Operating Systems - Review Questions 1

Deadline: Oct. 7, 2022

1. 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? Stack, Heap, or Shared memory segments?

   Answer: Only the shared memory segments are shared between the parent process and the newly forked child process. Copies of the stack and the heap are made for the newly created process.

---

2. Explain what the output will be at `LINE A`.

```c
int value = 15;
int main() {
  pid t pid;
  pid = fork();

  if (pid == 0) { /* child process */
    value += 10;
    return 0;
  } else if (pid > 0) { /* parent process */
    wait(NULL);
    printf("PARENT: value = %d",value); /* LINE A */
    return 0;
  }
}
```

   Answer: The result is still 15, as the child updates its copy of value. When control returns to the parent, its value remains at 15.

---

3. How many processes are created in the following code?

```c
int main() {
  fork();
  fork();
  fork();
}
```

   Answer: Eight processes are created.

---

4. See Section 4.1 of the "Operating Systems Concepts" book. Does the multithreaded web server described in that section exhibit task or data parallelism?

   Answer: Data parallelism. Each thread is performing the same task, but on different data.

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

Answer: User-level threads are unknown by the kernel, whereas the kernel is aware of kernel threads. On systems using either many-to-one or many-to-many model mapping, user threads are scheduled by the thread library, and the kernel schedules kernel threads. Kernel threads need not be associated with a process, whereas every user thread belongs to a process. Kernel threads are generally more expensive to maintain than user threads, as they must be represented with a kernel data structure.

---

6. Describe the actions taken by a kernel to context-switch between kernellevel threads.

Answer: Context switching between kernel threads typically requires saving the value of the CPU registers from the thread being switched out and restoring the CPU registers of the new thread being scheduled.

---

7. Explain the difference between preemptive and nonpreemptive scheduling.

Answer: Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Nonpreemptive scheduling ensures that a process relinquishes control of the CPU only when it nishes with its current CPU burst.

---

8. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

   a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?
   b. What is the average turnaround time for these processes with the SJF scheduling algorithm?
   c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling

Answer: (a) 10.53, (b) 9.53, and (c) 6.86. Remember that turnaround time is the finishing time minus arrival time, so you have to subtract the arrival times to compute the turnaround times. FCFS is 11 if you forget to subtract arrival time.

---

9. Consider the following set of processes, with the length of the CPU burst time given in milliseconds.

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

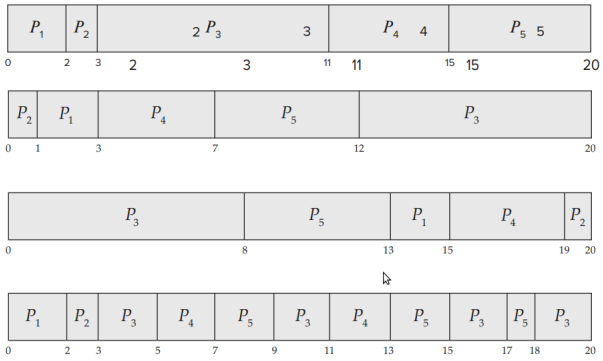
a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

c. What is the waiting time of each process for each of these scheduling algorithms?

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Answere: (a) The four Gantt chart:



(b) Turnaround time:

	FCFS	SJF	Priority	RR
P_1	2	3	15	2
P_2	3	1	20	3
P_3	11	20	8	20
P_4	15	7	19	13
P_5	20	12	13	18

(c) Waiting time (turnaround time minus burst time):

	FCFS	SJF	Priority	RR
P_1	0	1	13	0
P_2	2	0	19	2
P_3	3	12	0	12
P_4	11	3	15	9
P_5	15	7	8	13

(d) SJF has the shortest wait time.