

Name: Asis Rai

Student id: 6528683

Course: Computer Science

Module: 207SE Operating Systems, Security and Networks

Submission data: 10th April

Portfolio 2

## Lab Activity 12 – Jobs

- a) Comparison between multiprogramming and multitasking

[Here you comparison]

### MULTIPROGRAMMING & MULTITASKING

There was a time when there was a restriction in technology advances, there were older system, the prices were expensive and the computers were slow and processes needed to use peripheral devices. This meant that the CPU was sitting idle of a long period of time. With time and technological advances, a solution was developed and it is called 'Batch Processing' which is a new term used for modern operating systems. Multi-programming is an old term because in modern operating systems the whole program is not loaded completely into the main memory.

In a multiprogramming operating system, there are one or more programs (processes) resident in computer's main memory ready to execute. Only one program at a time can be executed in the CPU, while the others wait their turn to use the CPU. The whole idea of having a multi-programmed system is to optimize system utilization, more specifically CPU time. For example, the program currently being executed gets interrupted by the operating system between tasks and transfer control to another program in line. Running program keeps executing until it voluntarily gives the CPU back or when it blocks for IO. The design goal is very clear as processes waiting for IO should not block other processes which wastes CPU time. The idea is to keep the CPU busy, if there are processes ready to execute. (shown in diagram1)

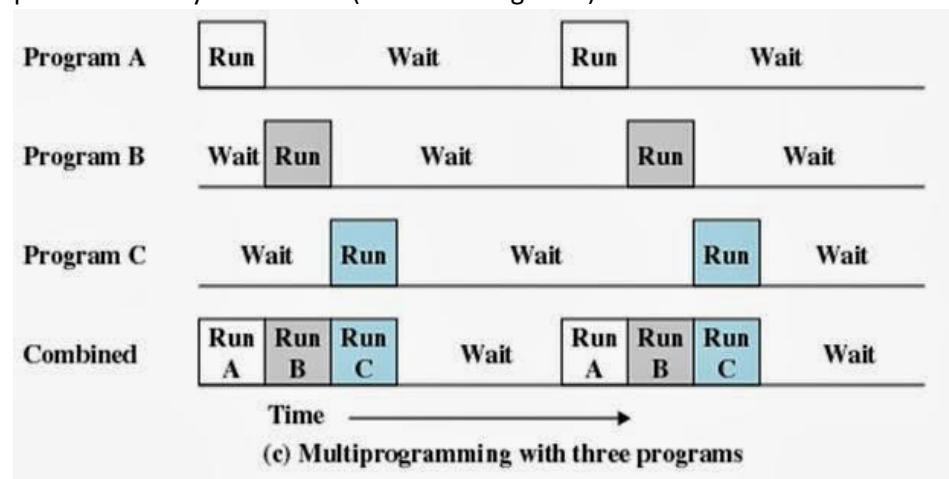


Diagram 1

Multitasking has the same meaning of multiprogramming but in a more general sense, as it refers to having multiple (programs, processes, tasks, threads) running at the same time. This term is used in modern operating systems when multiple tasks share a common processing resource (e.g., CPU and Memory). At any time, the CPU is executing one task only while other tasks waiting their turn. The illusion of parallelism is achieved when the CPU is reassigned to another task (i.e. process or thread context switching).

There are subtle differences between multitasking and multiprogramming. A task in a multitasking operating system is not a whole application program but it can also refer to a "thread of execution" when one process is divided into sub-tasks. Each smaller task does not hijack the CPU until it finishes like in the older multiprogramming but rather a fair share amount of the CPU time called quantum. Both multiprogramming and multitasking operating systems are (CPU) time sharing systems. However, while in multiprogramming (older OSs) one program keeps running until it blocks, in

multitasking (modern OSs) time sharing is best manifested because each running process takes only a fair quantum of the CPU time. (shown in diagram2)

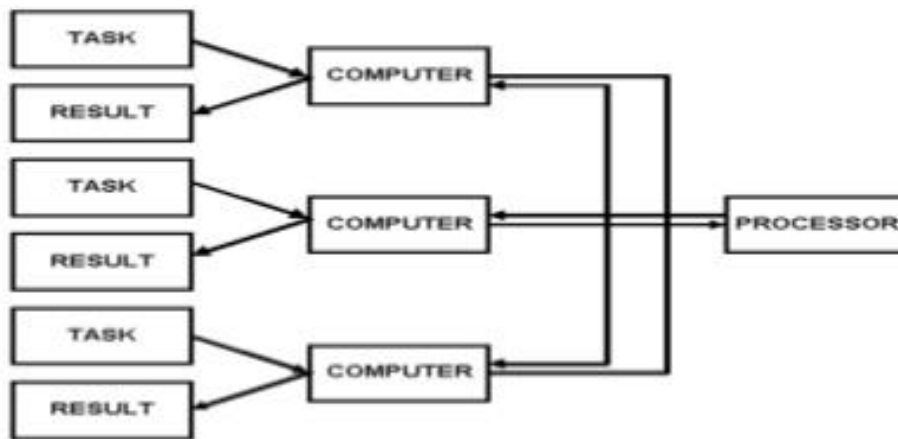


Diagram 2

In conclusion, Multiprogramming operating system allows multiple processes to reside in main memory where only one program is running. The running program keeps executing until it blocks for IO and the next program in line takes the turn for execution. The goal is to optimize CPU utilization by reducing CPU idle time. However, in multitasking time sharing is best manifested because each running process takes only a fair quantum of the CPU time.

b) What is a scheduler

[One paragraph on what is a scheduler]

Operating system scheduling is the process of controlling and prioritizing messages sent to a processor. An internal operating system program, called the scheduler, performs this task. The goal is maintaining a constant amount of work for the processor, eliminating highs and lows in the workload and making sure each process is completed within a reasonable time frame. Scheduling is typically broken down into three parts: long, mid, and short-term scheduling. Long-term scheduling revolves around admitting programs to the scheduling process. When a new program initiates, the long-term scheduler determines if there is enough space for the new entrant. If there isn't, then the scheduler delays the activation of the program until there is enough room. The midterm scheduler decides which processes have been idle and which are active. It leaves the active processes alone and writes idle ones to the hard drive. This frees up memory for other programs to come in through the long-term scheduler. When the mid- and long-term schedulers are combined, instead of delaying activation of a new process, the scheduler simply swaps it into storage. The short-term scheduler is the part that works directly with the processor. This portion activates processes, sets priorities and oversees the processor's workload. The short-term scheduler is constantly trying to anticipate computer needs to keep the processor running smoothly.

c) References

[Here your references for this report]

N.p., 2017. Web. 10 Apr. 2017.

"Batch Processing". *En.wikipedia.org*. N.p., 2017. Web. 10 Apr. 2017.

"Multiprogramming, Multiprocessing, Multitasking, And Multithreading". *Gabriele Tolomei*. N.p., 2017. Web. 10 Apr. 2017.

twitter., Follow. "Difference Between Multiprogramming, Multitasking, Multithreading And Multiprocessing". 8 BIT AVENUE. N.p., 2017. Web. 10 Apr. 2017.

"What Is Multiprogramming? - Definition From WhatIs.Com". *WhatIs.com*. N.p., 2017. Web. 10 Apr. 2017.

"What Is Operating System Scheduling? (With Pictures)". *wiseGEEK*. N.p., 2017. Web. 10 Apr. 2017.

## Lab Activity 14 – Linux command-line manipulation of processes

### a) Process manipulation

- Example(s) of how to start process

[How to started process]

The way to start a process on Linux shell is to command. I am going to start the 'sleep' process.

[Screenshot or section of Linux script showing process starting]

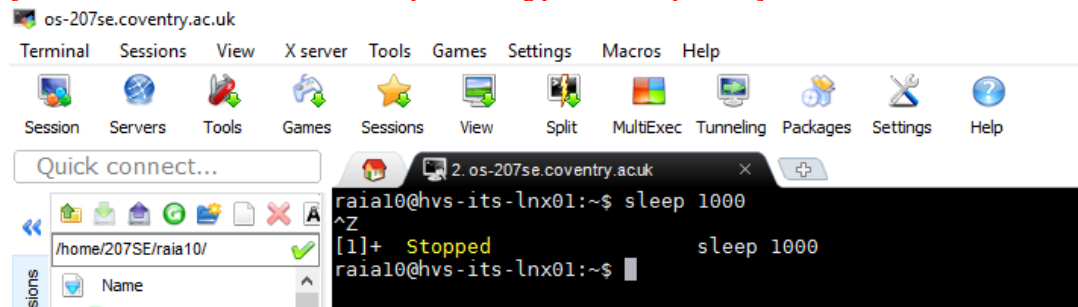


- Example(s) of how to suspend process

[How to suspended process]

To suspend a process : CTRL + Z

[Screenshot or section of Linux script showing process suspended]



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

[How to run process in background]

To run a process in background : Command &. For this task I am going to run the sleep process in background

[Screenshot or section of Linux script showing process running in background]



- Example(s) of how to run process in foreground and bring from background  
[\[How to run process in foreground and bring from background\]](#)  
 To bring a process in foreground from background : fg %1  
[\[Screenshot or section of Linux script showing process running in foreground or brought from background\]](#)

The screenshot shows a terminal window titled '3. os-207se.coventry.ac.uk'. The user 'raia10' runs the command 'sleep 1000 &' which returns '[1] 8029'. Then, they run 'jobs' which shows '[1]+ Running sleep 1000 &'. Finally, they run 'fg %1' which brings the process back to the foreground, showing 'sleep 1000'.

- Example(s) of how to kill a process  
[\[How to kill process\]](#)  
 There are different ways to kill a process: CTRL + C  
[\[Screenshot or section of Linux script showing process killed\]](#)

The screenshot shows a terminal window titled '3. os-207se.coventry.ac.uk'. The user 'raia10' runs 'sleep 1000' and presses Ctrl+C, which results in '[1]+ Stopped sleep 1000'. They then run 'jobs' which shows '[1]+ Stopped sleep 1000'. Next, they run 'fg %1' which brings the process back to the foreground, showing 'sleep 1000'. Finally, they run 'ps au |grep raia10' which shows the process details in a table:

USER	PID	%CPU	%MEM	VSZ	SSZ	PTS	TTY	TIME	COMMAND
raia10	7961	0.0	0.0	33888	5524	pts/5	Ss	16:39	0:00 -bash
raia10	8577	0.0	0.0	46384	3708	pts/5	R+	17:01	0:00 ps au
raia10	8578	0.0	0.0	33888	2032	pts/5	R+	17:01	0:00 -bash

## Demonstrating more ways

```

os-207se.coventry.ac.uk
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

Quick connect...
/home/207SE/raia10/

raia10@hvs-its-lnx01:~$ sleep 10000 &
[1] 8321
raia10@hvs-its-lnx01:~$ sleep 20000
^Z
[2]+  Stopped                  sleep 20000
raia10@hvs-its-lnx01:~$ jobs
[1]-  Running                  sleep 10000 &
[2]+  Stopped                  sleep 20000
raia10@hvs-its-lnx01:~$ kill %1
raia10@hvs-its-lnx01:~$ jobs
[1]-  Terminated              sleep 10000
[2]+  Stopped                  sleep 20000
raia10@hvs-its-lnx01:~$ kill -cont %2
raia10@hvs-its-lnx01:~$ jobs
[2]+  Running                  sleep 20000 &
raia10@hvs-its-lnx01:~$ ps au |grep raia10
raia10  7961  0.0  0.0  33888  5524 pts/5    Ss   16:39   0:00 -bash
raia10  8322  0.0  0.0   6016   816 pts/5    S    16:57   0:00 sleep 20000
raia10  8325  0.0  0.0  46384  3652 pts/5    R+   16:58   0:00 ps au
raia10  8326  0.0  0.0  12948  1020 pts/5    S+   16:58   0:00 grep --color=auto raia10
raia10@hvs-its-lnx01:~$ kill -9 8322
[2]+  Killed                  sleep 20000
raia10@hvs-its-lnx01:~$ jobs
raia10@hvs-its-lnx01:~$ ps au |grep raia10
raia10  7961  0.0  0.0  33888  5524 pts/5    Ss   16:39   0:00 -bash
raia10  8433  0.0  0.0  46384  3720 pts/5    R+   16:59   0:00 ps au
raia10  8434  0.0  0.0  33888  2032 pts/5    R+   16:59   0:00 -bash
raia10@hvs-its-lnx01:~$

```

- a. I ran the sleep process in the background.
  - b. I then started a sleep process in foreground.
  - c. I then suspended sleep process in foreground using **CTRL + Z**.
  - d. I then used **Jobs** command to show the two sleep processes.
  - e. I then killed the first sleep process using **Kill %1** which was running in background.
  - f. I then used **Jobs** command to show the first sleep process was terminated and the second was suspended.
  - g. I then used **kill -cont %2** to continue the second process in foreground.
  - h. I then used **Jobs** command to show the process was running continually in foreground.
  - i. I then used command **ps au |grep raia10** to show a list of all my processes running
  - j. I then killed the remaining sleep process using **kill -9 8322**
- b) Paragraph on disown and nohup command

[paragraph here]

### **NOHUP & Disown**

Nohup is a command which allows a process to continue executing even after you have logged out. Nohup also known as 'No hang up', when the Nohup command is executed, it ignores all the hang up signals in the terminal so that the user can exit the terminal and the process will still be running in the background. This is done by separating the command from the shell and therefore allowing the process to continue after the shell is terminated. By default, the standard output will be redirected to nohup.out file which contains the standard output and error messages from the script that you've executed using Nohup command. The command to run Nohup is '**nohup command &**'.

Disown on the other hand, removes the job from the shell's job list and all the sub points such as the process like SIGNUP sent by the shell don't apply and it is still connected to the terminal. If the terminal is destroyed and the controlling program is terminated by closing xterm or SSH connection, the program will fail as soon as it tries to read from standard input or write to standard output.

In conclusion, disown removes the process from the shell's job control but still leaves it connected to the terminal and Nohup disconnects the process from the terminal redirecting its output to nohup.out. Both commands suppress SIGHUP(hangup) signals so that the program isn't automatically killed when controlling terminal is closed. With Nohup, you must 'nohup' before a job/process begins however, if you did not then you can use 'disown' to modify the running job/process to keep executing even after you have logged out.

- c) Example of using watch command  
[example of nohup here]

**Watch** command executes a program periodically, showing output(changes) of a program. When the command is executed it runs repeatedly, displaying its output from the first screen full which allows users to watch the program output change overtime. By default, when the command is executed, it will run programs every 2 seconds. This can be changed to using **-n**( where n is time) or **-interval** to specify a different interval. The **-d** or **---differences** will highlight the differences between updates. **-cumulative** will run a display of all positions changed, **-t** or **-no—title** will turn off headers, time and blank line. **Watch** keeps running until interrupted.

#### Examples

To watch for mail = **watch -n 60 from**

To watch for directory change of a content = **watch -d ls -l**

Run **ls** command every 1 second = **watch -n 1 ls -l**

- d) References for this activity

Difference between nohup, disown and &. "Difference Between Nohup, Disown And &". *Unix.stackexchange.com*. N.p., 2017. Web. 10 Apr. 2017.

"Unix Nohup: Run A Command Or Shell-Script Even After You Logout". *Linux.101hacks.com*. N.p., 2017. Web. 10 Apr. 2017.

"Watch - Unix, Linux Command". *Tutorialspoint.com*. N.p., 2017. Web. 10 Apr. 2017.

## Lab Activity 15 IPC and Synchronisation

### a) Brief description of activity

In this activity to modify semaphore example code so that the two processes output the scene from Macbeth by William Shakespeare, so that one process says Macbeth's lines and the other says Lady Macbeth's lines. Furthermore, for advanced task I had to modify critical\_example2.c to write Lady Macbeth's part to stderr instead of stdout, as well as writing the lines to the screen, Macbeth's lines and Lady Macbeth's had to be redirected to a file of their own.

### b) Modified semaphore example code so that the two processes output the scene from MacBeth [Comment code here]

```
1. //sem.example.c
2. #include <sys/ipc.h>
3. #include <sys/sem.h>
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <unistd.h>
7.
8. #include "se207_sems.h"
9.
10. int main(int argc, char argv[]){
11.     //Use our source file as the "key"
12.     int id=se207_semget("sem_example.c",1);
13.     //int id1 =se207_semget("critical_example2.c",1);
14.
15.
16.     int pid=fork();
17.     if(pid){
18.         //Process 1 is LADY MACBETH
19.         while(1){
20.
21.             se207_wait(id);
22.             printf("LADY MACBETH: Donalbain.\n");
23.             rsleep();
24.
25.             se207_signal(id);
26.
27.             se207_wait(id);
28.             printf("LADY MACBETH: A foolish thought, to say a sorry sight.\n");
29.             rsleep();
30.
31.             se207_signal(id);
32.
33.             se207_wait(id);
34.             printf("LADY MACBETH: There are two lodged together.\n");
35.             rsleep();
36.
37.             se207_signal(id);
38.
39.             se207_wait(id);
40.             printf("LADY MACBETH: Consider it not so deeply.?\n");
41.             rsleep();
42.
43.             se207_signal(id);
44.
45.             se207_wait(id);
46.             printf("LADY MACBETH: These deeds must not be thought\n");
47.             rsleep();
48.             printf("\t After these ways; so, it will make us mad.\n");
49.             rsleep();
50.             rsleep();
```



```

51.
52.     se207_signal(id);
53.
54.         se207_wait(id);
55.         printf("LADY MACBETH: What do you mean?\n");
56.         rsleep();
57.
58.     se207_signal(id);
59.
60.
61.
62.     }
63. }else{
64.     //Process 2 is MACBETH
65.     while(1){
66.
67.         dup2(1,2);
68.
69.         se207_wait(id);
70.         printf("MACBETH: This is a sorry sight.\n");
71.         rsleep();
72.
73.         se207_signal(id);
74.
75.         se207_wait(id);
76.         printf("MACBETH: There's one did laugh in's sleep, and one cried\n");
77.         rsleep();
78.         printf("\t 'Murder!'\n");
79.         rsleep();
80.         printf("\t That they did wake each other: I stood and heard them:\n");
81.         rsleep();
82.         printf("\t But they did say their prayers, and address'd them'\n");
83.         rsleep();
84.         printf("\t Again to sleep.'\n");
85.         rsleep();
86.
87.         se207_signal(id);
88.
89.         se207_wait(id);
90.         printf("MACBETH: One cried 'God bless us!' and 'Amen' the other;\n");
91.         rsleep();
92.         printf("\t As they had seen me with these hangman's hands.\n");
93.         rsleep();
94.         printf("\t Listening their fear, I could not say 'Amen'\n");
95.         rsleep();
96.         printf("\t When they did say 'God bless us!'\n");
97.         rsleep();
98.
99.         se207_signal(id);
100.
101.             se207_wait(id);
102.             printf("MACBETH: But wherefore could not I pronounce 'Amen'? \n");
103.             rsleep();
104.             printf("\t I had most need of blessing, and 'Amen'\n");
105.             rsleep();
106.             printf("\t Stuck in my throat.\n");
107.             rsleep();
108.
109.             se207_signal(id);
110.
111.             se207_wait(id);
112.             printf("MACBETH: Methought I heard a voice cry 'Sleep no more!\n");
113.             rsleep();
114.             printf("\t Macbeth does murder sleep', the innocent sleep,\n");
115.             rsleep();
116.             printf("\t Sleep that knits up the ravell'd sleeve of care,\n");

```

```

117.         rsleep();
118.         printf("\t The death of each day's life, sore labour's bath,\n");
119.         rsleep();
120.         printf("\t Balm of hurt minds, great nature's second course,\n");
121.         rsleep();
122.         printf("\t Chief nourisher in life's feast,-- \n");
123.         rsleep();
124.
125.         se207_signal(id);
126.
127.     }
128.
129.
130.
131. }
132.
133.
134.
135. }

```

[Screenshot(s) showing code working]

**Command used :** 1. gcc -o sem\_example critical\_example2.c  
2. ./sem\_example

```

os-207se.coventry.ac.uk
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
/home/207SE/raia10/Portfolio2/L
critical_example2.c
se207_sems.h
sem_example
sem_example.c
Follow terminal folder
raia10@hvs-its-lnx01:~/Portfolio2/Lab15/lab15$ ./sem_example
Semaphore 17957823 initialized with path 'critical_example2.c'.
LADY MACBETH: Donalbain.
MACBETH: This is a sorry sight.
LADY MACBETH: A foolish thought, to say a sorry sight.
MACBETH: There's one did laugh 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.'
LADY 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 fear, I could not say 'Amen,'
When they did say 'God bless us!'
LADY MACBETH: Consider it not so deeply.?
MACBETH: But wherefore could not I pronounce 'Amen'?
I had most need of blessing, and 'Amen'
Stuck in my throat.
LADY MACBETH: These deeds must not be thought
After these ways; so, it will make us mad.
MACBETH: Methought I heard a voice cry 'Sleep no more!
Macbeth does murder sleep', the innocent sleep,
Sleep that knits up the ravell'd sleeve of care,
The death of each day's life, sore labour's bath, Balm of hurt minds, great nature's second course,
Chief nourisher in life's feast,--
LADY MACBETH: What do you mean?

```

- c) Modified code to write Lady MacBeth's part to stderr and redirect the two parts to different files.  
[Changes to code above here with comments]

```

1. //critical_example2.c
2. #include <sys/ipc.h>
3. #include <sys/sem.h>
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <unistd.h>
7.
8.

```

```

9. #include "se207_sems.h"
10.
11. int main(int argc, char argv[]){
12.     //Use our source file as the "key"
13.     int id=se207_semget("critical_example2.c",1);
14.     int countM = 0;
15.     int countL = 0;
16.     FILE *fp;
17.     FILE *fp2;
18.
19.     int pid=fork();
20.     if(pid){
21.         //P1
22.
23.         fp = freopen("ladyLines.txt", "w" , stderr);
24.
25.         while(countL != 6){
26.             se207_wait(id);
27.
28.             if (countL == 0) {
29.                 printf("LADY MACBETH Donalbain.\n");
30.                 fprintf(stderr, "LADY MACBETH Donalbain.\n");
31.             }
32.
33.             if (countL == 1) {
34.                 printf("LADY MACBETH A foolish thought, to say a sorry sight.\n");
35.                 fprintf(stderr, "LADY MACBETH A foolish thought, to say a sorry sight.\n");
36.             }
37.
38.             if (countL == 2) {
39.                 printf("LADY MACBETH There are two lodged together.\n");
40.                 fprintf(stderr, "LADY MACBETH There are two lodged together.\n");
41.             }
42.
43.             if (countL == 3) {
44.                 printf("LADY MACBETH Consider it not so deeply.\n");
45.                 fprintf(stderr, "LADY MACBETH Consider it not so deeply.\n");
46.             }
47.
48.             if (countL == 4) {
49.                 printf("LADY MACBETH These deeds must not be thought\n" "After these ways;
so, it will make us mad.\n");
50.                 fprintf(stderr, "LADY MACBETH These deeds must not be thought\n" "After the
se ways; so, it will make us mad.\n");
51.             }
52.
53.             if (countL == 5) {
54.                 printf("LADY MACBETH What do you mean?\n");
55.                 fprintf(stderr, "LADY MACBETH What do you mean?\n");
56.             }
57.
58.             rsleep();
59.             countL++;
60.             se207_signal(id);
61.         }
62.         fclose(stderr);
63.
64.     }else{
65.         //P2
66.
67.         fp2 = fopen("macbethLines.txt", "w");
68.
69.         while(countM != 5){
70.             se207_wait(id);
71.

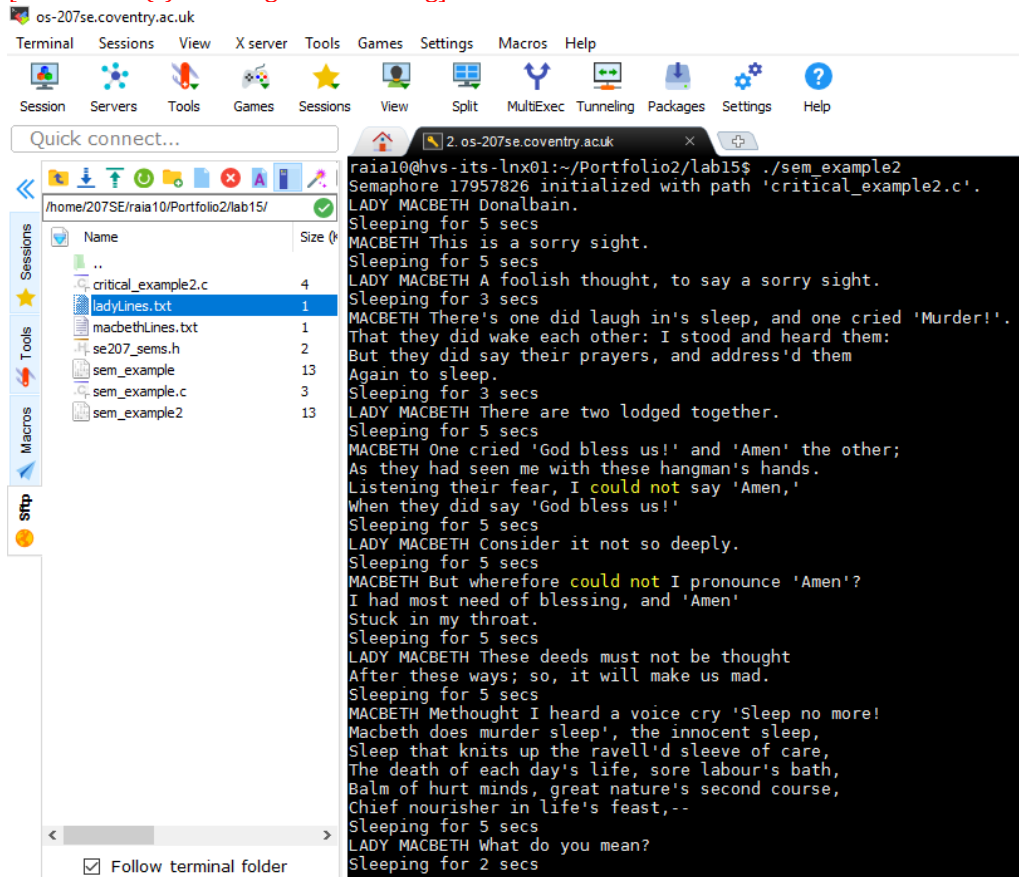
```

```

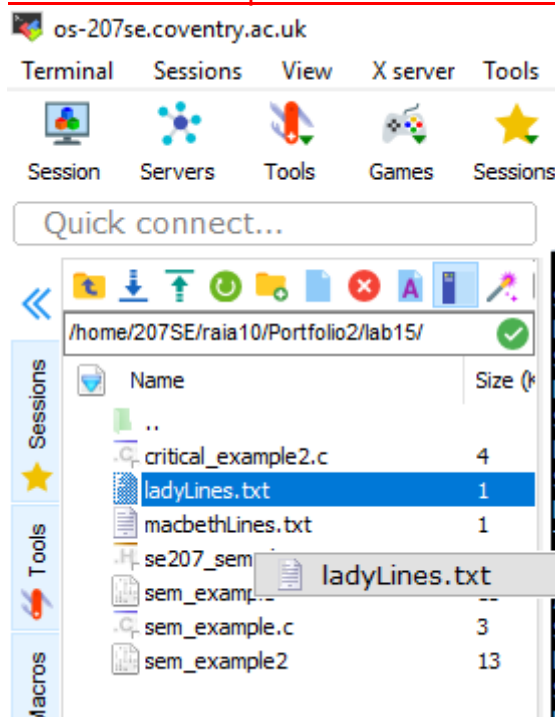
72.     if (countM == 0) {
73.         printf("MACBETH This is a sorry sight.\n");
74.         fprintf(fp2, "MACBETH This is a sorry sight.\n");
75.     }
76.
77.     if (countM == 1) {
78.         printf("MACBETH There's one did laugh in's sleep, and one cried 'Murder!'. \n"
79.             "That they did wake each other: I stood and heard them:\n" "But they did say the
80.             ir prayers, and address'd them\n""Again to sleep.\n");
81.         fprintf(fp2, "MACBETH There's one did laugh in's sleep, and one cried 'Murder!'. \n"
82.             "That they did wake each other: I stood and heard them:\n" "But they did say
83.             ay their prayers, and address'd them\n""Again to sleep.\n");
84.     }
85.
86.     if (countM == 2) {
87.         printf("MACBETH One cried 'God bless us!' and 'Amen' the other;\n" "As they
88.             had seen me with these hangman's hands.\n" "Listening their fear, I could not say
89.             'Amen,'\n" "When they did say 'God bless us!'\n");
90.         fprintf(fp2, "MACBETH One cried 'God bless us!' and 'Amen' the other;\n" "As
91.             they had seen me with these hangman's hands.\n" "Listening their fear, I could not
92.             say 'Amen,'\n" "When they did say 'God bless us!'\n");
93.     }
94.
95.     if (countM == 3) {
96.         printf("MACBETH But wherefore could not I pronounce 'Amen'? \n" "I had most
97.             need of blessing, and 'Amen'\n" "Stuck in my throat.\n");
98.         fprintf(fp2, "MACBETH But wherefore could not I pronounce 'Amen'? \n" "I had
99.             most need of blessing, and 'Amen'\n" "Stuck in my throat.\n");
100.    }
101.
102.     if (countM == 4) {
103.         printf("MACBETH Methought I heard a voice cry 'Sleep no more!\n" "Macbeth does
104.             murder sleep', the innocent sleep,\n" "Sleep that knits up the ravell'd sleeve
105.             of care,\n" "The death of each day's life, sore labour's bath,\n" "Balm of hurt
106.             minds, great nature's second course,\n" "Chief nourisher in life's feast,--\n");
107.         fprintf(fp2, "MACBETH Methought I heard a voice cry 'Sleep no more!\n" "Macbeth
108.             does murder sleep', the innocent sleep,\n" "Sleep that knits up the ravell'd sleeve
            of care,\n" "The death of each day's life, sore labour's bath,\n" "Balm of hurt
            minds, great nature's second course,\n" "Chief nourisher in life's feast,--\n");
109.     }
110.
111.     rsleep();
112.     countM++;
113.     se207_signal(id);
114. }
115. fclose(stdout);
116. }
117. }
118. }

```

[Screenshot(s) showing code working]



Redirect the two parts to different files.



#### Redirected Lady Macbeth's Lines to ladyLines.txt

```
raial0@hvs-its-lnx01:~/Portfolio2/lab15$ cat ladyLines.txt
LADY MACBETH Donalbain.
LADY MACBETH A foolish thought, to say a sorry sight.
LADY MACBETH There are two lodged together.
LADY MACBETH Consider it not so deeply.
LADY MACBETH These deeds must not be thought
After these ways; so, it will make us mad.
LADY MACBETH What do you mean?
```

#### Redirected Macbeth's Lines to macbethLines.txt

```
raial0@hvs-its-lnx01:~/Portfolio2/lab15$ cat macbethLines.txt
MACBETH This is a sorry sight.
MACBETH There's one did laugh 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.
MACBETH One cried 'God bless us!' and 'Amen' the other;
As they had seen me with these hangman's hands.
Listening their fear, I could not say 'Amen,'
When they did say 'God bless us!'
MACBETH But wherefore could not I pronounce 'Amen'?
I had most need of blessing, and 'Amen'
Stuck in my throat.
MACBETH Methought I heard a voice cry 'Sleep no more!
Macbeth does murder sleep', the innocent sleep,
Sleep that knits up the ravell'd sleeve of care,
The death of each day's life, sore labour's bath,
Balm of hurt minds, great nature's second course,
Chief nourisher in life's feast,--
raial0@hvs-its-lnx01:~/Portfolio2/lab15$ █
```

## Lab Activity 16: IPC and Synchronisation II

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

[Brief description here]

The producer/consumer problem, also known as bounded-buffer problem describes two processes, they both share a common fixed size buffer used as a queue. The producers job is to generate data, put it into the buffer and then start the process again. The consumer at the same time is consuming data such as removing it from the buffer, one piece at a time. The problem is that to make sure the producer does not add data into the buffer if the buffer is full and that the consumer does not try to delete or remove data from an empty buffer. Therefore, the producer only has two choices, either to go to sleep or discard data if it finds that the buffer is full, so when the consumer tries to remove an item from the buffer next time, the producer is notified about the activity and instantly start to fill the buffer again so that the buffer is not empty. Similarly, the consumer goes to sleep if the buffer is empty therefore when the producer adds data into the buffer, the consumer will be notified and starts to remove data from the buffer.

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

[Commented code here]

```
1. #include <sys/ipc.h>
2. #include <sys/sem.h>
3. #include <sys/shm.h>
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include "se207_sems.h"
7.
8. /* Remember to try reversing the timings...*/
9.
10. int bufferlength=8; //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.
19.     int consumerpos = bufferlength + 1;
20.     int producerpos = bufferlength + 2;
21.
22.     //Use our source file as the "key"
23.     int id=se207_semget("prodcon_example2.c",0);
24.
25.     char* data; //For our pointer to shared memory...
26.
27.     int pid=fork();
28.     if(pid){
29.         //P1 - CONSUMER
30.         shm_id=shmget(ftok("prodcon_example2.c",2),0,066);
31.
32.         //Attach the shared buffer
33.         data = shmat(shm_id, (void *)0, 0);
34.
35.         data[consumerpos] = 0;
36.         data[producerpos] = 0;
37.
38.         while(1){
39.             se207_wait(id);
40.             while (data[consumerpos] == data[producerpos]);
41.             printf("Consuming item number %d...\n",data[consumerpos]);
42.             sleep(1);
43. }
```

```

44.     char item=data[consumerpos];
45.
46.     printf("Consumed item number %d. Item value was %d\n",
47.         data[consumerpos],item);
48.     data[consumerpos] =(data[consumerpos] + 1) % bufferlength;
49. }
50.
51. //Detatch
52. shmdt(data);
53. printf("All done consuming.\n");
54.
55. wait(); //For child process so that we can
56.
57. //Delete the shared memory
58. printf("Child ended, removing shm\n");
59. shmctl(shm_id, IPC_RMID, NULL);
60. }else{
61.     //P2
62.     shm_id=shmget(ftok("prodcon_example2.c",2),0,006);
63.     //Attach the shared buffer
64.     data = shmat(shm_id, (void *)0, 0);
65.
66.     data[consumerpos] = 0;
67.     data[producerpos] = 0;
68.
69.     while(1){ //changed to true so loop runs forever
70.         while (((data[producerpos] + 1)%bufferlength) == data[consumerpos]);
71.         printf("Producing item number %d...\n",data[producerpos]);
72.         sleep(2);
73.         data[data[producerpos]]=data[producerpos]*2; //Simple data, easy to check.
74.         printf("Produced item number %d. Value is %d\n",
75.             data[producerpos],data[data[producerpos]]);
76.         data[producerpos] = (data[producerpos] + 1)%bufferlength;
77.         se207_signal(id);
78.     }
79.     //Detatch
80.     shmdt(data);
81.     printf("Producer finished.");
82. }
83. }

```



- Ring/Circular buffer with different buffer sizes

[Screenshot(s) showing code working as a circular buffer, with different buffer sizes]

### BUFFER SIZE = 8

```

raia10@hvs-its-lnx01:~/Portfolio2/lab16$ ./prodcon
Semaphore 17957853 initialized with path 'prodcon_example2.c'.
Producing item number 0...
Produced item number 0. Value is 0
Consuming item number 0...
Producing item number 1...
Consumed item number 0. Item value was 0
Produced item number 1. Value is 2
Consuming item number 1...
Producing item number 2...
Consumed item number 1. Item value was 1
Produced item number 2. Value is 4
Consuming item number 2...
Producing item number 3...
Consumed item number 2. Item value was 2
Produced item number 3. Value is 6
Consuming item number 3...
Producing item number 4...
Consumed item number 3. Item value was 3
Produced item number 4. Value is 8
Consuming item number 4...
Producing item number 5...
Consumed item number 4. Item value was 4
Produced item number 5. Value is 10
Consuming item number 5...
Producing item number 6...
Consumed item number 5. Item value was 5
Produced item number 6. Value is 12
Consuming item number 6...
Producing item number 7...
Consumed item number 6. Item value was 6
Produced item number 7. Value is 14
Consuming item number 7...
Producing item number 0...
Consumed item number 7. Item value was 7
Produced item number 0. Value is 0
Consuming item number 0...
Producing item number 1...

```

### BUFFER SIZE = 10

```

raia10@hvs-its-lnx01:~/Portfolio2/lab16$ ./prodcon
Semaphore 17957853 initialized with path 'prodcon_example2.c'.
Producing item number 0...
Produced item number 0. Value is 0
Consuming item number 0...
Producing item number 1...
Consumed item number 0. Item value was 0
Produced item number 1. Value is 2
Consuming item number 1...
Producing item number 2...
Consumed item number 1. Item value was 1
Produced item number 2. Value is 4
Consuming item number 2...
Producing item number 3...
Consumed item number 2. Item value was 2
Produced item number 3. Value is 6
Consuming item number 3...
Producing item number 4...
Consumed item number 3. Item value was 3
Produced item number 4. Value is 8
Consuming item number 4...
Producing item number 5...
Consumed item number 4. Item value was 4
Produced item number 5. Value is 10
Consuming item number 5...
Producing item number 6...
Consumed item number 5. Item value was 5
Produced item number 6. Value is 12
Consuming item number 6...
Producing item number 7...
Consumed item number 6. Item value was 6
Produced item number 7. Value is 14
Consuming item number 7...
Producing item number 8...
Consumed item number 7. Item value was 7
Produced item number 8. Value is 16
Consuming item number 8...
Producing item number 9...

```

- Producer/Consumer working at different speeds

[Screenshot(s) showing the producer and consumer working at different speeds]

**CONSUMER SPEED – 1**

**PRODUCER SPEED – 2**

```

raia10@hvs-its-lnx01:~/Portfolio2/lab16$ ./prodcon
Semaphore 17957853 initialized with path 'prodcon_example2.c'.
Producing item number 0...
Produced item number 0. Value is 0
Consuming item number 0...
Producing item number 1...
Consumed item number 0. Item value was 0
Produced item number 1. Value is 2
Consuming item number 1...
Producing item number 2...
Consumed item number 1. Item value was 1
Produced item number 2. Value is 4
Consuming item number 2...
Producing item number 3...
Consumed item number 2. Item value was 2
Produced item number 3. Value is 6
Consuming item number 3...
Producing item number 4...
Consumed item number 3. Item value was 3
Produced item number 4. Value is 8
Consuming item number 4...
Producing item number 5...
Consumed item number 4. Item value was 4
Produced item number 5. Value is 10
Consuming item number 5...
Producing item number 6...
Consumed item number 5. Item value was 5
Produced item number 6. Value is 12
Consuming item number 6...
Producing item number 7...
Consumed item number 6. Item value was 6
Produced item number 7. Value is 14
Consuming item number 7...
Producing item number 0...
Consumed item number 7. Item value was 7
Produced item number 0. Value is 0
Consuming item number 0...
Producing item number 1...

```

**CONSUMER SPEED – 1**

**PRODUCER SPEED – 3**

```

raia10@hvs-its-lnx01:~/Portfolio2/lab16$ ./prodcon
Semaphore 17957853 initialized with path 'prodcon_example2.c'.
Producing item number 0...
Produced item number 0. Value is 0
Consuming item number 0...
Producing item number 1...
Consumed item number 0. Item value was 0
Produced item number 1. Value is 2
Consuming item number 1...
Producing item number 2...
Consumed item number 1. Item value was 1
Produced item number 2. Value is 4
Consuming item number 2...
Producing item number 3...
Consumed item number 2. Item value was 2
Produced item number 3. Value is 6
Consuming item number 3...
Producing item number 4...
Consumed item number 3. Item value was 3
Produced item number 4. Value is 8
Consuming item number 4...
Producing item number 5...
Consumed item number 4. Item value was 4
Produced item number 5. Value is 10
Consuming item number 5...
Producing item number 6...
Consumed item number 5. Item value was 5
Produced item number 6. Value is 12
Consuming item number 6...
Producing item number 7...
Consumed item number 6. Item value was 6
Produced item number 7. Value is 14
Consuming item number 7...
Producing item number 0...
Consumed item number 7. Item value was 7
Produced item number 0. Value is 0
Consuming item number 0...
Producing item number 1...

```

## References for this activity

"Cite A Website - Cite This For Me". *Dcs.ed.ac.uk*. N.p., 2017. Web. 10 Apr. 2017.

"Producer–Consumer Problem". *En.wikipedia.org*. N.p., 2017. Web. 10 Apr. 2017.

"The Producer/Consumer Problem (Multithreaded Programming Guide)". *Docs.oracle.com*. N.p., 2017. Web. 10 Apr. 2017.

## Lab Activity 17 TCP Server

### a) Brief description of the TCP Server Activity

In this task, I had to modify the tcp\_server.cc and tcp\_client.cc code in order to create a simple RPN calculator server. Furthermore, I had to add additional code to both tcp\_client.cc and tcp\_server.cc in order to implement a stack for values and operators that apply to the top pair.

### b) Commented Code showing changing to tcp-server.cc tcp-client.cc [Commented code here]

## 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.
```

```

34. cout << "Enter server port number: ";
35. cin >> serverPort;
36. cin.get(c); // dispose of the newline
37.
38. // Create a socket. "AF_INET" means it will use the IPv4 protocol.
39. // "SOCK_STREAM" means it will be a reliable connection (i.e., TCP;
40. // for UDP use SOCK_DGRAM), and I'm not sure what the 0 for the last
41. // parameter means, but it seems to work.
42. socketDescriptor = socket(AF_INET, SOCK_STREAM, 0);
43. if (socketDescriptor < 0) {
44.     cerr << "cannot create socket\n";
45.     exit(1);
46. }
47.
48. // Connect to server. First we have to set some fields in the
49. // serverAddress structure. The system will assign me an arbitrary
50. // local port that is not in use.
51. serverAddress.sin_family = hostInfo->h_addrtype;
52. memcpy((char *) &serverAddress.sin_addr.s_addr,
53.         hostInfo->h_addr_list[0], hostInfo->h_length);
54. serverAddress.sin_port = htons(serverPort);
55.
56. if (connect(socketDescriptor,
57.             (struct sockaddr *) &serverAddress,
58.             sizeof(serverAddress)) < 0) {
59.     cerr << "cannot connect\n";
60.     exit(1);
61. }
62.
63. cout << "\nEnter an RPN Expression(use space in between):\n";
64.
65.
66.
67. // Prompt the user for input, then read in the input, up to MAX_LINE
68. // characters, and then dispose of the rest of the line, including
69. // the newline character.
70. cout << "Input: ";
71. cin.get(buf, MAX_LINE, '\n');
72. while (cin.get(c) && c != '\n')
73.     ; //Loop does nothing except consume the spare bytes
74.
75.
76. // Stop when the user inputs a line with just a dot.
77. while (strcmp(buf, ".") { //strcmp returns 0 when the two strings
78.     //are the same, so this continues when
79.     //they are different
80.     // Send the line to the server.
81.     if (send(socketDescriptor, buf, strlen(buf) + 1, 0) < 0) {
82.         cerr << "cannot send data ";
83.         close(socketDescriptor); //Note this is just like using files...
84.         exit(1);
85.     }
86.
87.     // Zero out the buffer.
88.     memset(buf, 0x0, LINE_ARRAY_SIZE);
89.
90.     // Read the modified line back from the server.
91.     if (recv(socketDescriptor, buf, MAX_LINE, 0) < 0) {
92.         cerr << "didn't get response from server?";
93.         close(socketDescriptor);
94.         exit(1);
95.     }
96.
97.     cout << "Answer: " << buf << "\n";
98.
99.     // Prompt the user for input, then read in the input, up to MAX_LINE

```

```

100.         // characters, and then dispose of the rest of the line, including
101.         // the newline character. As above.
102.         cout << "Input: ";
103.         cin.get(buf, MAX_LINE, '\n');
104.         while (cin.get(c) && c != '\n')
105.             ; //Chomp chomp chomp
106.     }
107.
108.     close(socketDescriptor);
109.     return 0;
110. }

```

## 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. #include <stdio.h>
10.
11. #define MAX_MSG 100
12. #define LINE_ARRAY_SIZE (MAX_MSG+1)
13.
14. using namespace std;
15.
16. //Create stack, variable and
17. //top of stack variable.
18. int stack[MAX_MSG];
19. string str_temp;
20. int top;
21.
22. //function to push a value.
23. void push(int x){ stack[top++]=x; }
24.
25. //function to pop a value.
26. int pop(){
27.
28.     int t = stack[--top];
29.     stack[top]=0;
30.     return t;
31.
32. }
33.
34. int main()
35. {
36.     int listenSocket, connectSocket, i;
37.     unsigned short int listenPort;
38.     socklen_t clientAddressLength;
39.     struct sockaddr_in clientAddress, serverAddress;
40.     char line[LINE_ARRAY_SIZE];
41.
42.     cout << "Enter port number to listen on (between 1500 and 65000): ";
43.     cin >> listenPort;
44.
45.     // Create socket for listening for client connection
46.     // requests.
47.     listenSocket = socket(AF_INET, SOCK_STREAM, 0);
48.     if (listenSocket < 0) {
49.         cerr << "cannot create listen socket";
50.         exit(1);

```

```

51. }
52.
53. // Bind listen socket to listen port. First set various
54. // fields in the serverAddress structure, then call
55. // bind().
56.
57. // htonl() and htons() convert long integers and short
58. // integers (respectively) from host byte order (on x86
59. // this is Least Significant Byte first) to network byte
60. // order (Most Significant Byte first).
61. serverAddress.sin_family = AF_INET;
62. serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
63. serverAddress.sin_port = htons(listenPort);
64.
65. if (bind(listenSocket,
66.         (struct sockaddr *) &serverAddress,
67.         sizeof(serverAddress)) < 0) {
68.     cerr << "cannot bind socket";
69.     exit(1);
70. }
71.
72. // Wait for connections from clients. This is a
73. // non-blocking call; i.e., it registers this program with
74. // the system as expecting connections on this socket, and
75. // then this thread of execution continues on.
76. listen(listenSocket, 5);
77.
78. while (1) {
79.     cout << "Waiting for TCP connection on port " << listenPort << " ...\n";
80.
81.     // Accept a connection with a client that is requesting
82.     // one. The accept() call is a blocking call; i.e., this
83.     // thread of execution stops until a connection comes
84.     // in. connectSocket is a new socket that the system
85.     // provides, separate from listenSocket. We *could*
86.     // accept more connections on listenSocket, before
87.     // connectSocket is closed, but this program doesn't do
88.     // that.
89.     clientAddressLength = sizeof(clientAddress);
90.     connectSocket = accept(listenSocket,
91.                            (struct sockaddr *) &clientAddress,
92.                            &clientAddressLength);
93.     if (connectSocket < 0) {
94.         cerr << "cannot accept connection ";
95.         exit(1);
96.     }
97.
98.     // Show the IP address of the client.
99.     // inet_ntoa() converts an IP address from binary form to the
100.    // standard "numbers and dots" notation.
101.    cout << " connected to " << inet_ntoa(clientAddress.sin_addr);
102.
103.    // Show the client's port number.
104.    // ntohs() converts a short int from network byte order (which is
105.    // Most Significant Byte first) to host byte order (which on x86,
106.    // for example, is Least Significant Byte first).
107.    cout << ":" << ntohs(clientAddress.sin_port) << "\n";
108.
109.    // Read lines from socket, using recv(), storing them in the line
110.    // array. If no messages are currently available, recv() blocks
111.    // until one arrives.
112.    // First set line to all zeroes, so we'll know where the end of
113.    // the string is.
114.    memset(line, 0x0, LINE_ARRAY_SIZE);
115.    while (recv(connectSocket, line, MAX_MSG, 0) > 0) {
116.

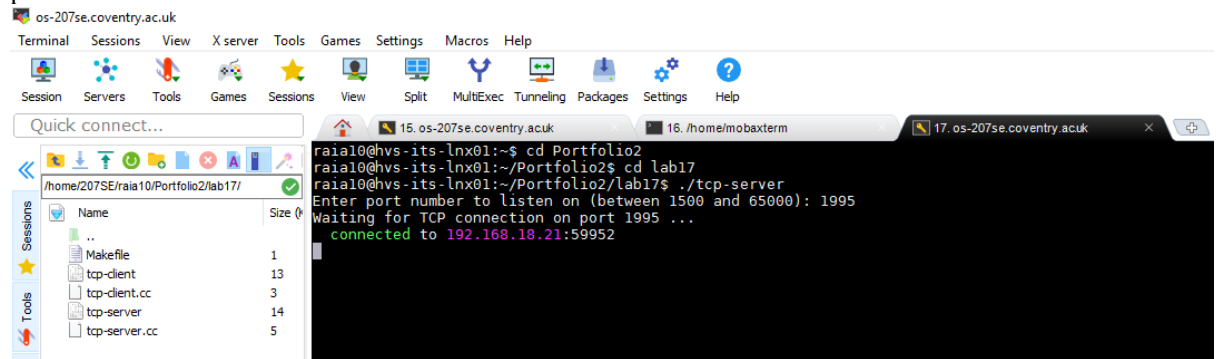
```

```

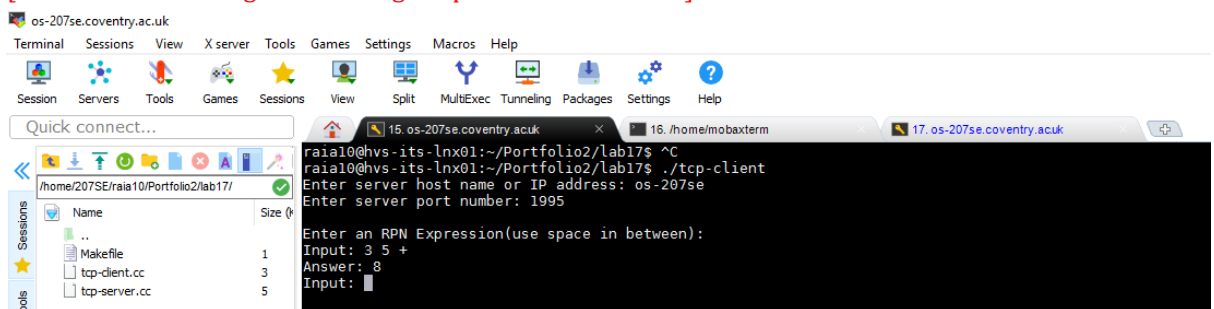
117.         // RPN evaluation loop.
118.         for (i =0; i<LINE_ARRAY_SIZE;i++)
119.         {
120.             //If number then add it to str_temp
121.             if(line[i] >='0' && line[i] <= '9'){
122.                 str_temp+=line[i];
123.             }
124.             // if separator char (SPACE)
125.             else if(line[i] == ' ')
126.             {
127.                 // if str_temp has a value in it,
128.                 // then push and clear the string.
129.                 if(!str_temp.empty()){
130.                     push(atoi(str_temp.c_str()));
131.                     str_temp.clear();}
132.                 // else go to next char in input string.
133.             else
134.                 continue;
135.
136.             }
137.             // if operator
138.             else if(line[i] == '+' || line[i] == '-'
139.                 ||line[i] == '*' ||line[i] == '/') {
140.
141.                 // pop two value from stack.
142.                 int a = pop();
143.                 int b = pop();
144.
145.                 // Do operation depending on operator
146.                 // push result to stack.
147.                 switch (line[i]){
148.
149.                 case '+':
150.                     push(a+b);
151.                     break;
152.
153.                 case '-':
154.                     push(b-a);
155.                     break;
156.
157.                 case '*':
158.                     push(a*b);
159.                     break;
160.
161.                 case '/':
162.                     push(b/a);
163.                     break;
164.
165.                 }
166.
167.             }
168.
169.         }
170.
171.         // Only value left in stack is the answer.
172.         sprintf(line,"%d",pop());
173.
174.         // Send converted line back to client.
175.         if (send(connectSocket, line, strlen(line) + 1, 0) < 0)
176.             cerr << "Error: cannot send modified data";
177.
178.         memset(line, 0x0, LINE_ARRAY_SIZE); // set line to all zeroes
179.     }
180. }
181. }

```

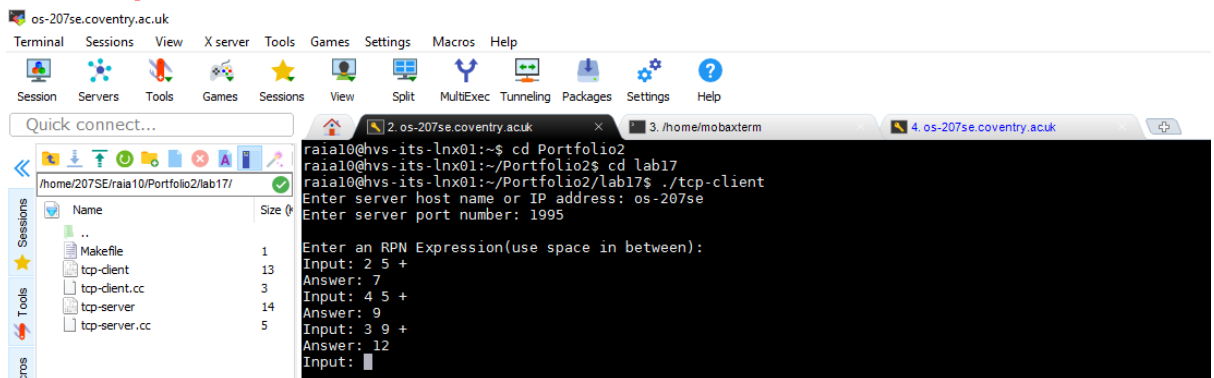
Before opening tcp-client.cc, I needed to open the server first so the client could connect to the port.



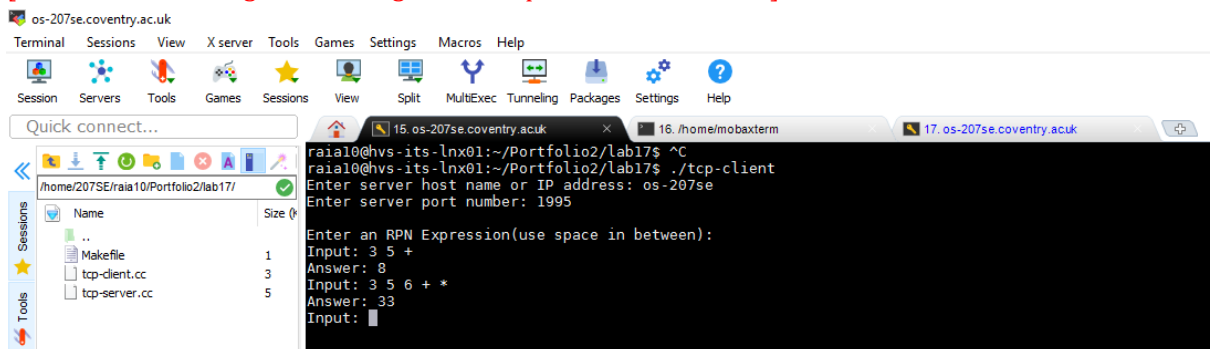
- c) Examples of TCP server doing simple RPN calculations (2 numbers 1 arithmetic operation)  
 [Screenshots showing server doing simple RPN calculations]



More examples:

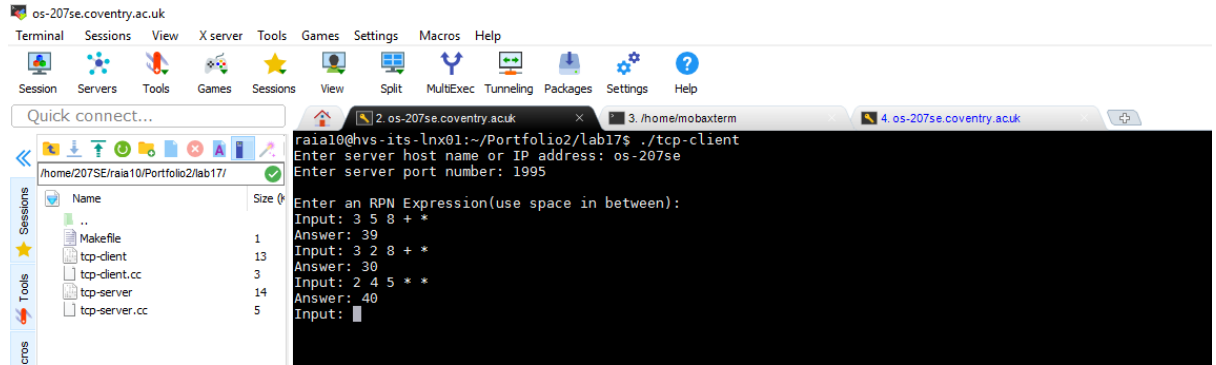


- d) Examples of TCP server doing more complex RPN calculations with stack or stack-like structure.  
 [Screenshots showing server doing more complex RPN calculations]





## More examples:



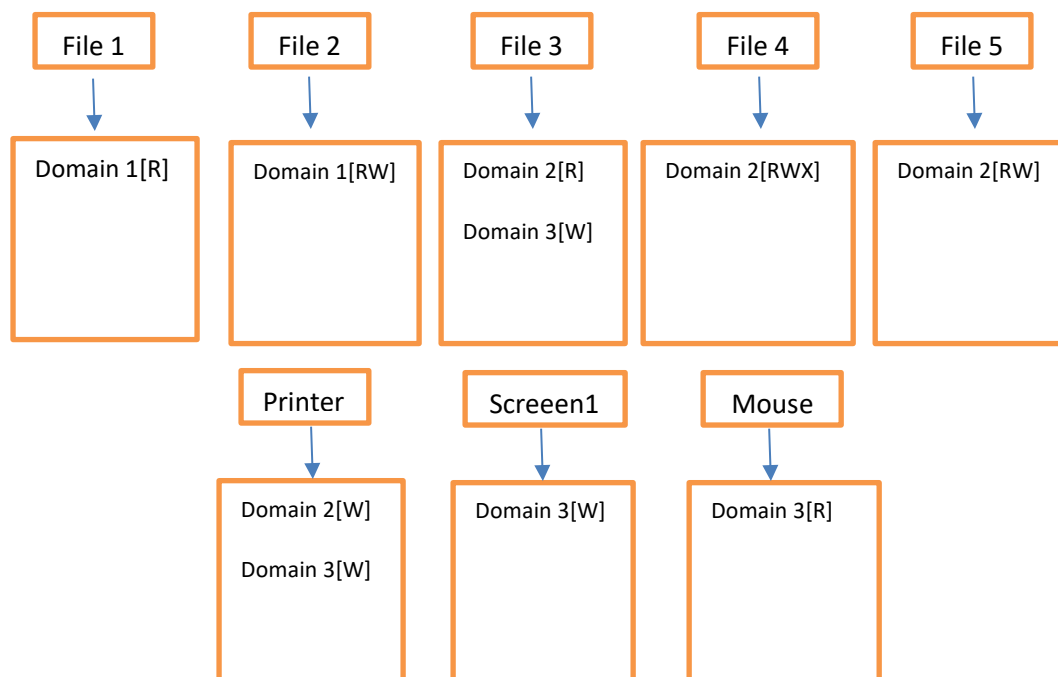
## Lab Activity 19 Security

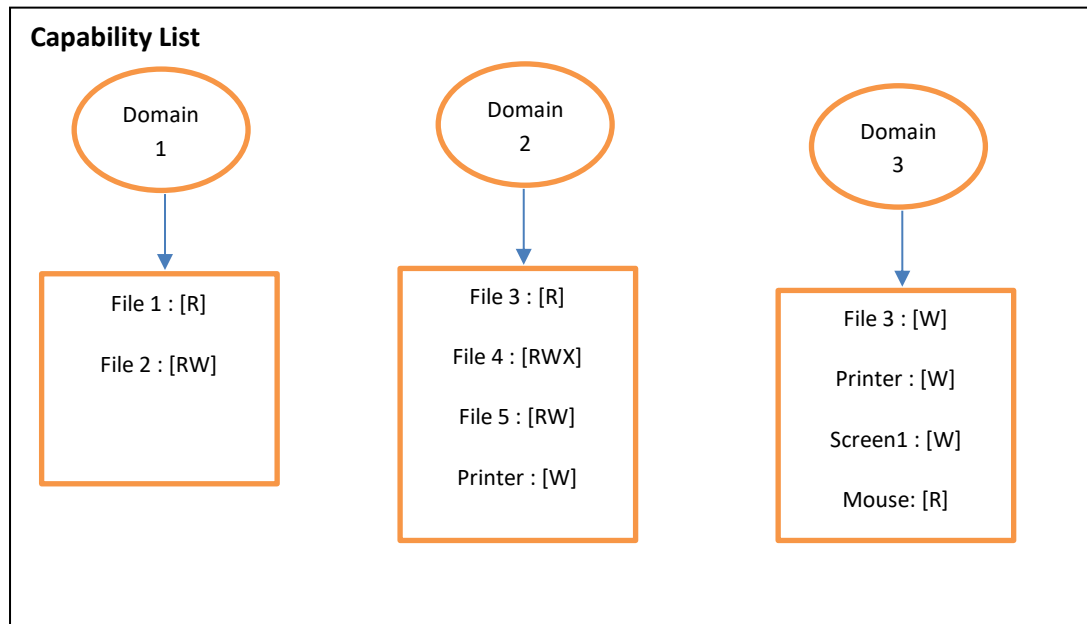
- a) Create a protection domain matrix, access list and capability list for the diagram below  
[Diagrams here]



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

### Access List (column)





b) Commented Code showing hash function

[Commented code here]

Program used to create hash function: Python

### Python Code

[Commented code here]

```

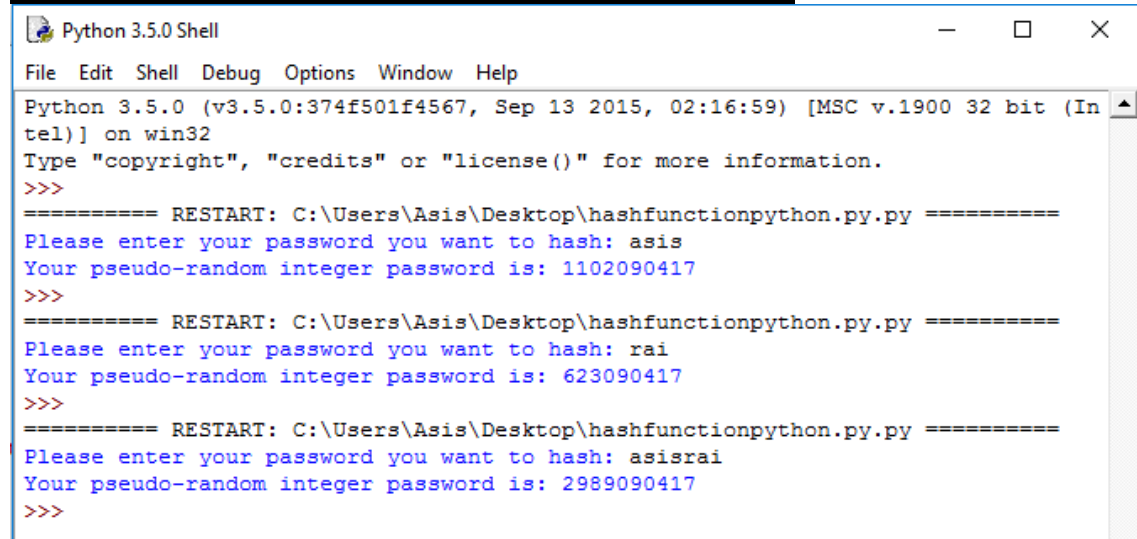
1. """a basic salted-hash function in a programming language of your
2. choice that takes a password as a string and produces a pseudo-random
3. integer based on that string"""
4.
5. import time
6.
7. #input
8. password=input("Please enter your password you want to hash: ")
9. #putting password in a list
10. password=list(password)
11. #new list
12. hashpassword=[]
13.
14. #asciivalue function, take each letter of the string and converting it to its ascii
    value
15. def asciivalue():
16.     for i in password:
17.         j=ord(i)
18.         hashpassword.append(j)
19.     return(hashpassword)
20.
21. #multiply function, multiple it by a specific number based on its position in the s
    tring sequence
22. def multiply(hashpassword):
23.     Counter=1
24.     integer=0
25.     while integer<len(hashpassword):
26.         hashpassword[integer]=hashpassword[integer]*Counter
27.         Counter=Counter+1
28.         integer=integer+1
29.     return(hashpassword)
30.
  
```

```

31. #add and salt function, add the values together and add salt in such as the current
    date.
32. def addnumberandsalt(hashpassword):
33.     total=sum(hashpassword)
34.     total=str(total)+time.strftime("%d%m%y")
35.     return(total)
36.
37. #run functions
38. asciivalue()
39. multiply(hashpassword)
40. total=addnumberandsalt(hashpassword)
41. print("Your pseudo-random integer password is: " + total)

```

### **Output showing your salted- hash function working**



```

Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Asis\Desktop\hashfunctionpython.py.py =====
Please enter your password you want to hash: asis
Your pseudo-random integer password is: 1102090417
>>>
===== RESTART: C:\Users\Asis\Desktop\hashfunctionpython.py.py =====
Please enter your password you want to hash: rai
Your pseudo-random integer password is: 623090417
>>>
===== RESTART: C:\Users\Asis\Desktop\hashfunctionpython.py.py =====
Please enter your password you want to hash: asisrai
Your pseudo-random integer password is: 2989090417
>>>

```

## **References**

"2. Built-In Functions — Python 2.7.13 Documentation". *Docs.python.org*. N.p., 2017. Web. 10 Apr. 2017.

"Hash Function". *En.wikipedia.org*. N.p., 2017. Web. 10 Apr. 2017.

python, hash. "Hash Function In Python". *Stackoverflow.com*. N.p., 2017. Web. 10 Apr. 2017.