CS 471: Operating System Concepts
Fall 2006
Examination I
Points: 150
September 30, 2006
Time: 8:30-11:30 AM
CLOSED BOOK
SOLUTION

Turning in this exam under your name confirms your continued support for the honor code of Old Dominion University and further indicates that you have neither received nor given assistance in completing it.

Name: _____

CS Unix ID: _____@cs.odu.edu

Question #	Points	
	Maximum	Obtained
1	30	
2	30	
3	30	
4	30	
5	30	
Total	150	

Question 1.

(a) List the sequence of activities that occur when a currently running process P1 has executed an I/O instruction. Currently, three processes---P2,P3,P4---are in the ready queue.

Answer:

1. PCB of P1 will be saved.
2. P1 will be placed in the Turnaround state and in the queue at the I/O device.
3. One of the current processes---P2,P3,P4---will be chosen by the scheduler for running.
4. The PCB of that process will be loaded onto the system.
5. The selected process will be placed in run state.
6. When P1's I/O is completed, it will be placed in the ready queue.

(b) What is printed by the following program?

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int value=20;
int main (int argvc, char **argv)
{
    pid_t pid;
    value++;
    pid = fork();

    if (pid == 0) {
        value++;
        printf("I am the child  %d\n", value);
    }
    else {
        value +=5;
        printf("I am the parent  %d\n", value);
    }
    printf("exiting\n");
    exit(0);
}
```

Answer:

I am the child 22
exiting
I am the parent 26
Exiting

(c) Suppose processes P1-P5 have **terminated**, P6 is **running**, P7-P15 are **waiting** at different I/O queues, P16-P20 are in the **ready** queue, and P21-P25 are in the **new** state, determine the degree of multiprogramming of the system.

Answer:

P6 + (P7-P15) + (P16-P20) = 1 + 9 + 5 = 15

(d) What is the role of a medium-term scheduler?

Answer:

Medium-term scheduler can change the mix of the current processes in the system based on the utilization of the CPU and I/O queues. It can increase or decrease degree of multiprogramming also.

Question 2.

(a) Given the following set of processes---with the specified length of the CPU burst, arrival time, and priority---compute **average turnaround time** assuming **preemptive priority** scheduling Show the Gantt chart and other working details.

Process ID	CPU Burst time	Arrival time	Priority
1	13	5	4
2	10	10	3
3	7	15	2
4	12	20	1

Answer:
5---10---15---20---25---32---34---39---47
P1 P2 P3 P4 P4 P3 P2 P1

Turnaround time for P1 = 47-5 = 42
Turnaround time for P2 = 39-10 = 29
Turnaround time for P3 = 34-15 = 19
Turnaround time for P4 = 32-20=12
Average turnaround time = (42+29+19+12)/4 = 25.2

(b) Answer question (a) assuming **shortest-job first scheduling** without preemption.

Answer:
5----18---25---35---47
P1 P3 P2 P4

Turnaround time for P1 = 18-5=13
Turnaround time for P2 = 35-10 = 25
Turnaround time for P3 = 25-15 = 10
Turnaround time for P4 = 47-20=17
Average turnaround time = (13+25+10+17)/4 = 18.75

(c) Consider the exponential average formula used to predict the length of the next CPU burst of a process. The initial estimate of the CPU burst time is $\tau 0 = 150$ milliseconds and $\alpha = 0.7$. The following are the actual CPU burst observed. t_0=60 msec; t1=80 msec; t2=100 msec; t3=20 msec. Compute τ_1, τ_2, τ_3, and τ_4.
(Hint: $\boldsymbol{\tau_{n+1} = \alpha\, t_n + (1-\alpha)\, \tau_n}$)

Answer:
$\tau 0 = 150$;
t0=60; $\tau 1 = 0.7*60+0.3*150 = 97$;
t1=80; $\tau 2 = 0.7*80+0.3*87 = 82.1$;
t2=100; $\tau 3 = 0.7*100+0.3*82.1 = 94.63$;
t3=20; $\tau 4 = 0.7*20+0.3*94.63 = 42.39$;

Question 3.

(a) Given the following solution for the critical section problem (the code is for processes P1 and P2), show why it does not satisfy the mutual exclusion property for processes P1 and P2. Here, **key** is a **local** variable and **unlock** is a **shared** variable (initialized to FALSE). Swap is an atomic operation. (Hint: Indicate one scenario where the mutual exclusion property is violated.)

```
do {
        key=FALSE;
        unlock= TRUE;
        while (key = =FALSE)  Swap(&unlock, &key);

        Critical section

        unlock= FALSE;

        Remainder section
}
```

Answer:

P1: key=FALSE
unlock = TRUE
Swap(&unlock, &key); key=TRUE; unlock = FALSE;
P2: key=FALSE
Unlock = TRUE
Swap(&unlock, &key); key=TRUE; unlock = FALSE;

Both P1 and P2 enter critical section

(b) Suppose TestAndSet instruction is implemented in software instead of as a hardware instruction, what would be the impact on the following solution to the critical section? (Hint: lock is a shared variable initially set to FALSE; consider P1 and P2 as the concurrent processes using TestAndSetLock to enforce critical section on a resource.)

Critical section solution:

```
do {
    while (TestAndSetLock(&lock);  //do nothing
        //critical section

     lock = FALSE;
        //reminder section
  } while (TRUE);
```

Code for TestAndSet:

```
boolean TestAndSet(Boolean *target) {
            boolean rv = *target;
            *target = TRUE;
            return rv;
          }
```

Answer:

P1: TestSndSet(..) rv = FALSE;
P2: TestAndSet(..) rv=FALSE; target=TRUE; return; enter CS
P1: target=TRUE; return; enter CS
Mutual exclusion is violated

(c) Given the following monitor solution for mutually exclusive access of a single instance resource R, answer the following questions.

```
monitor RA
{
    boolean busy;
    condition x;

    void acquire() {
          if (busy)
             x.wait();
          busy = TRUE;
    }

     void release() {
            busy = FALSE;
            x.signal();
     }

     Initialization_code() {
            busy = FALSE;
       }
}
```

Answer each of the following questions independently.

(i) What are the consequences of two processes P1 and P2 simultaneously executing the statement RA.acquire() in their own code? Assume that prior to this, the resource is not acquired by any one.

Answer: One of the processes enters the monitor, the other waits outside the monitor.

(ii) What are the consequences, if any, of a process P3 incorrectly using the sequence:

RA.release(); critical section; RA.acquire();?

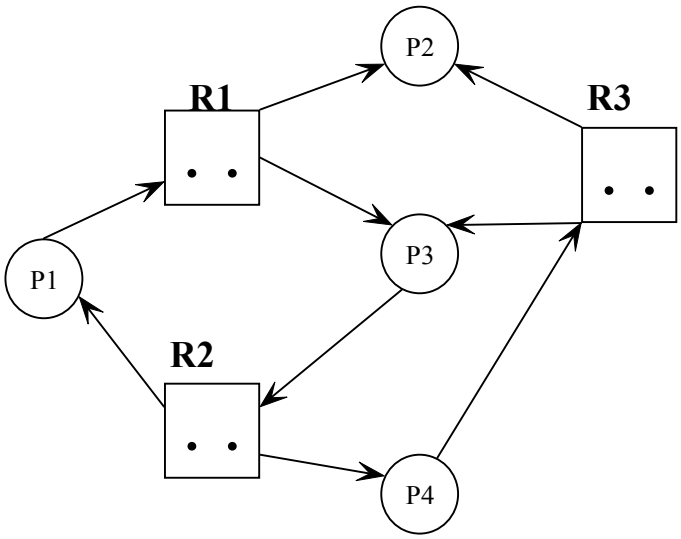
Answer:

RA.release {Critical section} RA.acquire().
When RA.release occurs, if another rprocess is waiting for a resource, it will incorrectly acquire it since a third process has already acquired it and is in its critical section.

When RA.acquire is executed, P3 will incorrectly acquire the resource (if free) or be blocked (if already held by another process). This will lead to either unavailability of the resource for other processes or blocking of this process.

Question 4.

(a) Given the following resource-allocation diagram, (i) Draw the wait-for graph (ii) Determine whether or not there is a deadlock. If yes, justify clearly indicating the reason. If no, explain why there is no deadlock.



Answer:

(i) wait-for-graph:
Dependencies: P1→P3 (R1); P1→ P2 (R1); P3 → P1 (R2); P3 → P4 (R2); P4 → P2 (R2); P4 → P2 (R3)

(ii) No deadlock: Because:
P2 does not depend on any other process;
When P2 terminates, it releases R1 and R3.
P1 can acquire R1 and execute; P4 may acquire R3;
P1 when terminated releases R1 and R2;
P4 acquires R3 and executes and terminates;
P3 may acquire R1 and R3 and terminate.

(b) Given the following snapshot of a system, determine whether or not the system is in a safety state. SHOW YOUR WORK justifying your answer. (Use Banker's algorithm)

Process ID	Maximum need			Current allocation			Available		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	3	2	3	2	2	2	1	1	2
P2	1	3	2	0	1	2			
P3	4	4	4	0	2	2			
P4	1	6	1	1	0	1			

Answer:

Process ID	Additional need			Current allocation			Available		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	1	0	1	2	2	2	1	1	2
P2	1	2	0	0	1	2			
P3	4	2	2	0	2	2			
P4	0	6	0	1	0	1			

P1 finishes (since [1 0 1] >= [1 1 2]); releases all; new Available[3 3 4]
P2 finsihes (since [3 3 4] >= [1 2 0]); release all; new Available [3 4 6])
No other process can finish. So it is in **unsafe** state.

(c) Given the following resource-allocation state of a system at time T0, determine whether or not the system is in deadlock at T0. Justify your answer.

Process ID	**New request**			**Current allocation**			**Currently available**		
	R1	**R2**	**R3**	**R1**	**R2**	**R3**	**R1**	**R2**	**R3**
A	2	3	1	1	1	1	1	0	2
B	0	2	2	0	2	0			
C	3	4	4	4	2	4			
D	1	0	1	1	2	1			

Answer:
D can finish (since [1 0 2] >= [1 0 1]); release; new Available [2 2 3]
B can finish (since [2 2 3] >= 0 2 2]); releases; new Available [2 4 3]
A can finish (since [2 4 3] >= [2 3 1]); release; new Available [4 7 4]
C finishes (since [4 7 4] >= [3 4 4]))
NO DEADLOCK

Question 5.

(a) Given the following page table (page size = 512 bytes) of a process with size 2724 bytes, answer the following:

4
20
9
11
15
27

Page table

(i) How much memory was wasted due to internal fragmentation?
(ii) How much memory is wasted due to external fragmentation?
(iii) What steps are taken when the CPU is presented with logical address 2120?
(iv) What is the physical address (in memory) corresponding to the logical address 1265?

Answer:

(i) The process is allocated 6 pages or 6*512=3072 bytes; it is using 2724 bytes; so internal fragmentation= 3072-2174=348 bytes.

(ii)No memory is wasted due to external fragmentation as it uses paging.

(iii)2120 → (4,72) → page table look up → Physical address = [15,72] → 7752 in main memory
(iv) 1265 → (2,241) → page table lookup → Physical address = [9,241] = 4849 in main memory

(b) A paging hardware uses TLB. The access time for TLB is 30 nanoseconds. The main memory access time is 100 nanoseconds. Currently TLB has the following entries:

Page#	Frame #
2	10
5	25
3	15

Suppose the process in (a) above is currently executing. Answer the following:

(i) What is the memory access time if the logical address is 1433.
1433 → (2,409) → TLB look up Hit → Memory access [10,409]
Access time = 30 + 100 = 130 nanoseconds

(ii) What is the memory access time if the logical address is 1000.
1000 → (1,488) → TLB lookup Miss → Access page table in memory → Access [20,488] in main memory
Access time = 30 + 100 + 100 = 230 nanoseconds

(iii) If on the average 75% of the memory references are found in the TLB, what is the effective memory access time?
Effective access time = 0.75*130+0.25*230 = 155 nanoseconds

(c) Given the following **segment table**, map the logical addresses to physical addresses.

Segment #	Limit	Base
0	1000	2500
1	3000	30000
2	10500	5000
3	500	40000

Logical address <seg#, offset>	Physical address
<3, 122>	40122
<2, 4500>	9500
<1, 2500>	32500
<0, 750>	3250
<4,50>	Illegal reference