北京邮电大学 2019——2020 学年第一学期

《操作系统》期中考试试题

1. FILL IN BLANKS (10 points, 须用英文应答，中文答对得一半分)

(1) The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.

(2) Modern operating system is driven by interrupt .

(3) If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

(4) Considering OS interfaces, an application program can utilize system calls to acquire services provided by the OS.

(5) Programs loaded into and executing in memory refers to __process_ . It needs certain resources, including CPU, memory, files, and I/O devices to accomplish its task.

(6) Three general methods used to pass parameters to the operating system are registers, memory block, and stack.

(7) In When CPU executes the instructions of operating systems, it is said that CPU is in kernel / system / supervisor / privileged ____mode.

(8) Two communication methods between processes are shared memory and message passing 。

(9) Operations on semaphores are initialization, wait() , and signal() 。

(10) 3 conditions that a good solution for critical section problems should satisfy are Mutual Exclusion, Progress , and bounded waiting.

2. CHOICE (10 points)

(1) ____D_____ is not included in the context of process?

A.PCB B. code C. kernel stack D. interrupt vector

(2) Among the following migrations, ___D____is impossible?

A. running→ready B. running→waiting

C. ready→running D. waiting→running

(3) When does a process migrate from waiting state to ready state?__C____

A. the process is waiting for an event B. process is selected by scheduler

C. event that the process is waiting for occurs D. time slice is used up

(4) _B _is the interval from the time of submission of a process to the time of completion.

A. Waiting time B. Turnaround time C. Response time D. Throughput

(5) A starvation-free scheduling policy guarantees that no process waits indefinitely for service. Which of the following scheduling policies is starvation free? ___A____

A. Round Robin B. Priority

C. Shortest Job First D. None of the above

(6) In multiprogramming system, in order to guarantee the integrality of shared variable, processes should enter their critical section mutual exclusively. Critical section refers to C________.

A. a buffer B. a data segment

C. a code segment D. synchronous mechanism

(7) A deadlock situation can arise if the four necessary conditions hold simultaneously in a system. Which one of the following is not the necessary conditions? C________

A. mutual exclusion B. hold and wait C. preemption D. circular wait

(8) Which handling procedures of the following situations will not switch into monitor mode? __A_____

A. procedure calls B. system calls C. interrupts D. traps

(9) In operating systems, the semaphore stands for instances of resource, it is a integer variable relevant to a queue, its value can only be changed by operation wait() and signal(). If a semaphore S is initialized to 5, now it's value is 2, how many processes is or are waiting in the queue relevant to S. D________

A. 3 B. 2 C. 1 D. 0

(10) In the following comments on processes and threads, only ______B____ are correct.

① As the result of I/O completion, a process switches from the waiting state to the ready state, and the CPU scheduling may take place.

② In a system with the operating system supporting kernel-level threads, the process is the basic unit for resources allocation, while the thread is the basic unit for CPU scheduling.

③ For several threads created by one process, one thread's stack and registers can be shared by the others.

④ The round robin algorithm will never result in process starvation.

⑤ PCB contains the process state, the program counter, CPU registers, and user data.

A. ①②③ B. ①②④ C. ②③⑤ D. ②④⑤

3. (12 points) In a system with 2 processors, there are 10 concurrent processes sharing a type of resource based on a semaphore *S*, if each time,

(a) only one process is permitted to enter its critical section to use the resource, or

(b) at most 3 processes are allowed to enter their critical sections to use the resource,

then answer the following questions.

(1) in these two cases, what are the initial, maximum, and minimum values for the semaphore *S* respectively?

Case	initial value	maximum value	minimum value
(a)	1	1	-9
(b)	3	3	-7

(2) what are the maximum, and minimum numbers of processes in ready, running, and waiting state?

number of processes	ready	running	waiting
maximum	8	2	10
minimum	0	0	0

4. （10 points） In a computer system with only one processor, one input device and one printer. Processes A and B enter the system sequentially at time 0, and A is scheduled by the CPU scheduler at first. The execution tracks of A and B are as follows:

A: CPU burst lasting 20ms, then I/O burst of 100ms on the input device, and then CPU burst lasting 50ms, exiting.

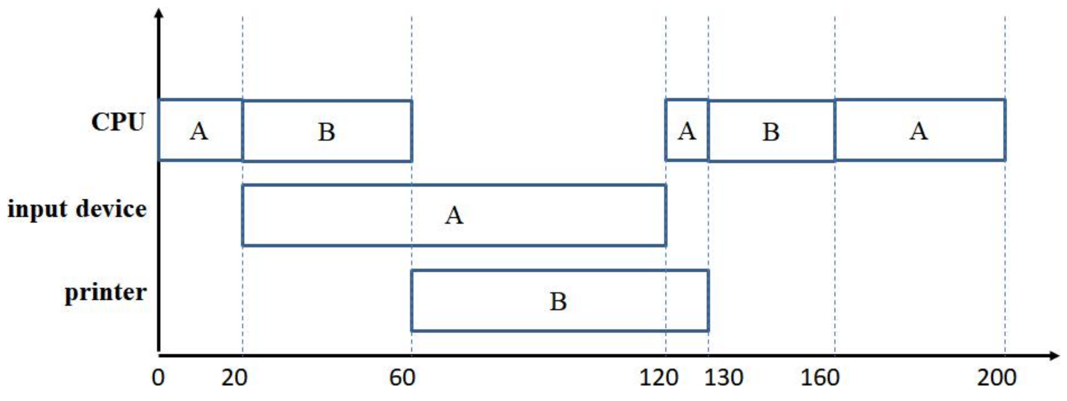
B: CPU burst lasting 40ms, then I/O burst of 70ms on the printer, and then CPU burst lasting 30ms, exiting.

(a) Suppose that preemptive SJF scheduling algorithm (SRTF) is employed, draw the Gantt chart to describe the resource usage of A and B on the CPU, the input device and the printer.

(b) Calculate the waiting time and turnaround time for process A and B respectively.

Answer:

(1)



(2) for process A:	**waiting time:**	**30ms**	**turnaround time:**	**200 ms**
for process B:	**waiting time:**	**20ms**	**turnaround time:**	**160 ms**

5. (20 points) Consider the following set of processes, their arrival time, CPU burst time, and priority numbers are as following.

The length of the CPU burst given in millisenconds, and larger priority numbers imply higher priority.

Process	**Arrival Time**	**Burst Time**	**Priority number**
P1	**0**	**5**	**1**
P2	**1**	**3**	**3**
P3	**2**	**3**	**7**
P4	**4**	**6**	**5**

(1). Suppose that priority-based preemptive scheduling is employed,

(a) Draw a Gantt chart illustrating the execution of these processes;

(b) Calculate the average waiting time.

(c) Calculate the average turnaround time.

(2). Suppose that priority-based non-preemptive scheduling is employed,

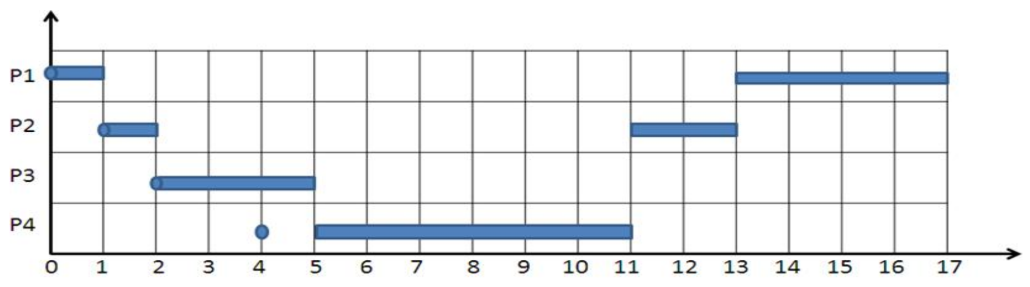
(a) Draw a Gantt chart illustrating the execution of these processes;

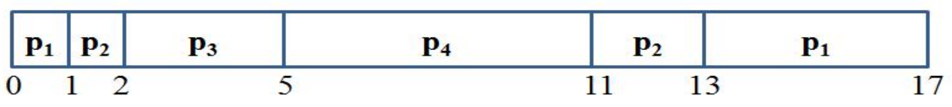
(b) Calculate the average waiting time.

(c) Calculate the average turnaround time.

(1) *preemptive* priority scheduling algorithms.

(a)



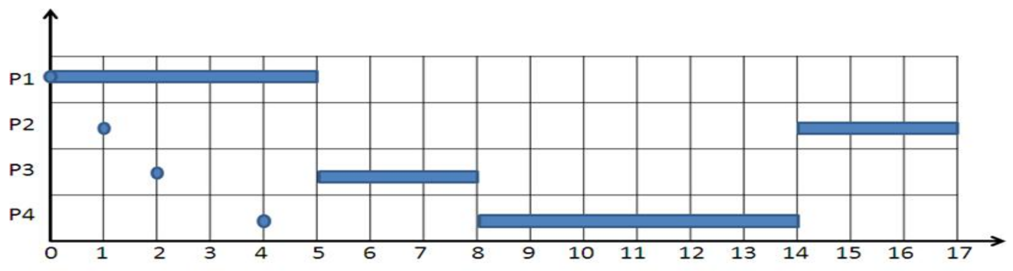


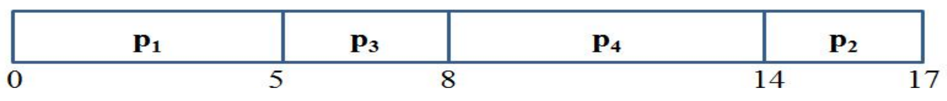
(b) the average waiting time: (12+9+0+1)/4=5.5

(c) the average turnaround time: (17+12+3+7)/4=9.75

(2) *non-preemptive* priority scheduling algorithms:

(a)





(b) the average waiting time: (0+13+3+4)/4=5

(c) the average turnaround time: (5+16+6+10)/4=9.25

6. (20 points) There is a plate that can hold only one apple or can hold three oranges. Father put an apple once a time into the plate; mother put an orange once a time into the plate. If there is already one or two oranges on the plate, the mother can put another orange into the plate.
Son takes an apple from the plate and eats. Daughter takes an orange from the plate once a time and eats.
The processes for the father, mother, son, and daughter are shown as followings.
In order to synchronize these processes, please design semaphores and complete these processes by using wait() and signal() operations on semaphores.
(1) Define semaphores needed and initialize them.

SEMAPHORES

Splate=1;	**Used for mutual exclusion use of the plate.**
Sapple=0;	**Used for synchronization between the process father and son.**
Sorange=0;	**Used for synchronization between the process mother and daughter.**
MUTEX_OC=1;	**Used for mutual exclusion operation on variable orange_count.**
MUTEX_MD=1;	**Used for mutual exclusion operation on plate between mother and daughter.**
Orange_ROOM=3;	**Used to record the rooms left for keeping oranges.**

VARIABLE

orange_count=0; Used to record the number of oranges kept in the plate.

(2) Write appropriate code segmentation for each place marked by number from (1) to (8).

Father:
while(true){
(1) wait(Splate);
Put an apple into the plate;
(2) signal(Sapple);
}
	Son:
	while(true){
	(3) wait(Sapple);
	Take the apple from the plate;
	(4) signal(Splate);
	eats the apple;
	}

Mother:
while(true){
5) wait(orange_ROOM);
wait(MUTEX_OC);
orange_count++;
if (orange_count==1) wait(Splate);
signal(MUTEX_OC);
wait(MUTEX_MD);
Put an orange into the plate;
signal(MUTEX_MD);
(6) signal(Sorange);
}
	Daughter:
	while(true){
	(7) wait(Sorange);
	wait(MUTEX_MD);
	Take an orange from the plate;
	signal(MUTEX_MD);
	(8) wait(MUTEX_OC);
	orange_count--;
	if (orange_count==0) signal(Splate);
	signal(MUTEX_OC);
	signal(orange_ROOM);
	eats the orange;
	}

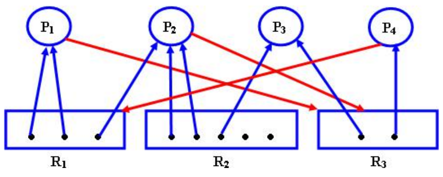
7. (18 points) For the system described in the table below

process	Current allocation			Maximum needs			outstanding requests			Available		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	0	0	2	0	1	0	0	1	0	2	0
P_2	1	2	0	2	5	2	0	0	1			
P_3	0	1	1	1	4	2	0	0	0			
P_4	0	0	1	2	0	1	1	0	0			

(1) How many instances are there for each type of resources?

Total resources in the system are (R1, R2, R3) = (3, 5, 2),

(2) Draw the resource-allocation graph



(3) Is the system in a safe or unsafe state? Specify your judging procedure.

System is in an unsafe state

Need=

	needs		
	R_1	R_2	R_3
P_1	0	0	1
P_2	1	3	2
P_3	1	3	1
P_4	2	0	0

Resources available=(0, 2, 0) can not satisfy the requirements need from any processes, so there exists no safe sequence of processes, so system is in an unsafe state.

(4) Is the system deadlocked? Specify your judging procedure.

System is not deadlocked.

According to deadlock detection algorithm,

Work=(0,2,0) finish[i]=false for i=1,2,3,4

Because request3=(0,0,0)<need3 and request3<work

So, finish[3]=true, work=(0,3,1)

Because request1=(0,0,1)<=need3 and request1<work

So, finish[1]=true, work=(2,3,1)

Because request2=(0,0,1)<need2 and request2<work

So, finish[2]=true, work=(3,5,1)

Because request4=(1,0,0)<need4 and request4<work

So, finish[4]=true, work=(3,5,2)

Finish[i]=true for all i, so the system is not deadlock.