CS471: Operating System Concepts
Fall 2005
(Lecture: W 8:00-10:40 AM)
Homework #1
Points: 20
Due: September 7, 2005
Solution

1. Using the program below, explain what will be output at Line A.

```
#include <sys/types.h>
#include <stdio.h>
#include<unistd.h>
int value = 5;
int main()
{
pid_t pid;
  pid = fork();
if(pid==0)/*child process*/
{
value +=15;
}
 else if(pid>0)/*parents process*/
{
  wait(NULL);
printf("parent: value =%d", value);/*Line A*/
exit(0);
}
}
```

Answer: The output at line A is Parent: value = 5;

Explanation: When a process forks a child process, the child and the parent have different copies of the same variables in the code. So even if the ***value***'s value is changed from 5 to 20 in the child process, it is unchanged in the parent process. So when the parent process executes line A, it prints 5.

2. Which of the following components of program state are shared across threads in a multithreaded process?
   a. Register Values
   b. Heap memory
   c. Global variables
   d. Stack memory

Answer: The Global values and heap memory are **shared** across a multithreaded process. Register values and stack memory are **private** to each thread.

Explanation: See Figure 4.1.

3. Consider the following set of processes, with the length of the CPU burst given in milliseconds:

process	Burst Time	Priority

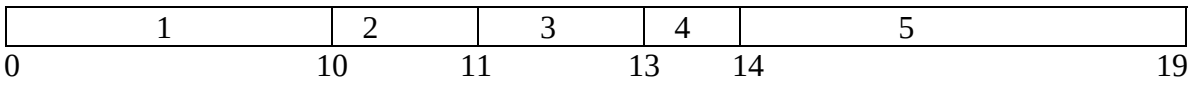
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

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

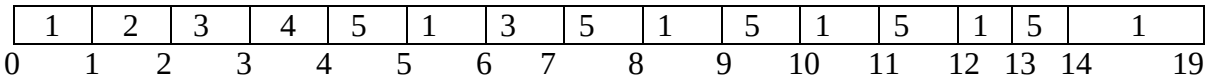
a. Draw four Gantt charts that illustrates the execution of these processes using the following scheduling algorithms: FCFS, SFJ, non-preemptive priority (a smaller priority number implies a higher priority), RR (quantum =1)
b. What is the turn around time of each process for each of the scheduling algorithms in part a?
c. What is the waiting time if each process for each scheduling algorithms in part a?
d. Which of the algorithm in part a minimum average waiting time(over all processes)?

Sol: a. **Gantt charts**

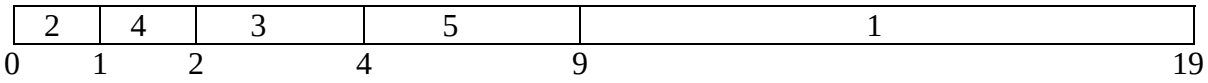
FCFS



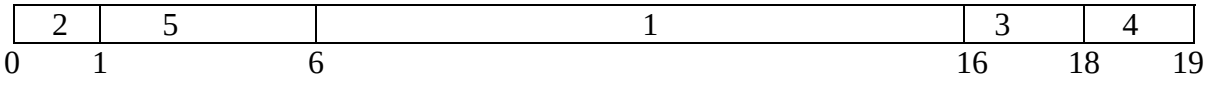
RR



SJF



Priority



b. **Turnaround Time**

	FCFS	RR	SJF	priority

P1	10	19	19	16
P2	11	2	1	1
P3	13	7	4	18
P4	14	4	2	19
P5	19	14	9	6

c. waiting time (turn around time in burst time)

	FCFS	RR	SJF	Priority
P1	0	9	9	6
P2	10	1	1	0
P3	11	5	2	16
P4	13	3	1	4
P5	14	9	4	1

d. Shortest job first .

4. Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation one for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context switching overhead is 0.1 millisecond and that all processes are long running tasks. What is the CPU utilization for a Round-Robin scheduler when:

(a) The time quantum is 1 msec.
(b) The time quantum is 10 msec.

Answer: a. The scheduler incurs a 0.1 msec. of context-switching cost for every context switch. There is 1 unit of CPU time followed by 0.1 units of context switch for both CPU and I/O bound jobs. So CPU utilization = 1.0/(1.0+0.1) = 1/1.1=0.91 or 91%.

Explanation: Let us assume that at time 0 all 11 processes are in ready queue and that CPU-bound job is at the front of the queue. As stated in the problem, we assume that all jobs are long-duration jobs. The CPU scheduling could be as follows:
0-1 C1
1-1.1 – Swap
1.1-2.1 I/O1---Go to I/O queue and start I/O---Finishes at 12.1
2.1-2.2 Swap
2.2-3.2 I/O2 --- Go to I/O queue----Starts at 12.1---finishes at 22.1
3.2-3.3 Swap
…
9.8-9.9 Swap
9.9-10.9 I/O9
10.9-11.0 Swap
11-12.0 I/O10
12-12.1 Swap
12.1-13.1 C1
13.1-13.2 Swap
13.2-14.2 I/O1 --- Go to I/O queue
14.2-14.3 Swap
14.3-15.3 C1
15.3-16.3

b. This is slightly complicated because I/O jobs take only 1 msec of CPU time while CPU uses 10 msec in each turn. Since time quantum is 10 units, after CPU finishes 10 msec of CPU and is context switched there will be exactly one I/O job waiting in the ready queue. However, it would only use 1 msec. After this CPU-bound job gets the CPU again. This way, on the long run, for every one turn of CPU-bound job, I/O bound job gets a turn. So for every 11 units of CPU usage, 0.2 units are overhead. CPU utilization=11/11.2 = 0.982 or 98.2%.