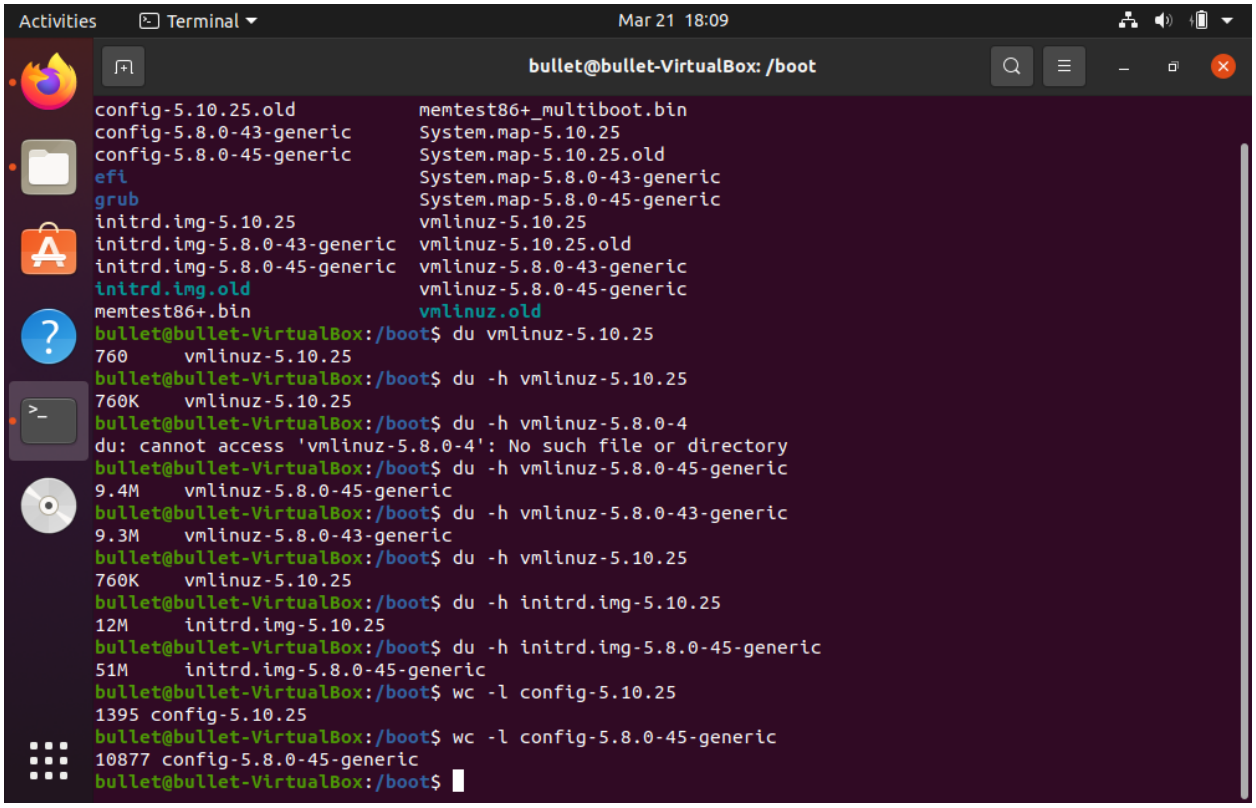
By:
Saatvik Chugh b18084
Manav Mehta b18175

1.

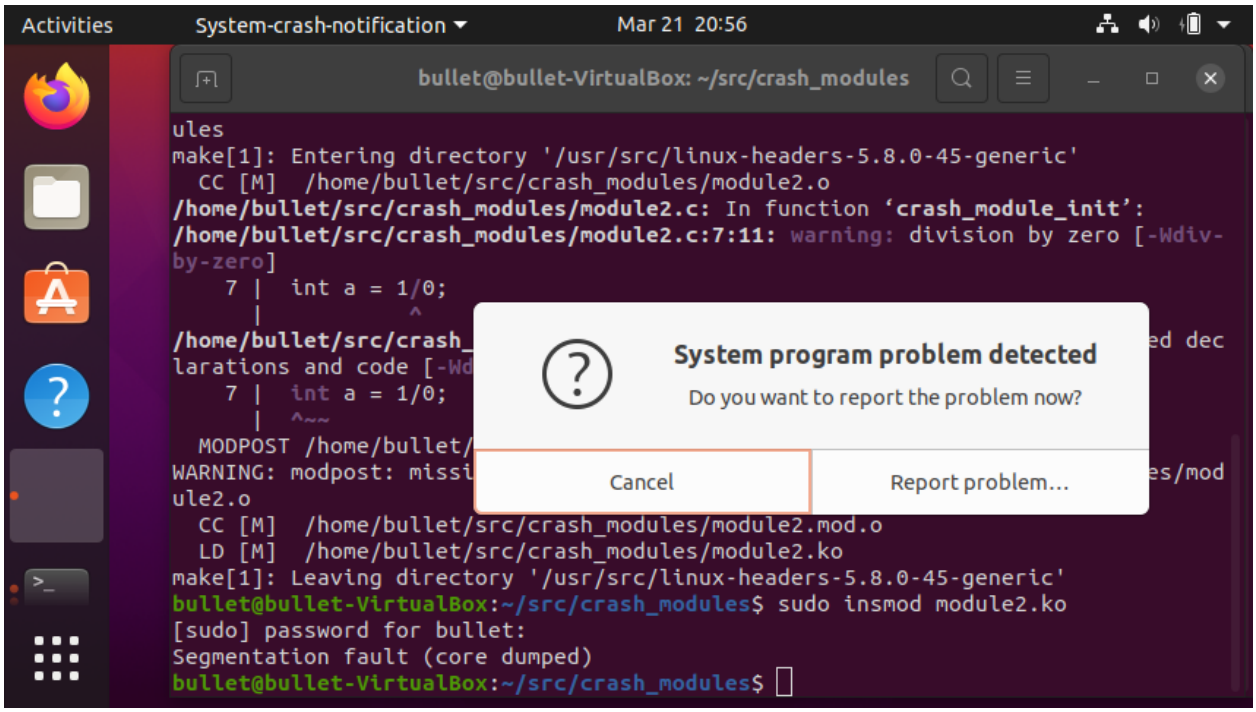
Linux kernel version - 5.10.25

Flags marked:
64 bit kernel
Enable TTY (and all accompanying flags checked)
Enable loadable module support
initramfs/initrd support

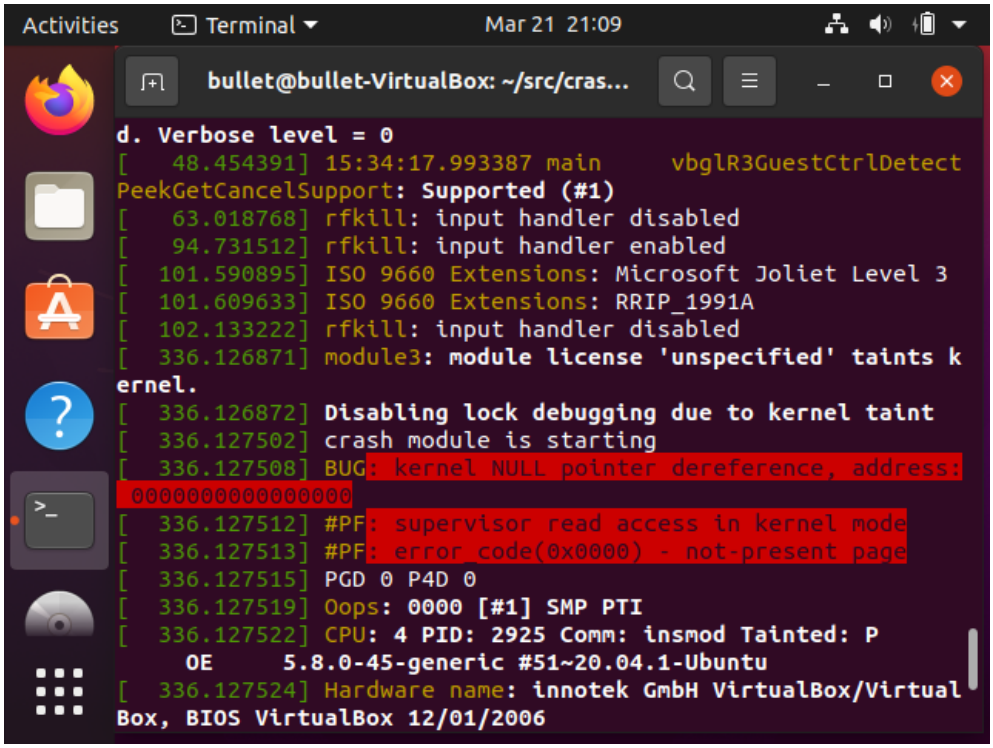


The size is reduced to 760 K
Modules:
Module-1 : Panic function causes the system to freeze.

Module-2: Division by zero does not cause the system to freeze rather gives segmentation fault to the console.



Module-3: Dereferencing a null pointer similarly gives an entry in the kernel buffer.



2.

2. Round Robin Scheduling
About the algorithm:
The array pe_burst is pending burst times of the processes.
Here the algorithm runs over the array elements one by one - till the whole array is nullified - and seeks the processes
which have arrived at time 't' and are still pending. If the pending burst time is more than time quantum then they are partially executed.
Else if they are equal to or less than time quantum, then that process is finished and time is increased as required and completion time is calculated.

Number of context switches are also calculated as a bonus.

Other parameters are calculated as follows
* Turnaround Time = Completion - Arrival times
* Waiting Time = Turnaround - Burst times
* Response Time = Start - Arrival times

The averages are calculated simply using the STL accumulate function

Round robin is starvation proof as all of the processes are getting their fair share of CPU.

For input:
n: 6
arrival: 5 4 3 1 2 6
burst: 5 6 7 9 2 3

When quantum is increased, lesser number of time is a process scheduled

Process #		Start	Complete	Turnaround	Waiting	Response
1	5	25	20	15	0	
2	6	29	25	19	2	
3	3	30	27	20	0	
4	2	31	30	21	1	
5	3	3	1	-1	1	
6	9	20	14	11	3	

ATT: 19.5
AWT: 14.1667
ART: 1.16667
Context Switches: 32

Process #		Start	Complete	Turnaround	Waiting	Response
1	8	26	21	16	3	
2	10	28	24	18	6	
3	12	33	30	23	9	
4	2	34	33	24	1	
5	4	6	4	2	2	
6	6	17	11	8	0	

ATT: 20.5
AWT: 15.1667
ART: 3.5
Context Switches: 18

Process #		Start	Complete	Turnaround	Waiting	Response
1	10	24	19	14	5	
2	13	27	23	17	9	
3	16	34	31	24	13	
4	2	33	32	23	1	
5	5	7	5	3	3	
6	7	10	4	1	1	

ATT: 19
AWT: 13.6667
ART: 5.33333

Context Switches: 12

Process #	Start	Complete	Turnaround	Waiting	Response
1	11	28	23	18	6
2	15	30	26	20	11
3	19	33	30	23	16
4	2	34	33	24	1
5	6	8	6	4	4
6	8	11	5	2	2

ATT: 20.5
AWT: 15.1667
ART: 6.66667
Context Switches: 11

Process #	Start	Complete	Turnaround	Waiting	Response
1	12	17	12	7	7
2	17	32	28	22	13
3	22	34	31	24	19
4	2	31	30	21	1
5	7	9	7	5	5
6	9	12	6	3	3

ATT: 19
AWT: 13.6667
ART: 8
Context Switches: 9

Process #	Start	Complete	Turnaround	Waiting	Response
1	13	18	13	8	8
2	18	24	20	14	14
3	24	34	31	24	21
4	2	33	32	23	1
5	8	10	8	6	6
6	10	13	7	4	4

ATT: 18.5
AWT: 13.1667
ART: 9
Context Switches: 8

Process #	Start	Complete	Turnaround	Waiting	Response
1	14	19	14	9	9
2	19	25	21	15	15
3	25	32	29	22	22
4	2	34	33	24	1
5	9	11	9	7	7

6	11	14	8	5	5

ATT: 19
AWT: 13.6667
ART: 9.83333
Context Switches: 7

Process #	Start	Complete	Turnaround	Waiting	Response
1	15	20	15	10	10
2	20	26	22	16	16
3	26	33	30	23	23
4	2	34	33	24	1
5	10	12	10	8	8
6	12	15	9	6	6

ATT: 19.8333
AWT: 14.5
ART: 10.6667
Context Switches: 7

Process #	Start	Complete	Turnaround	Waiting	Response
1	16	21	16	11	11
2	21	27	23	17	17
3	27	34	31	24	24
4	2	11	10	1	1
5	11	13	11	9	9
6	13	16	10	7	7

ATT: 16.8333
AWT: 11.5
ART: 11.5
Context Switches: 6

Process #	Start	Complete	Turnaround	Waiting	Response
1	16	21	16	11	11
2	21	27	23	17	17
3	27	34	31	24	24
4	2	11	10	1	1
5	11	13	11	9	9
6	13	16	10	7	7

ATT: 16.8333
AWT: 11.5
ART: 11.5
Context Switches: 6

---

* To upwards of time quantum (near the max time quantum supplied), ATT decreases significantly, else is similar.
* Same trend for AWT.
* ART increases with increase in time quantum - trivially because more a particular late arriving process has to be queued.
* The number of context switches decrease with increase in time quantum.

3.

Merge sort which is in-place is an $O(n^2)$ algorithm. Hence the merge function for standard merge-sort is also altered to make it $O(n^2)$ and compare the performance with and without the threads.

Note: The number of threads are rounded off to the nearest power of 2.

Find graph below,

Size of array being from [0, 1000] in steps of 10
And for the merge sort with threads, 25 threads are being used.