2021智科操作系统期中大题回忆

一、简答（每个5分）

（1）进程通信的两种方式并解释

共享存储和消息传递，解释我编的

（2）临界区问题需要满足的条件并解释

互斥、process、bounded waiting

二、2020三大班卷子里的改编题

题干中说是3处理机系统，有20个进程，设信号量S

(a)同

(b)最多有五个进程允许进入临界区（？这应该初始化s为3还是5啊）

(2)（6 points）There are 10 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 in these two cases, what are the initial, maximum, and minimum values for the semaphore *S* respectively?
Answer:

	initial value	maximum value	minimum value
(a)			
(b)			

三、甘特图的题，有ABC三个进程同时到达，一个CPU，两个I/O，采用FCFS，画出甘特图，计算CPU利用率

类似下面这道

(3)（8 points）A computer system has one *CPU*, one *input processor* and one *printer*. Processes A and B enter the system sequentially, 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 processor*, and then CPU burst lasting 40ms, exiting.
B: CPU burst lasting 40ms, then I/O burst of 70ms on the *printer*, and then CPU burst lasting 50ms, exiting.
Answer the following questions:

2

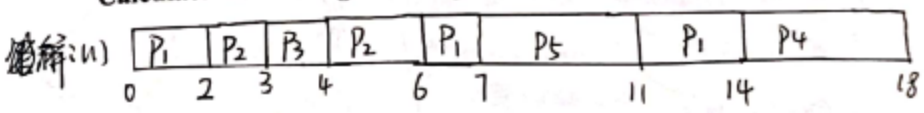
(a) Suppose that FCFS scheduling algorithm is employed, draw the Gantt chart to describe the resource usage of A and B on the *CPU*, the *input processor* and the *printer*.
(b) Calculate the waiting time and turnaround time for process A and B respectively.

四、甘特图，几乎原题

四、（20 points）Given processes as following:

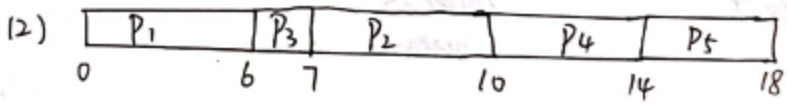
process	Arrival time	priority	CPU burst time
P1	0	3	6
P2	2	1	3
P3	3	0	1
P4	5	4	4
P5	7	2	4

1) Suppose the priority-based preemptive scheduling is employed (assume that low numbers represent high priority).
Draw a Gantt chart illustrating the execution of these processes.
Calculate the average waiting time and the average turnaround time.
2) Suppose the non preemptive SJF scheduling is employed.
Draw a Gantt chart illustrating the execution of these processes.
Calculate the average waiting time and the average turnaround time.
3)Suppose the FCFS scheduling is employed.
Draw a Gantt chart illustrating the execution of these processes.
Calculate the average waiting time and the average turnaround time.



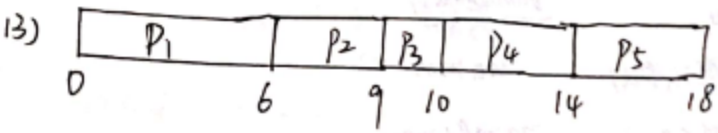
平均等待时间 = (14+6+4+18+11-6-3-4-4-0-2-3-5-7)/5 = 3.6

平均周转时间 = (14+6+4+18+11-0-2-3-5-7)/5 = 7.2



平均等待时间 = (6+7+10+14+18-0-2-3-5-7-18)/5 = 4.0

平均周转时间 = (6+7+10+14+18-0-2-3-5-7)/5 = 7.6



平均等待时间 = (6+9+10+14+18-0-2-3-5-7-18)/5 = 4.4

平均周转时间 = (6+9+10+14+18-0-2-3-5-7)/5 = 8.0

五、多生产者-多消费者问题

把下面这道改了，改成盘子可以装2个苹果或3个橘子，父亲每次放2个苹果，儿子每次取一个苹果；母亲女儿和这个方式一样

5. (20 points) There is a plate that can hold only one apple or 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 one or two orange on the plate, another two or one oranges are allowed to be put 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.
(2) Write appropriate code segmentation for each process.

5. (20 points)

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=1; Used for mutual exclusion operation on variable orange_count.
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.

CODE SECTIONS:

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

母女的代码完全一样的，父亲儿子稍作修改：

```
增设信号量mutex2=1;

计数器int apple_count=0;

father:
while(1){
    wait(splate);
    wait(mutex2);
    apple_count+=2;
    put 2 apples;
    signal(mutex2);
    signal(apple);
}
son:
while(1){
    wait(apple);
    wait(mutex2);
```

```
    if(apple_count>0)apple_count--;
    if(apple_count==0)signal(splate);
    signal(mutex2);
    eat the apple;
}
```

六、死锁检测，几乎和下面题目一样，改了数据

三. (13 points)

Consider a system with four processes P_1 through P_4 and four resource types A, B, C and D. the system snapshot at time T_0 is shown as following table.

	Allocation				Request				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_1	0	0	1	0	2	0	0	1	2	1	0	0
P_2	2	0	0	1	1	0	1	0				
P_3	0	1	2	0	2	1	0	0				
P_4	0	0	0	0	0	0	3	0				

Answer the following questions on the basis of the deadlock-detecting algorithm.

(1) What are the total numbers of the instances for the type A, B, C and D, respectively?

(2) Is the system in a deadlock state, and why?

(3) If P_2's request vector is changed into (2, 1, 0, 1), is there a deadlock in the system? And if the system is deadlocked, indicate the process/processes involved in deadlock.

三. (13 points)

(1) (2 points) (A, B, C, D)=(4, 2, 3, 1)

(2) (1+4 points) 无死锁，可以按照 P_3、P_2、P_1、P_4 的顺序依次执行完

只回答有无死锁，没有给出正确原因（即进程执行顺序）扣分。如果出现安全序列，概念错误，扣 1 分

(3) (2+4 points) 当的 P_2 的请求向量变为(2, 1, 0, 1)时，只能首先执行 P_3，之后空闲可用资源向量变为(2, 2, 2, 0)，无法满足 P_1、P_2、P_4 的资源需求，因此这 3 个进程处于死锁。

没有正确指出哪些进程处于死锁，扣分。

我是按下面这个算法的步骤写的

(4)

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.