Assignment 1

Unix-based systems Practice Assignment

REPORT PDF

Department of Systems and Computer Engineering

SYSC 4001

Operating Systems

FALL 2025

Professor Gabriel Wainer

Student 1: Seham Khalifa 101295726

Student 2: Pardis Ehsani 101300400

Submitted: October 6th, 2025

You should execute at least 20 simulation cases (more are preferred), analyze the results, and write a report (1-3 pages long), in pdf format. Submit the report in a report.pdf file.

Test Cases (used trace_3.txt)	Context Restore (ms)	Context Save (ms)	ISR Time (ms)	Total Execution Time (ms)
1	10	10	40	746
2	20	10	40	806
3	30	10	40	866
4	10	20	40	806
5	10	30	40	866
6	10	10	50	806
7	10	10	60	866
8	10	10	70	926
9	10	10	80	986
10	10	10	90	1046
11	10	10	100	1106
12	10	10	150	1406
13	10	10	200	1706
14	20	20	60	986
15	30	30	60	1106
16	30	30	80	1226
17	30	30	90	1286
18	30	30	100	1346
19	30	30	120	1466
20	30	30	140	1586
21	30	30	160	1706
22	30	30	165	1736
23	30	30	170	1766
24	30	30	175	1796
25	30	30	180	1826
26	30	30	190	1886

Change the value of the save/restore context time from 10, to 20, to 30ms. What do you observe?

When changing the store/restore context times during the interrupt mechanism process, it is shown that the longer it takes to restore or to save context, the longer the process will take. For example, if the first test case is being looked at, both the store and restore contexts are 10ms to save and store context into registers during the interrupt mechanism. When the ISR (interrupt service routine) takes 40ms to do its routine, the total time that it takes to complete the instructions from the second trace file 746ms. If this total time is compared with 10ms to store and 20ms to save into registers and the ISR take 40ms also, its

total elapsed time becomes 806ms. This is due to the number of loops the program must do. If storing takes a long time, then the entire interrupt mechanism takes longer. This means that when the time to store/save context int registers, this directly is proportional to the amount of time it takes to follow through the interrupt mechanism. Although these might be negligible in smaller programs, it adds up for programs that require many processes to store and save contexes into registers when dealing with interrupts.

Vary the ISR activity time from between 40 and 200, what happens when the ISR execution takes too long?

Looking at test cases from test case 15 to 26, it is shown that the longe the ISR activity takes to do its service routine, the overall longer the program will take to run. In this case using trace_3.txt to do these test cases, even a 5ms increase to run through the interrupt service routine activity will result on average of a 30ms increase in the over timings. This adds up very quickly over a short amount of time and can be crucial when wanting to run fast programs. To ensure that the program is as efficient as possible, the ISR execution must take as little time as possible.

How does the difference in speed of these steps affect the overall execution time of the process?

The overall spell of execution processes depends on the time it takes to run these processes, meaning, the amount of delay or processing speed or processing time is taken for the interrupt mechanism. As the overall execution of the times for the interrupt mechanisms increased, the time it took to go through the lines in the test cases also increased. Although sometimes too small to notice, when running longer and longer programs, it can get quite frustrating. It is important to get things done as quickly as possible meaning smaller store/save context times and ISR activity times.

- Ask yourselves other interesting questions and try to answer them through simulations. For instance: what happens if we have addresses of 4 bytes instead of 2? What if we have a faster CPU

Currently, each vector is around 2 bytes long (10 chars). Which means if it was 4 bytes long, the vector that is a t position 5 would be now at 10 because the amount that can be written as addresses for them has grown too making more possibilities possible. Everything will now be doubled, and the positions of the vectors will be changed. This would require a hash function to map the vectors to their corresponding index again properly. If the CPU was faster, the processes would also be done quicker.