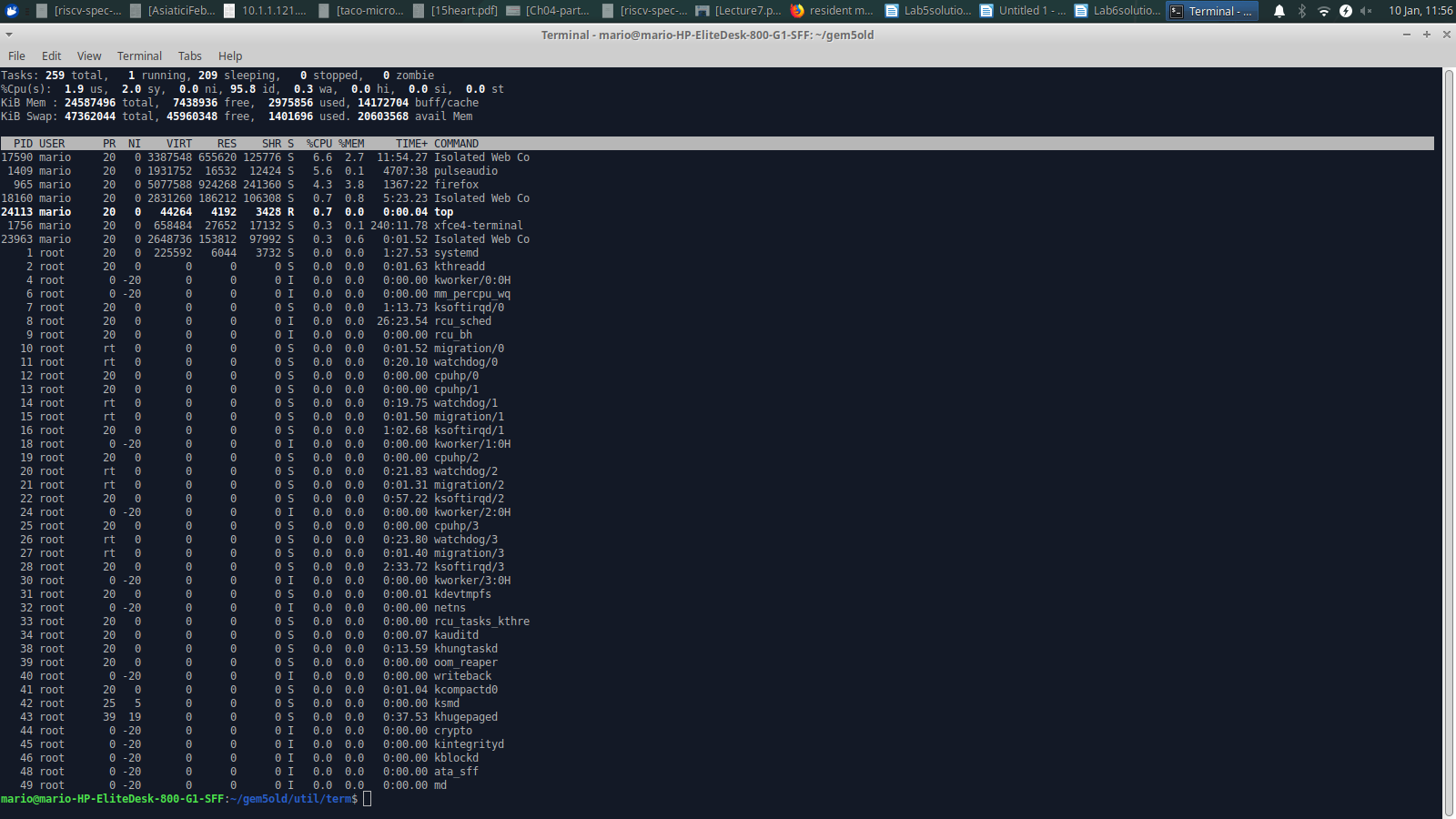
Exam 2 – OSP – 19/01/2023 – Duration 1:45min

Name: Anush Mali

ID: C729794



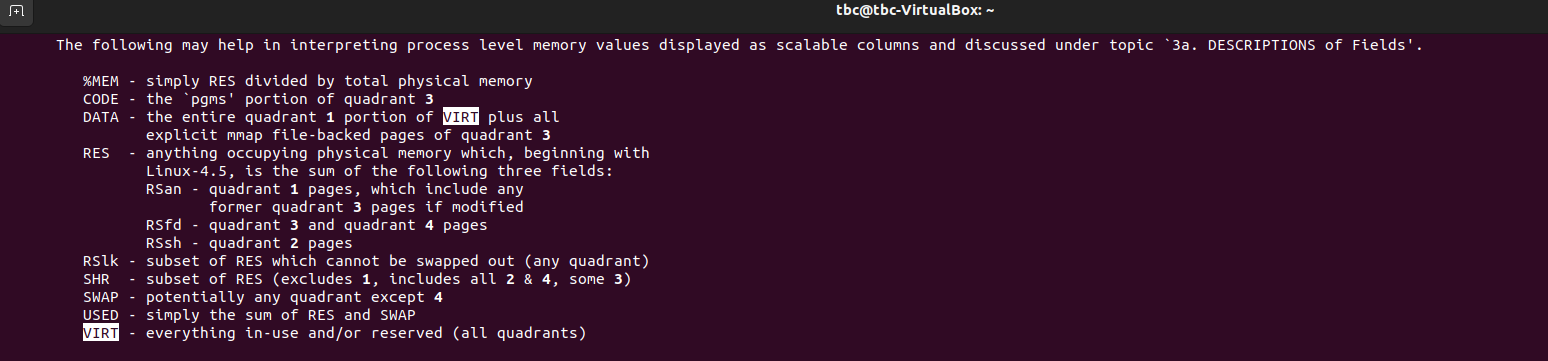
Question 1 (total: 50 marks)

Inspect the screenshot above and respond to the following questions:

1a. RES is the amount of physical memory a process is using. For the PID 17590, how much memory is being used? (5 marks)

Answer: for PID 17590 655620 KiB memory is being used.

1b. You can notice that PID 17590 shows 3387548 on the as VIRT. Do man top and find what the VIRT field represents in terms of memory. Provide a screenshot. (5 marks)

Answer: VIRT is everything in use and/or reserved

1c. On top’s upper left side, you find that this computer has Kib Mem 24587496 or ~24.59 GB (gigabytes) of physical memory (RAM). Supposing you **only** had RAM, what would be the maximum size of program that would be possible to be run? Explain. (5 marks)

Answer: The maximum size of program that would be possible to be run is 10,414,792 KiB.

1d. On top’s upper left side, you find that top shows Kib Swap 47362044 (~47.36GB - gigabytes).

Assume you have a program running on this computer and this program is using 30GB of memory. Briefly explain how the OS would manage its execution since the amount of memory it uses is larger than the RAM size (24.59GB)? (15 marks)

Answer: The OS manages the execution of program by using virtual memory and swapping techniques.

Virtual memory allows the OS to use disk space to simulate additional memory when the amount of physical memory is not sufficient. The OS uses the portion of hard drive as an extention of physical memory called swap.

Swapping is a process of moving entire process from memory to disk when the system is low in memory.

1e. The OS uses the concept of pages as units of management of its virtual memory. To facilitate management, pages are used instead of addresses. These pages are organized in page tables kept in the PCB. As other PCB elements, page tables are maintained in memory. Explain how many memory accesses are required to access a generic page (that is placed in the page table) (2.5 marks) and what the OS does do to speed them up? (2.5 marks) (total 5 marks)

Answer: Three memory accesses are required to access a generic page. First when the CPU looks up the page table in PCB to find the physical address of the page. Another memory access is required for the CPU to access the page table entry (PTE) that corresponds to page. The PTE contains physical address of the page in memory. Another memory access is required when CPU uses this address to access the page in memory.

To speed up this process OS uses technique called page caching. The page table is often stored in a special memory area called Translation Lookaside Buffer (TLB) which is a small, fast code that holds recently accessed page table entries. When page is accessed, CPU first checks the TLB to see if page table entry already in the cache, if it is then it reduces the time required as CPU can access page directly without having to looks up page table in memory.

1f. The virtual memory address space is managed via segments, and these are very useful to the programmer. List four different segments of the virtual address space (5 marks).

Answer: Four different segments of the virtual address space are:

1. Data segment
2. Code segment
3. Stack segment
4. Heap segment

1g. From the computer where top was run, you find below part of the of the output of the command cat /proc/cpuinfo:

address sizes: 48 bits virtual

Using the above info, indicate how you calculate the virtual address space of this computer. (5 marks)

Answer: using above info the virtual address space can be calculated as

2^48 address

2^48 address = 2^48 \* 1bytes = 6 Gbytes

1h. Explain the job of the OS in the following sentence: the virtual address space needs to be backed up by physical media. (5 marks)

Answer: The virtual address space is a logical construct that is mapped to the physical memory address by OS. It actually does not exist, unless backed up by RAM or hard drive/ssd. Hence, the virtual address space needs to be backed up by physical media.

Question 2 (25 marks)

In single-user dedicated systems, user can debug and find out when a program goes into an infinite loop. But in multiuser systems running large numbers of processes, user cannot easily determine that an individual process is not progressing.

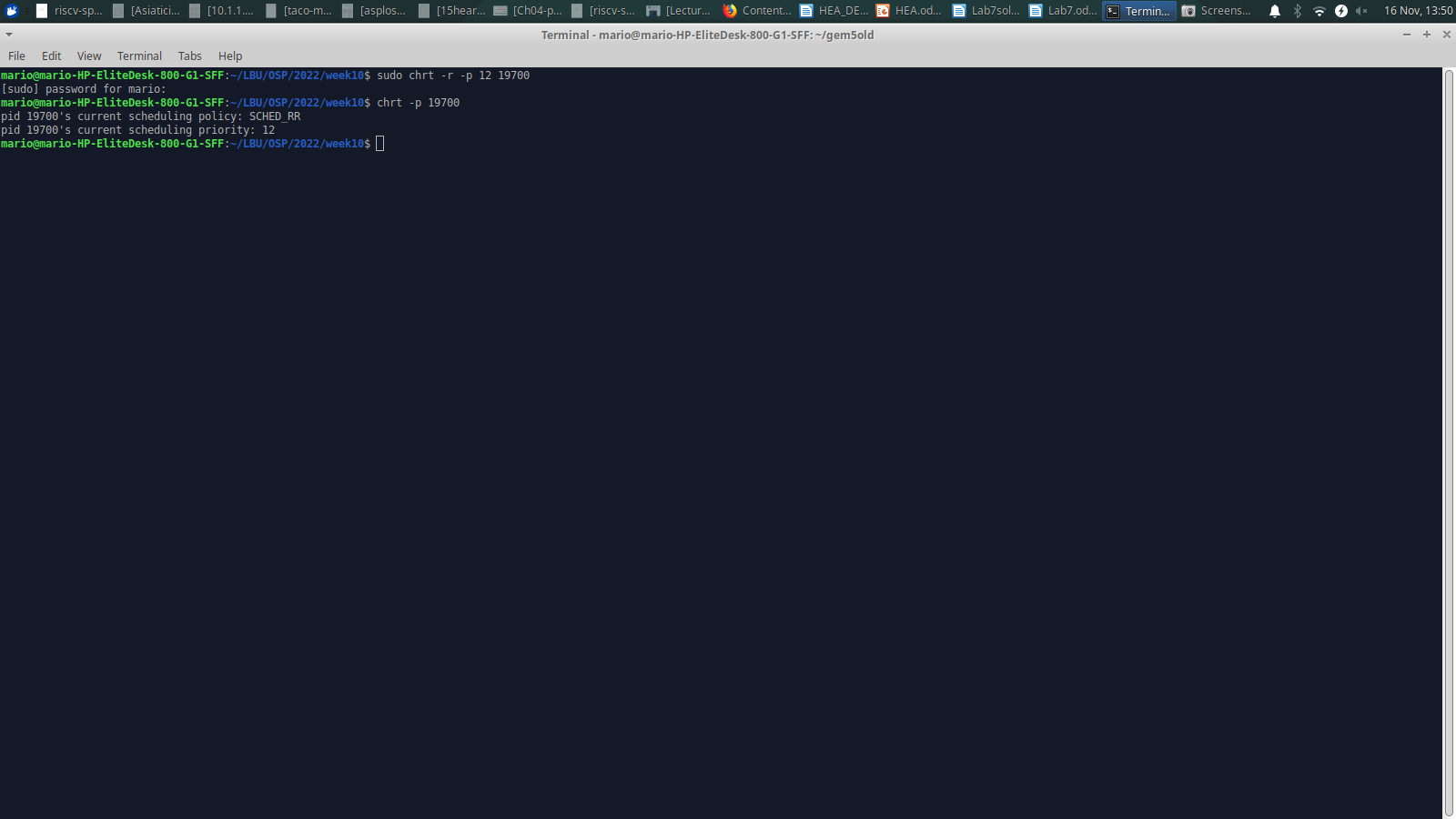
2a. Can the OS determine whether a process is in an infinite loop? (5 marks)

Answer: No, the OS cannot determine whether a process is in an infinite loop. We need to implement some process in the OS so that it would prevent processes in infinite loops.

2b. If you were implementing the OS, what would you use to to prevent processes in infinite loops from running indefinitely? (5 marks)

Answer: If I were implementing the OS, I would use Round Robin Scheduling to set maximum CPU usage limit for each process, and if a process exceeds this limit, the OS would assume it is an infinite loop and terminate it.

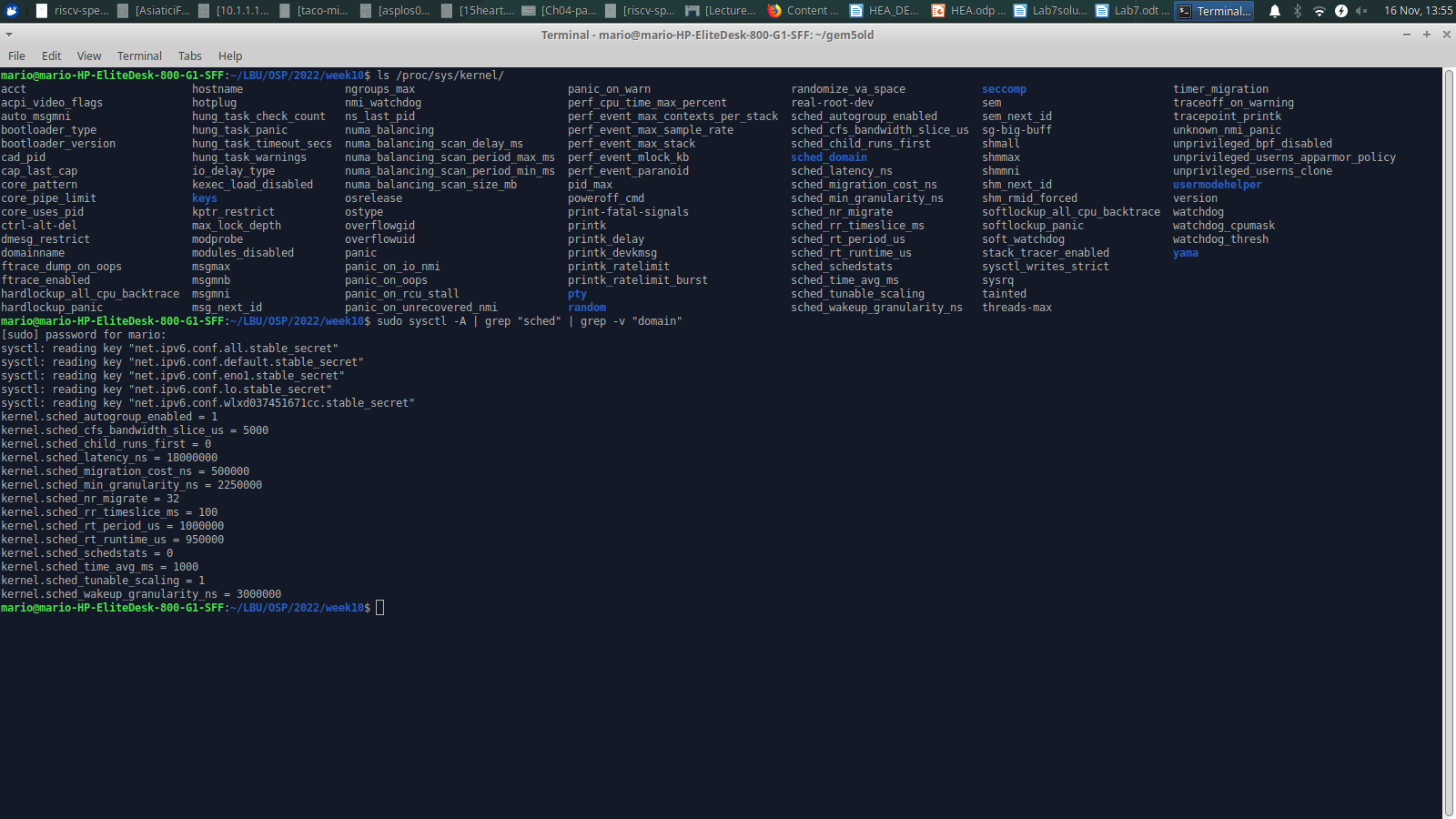
2c. Consider the screenshot below.



Looking at the screenshot above, what is the scheduling policy used? (5 marks)

Answer: The scheduling policy used is Round Robin scheduling.

2d. Assume the following screenshot is valid.

Consider you have one CPU and assume the following set of processes (table given below) are about to run, with the length of the CPU-burst time given in milliseconds (ms):

|  |  |  |
| --- | --- | --- |
| Process | Arrival time (ms) | Burst Time (ms) |
| P1 | 0 | 120 |
| P2 | 0 | 80 |
| P3 | 120 | 60 |
| P4 | 140 | 40 |
| P5 | 160 | 40 |

Using the policy you found in item 2c, draw one Gantt chart (draw it as a table with one line and multiple columns) illustrating the execution of the above processes. (10 marks)

Answer:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 | P1 |

0 100 180 240 280 320 340

Question 3

Consider the program below:

// Source code modified from: http://www.informit.com

#include <fcntl.h>

#include <signal.h>

#include <stdio.h>

#include <string.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int size;

char\* array;

int main ()

{

int fd;

struct sigaction sa;

size = getpagesize ();

fd = open ("file.txt", O\_RDWR);

array = mmap (NULL, 10 \* size, PROT\_NONE, MAP\_SHARED, fd, 0);

mprotect (array, 8 \* size, PROT\_READ);

printf("array[30000] = %c\n", array[30000]) ;

printf("writing to array: press enter");

getchar();

array[30000] = 'x';

close (fd);

printf ("all done\n");

munmap (array, 10\*size);

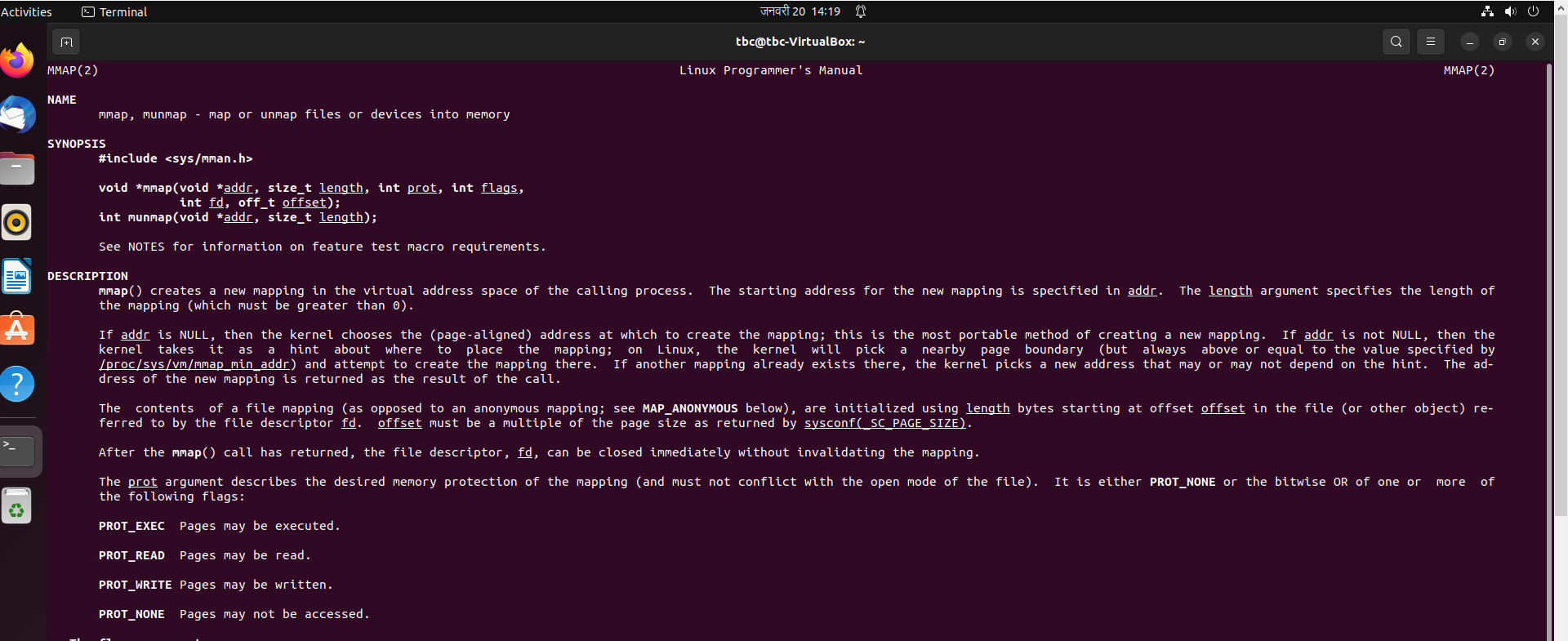
return 0;

}

Question3.c does mimic some OS operations.When question3 process is running, it extends its virtual memory space via the mmap system call.

3a. In terms of number of pages, what is the size of the area created by mmap? (2.5 marks) What is the initial protection of the area set by mmap? (2.5 marks) Hint: do man mmap <enter> to find about its parameters are. (total 5 marks)

Answers: There are 10 pages created. The size of area created by mmap is 10\*4096 = 40,960

The initial protection of the area set by mmap is PROT\_NONE i.e. pages may not be accessed.

3b. The OS can change protection of areas accordingly. In terms of number of pages, what is the size of the area changed write after mprotect? (2.5 marks) What is the protection of this area right after mprotect is executed? (2.5 marks) Hint: do man mprotect <enter> to find about its parameters are. (total 5 marks)

Answers: There are 8 pages. The size of the area after mprotect is 8\*size i.e. 8\*4096 = 32,768.

The protection of the area after mprotect is PROT\_READ and PROT\_NONE. 8 pages with PROT\_READ and 2 pages with PROT\_NONE.

3c. Modify the sample of table below and set the number of pages with respective protection of the program, right after mprotect. Hint: Use the number of pages you got in item 3a to set the number of pages required below. (10 marks)

Answer:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Page 0:  PROT\_READ | Page 1:  PROT\_READ | Page 2:  PROT\_READ | Page 3:  PROT\_READ | Page 4:  PROT\_READ | Page 5:  PROT\_READ | Page 6:  PROT\_READ | Page 7:  PROT\_READ | Page 8:  PROT\_NONE | Page 9:  PROT\_NONE |

3d. Explain what is expected to happen after you press enter at the getchar(), highlighted below below. (2.5 marks). Explain what has to be done to have the program executing as it is until the end **keeping mmap and mprotect parameters as they are**. (2.5 marks)

Note: You don’t have to either compile or run the program to explain it, but if you prefer, you can do so; in the latter make **sure** to have downloaded **question3.c** and **file.txt** files. (total 5 marks).

printf("writing to array: press enter");

getchar();

array[30000] = 'x';

Answer: We will get segmentation fault as we don’t have permission to write.