdo
Student ID: 6179614

Lab Activity 14 – Linux command-line manipulation of processes

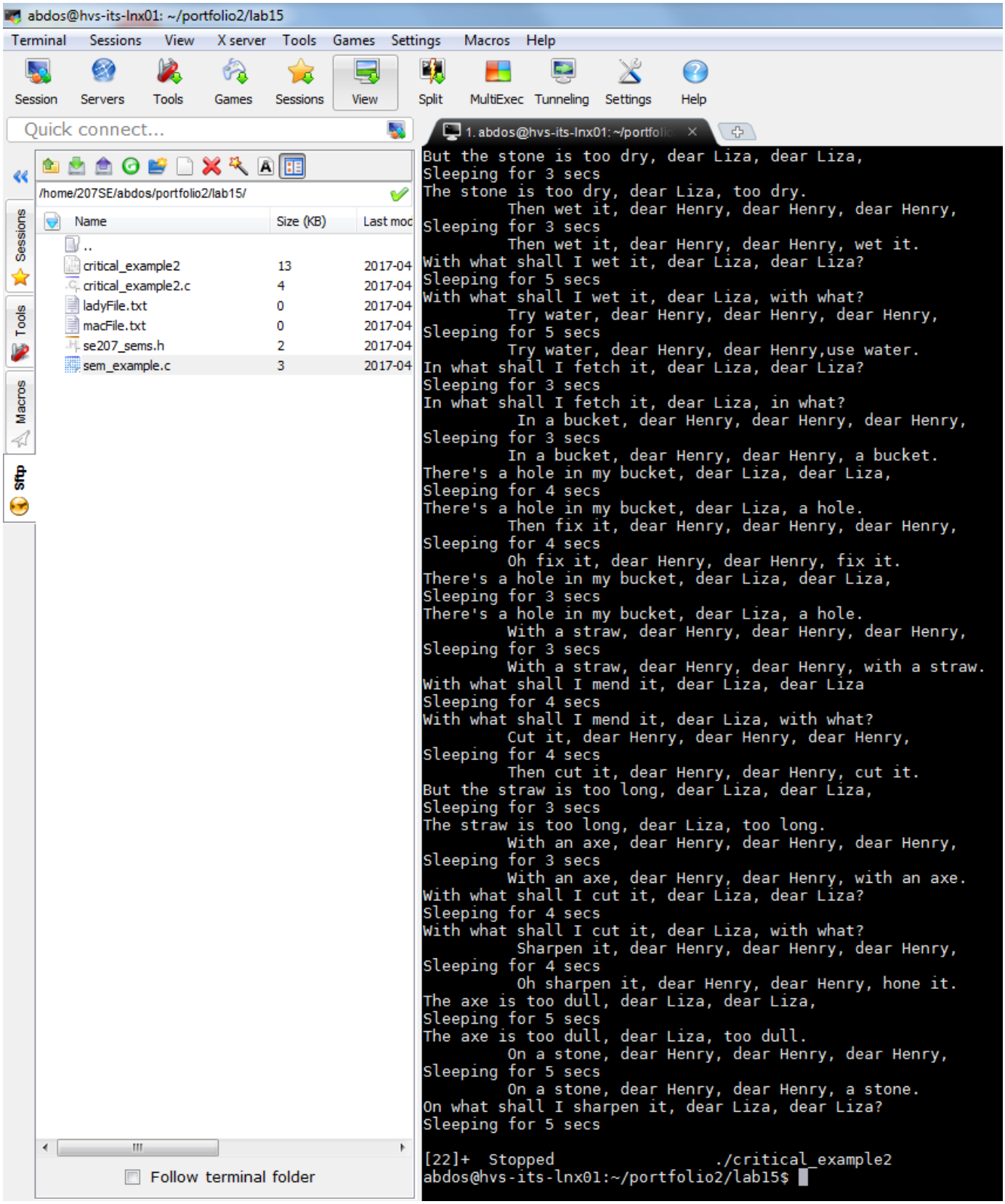
a) Process manipulation

- Example(s) of how to start process
  ./critical_example



Salah Abdo
Student ID: 6179614

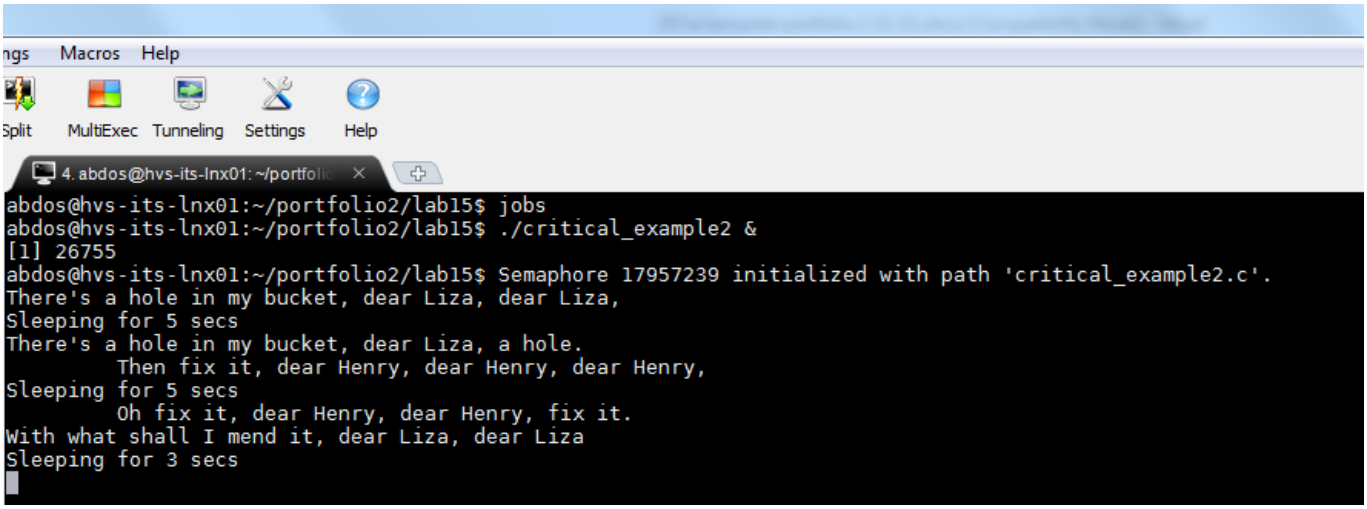
- Example(s) of how to suspend process
  Ctrl-Z



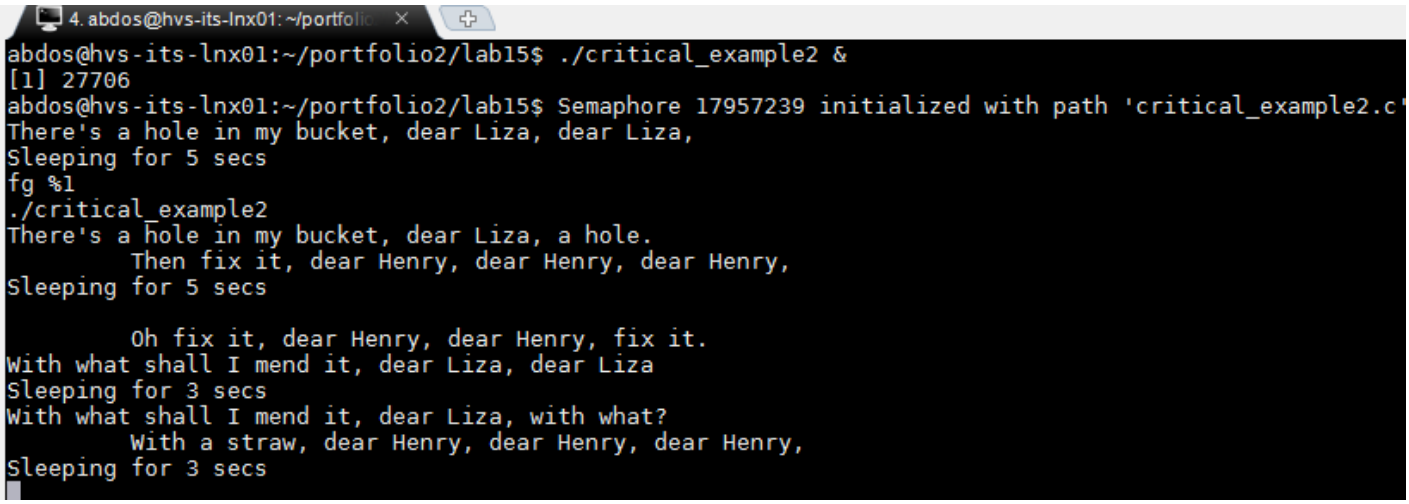
Salah Abdo
Student ID: 6179614

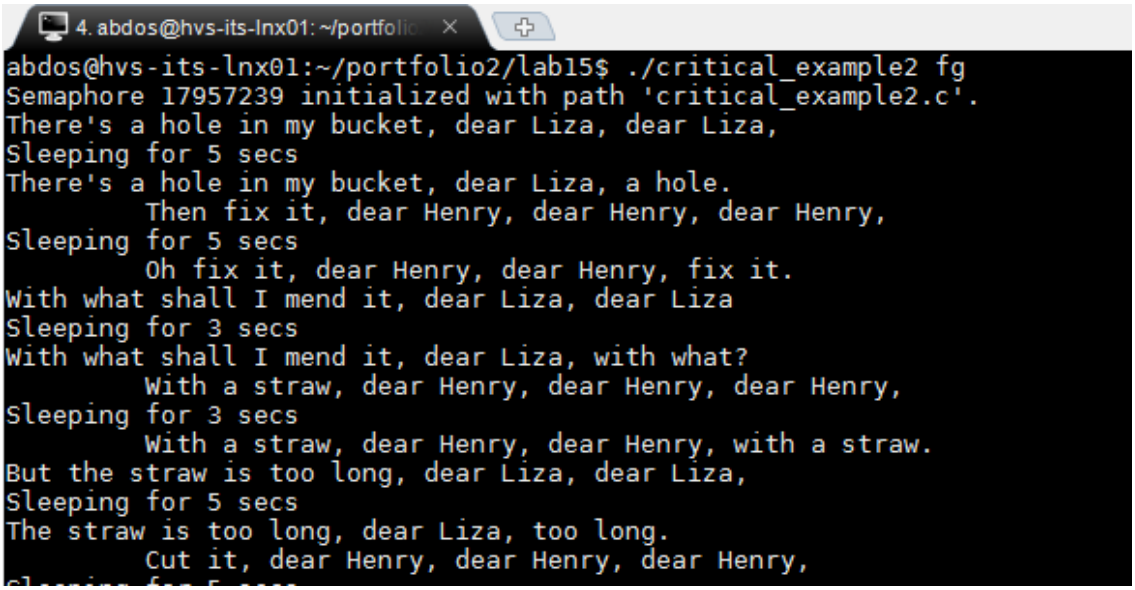
- Example(s) of how to run process in background

  ./critical_example2 & or ./critical_example2 bg



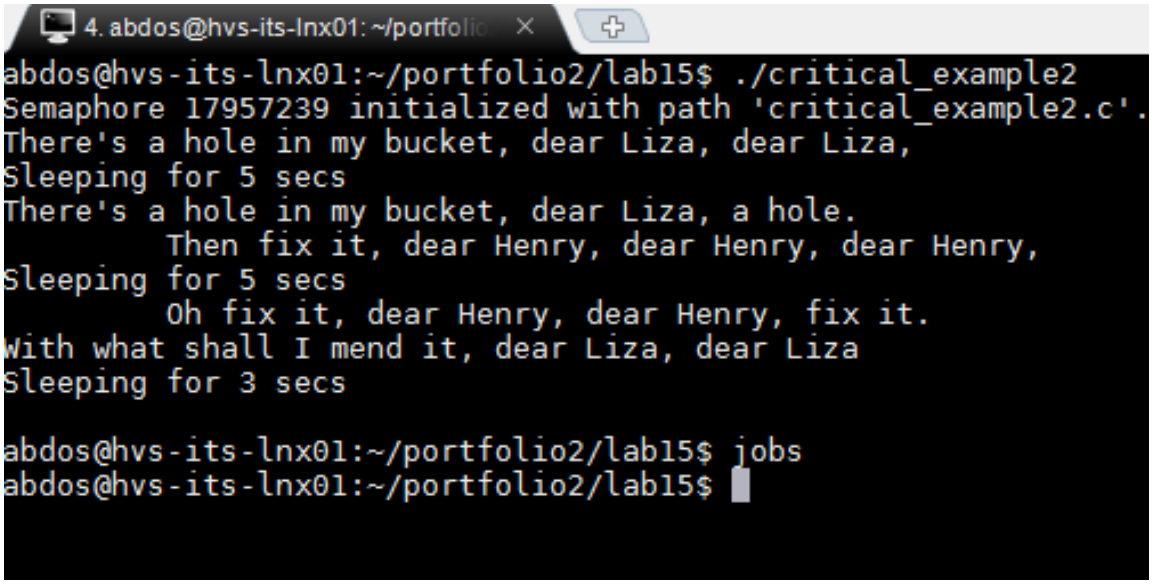
- Example(s) of how to run process in foreground and bring from background
  fg %1 to bring a background process to the foreground or ./critical_example2 fg

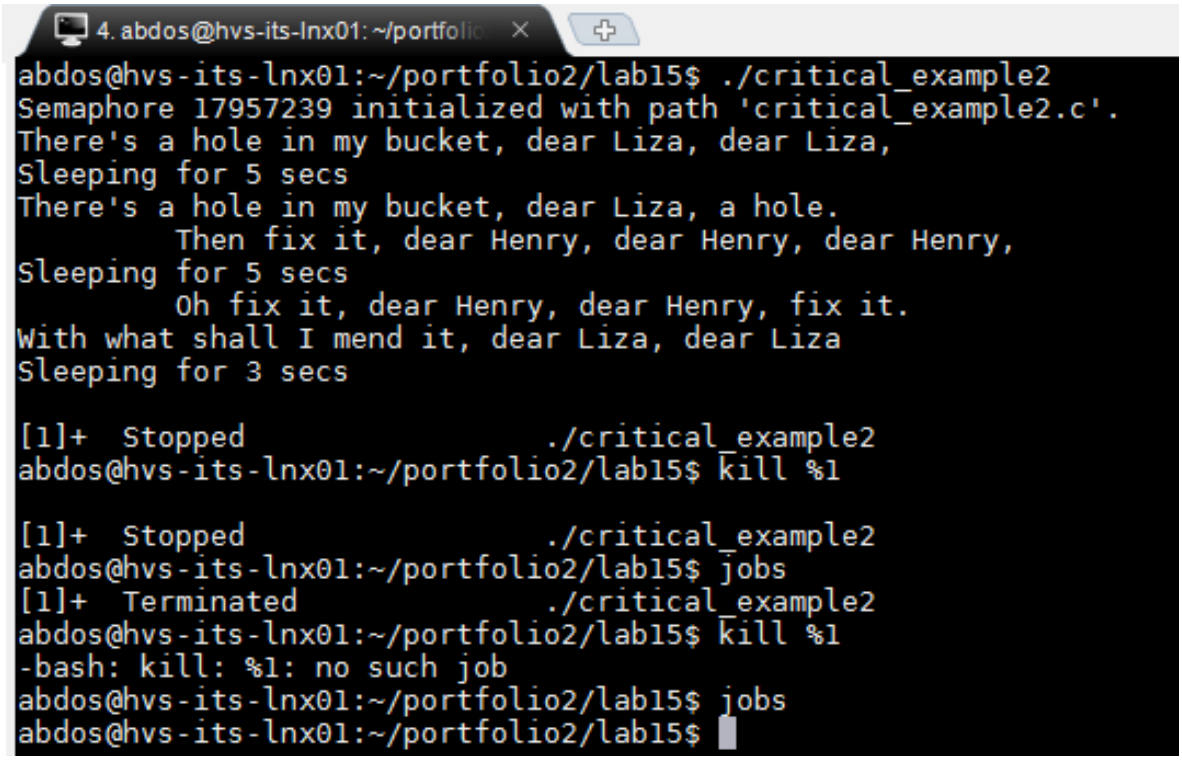




Salah Abdo
Student ID: 6179614

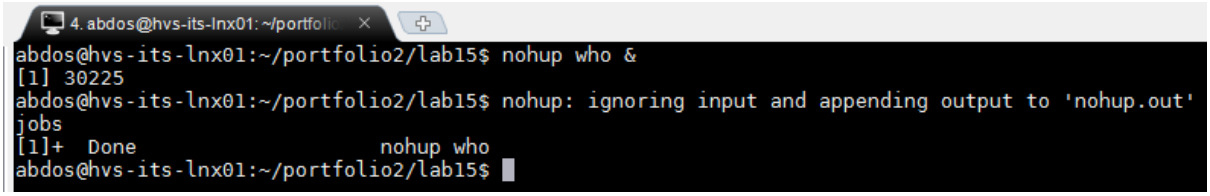
- Example(s) of how to kill a process
  Ctrl C can kill a process or you can do kill %1 (number would be the process you would want to kill)

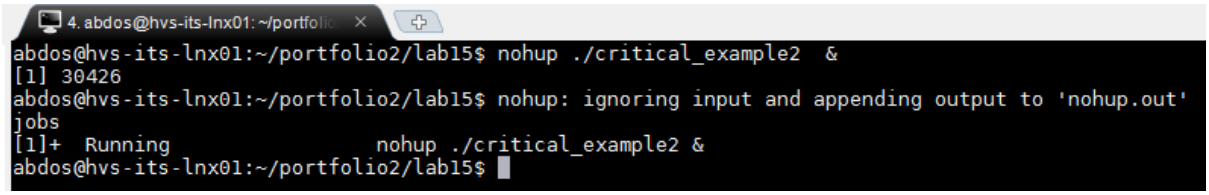




b) Paragraph on disown and nohup command

What disown does is that it removes jobs from the shell's job list. This is so that all the sub points above don't apply. But, it's still connected to the terminal, if anything where to happen to the terminal, the program will fail when it tries to read from the standard output (unix.stackexchange 2014). What nohup is a command that is used to run another command while suppressing of the hang up signal command, which keeps running after the user who executed the command has logged off (developerfeed 2010). If a process runs in the background, it's places into a list of background jobs that the shells managing. However, it's still connected to shell and if the shell closes so does the process as it gets terminated. What noHup does is that it effectively separates the command from the shell, allowing it to close and the process to continue.





c) Example of using watch command

The watch command runs command repeatedly displaying its output. By doing it allows the user to watch the program output change over time. By default, the program is run every 2 seconds: use -n or –interval to specify a different interval (tutorialpoint 2014).

To watch for mail
watch -n 60 from

To watch the contents of a directory change
watch -d ls -l

If you're only interested in files owned by user salah
watch -d 'ls -l | fgrep salah'

d) References for this activity

celtschk (2014) 'difference between nohup,disown and &'6/08/20014] Unix &Linux [online]. available from < http://unix.stackexchange.com/questions/3886/difference-between-nohup-disown-and> [09/04/2017]

developerfeed(2010) *what is Nohup and how is it used?*. [online] Available from <https://www.developerfeed.com/what-nohup-and-how-it-used/> [09/04/2017]

tutorialspoint (2014) *Watch-unix, linux command.* [online] Available from <http://www.tutorialspoint.com/unix_commands/watch.htm> [09/04/2017]

Salah Abdo
Student ID: 6179614

Lab Activity 15 IPC and Synchronisation

a) Brief description of activity

For this activity, I had to modify the semaphore provided with the activity. What I needed to modify was that the two processes had to output the scene from Macbeth by William Shakespeare, so that one process says Macbeth's lines and the other says Lady Macbeth's lines. The second part of this activity I had to write Lady Macbeth's part to stderr instead of stfout. As well as writing the lines to the screen I had to make sure that it redirected Macbeth's lines to one file and Lady Macbeth's lines to another.

Salah Abdo
Student ID: 6179614

b) Modified semaphore example code so that the two processes output the scene from MacBeth

```c
//critical_example2.c
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdio.h>
#include <stdlib.h>

#include "se207_sems.h"

int main(int argc, char argv[]){
  //Use our source file as the "key"
  int id=se207_semget("critical_example2.c",1);
 //int id1 =se207_semget("critcal_example2.c",1);


  int pid=fork();
  if(pid){
    //P1
    while(1){


      se207_wait(id);
      printf("There's a hole in my bucket, dear Liza, dear Liza,\n");
      rsleep();
      printf("There's a hole in my bucket, dear Liza, a hole.\n");
    se207_signal(id);

   se207_wait(id);
    printf("With what shall I mend it, dear Liza, dear Liza\n");
    rsleep();
    printf("With what shall I mend it, dear Liza, with what?\n");

    se207_signal(id);

    se207_wait(id);
    printf("But the straw is too long, dear Liza, dear Liza,\n");
    rsleep();
    printf("The straw is too long, dear Liza, too long.\n");

    se207_signal(id);

    se207_wait(id);
    printf("With what shall I cut it, dear Liza, dear Liza?\n");
    rsleep();
    printf("With what shall I cut it, dear Liza, with what?\n");

    se207_signal(id);

    se207_wait(id);
    printf("The axe is too dull, dear Liza, dear Liza,\n");
    rsleep();
    printf("The axe is too dull, dear Liza, too dull.\n");

    se207_signal(id);

    se207_wait(id);
    printf("On what shall I sharpen it, dear Liza, dear Liza?\n");
    rsleep();
    printf("On what shall I sharpen it, dear Liza, with what?\n");

    se207_signal(id);

    se207_wait(id);
```

```
63.     printf("But the stone is too dry, dear Liza, dear Liza,\n");
64.     rsleep();
65.     printf("The stone is too dry, dear Liza, too dry.\n");
66.
67.     se207_signal(id);
68.
69.     se207_wait(id);
70.     printf("With what shall I wet it, dear Liza, dear Liza?\n");
71.     rsleep();
72.     printf("With what shall I wet it, dear Liza, with what?\n");
73.
74.     se207_signal(id);
75.
76.     se207_wait(id);
77.     printf("In what shall I fetch it, dear Liza, dear Liza?\n");
78.     rsleep();
79.     printf("In what shall I fetch it, dear Liza, in what?\n");
80.
81.     se207_signal(id);
82.
83.     se207_wait(id);
84.     printf("There's a hole in my bucket, dear Liza, dear Liza,\n");
85.     rsleep();
86.     printf("There's a hole in my bucket, dear Liza, a hole.\n");
87.
88.     se207_signal(id);
89.
90.
91.     }
92.   }else{
93.     //P2
94.     while(1){
95.
96.     dup2(1,2);
97.
98.       se207_wait(id);
99.       printf("\t Then fix it, dear Henry, dear Henry, dear Henry,\n");
100.             rsleep();
101.             printf(" \t Oh fix it, dear Henry, dear Henry, fix it. \n");
102.             se207_signal(id);
103.
104.          se207_wait(id);
105.          printf("\t With a straw, dear Henry, dear Henry, dear Henry,\n");
106.          rsleep();
107.          printf("\t With a straw, dear Henry, dear Henry, with a straw.\n");
108.
109.          se207_signal(id);
110.
111.          se207_wait(id);
112.          printf("\t Cut it, dear Henry, dear Henry, dear Henry,\n");
113.          rsleep();
114.          printf("\t Then cut it, dear Henry, dear Henry, cut it.\n");
115.
116.          se207_signal(id);
117.
118.          se207_wait(id);
119.          printf("\t With an axe, dear Henry, dear Henry, dear Henry,\n");
120.          rsleep();
121.          printf("\t With an axe, dear Henry, dear Henry, with an axe.\n");
122.
123.          se207_signal(id);
124.
125.          se207_wait(id);
126.          printf("\t  Sharpen it, dear Henry, dear Henry, dear Henry,\n");
127.          rsleep();
```

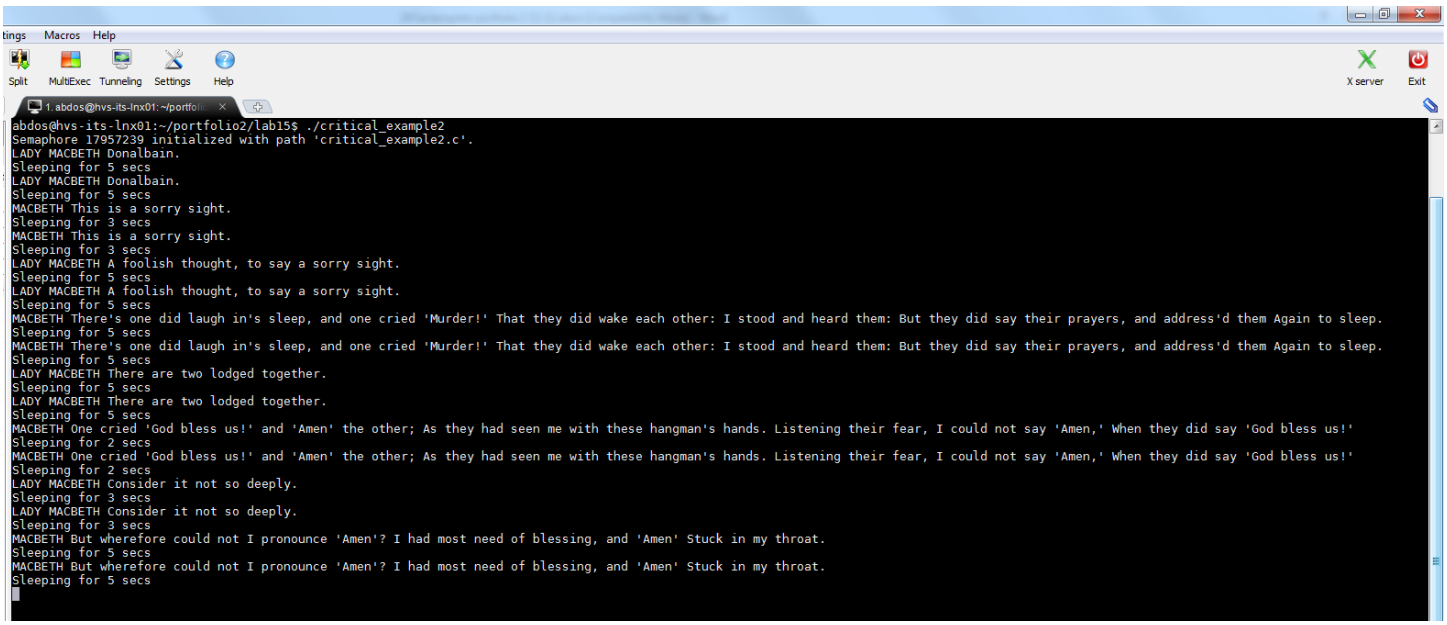
```
128.            printf("\t  Oh sharpen it, dear Henry, dear Henry, hone it.\n");
129.
130.            se207_signal(id);
131.
132.            se207_wait(id);
133.            printf("\t On a stone, dear Henry, dear Henry, dear Henry,\n");
134.            rsleep();
135.            printf("\t On a stone, dear Henry, dear Henry, a stone.\n");
136.
137.            se207_signal(id);
138.
139.            se207_wait(id);
140.            printf("\t Then wet it, dear Henry, dear Henry, dear Henry,\n");
141.            rsleep();
142.            printf("\t Then wet it, dear Henry, dear Henry, wet it.\n");
143.
144.            se207_signal(id);
145.
146.            se207_wait(id);
147.            printf("\t Try water, dear Henry, dear Henry, dear Henry,\n");
148.            rsleep();
149.            printf("\t Try water, dear Henry, dear Henry,use water.\n");
150.
151.            se207_signal(id);
152.
153.            se207_wait(id);
154.            printf("\t  In a bucket, dear Henry, dear Henry, dear Henry,\n");
155.            rsleep();
156.            printf("\t In a bucket, dear Henry, dear Henry, a bucket.\n");
157.
158.            se207_signal(id);
159.            }
160.
161.
162.
163.          }
164.
165.
166.
167.        }
```

Salah Abdo
Student ID: 6179614

c) Modified code to write Lady MacBeth's part to stderr and redirect the two parts to different files.

```
1.  //critical_example2.c
2.  #include <sys/ipc.h>
3.  #include <sys/sem.h>
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.
7.  #include "se207_sems.h"
8.
9.  int main(int argc, char argv[]){
10.   //Use our source file as the "key"
11.   int count = 0;
12.   //the line below is the array that stores the convertation between Mavbeth and La
    dy Macbeth
13.   const char *lines[] = {"LADY MACBETH Donalbain.","MACBETH This is a sorry sight."
    ,"LADY MACBETH A foolish thought, to say a sorry sight.","MACBETH There's one did l
    augh in's sleep, and one cried 'Murder!' That they did wake each other: I stood and
     heard them: But they did say their prayers, and address'd them Again to sleep.","L
    ADY MACBETH There are two lodged together.","MACBETH One cried 'God bless us!' and
    'Amen' the other; As they had seen me with these hangman's hands. Listening their f
    ear, I could not say 'Amen,' When they did say 'God bless us!'","LADY MACBETH Consi
    der it not so deeply.","MACBETH But wherefore could not I pronounce 'Amen'? I had m
    ost need of blessing, and 'Amen' Stuck in my throat.","LADY MACBETH These deeds mus
    t not be thought After these ways; so, it will ake us mad.","MACBETH Methought I he
    ard a voice cry 'Sleep no more! Macbeth does murder sleep', the innocent sleep, Sle
    ep that knits up the ravell'd sleeve of care, The death of each day's life, sore la
    bour's bath, Balm of hurt minds, great nature's second course, Chief nourisher in l
    ife's feast,--", "LADY MACBETH What do you mean?"};
14.
15.   int id=se207_semget("critical_example2.c",1);
16.
17.   FILE * ladyFile; //makes a file pointer for the ladyFile
18.   FILE * macFile; //makes a file pointer for the macFile
19.
20.   ladyFile = fopen ("ladyFile.txt","w"); //opens the ladyFile.txt file
21.   macFile = fopen ("macFile.txt","w"); //opens the macFile.txt file
22.
23.   int pid=fork();  //forks the process
24.   if(pid){//if it's process 1
25.     //P1
26.     while(1){//loops
27.       se207_wait(id);
28.       printf("%s\n",lines[count]);//prints the current item in the array
29.       fprintf(ladyFile, "%s\n",lines[count]);//prints to the lady macbeth file
30.       count++;//increases the array count
31.       rsleep();
32.       se207_signal(id);
33.     }
34.   }else{//Else it's process 2
35.     //P2
36.     while(1){//loops
37.       se207_wait(id);
38.       printf("%s\n",lines[count]);//prints the current item in the array
39.       fprintf(macFile, "%s\n",lines[count]); //prints to the macbeth file
40.       count++;//increases the array count
41.       rsleep();
42.       se207_signal(id);
43.     }
44.
45.   }
46.   fclose(ladyFile); //closes, lady macbeth file
47.   fclose(macFile); //closes, macbeth file
48. }
```

Salah Abdo
Student ID: 6179614



Salah Abdo
Student ID: 6179614

Lab Activity 16: IPC and Synchronisation II

a) Brief description of what the producer/consumer problem is.

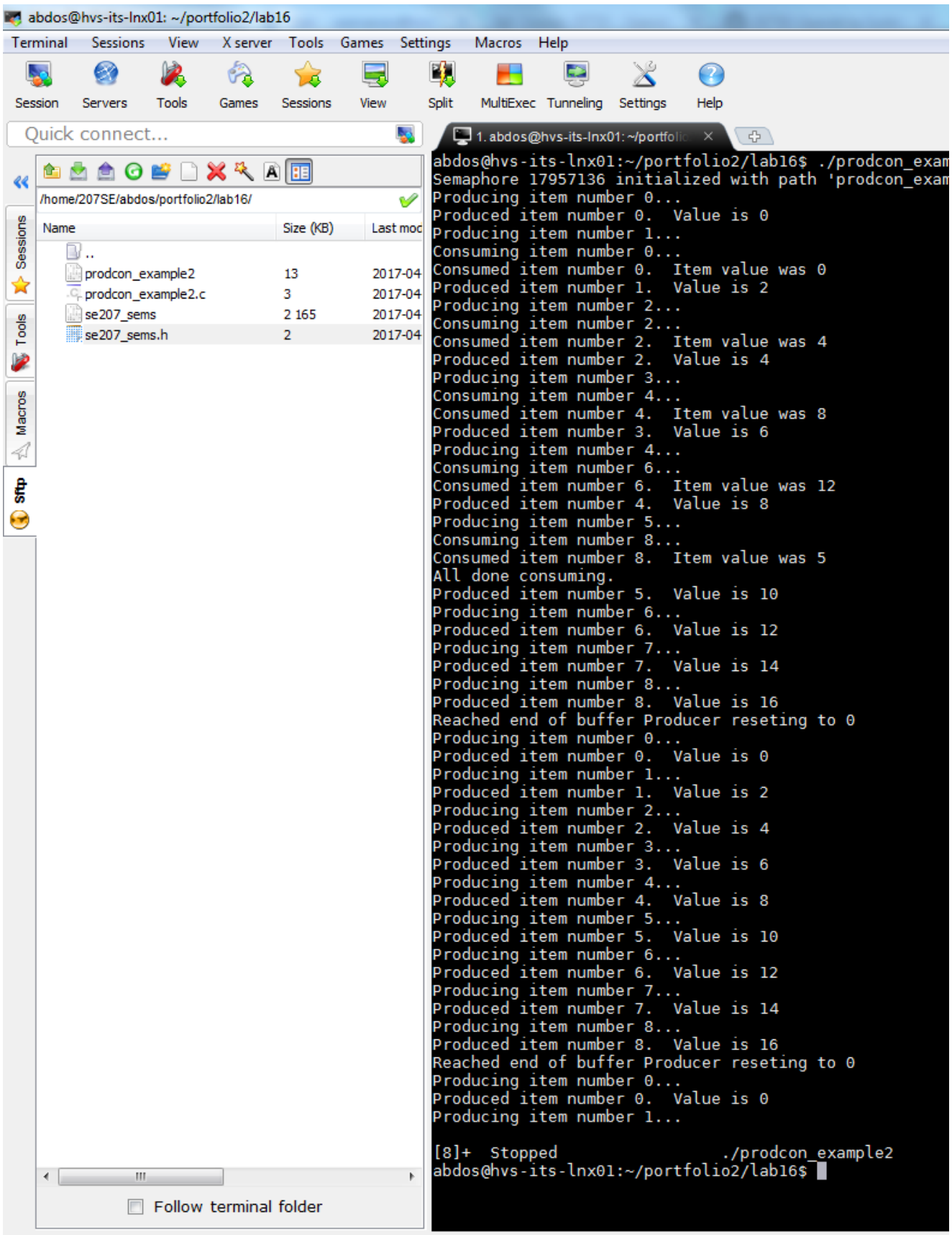
The producer-consumer problem is problem that describes two different process. It's when two process are trying to access the same resource at the exact same time meaning one will fail to do its process. This is because another process is using that resource. This is the produce and consumer problem.

b) Modified commented code that creates a ring/circular buffer, ensures data is not corrupted and works with different buffer sizes.

```
1.  include <sys/sem.h>
2.  #include <sys/shm.h>
3.  #include <stdio.h>
4.  #include <stdlib.h>
5.  #include <unistd.h>
6.  #include "se207_sems.h"
7.
8.  /* Remember to try reversing the timings...*/
9.
10. int bufferlength=10; //Limited buffer length
11. //what could we do about this?
12.
13. int main(int argc, char argv[]){
14.
15.   //Create shared memory segment
16.   int shm_id=shmget(ftok("prodcon_example2.c",2),bufferlength,
17.              0666|IPC_CREAT);
18.   //Use our source file as the "key"
19.   int id=se207_semget("prodcon_example2.c",0);
20.
21.   char* data; //For our pointer to shared memory...
22.   int pid=fork();
23.   if(pid){
24.     //P1 - CONSUMER
25.     shm_id=shmget(ftok("prodcon_example2.c",2),0,006);
26.
27.     //Attach the shared buffer
28.     data = shmat(shm_id, (void *)0, 0);
29.     int consumed=0;
30.     while(consumed<bufferlength){
31.       if (consumed>=bufferlength-
    1){//if the buffer = 2, the final positions in the buffer it reset the position to
    0 so it goes back to the begining of the buffer
32.
33.         printf("Reached end buffer Consumer reseting to 0\n");
34.         consumed=0;//resets the consumer
35.       }
36.       data[bufferlength-
    1]=consumed;//stores the current location of the consumer in the final position of
    the buffer
37.       if (data[bufferlength]-1 != data[bufferlength-
    2]){//  if the consumer is not overruning into the producer
38.         se207_wait(id);
39.         printf("Consuming item number %d...\n",consumed);
40.         sleep(1);
41.         char item=data[consumed];
42.
43.         printf("Consumed item number %d.  Item value was %d\n",
44.           consumed,item);
45.         consumed+=2;//consumer increment has been changed to 2
46.       }
47.       else{//if the consumer is going to over run the producer the consumer will do
     nothing untill the producer has moved
48.         printf("Stopping Consumer over running Producer\n");
49.       }
50.     }
51.
52. //Detatch
53.     shmdt(data);
54.     printf("All done consuming.\n");
```

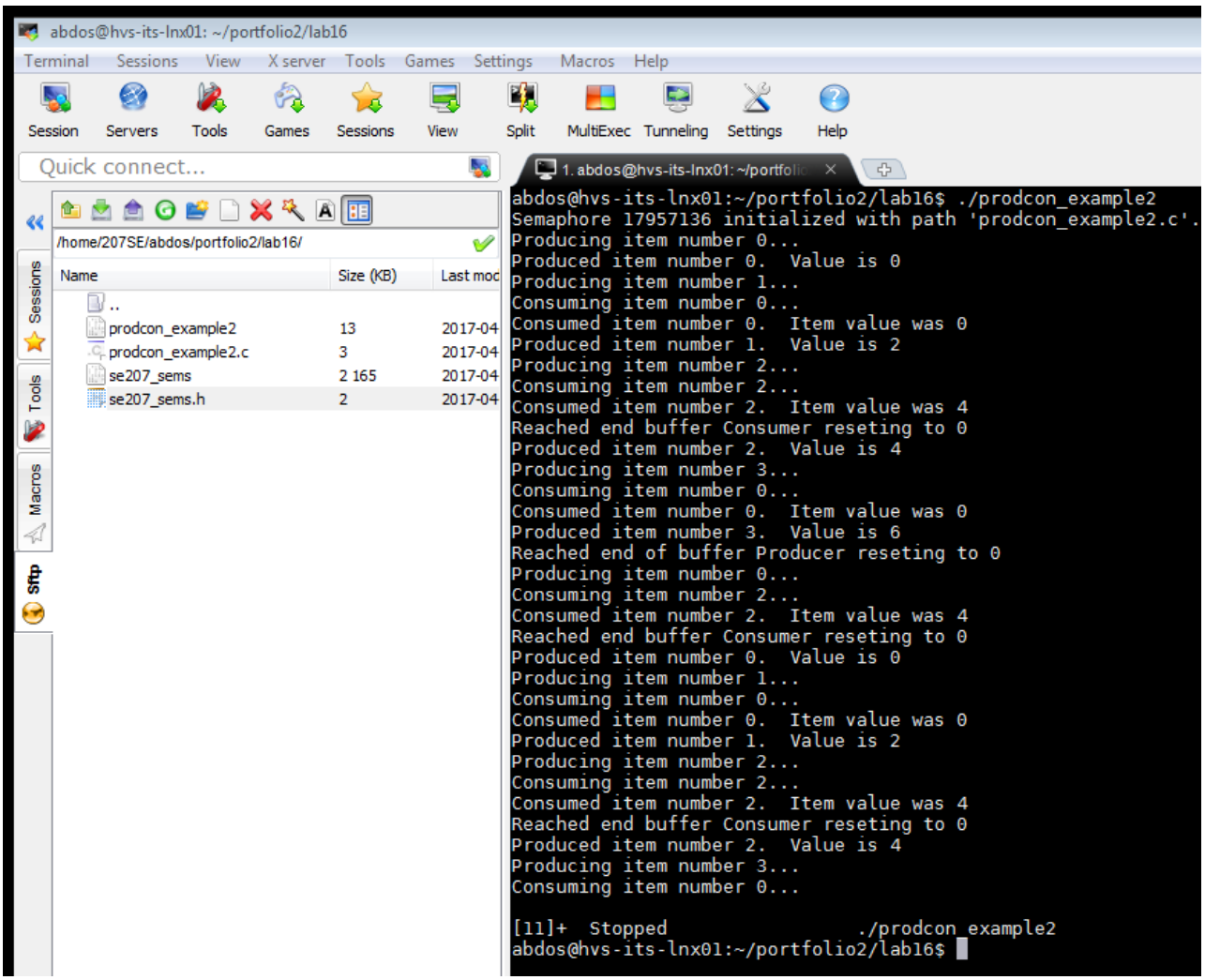
```
55.
56.     wait(); //For child process so that we can
57.
58.     //Delete the shared memory
59.     printf("Child ended, removing shm\n");
60.     shmctl(shm_id, IPC_RMID, NULL);
61.   }else{
62.     //P2
63.     shm_id=shmget(ftok("prodcon_example2.c",2),0,006);
64.     //Attach the shared buffer
65.     data = shmat(shm_id, (void *)0, 0);
66.
67.     int produced=0;
68.     while(produced<bufferlength){
69.       if (produced>=bufferlength-
    1){//if the buffer = 2, the final positions in the buffer it reset the position to
    0 so it goes back to the begining of the buffer
70.
71.         printf("Reached end of buffer Producer reseting to 0\n");
72.         produced=0;//resetes the producer
73.       }
74.       data[bufferlength-
    2]=produced;//sets the second from final postition in the buffer to the producer
75.       if (data[bufferlength]-1 != data[bufferlength-
    2]){//if the producer is not over runnning the consumer
76.         printf("Producing item number %d...\n",produced);
77.         sleep(2);
78.         data[produced]=produced*2; //Simple data, easy to check.
79.         printf("Produced item number %d.  Value is %d\n",
80.          produced,data[produced]);
81.         se207_signal(id);
82.         produced++;
83.       }
84.       else{//if the producer is over running the consumer it will do nothing until
    the consumer has moved
85.         printf("Stopping Producer over running Producer\n");
86.       }
87.     }
88.     //Detatch
89.     shmdt(data);
90.     printf("Producer finished.");
91.   }
92. }
```

Salah Abdo
Student ID: 6179614



Salah Abdo
Student ID: 6179614

- Ring/Circular buffer with different buffer sizes
- Producer/Consumer working at different speeds



Salah Abdo
Student ID: 6179614

Lab Activity 17 TCP Server

a) Brief description of the TCP Server Activity

For this activity, I had to make a simple reverse polish notation calculator server, which allowed addition, subtraction, multiplication and division. Also, I make sure that the program allows more general calculation by implementing a stack for values and operators that apply to the top pair.

Salah Abdo
Student ID: 6179614

b) Commented Code showing changing to tcp-server.cc tcp-client.cc

tcp-server.cc

```
1.  #include <arpa/inet.h>
2.
3.  #include <netdb.h>
4.  #include <netinet/in.h>
5.  #include <unistd.h>
6.  #include <iostream>
7.  #include <cstring>
8.  #include <stdlib.h>
9.
10.
11. #define MAX_MSG 100
12. #define LINE_ARRAY_SIZE (MAX_MSG+1)
13.
14. using namespace std;
15.
16. int main()
17. {
18.   int listenSocket, connectSocket, i;
19.   unsigned short int listenPort;
20.   socklen_t clientAddressLength;
21.   struct sockaddr_in clientAddress, serverAddress;
22.   char line[LINE_ARRAY_SIZE];
23.   int num1; //First Input
24.   int num2; //second Input
25.   char sign; //Operator selected
26.   char output; //output to be put in the line array so it can be sent to the client

27.   int temp; //temp that is used to store the value that is make by the sum of the u
    sers RPN before it is converted to a char(output) to be put in line
28.
29.   cout << "Enter port number to listen on (between 1500 and 65000): ";
30.   cin >> listenPort;
31.
32.   // Create socket for listening for client connection
33.   // requests.
34.   listenSocket = socket(AF_INET, SOCK_STREAM, 0);
35.   if (listenSocket < 0) {
36.     cerr << "cannot create listen socket";
37.     exit(1);
38.   }
39.
40.   // Bind listen socket to listen port. First set various
41.   // fields in the serverAddress structure, then call
42.   // bind().
43.
44.   // htonl() and htons() convert long integers and short
45.   // integers (respectively) from host byte order (on x86
46.   // this is Least Significant Byte first) to network byte
47.   // order (Most Significant Byte first).
48.
49.   serverAddress.sin_family = AF_INET;
50.   serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
51.   serverAddress.sin_port = htons(listenPort);
52. if (bind(listenSocket,
53.            (struct sockaddr *) &serverAddress,
54.            sizeof(serverAddress)) < 0) {
55.     cerr << "cannot bind socket";
56.     exit(1);
57.   }
58.
```

```
59.   // Wait for connections from clients. This is a
60.   // non-blocking call; i.e., it registers this program with
61.   // the system as expecting connections on this socket, and
62.   // then this thread of execution continues on.
63.   listen(listenSocket, 5);
64.
65.   while (1) {
66.     cout << "Waiting for TCP connection on port " << listenPort << " ...\n";
67.
68.     // Accept a connection with a client that is requesting
69.     // one. The accept() call is a blocking call; i.e., this
70.     // thread of execution stops until a connection comes
71.     // in. connectSocket is a new socket that the system
72.     // provides, separate from listenSocket. We *could*
73.     // accept more connections on listenSocket, before
74.     // connectSocket is closed, but this program doesn't do
75.     // that.
76.     clientAddressLength = sizeof(clientAddress);
77.     connectSocket = accept(listenSocket,
78.                            (struct sockaddr *) &clientAddress,
79.                            &clientAddressLength);
80.     if (connectSocket < 0) {
81.       cerr << "cannot accept connection ";
82.       exit(1);
83.     }
84.     // Show the IP address of the client.
85.     // inet_ntoa() converts an IP address from binary form to the
86.     // standard "numbers and dots" notation.
87.     cout << "  connected to " << inet_ntoa(clientAddress.sin_addr);
88.
89.     // Show the client's port number.
90.     // ntohs() converts a short int from network byte order (which is
91.     // Most Significant Byte first) to host byte order (which on x86,
92.     // for example, is Least Significant Byte first).
93.     cout << ":" << ntohs(clientAddress.sin_port) << "\n";
94.
95.     // Read lines from socket, using recv(), storing them in the line
96.     // array.  If no messages are currently available, recv() blocks
97.     // until one arrives.
98.     // First set line to all zeroes, so we'll know where the end of
99.     // the string is.
100.          memset(line, 0x0, LINE_ARRAY_SIZE);
101.          while (recv(connectSocket, line, MAX_MSG, 0) > 0) {
102.            cout << "  --  " << line << "\n";
103.
104.       num1 = line[0] -
    '0'; //converts the first value to an integer and stores it in num1
105.            num2 = line[1] -
    '0'; //converts the second value to an integer and stores ut in num2
106.            sign = (char)line[2]; //third value is the operator
107.            cout << num1 << endl; // this line and the next 2 output what has been
     put in the num1, num2 and sign to the server screen to make sure it is correct
108.            cout << num2 << endl;
109.            cout << sign << endl;
110.
111.            if (sign == '+') //if the user has entered "+" then it does the follow
    ing
112.            {
113.              temp = num1+num2; //adds the 2 numbers the user entered and stores i
    t in temp
114.            }
115.            else if(sign == '-') //if the user has entered "-
    " then it does the following
116.            {
```

```
117.                temp = num1-
    num2; //subtracts the 2 numbers the user entered and stores it in temp
118.              }
119.              else if(sign == '*') //if the user has entered "*" then it does the fo
    llowing
120.              {
121.                temp = num1*num2; //multiplies the 2 numbers the user entered and st
    ores it in temp
122.              }
123.              else if(sign == '/') //if the user has entered "/" then it does the fo
    llowing
124.              {
125.                temp = num1/num2; //divides the 2 numbers the user entered and store
    s it in temp
126.              }
127.              else //if the user hasn't used a valid operator
128.              {
129.                cout << "User entered invalid entry\n";
130.              }
131.              output = temp + '0';//temp is converted to the char output so it can b
    e stored in line
132.              line[0] = output;//sets the first value in line to the sum of the user
    s equation. The integer has been converted to a char so it can be sent to the clien
    t
133.              line[1] = '\0';//this and the next line empty the array
134.              line[2] = '\0';
135.              cout << "temp: " << temp << endl; //this and the next 2 lines output t
    he 3 variable temp. output and line to the server screen so you can see they have b
    een converted from int to char to char* correctly.
136.              cout << "output: " << output << endl;
137.              cout << "line: " << line << endl;
138.
139.              // Send converted line back to client.
140.              if (send(connectSocket, line, strlen(line) + 1, 0) < 0)
141.                cerr << "Error: cannot send modified data";
142.
143.              memset(line, 0x0, LINE_ARRAY_SIZE);  // set line to all zeroes
144.            }
145.          }
146.        }
```

Salah Abdo
Student ID: 6179614

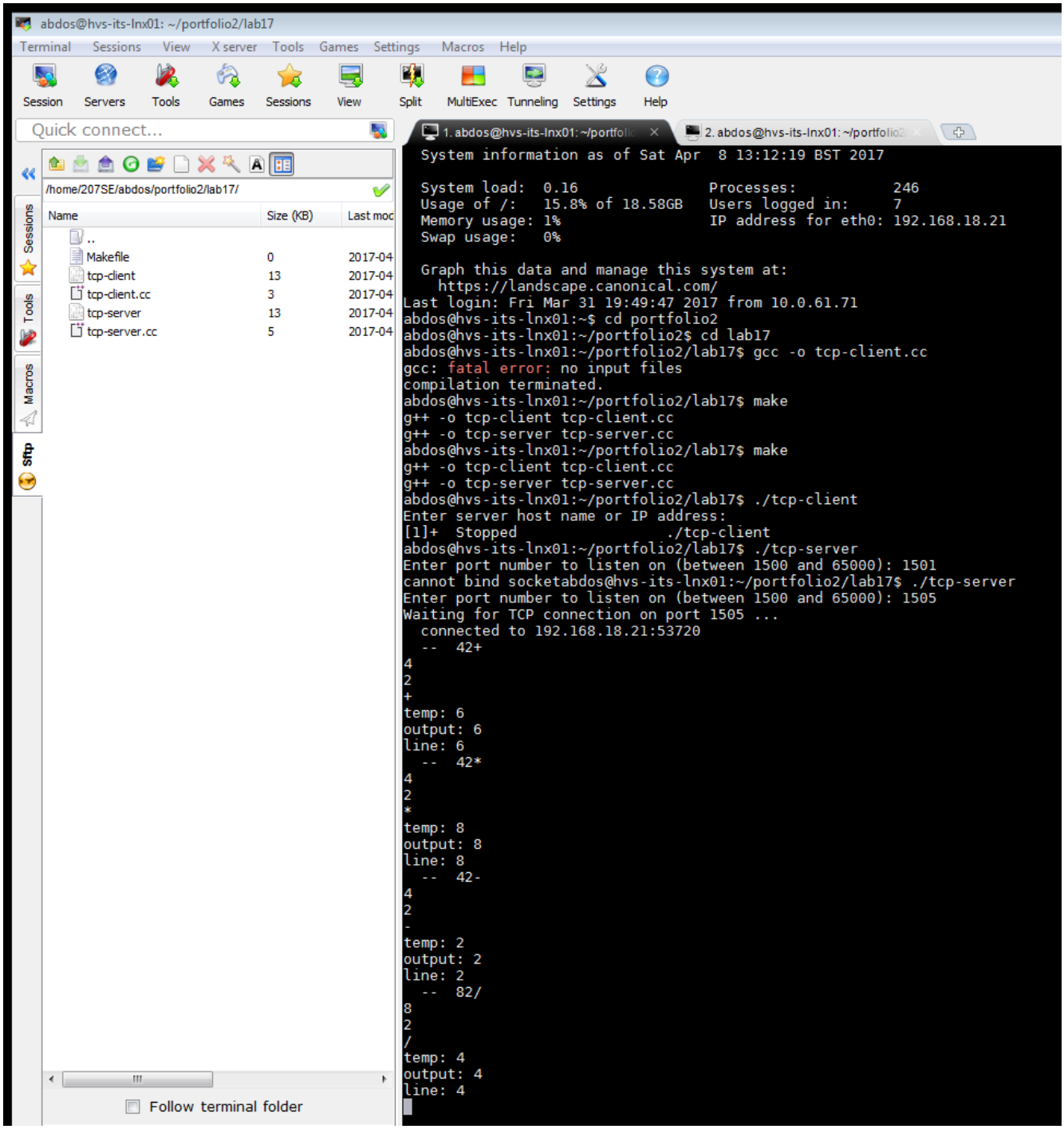
tcp-client.cc

```
1.  #include <netdb.h>
2.  #include <netinet/in.h>
3.  #include <unistd.h>
4.  #include <iostream>
5.  #include <cstring>
6.  #include <stdlib.h>
7.
8.  #define MAX_LINE 100
9.  #define LINE_ARRAY_SIZE (MAX_LINE+1)
10.
11. using namespace std;
12.
13. int main()
14. {
15.   int socketDescriptor;
16.   unsigned short int serverPort;
17.   struct sockaddr_in serverAddress;
18.   struct hostent *hostInfo;
19.   char buf[LINE_ARRAY_SIZE], c;
20.
21.   cout << "Enter server host name or IP address: ";
22.   cin.get(buf, MAX_LINE, '\n');
23.
24.   // gethostbyname() takes a host name or ip address in "numbers and
25.   // dots" notation, and returns a pointer to a hostent structure,
26.   // which we'll need later.  It's not important for us what this
27.   // structure is actually composed of.
28.   hostInfo = gethostbyname(buf);
29.   if (hostInfo == NULL) {
30.     cout << "problem interpreting host: " << buf << "\n";
31.     exit(1);
32.   }
33. cout << "Enter server port number: ";
34.   cin >> serverPort;
35.   cin.get(c); // dispose of the newline
36.
37.   // Create a socket.  "AF_INET" means it will use the IPv4 protocol.
38.   // "SOCK_STREAM" means it will be a reliable connection (i.e., TCP;
39.   // for UDP use SOCK_DGRAM), and I'm not sure what the 0 for the last
40.   // parameter means, but it seems to work.
41.   socketDescriptor = socket(AF_INET, SOCK_STREAM, 0);
42.   if (socketDescriptor < 0) {
43.     cerr << "cannot create socket\n";
44.     exit(1);
45.   }
46.
47.   // Connect to server.  First we have to set some fields in the
48.   // serverAddress structure.  The system will assign me an arbitrary
49.   // local port that is not in use.
50.   serverAddress.sin_family = hostInfo->h_addrtype;
51.   memcpy((char *) &serverAddress.sin_addr.s_addr,
52.          hostInfo->h_addr_list[0], hostInfo->h_length);
53.   serverAddress.sin_port = htons(serverPort);
54. if (connect(socketDescriptor,
55.                (struct sockaddr *) &serverAddress,
56.                sizeof(serverAddress)) < 0) {
57.     cerr << "cannot connect\n";
58.     exit(1);
59.   }
60.
61.   cout << "\n Please enter 3 characters, 2 number followed by 1 operator ---->(+,-
    ,*,/):\n";
62.
```

```
63.   //Changed the output that tells the user what to do to be more relivant to the ta
    sk
64.
65.   //cout << "\nEnter some lines, and the server will modify them and\n";
66.   //cout << "send them back.  When you are done, enter a line with\n";
67.   //cout << "just a dot, and nothing else.\n";
68.   //cout << "If a line is more than " << MAX_LINE << " characters, then\n";
69.   //cout << "only the first " << MAX_LINE << " characters will be used.\n\n";
70.
71.   // Prompt the user for input, then read in the input, up to MAX_LINE
72.   // charactars, and then dispose of the rest of the line, including
73.   // the newline character.
74.   cout << "Input: ";
75.   cin.get(buf, MAX_LINE, '\n');
76.   while (cin.get(c) && c != '\n')
77.     ; //Loop does nothing except consume the spare bytes
78.
79. // Stop when the user inputs a line with just a dot.
80.   while (strcmp(buf, ".")) { //strcmp returns 0 when the two strings
81.                   //are the same, so this continues when
82.                   //they are different
83.     // Send the line to the server.
84.     if (send(socketDescriptor, buf, strlen(buf) + 1, 0) < 0) {
85.       cerr << "cannot send data ";
86.       close(socketDescriptor); //Note this is just like using files...
87.       exit(1);
88.     }
89.
90.     // Zero out the buffer.
91.     memset(buf, 0x0, LINE_ARRAY_SIZE);
92.
93. // Read the modified line back from the server.
94.     if (recv(socketDescriptor, buf, MAX_LINE, 0) < 0) {
95.       cerr << "didn't get response from server?";
96.       close(socketDescriptor);
97.       exit(1);
98.     }
99.     //int out = (int)buf;
100.           cout << "Modified: " << buf << "\n";
101.
102.       // Prompt the user for input, then read in the input, up to MAX_LINE
103.           // charactars, and then dispose of the rest of the line, including
104.           // the newline character.  As above.
105.           cout << "Input: ";
106.           cin.get(buf, MAX_LINE, '\n');
107.           while (cin.get(c) && c != '\n')
108.             ; //Chomp chomp chomp
109.         }
110.
111.         close(socketDescriptor);
112.         return 0;
113.       }
```

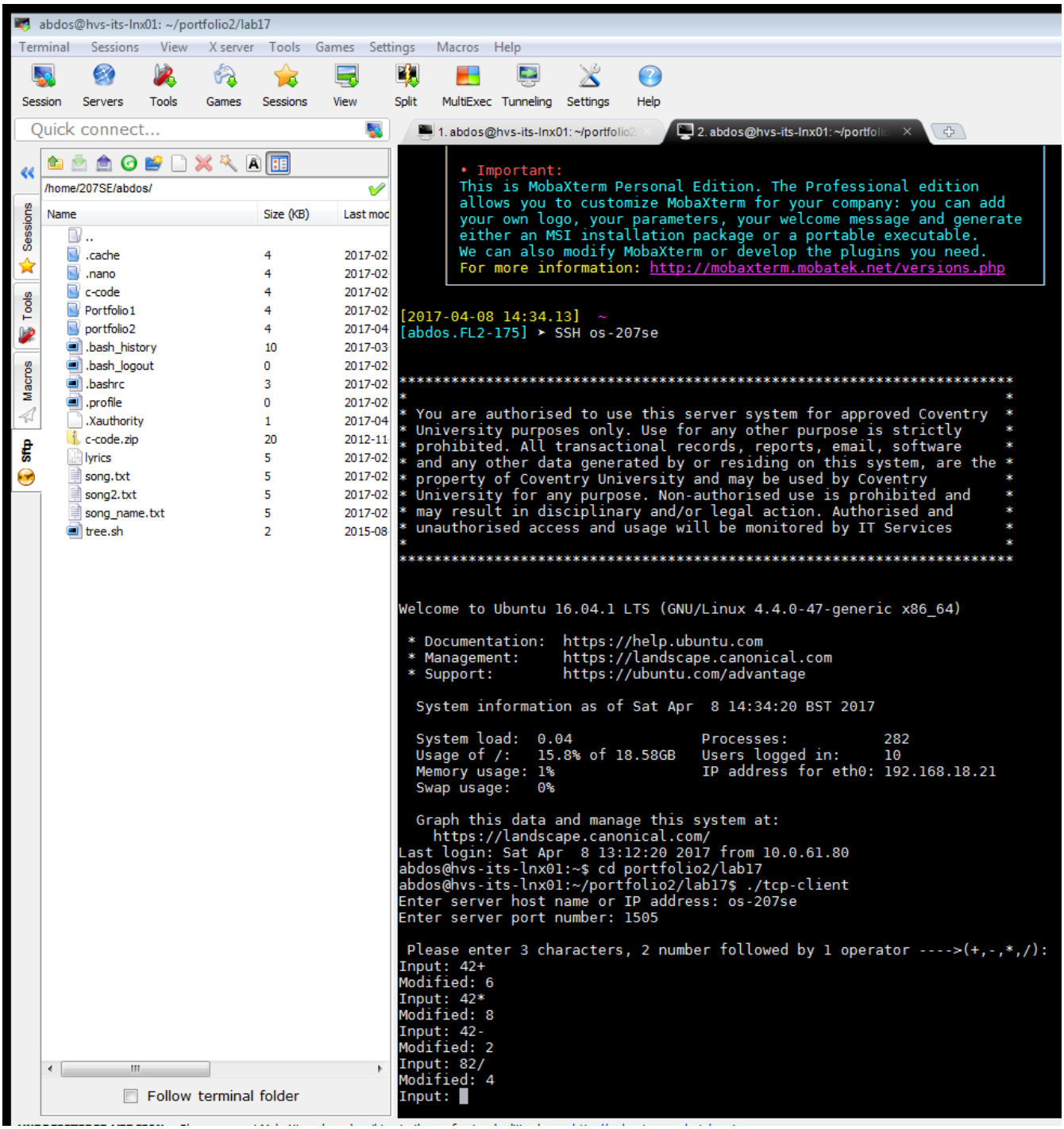
Salah Abdo
Student ID: 6179614

c) Examples of TCP server doing simple RPN calculations (2 numbers 1 arithmetic operation)



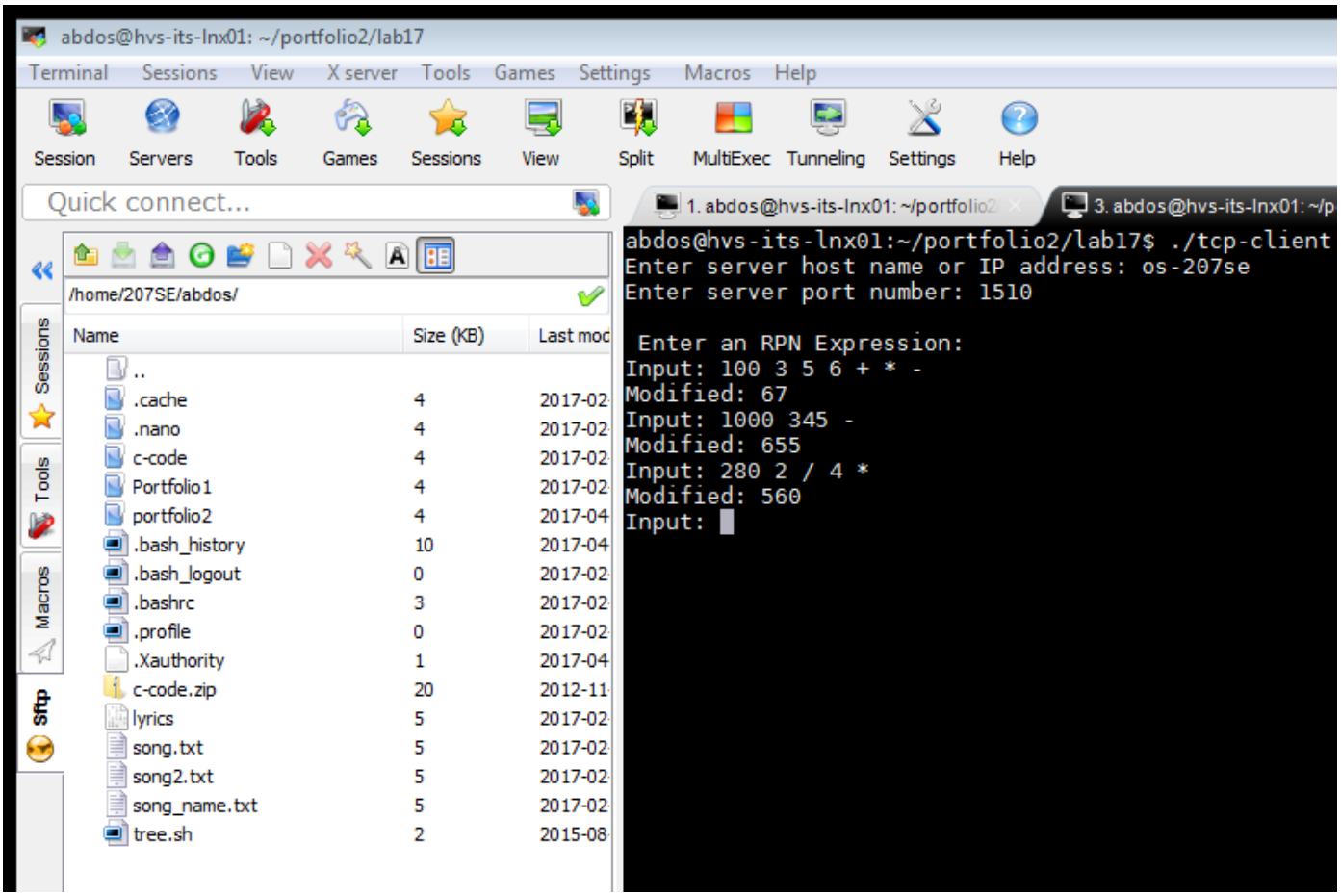
Salah Abdo
Student ID: 6179614

d) Examples of TCP server doing more complex RPN calculations with stack or stack-like structure.



Salah Abdo
Student ID: 6179614

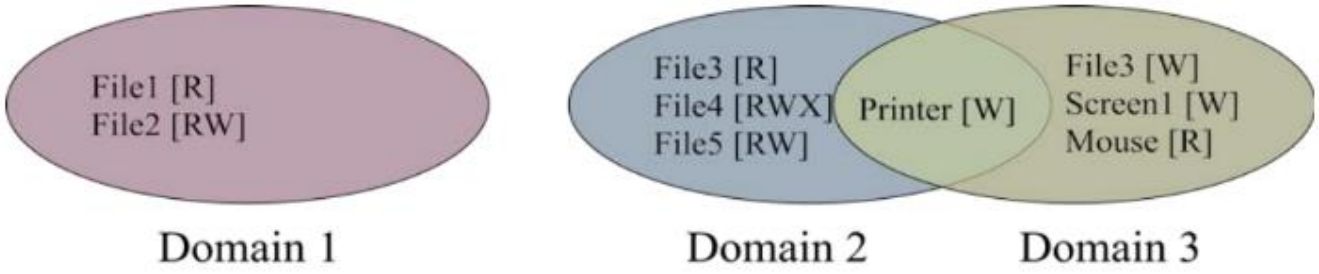
d) Examples of TCP server doing more complex RPN calculations with stack or stack-like structure.



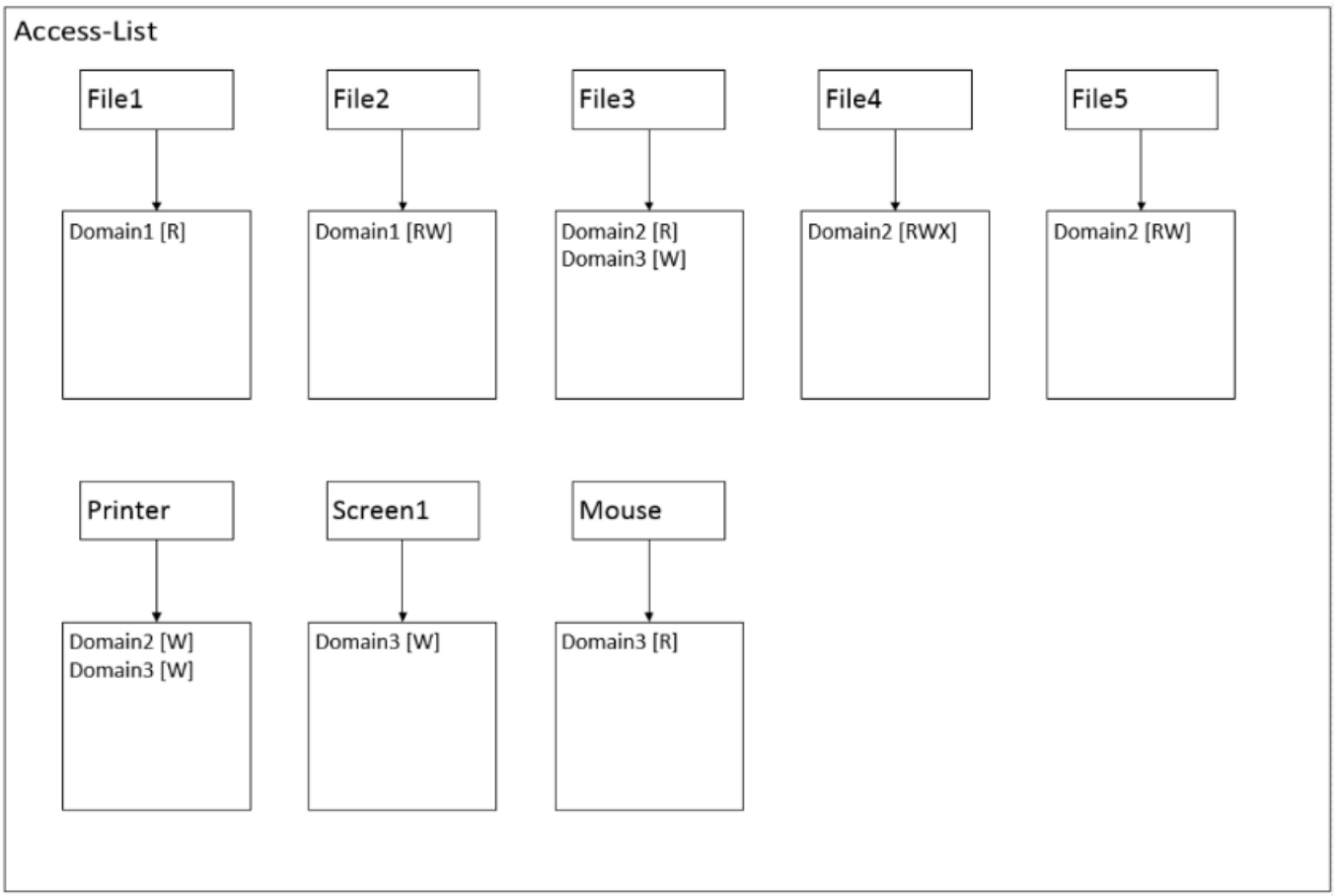
Salah Abdo
Student ID: 6179614

Lab Activity 19 Security

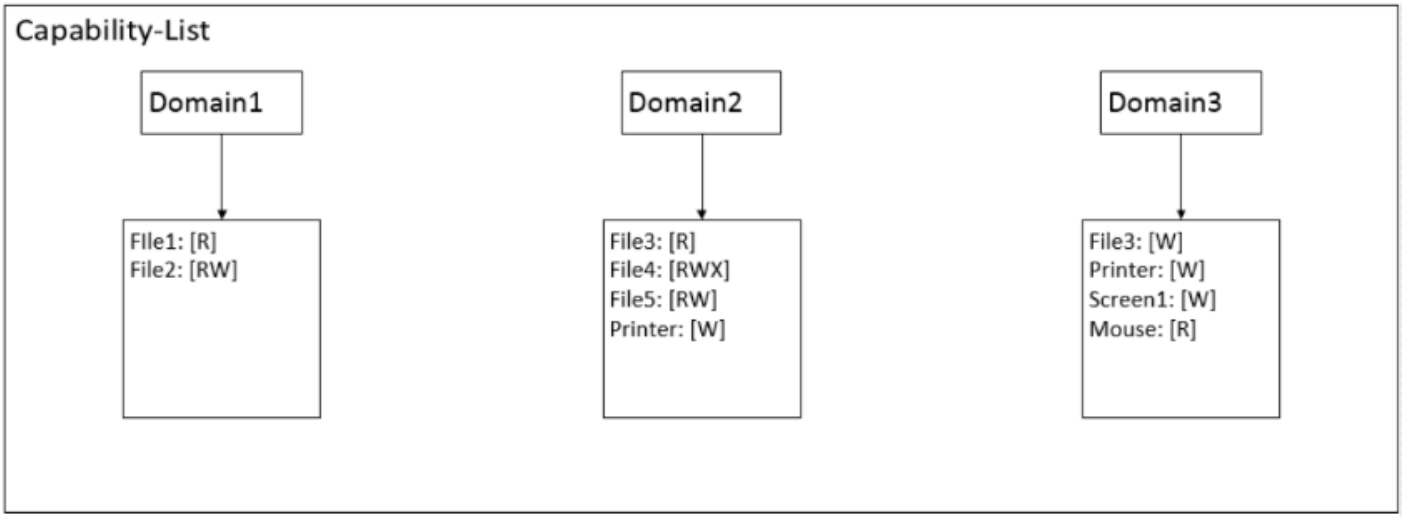
a) Create a protection domain matrix, access list and capability list for the diagram below



Protection domain matrix								
	File1	File2	File3	File4	File5	Printer	Screen1	mouse
D1	R	RW						
D2			R	RWX	RW	W		
D3			W			W	W	R



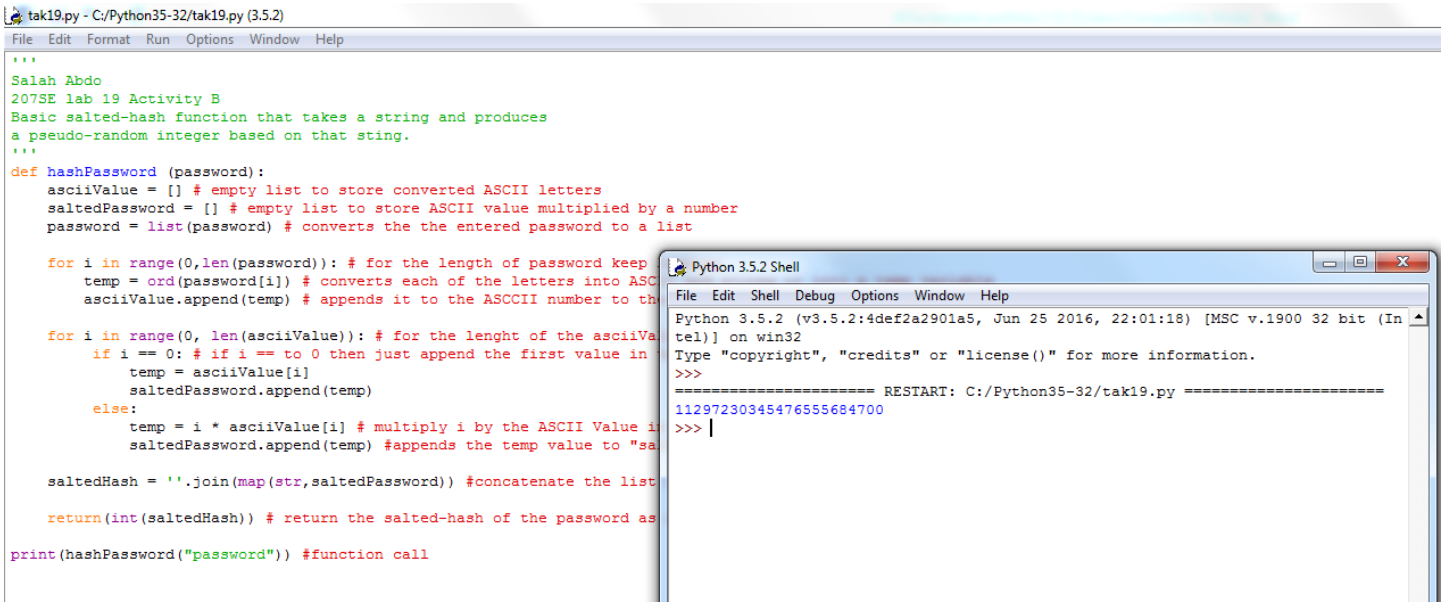
Salah Abdo
Student ID: 6179614



Salah Abdo
Student ID: 6179614

b) Commented Code showing hash function

```
'''''
Salah Abdo
207SE lab 19 Activity B
Basic salted-hash function that takes a string and produces
a pseudo-random integer based on that sting.
'''
def hashPassword (password):
    asciiValue = [] # empty list to store converted ASCII letters
    saltedPassword = [] # empty list to store ASCII value multiplied by a number
    password = list(password) # converts the the entered password to a list

    for i in range(0,len(password)): # for the length of password keep looping
        temp = ord(password[i]) # converts each of the letters into ASCII and store
s it into a temp variable
        asciiValue.append(temp) # appends it to the ASCCII number to the list "asci
iValue"

    for i in range(0, len(asciiValue)): # for the lenght of the asciiValue list kee
p looping
         if i == 0: # if i == to 0 then just append the first value in the list to
"saltedPassword"
             temp = asciiValue[i]
             saltedPassword.append(temp)
         else:
             temp = i * asciiValue[i] # multiply i by the ASCII Value in "asciiValu
e list and store it into a temp variable
             saltedPassword.append(temp) #appends the temp value to "saltedPassword
"

    saltedHash = ''.join(map(str,saltedPassword)) #concatenate the list of integers
 in "saltedPassword"

    return(int(saltedHash)) # return the salted-hash of the password as integer

print(hashPassword("password")) #function call
```



References

celtschk (2014) ‘difference between nohup,disown and &’6/08/20014] Unix &Linux [online]. available from < http://unix.stackexchange.com/questions/3886/difference-between-nohup-disown-and> [09/04/2017]

developerfeed(2010) *what is Nohup and how is it used?*. [online] Available from <https://www.developerfeed.com/what-nohup-and-how-it-used/> [09/04/2017]

tutorialspoint (2014) *Watch-unix, linux command.* [online] Available from <http://www.tutorialspoint.com/unix_commands/watch.htm> [09/04/2017]

techopedia (n.d) *scheduler.* [online] Available from <https://www.techopedia.com/definition/25078/scheduler> [09/04/2017]

gabrieletolomei.wordpress (n.d) *multiprogramming,Multiprocessing, Multitasking and multithreading* [online] Available from <https://gabrieletolomei.wordpress.com/miscellanea/operating-systems/multiprogramming-multiprocessing-multitasking-multithreading/ > [09/04/2017]