Operating Systems (INFR09047)

Revision 1

February 2020

2.1
What is the purpose of system calls?

Answer:
They provide a standard interface for the user to access services offered by the OS.

2.5
What is the main advantage of the layered approach to system design?
What are the disadvantages of the layered approach?

Answer:
Advantages
- Easier to adapt/modify/debug/test/extend (flexibility) since its layer is fairly isolated
- Easier to comprehend (if done properly)
- Associated on the above: reduced cost, construction time, etc

Disadvantages
- Disjoint from reality/gap of model from implementation
- Overhead caused by the inter-layer communication
- Unclear/difficult division when two layers are tightly coupled

4.1

Provide three programming examples in which multithreading provides better performance than a single-threaded solution.

Answer:

- Web server handling incoming client requests in a separate thread.
- Some kind of computational kernel or inherently parallel algorithm (e.g. graph traversal with update). Real-world applications such as image processing.
- Program compilation or even various tasks on a graphical IDE (e.g. text processing, dispatching external commands, linting, etc).

4.4

What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

Answer:

Differences:

1. User-level threads are not known by the kernel.
2. User-level threads are part of a process, while kernel-level ones do not need to be associated with a process.

Circumstances:

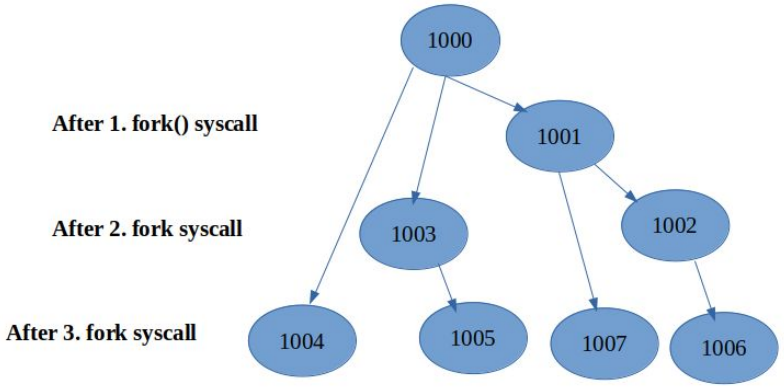
User-level threads are better when management/fast development/portability are required. Kernel-level threads are better when no context switching is required from userspace to the kernel (e.g. kernel drivers, etc).

3.1 Using the program shown in Figure 3.30, explain what the output will be at LINE A.

Answer:
Each process has its own address space. After the fork system call, there will be two identical processes. Each of them has its **value = 5** variable. Only the child process changes the value and it only changes its own **value** variable so it does not affect the parent process. Parent process still has **value = 5**.

3.2 Including the initial parent process, how many processes are created by the program shown in Figure 3.31?

Answer:



3.5 When a process creates a new process using the fork() operation, which of the following states is shared between the parent process and the child process?

- **a. Stack**
- **b. Heap**
- **c. Shared memory segments**

Answer:
Just shared memory segments if shared.

5.3

Answer:

FCFS (First-Come-First-Serve)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P1															
P2															
P3															

Turnaround Time (Time from submission of task to completion)

- **Tp1 = 8 - 0 = 8**
- **Tp2 = 12 - 0.4 = 11.6**
- **Tp3 = 13 - 1 = 12**
- **Avg Turnaround Time = (8+11.6+12)/3 = 31.6/3 = 10.5**

SJF (Shortest-Job-First)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P1															
P2															
P3															

Turnaround Time (Time from submission of task to completion)

- **Tp1 = 8 - 0 = 8**
- **Tp2 = 13 - 0.4 = 12.6**
- **Tp3 = 9 - 1 = 8**
- **Avg Turnaround Time = (8+12.6+8)/3 = 28.6/3 = 9.4**

5.3 (continued)

Future Knowledge Scheduling

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P1															
P2															
P3															

<u>Turnaround Time (Time from submission of task to completion)</u>

- **Tp1 = 14 - 0 = 14**
- **Tp2 = 6 - 0.4 = 5.6**
- **Tp3 = 2 - 1 = 1**
- **Avg Turnaround Time = (14+5.6+1)/3 = 20.6/3 = 6.9**

5.4

Answer:

FCFS

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				
P5																				

TAT(P1) = 2 - 0 = 2 TAT(P2) = 3 - 0 = 3 TAT(P3) = 11 - 0 = 11 TAT(P4) = 15 - 0 = 15 TAT(P5) = 20 - 0 = 20 AVG(TAT) = 51 / 5 = 10.2	WT(P1) = 0 WT(P2) = 2 WT(P3) = 3 WT(P4) = 11 WT(P5) = 15 AVG(WT) = 31 / 5 = 6.2

SJF

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				
P5																				

TAT(P1) = 3 TAT(P2) = 1 TAT(P3) = 20 TAT(P4) = 7 TAT(P5) = 12 AVG(TAT) = 43 / 5 = 8.6	WT(P1) = 1 WT(P2) = 0 WT(P3) = 12 WT(P4) = 3 WT(P5) = 7 AVG(WT) = 23 / 5 = 4.6

5.4 (continued)

NP-Prio

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				
P5																				

TAT(P1) = 15 TAT(P2) = 20 TAT(P3) = 8 TAT(P4) = 19 TAT(P5) = 13 AVG(TAT) = 75 / 5 = 15	WT(P1) = 13 WT(P2) = 19 WT(P3) = 0 WT(P4) = 15 WT(P5) = 8 AVG(WT) = 55 / 5 = 11

RR with q = 2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				
P5																				

TAT(P1) = 2 TAT(P2) = 3 TAT(P3) = 20 TAT(P4) = 13 TAT(P5) = 18 AVG(TAT) = 56 / 5 = 11.2	WT(P1) = 0 WT(P2) = 2 WT(P3) = 3 WT(P4) = 5 WT(P5) = 7 AVG(WT) = 23 / 5 = 4.6