CS342302 Operating Systems Mid-term Examination (I) (11/1/2010) 10:10-11:50am

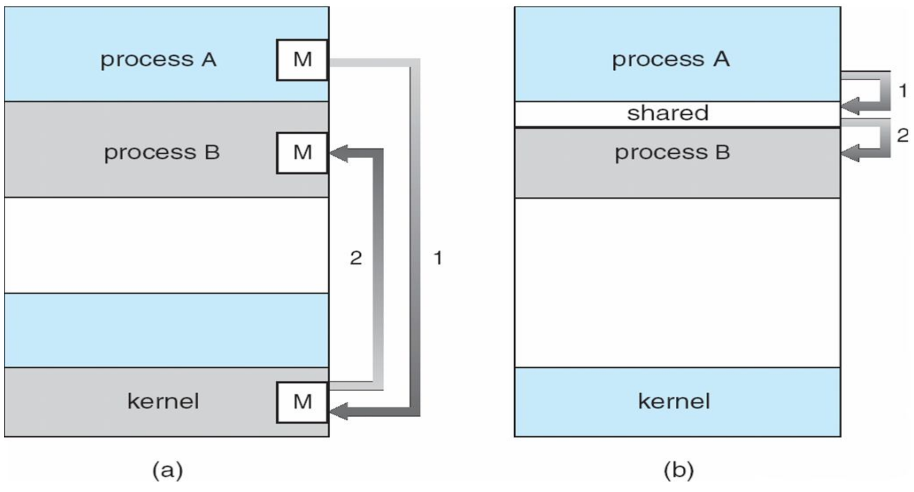
1. (20%) Explain the following terms as details as possible:

(a) DMA (Direct Memory Access),

Afer setting up buffers, pointers, and counters for the I/O, the device controller transfer an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU. Only one interrupt is generated per block, to tell the device driver that the operation is completed, rather than the one interrupt per byte generated for lower-speed devices.

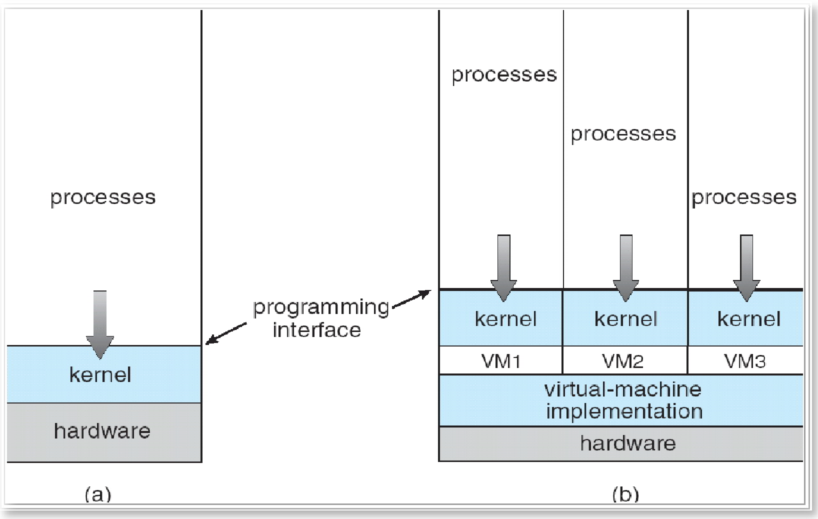
(b) Inter-process communications (IPC)

They are two fundamental models of interprocess communication: (1)shared memory : (2)message passing.



(c) Virtual Machine (VM)

1. A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware
2. A virtual machine provides an interface *identical* to the underlying bare hardware
3. The operating system host creates the illusion that a process has its own processor and (virtual memory)
4. Each guest provided with a (virtual) copy of underlying computer



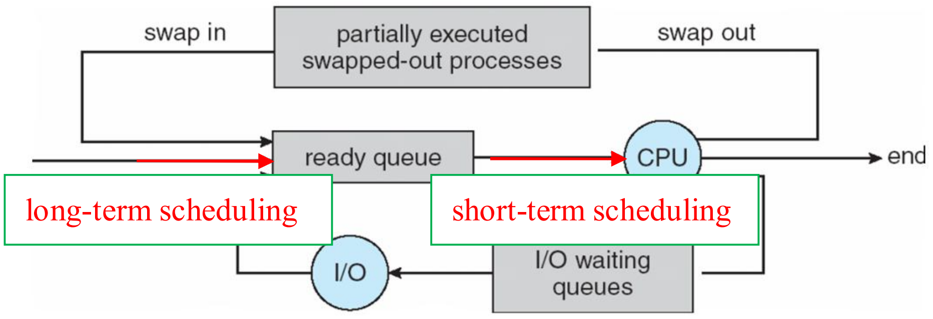
(d) Context switch

1. When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a context switch
2. Context of a process represented in the PCB
3. Context-switch time is overhead; the system does no useful work while switching
4. Time dependent on hardware support

(e) Multi-level feedback queue scheduling.

1. Allow process to move between queues.
2. To separate processed according to the characteristics of their CPU burtst time.
3. It lowers the priority. But when a process waited to long, it will be move to a higher-priority queue.(prevents starvation)
4. Queue 2's process will be executed by CPU only when queue 1 and queue 0 are empty. The rest likewise.

2.(10%) **E**xplain the following queueing-diagram representation of process scheduling. Where are the long-term and short-term scheduling algorithms in this diagram? What are their function goals ?



1. As the shown in the graph
2. **Long-term scheduler** (or job scheduler) – selects which processes should be brought into the ready queue

   Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU

3.(10%) 1.What are the differences between user-level threads and kernel-level threads ?2. Under what circumstances is one type better than the other ? (8%) (2%)

1.

	User-level	Kernel-level
Managed by	Thread library/without kernel support.	Kernel
Disadvantage	Can't be parallel.	Cost more
Platform	Portable	Platform specific
When Blocked	Has to wait	Switchable

2.

The major difference can be seen when using multiprocessor systems, user threads completely managed by the threading library can't be ran in parallel on the different CPUs, although this means they will run fine on uniprocessor systems. Since kernel threads use the kernel scheduler, different kernel threads can run on different CPUs.

If one answered the second question by the first's answer, he can't get the full score, which is 5points here. For one was asked to describe a circumstance, not the difference. And notice that there must be a relationship between user threads and kernel threads.

4.(5%) What resources are used when a thread is created ? How do they differ from those used when a process is created ?

Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, list of open files, and environment variables. Allocating and managing the memory map is typically the most time-consuming activity. Creating either a user or kernel thread involves allocating a small data structure to hold a register set, stack, and priority.

5.(10%) Explain the three common ways of establishing the relationship between user threads and kernel threads:

TA: Students have to specify the differences of these model.

(a) Many-to-one model (3%)

Managed by thread library, which is efficient- but will have problem when one thread is blocked.

(b) One-to-one model (3%)

Good at concurrency and while implemented among multiprocessors. But cost more kernel-level threads.

(c) Many-to-many model (4%)

Suffers from neither of the above shortcomings. Can create as many user threads as necessary, and the corresponding kernel threads can run in parallel.

6.(10%) Explain the following issues we need to consider with multithreaded programs:

(a) The fock() and exec() System calls

If exec() is called immediately after forking, then duplicating all threads is unnecessary, as the program specified in the parameters to exec() will replace the process. In this case, duplicating only the calling thread is appropriate.

However, if the separate process does not call exec() after forking, the separate process should duplicate all threads.

(b) Cancellation

Terminating a thread before it has finished, with 2 general approaches:

Asynchronous cancellation: terminates the target thread immediately

Deferred cancellation: allows the target thread to periodically check if it should be cancelled

(c) Signal handling

Signals are used to notify a process that a particular event has occurred

(d) Thread Pools

Create a number of threads in a pool where they await work, Allows the number of threads in the application(s) to be bound to the size of the pool

7.(10%) The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: the higher the number, the lower the priority. The scheduler recalculates process priorities once per second using the following function:

Priority = (recent CPU usage /2) + base

Where base = 60 and recent CPU usage refers to a value indicating how often a process has used the CPU since priorities were last recalculated.

Assume that recent CPU usage for process P1 is 40, for process P2 is 18, and for process P3 is 10. What will be the new priorities for these three processes when priorities are recalculated ? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process ?

80, 69, 65.   Lower.

8.(10%) Consider the following set of processes, with the length of the CPU burst given in milliseconds.

Process	Burst time	Priority
P1	12	3
P2	1	1
P3	2	3
P4	1	4
P5	6	2

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

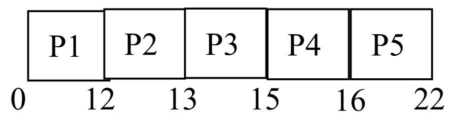
(a) Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum =2)

(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 of these scheduling algorithms ?

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

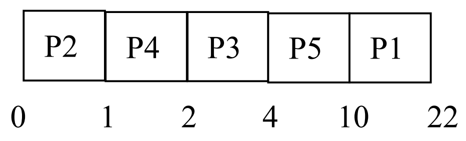
FCFS



turnaround time P1=12   P2=13   P3=15   P4=16   P5=22

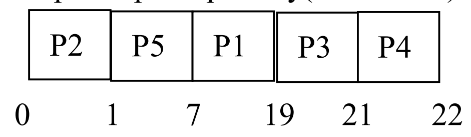
waiting time P1=0   P2=12   P3=13   P4=15   P5=16   total=56

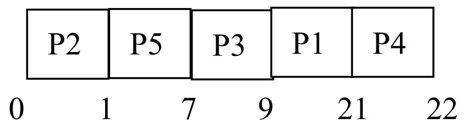
SJF



turnaround time P1=22   P2=1   P3=4   P4=2   P5=10
waiting time P1=10   P2=0   P3=2   P4=1   P5=4   total=17

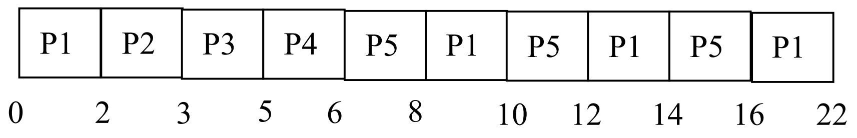
nonpreemptive priority(use FCFS)   (use SJF)





turnaround time P1=19(21)   P2=1   P3=21(9)   P4=22   P5=7
waiting time P1=7(9)   P2=0   P3=19(7)   P4=21   P5=1   total=48(38)

RR



turnaround time P1=22   P2=3   P3=5   P4=6   P5=16
waiting time P1=10   P2=2   P3=3   P4=5   P5=10   total=30

Average waiting time
FCFS: (0+12+13+15+16)/5=56/5=11.2
SJF: (10+ 0+2+1+4)/5=17/5=3.4
priority: (7+0+19+21+1)/5=48/5=9.6
RR: (10+2+3+5+10)/5=30/5=6

minimum average waiting time => SJF

9.(5%) Consider the following preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α; when it is running, its priority changes at a rate β. All processes are given a priority 0 when they enter the ready queue. The parameters α and β can be set to give many different scheduling algorithms.

(a) What is the algorithm that results from $\beta > \alpha > 0$?

FCFS

(b) What is the algorithm that results from $\beta < \alpha < 0$?

Multilevel feedback queue. **RR(-1)**

10.(10%) Explain the following methods usually used to evaluate algorithms:

(a) Deterministic modeling

Takes a particular predetermined workload and defines the performance of each algorithm for that workload

(b) Queueing models

The computer system is described as a network of servers. Each server has a queue of waiting processes. Knowing arrive rates and service rates, we can compute utilization, average queue length, average wait time

(c) Simulations

Programming a model of computer system, simulate the algorithms and show the performance

(d) Implementation

To code the algorithm, run it in the OS and see how it works