CS 471: Operating System Concepts
Fall 2007
Examination I
Points: 150
September 29, 2007
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) What is printed by the following program? (10 points)

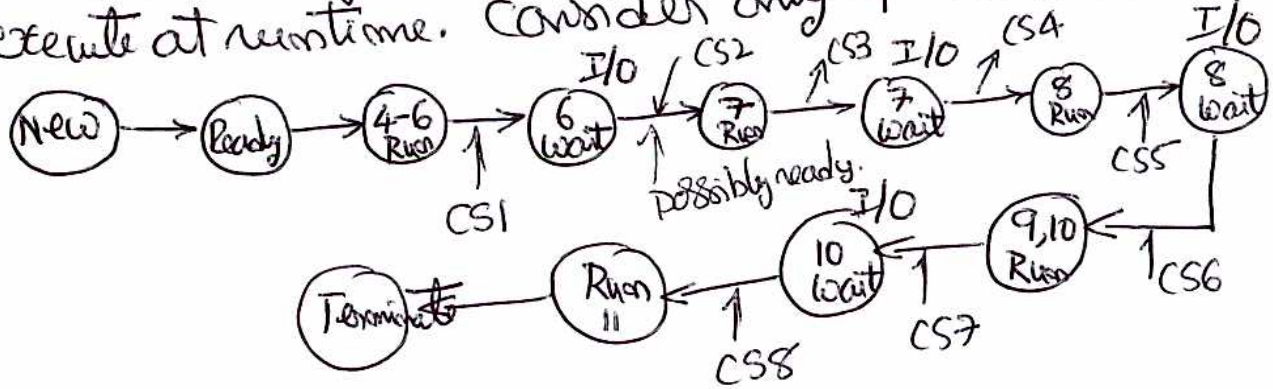
```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int value=20;
int main ()
{
    pid_t pid1, pid2;
    pid1 = fork();
    if (pid1 == 0) {
        value += 10;
        printf("Value1 = %d\n", value);
        exit(0);
    }
    else
    {   pid2=fork();
        if (pid2==0) {
            value += 15;
            printf("Value2 = %d\n", value);
            exit(0);
         }
        wait(NULL); // wait for children to finish
        value +=5;
        printf("Value3 = %d\n", value);
    }
    printf("New value: %d\n", value);
    exit(0);
}
```

Value 1 = 30
Value 2 = 35
Value 3 = 25
New value: 25

(b) Show the state transition diagram for of the following process. Assume that it is the only user process currently in the system. The executable is already loaded into the system. For each state, show the statement it will be executing. (If a particular state occurs more than once, show it multiple times. Consider each I/O statement as a single instruction for CPU and a single I/O operation.) Points 10

```
1  int main () {
2    int i,j;
3    ofstream myfile;
4    i=100;
5    j = 2*i;
6    myfile.open ("example.txt");
7    myfile << "Writing this to a file.\n";
8    myfile.close();
9    j = j*2;
10   cout << j << endl;
11   return 0;
12 }
```

Lines 1-3 may be set at compile time and no need to execute at runtime. Consider only 4-10 lines.



(c) Suppose each C++ instruction takes 1 unit of time, each context switch takes 5 units of time, and each I/O instruction takes 20 units of time, what would be the total time **for the above process** from the time it is loaded to the time it finishes.

#of CPU instructions (counting 4-11); 8 each unit = 8 unit

I/O instructions + Context switch: 4 x 20 + 8 x 5 = 120

128 units

If you include 2-3, it would be 130 units.

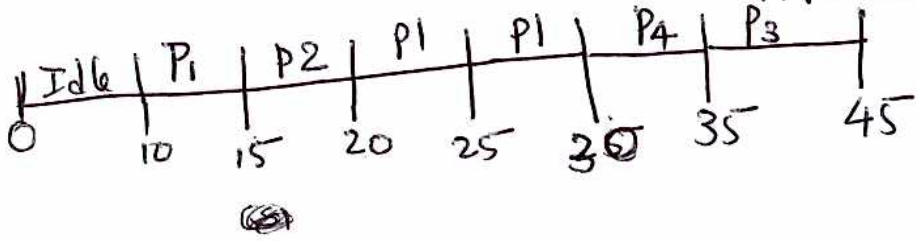
Question 2.

(a) Given the following set of processes---with the specified length of the CPU burst, arrival time, and priority---compute **average waiting time** with **preemptive SJF** scheduling Show the Gantt chart and other working details. (If there is a tie, give priority to the currently executing one.)

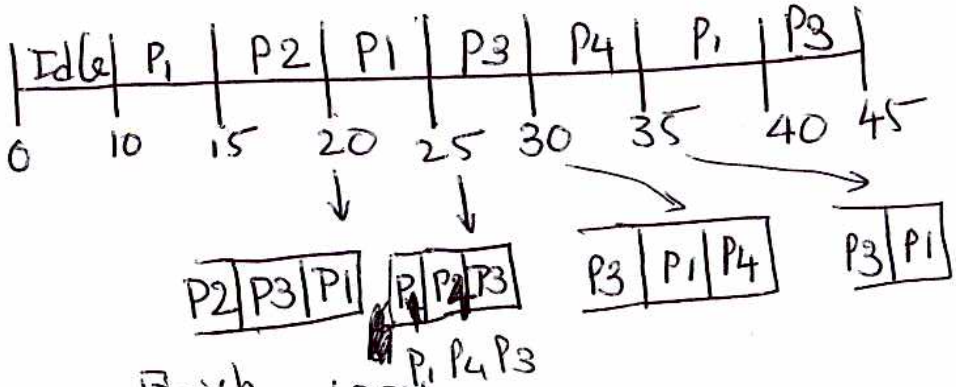
Process ID	CPU Burst time	Arrival time	Priority
1	15	10	4
2	5	15	3
3	10	20	2
4	5	25	1

Bursh	wait
30	5
20	0
45	15
35	5
	25 total

Ave wait time = $\frac{25}{4}$ = 6,25



(b) Answer question (a) assuming **round-robin** with time quantum=5 units.



P1 P4 P3

	Bursh	wait
P1	40	15
P2	20	0
P3	45	15
P4	35	5

Ave. wait time = $\frac{35}{4}$ = 8,75

(c) Suppose each context switch takes 0.1 units of time and you are asked to decide between a RR scheduling (in 2b) and FCFS scheduling for the data in (2a). Compute the percentage overhead due to context switching in both cases. When (under what conditions) should we choose the one with higher percentage overhead?

2a. SJF: #CS: 4 $P_1 \to P_2 \to P_1 \to P_4 \to P_3$

RR: #CS: 6 $P_1 \to P_2 \to P_1 \to P_3 \to P_4 \to P_1 \to P_3$

FCFS: #CS: 3 $P_1 \to P_2 \to P_3 \to P_4$

SJF: overhead 0.4 $\%OH = \frac{0.4}{35.4} \times 100 = 1.13\%$

RR: overhead 0.6 $\%OH = \frac{0.6}{35.6} \times 100 = 1.69\%$

FCFS: overhead 0.3 $\%OH = \frac{0.3}{35.3} \times 100 = 0.85\%$

(2b) RR has overhead, but it's preferred in a time-sharing environment where multiple users' processes need to be given good response time.

Question 3.

(a) Given the following solution for the critical section problem (the code is for processes P1 and P2), answer the following questions. Here, flag and turn are **shared** variable. flag is initialized to FALSE and turn is initialized to 1. (¬is a logical NOT operator similar to !)

```
do {
        flag[i] = TRUE;
        turn = i;
        while ¬(flag[i] && turn == i);

        Critical section

        flag[i] = FALSE;

        Remainder section
}
```

(i) several alternates were suggested.
(a) turn=i → turn=j
(b) while ¬( ) → while ( )
remove ¬
(c) while ¬( ) → while ¬(flag[i] && turn==j);

Each solves this problem, but may suffer from progress.

(i) Give an example scenario to show why this violates mutual exclusion condition.
(ii) What **change** in a **single statement** in the above solution would rectify the problem?

(a) P1

flag[1] = TRUE;
turn = 1;
while(flag[1] && turn == 1);

→ CS

P2

flag[2] = True;
turn = 2
while ¬(flag[2] && turn == 2);

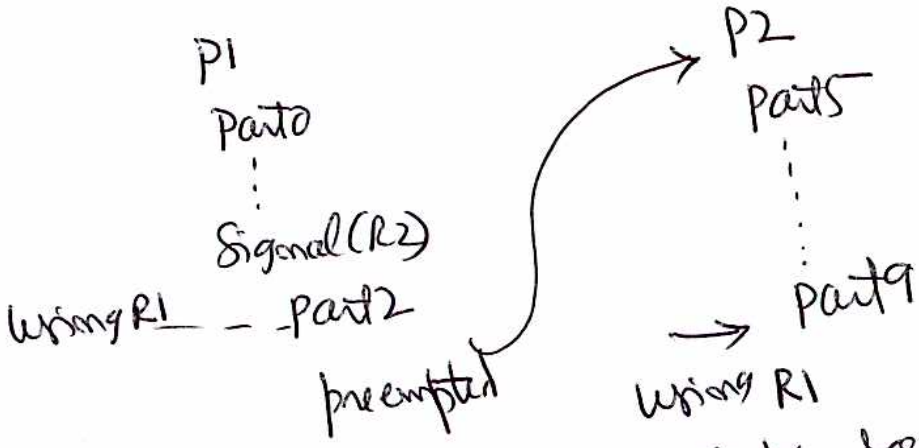
time ↓

Mutual Exclusion violated → critical section

(b) Following is the code of two processes P1 and P2 that are sharing resources R1 and R2. wait and signal are implemented using binary semaphores. It has at least two problems: (i) mutual exclusion violation (ii) starvation (due to no progress). For each problem, show how this is likely to occur. *Part* identifies some computation or use of the resource acquired.

(i)

P1	**P2**
…Part0	*…Part5*
wait(R1)	**wait(R2)**
….. Part1	*….Part6*
signal (R2)	**signal (R1)**
….Part2	*…Part7*
signal (R1)	**signal(R2)**
…….Part3	*…Part8*
wait (R2)	**wait(R1)**
….Part4	*…Part9*



Both have R1 at the same time.

(ii) Several scenarios. One scenario:

P2 Quits after part9

P1 can never get R1 again. Starvation

(c) Given the attached monitor solution to the dining philosophers problem, answer the following question. "Suppose currently all philosophers are in thinking state, and suddenly philosophers 1 and 2 decide to eat and execute methods pickup(1) and pickup(2), respectively. Describe the sequence of actions that take place and the resulting outcome." You need to describe only one possible scenario.

```
monitor dp
{
   enum {THINKING, HUNGRY, EATING}state[5];
   condition self[5];

   void pickup(int i) {
      state[i] = HUNGRY;
      test(i);
      if (state[i] != EATING)
         self[i].wait();
   }

   void putdown(int i) {
      state[i] = THINKING;
      test((i + 4) % 5);
      test((i + 1) % 5);
   }

   void test(int i) {
      if ((state[(i + 4) % 5] != EATING) &&
       (state[i] == HUNGRY) &&
       (state[(i + 1) % 5] != EATING)) {
          state[i] = EATING;
          self[i].signal();
       }
    }

    initialization_code() {
       for (int i = 0; i < 5; i++)
          state[i] = THINKING;
    }
}
```

Figure 6.19 A monitor solution to the dining-philosopher problem.

One Scenario:

pickup(1): state[1] = Hungry;
test(1) → state[1] = Eating
self[1].signal();

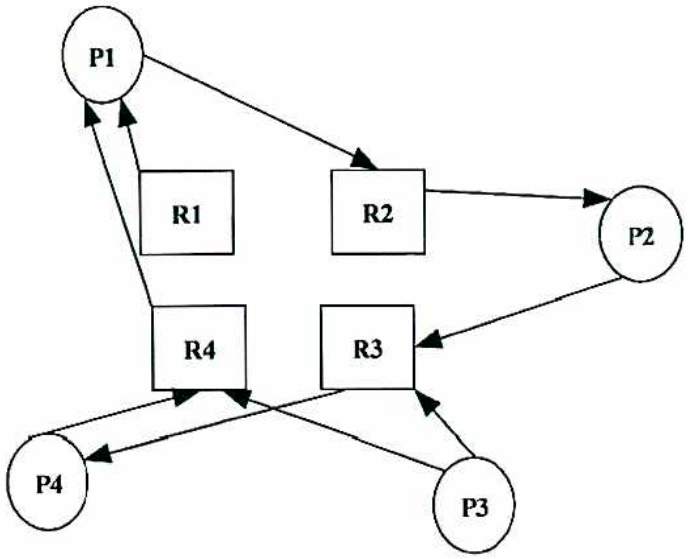
time

pickup(2): state[2] = Hungry;
test(2) → Nothing
self[2].wait;

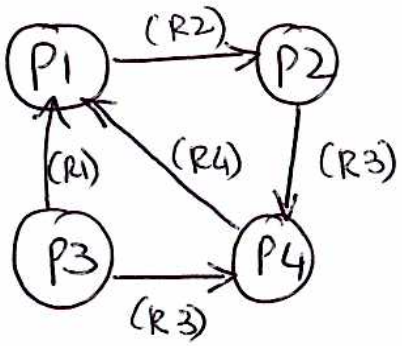
When philosopher 1 executes putdown(1),

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 and suggest one way to resolve it. If no, explain why there is no deadlock. (There is a single instance of each resource.)



(i)



WFG

(ii) There is a cycle in the WFG P1 → P2 → P4 → P1 and each resource is single instance. So there is a deadlock involving P1, P2, P4.

Solution: Kill one of them say P1. Then it will release R1, R4; Then P4 can finish; then P2 & P3.

(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	5	7	5	2	1	0	2	2	0
P2	3	4	3	1	1	1			
P3	3	3	3	0	2	3			
P4	1	4	2	0	2	2			

(Need) Required

R1	R2	R3
3	6	5
2	3	2
3	1	0
1	2	0

completion

Available

2 2 0

P4 → 2 4 2

P2 → 3 5 3

P3 → 3 7 6

P1 → 5 8 6

Safe 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
P1	3	5	3	2	1	0	2	2	0
P2	2	4	1	1	1	1			
P3	3	3	3	0	2	3			
P4	1	2	0	0	2	2			

Possible completion

Available

2 2 0

P4 → 2 4 2

P2 → 3 5 3

P3 → 3 7 6

P1 → 5 8 6

NO deadlock.

Question 5.

(a) Given a frame size of 1024 bytes, and a 10234-byte process, answer the following:
- (i) How many frames of main memory are needed need for this process?
- (ii) How much memory (in bytes) is wasted due to internal fragmentation?

(i) $\lceil \frac{10234}{1024} \rceil = 10$ Frames

(ii) Wasted $= 10240 - 10234 = 6$ bytes

(b) Given the following page table for a process P1 (with page size=1024 bytes), determine
- (i) The physical address corresponding to logical address 4567
- (ii) The logical address corresponding to physical address 11397

(i) 4567 $\rightarrow$ <4,471> $\Rightarrow$ (Map) <9,471> $\rightarrow$ 9687 Phys. Add.

0	10
1	5
2	15
3	25
4	9
5	11

Page table

(ii) 11397 $\rightarrow$ <11,133> $\Rightarrow$ Map <5,133> $\rightarrow$ 5253 Logical add

(c) A paging hardware uses TLB. The access time for TLB is 10 nanoseconds. The main memory access time is 150 nanoseconds. Consider the process P1 with page table in (b) above. Currently, entries for pages 0, 2, and 4 are in the TLB. Answer the following:

- (i) What is the memory access time if the logical address is 2045 $\rightarrow$ <1,1021>

  $10 + 300 = 310$ nanosec.
- (ii) What is the memory access time if the logical address is 2761. $\rightarrow$ <2,713>

  $10 + 150 = 160$ nanosec
- (iii) If on the average 80% of the memory references are found in the TLB, what is the effective memory access time?

  $0.8 \times 160 + 0.2 \times 310 = 190$ nanosec.

(d) Suppose a system supports a page table with up to 16 entries. Frame size is 2 Kbytes.
- (i) What is the maximum size of a process that only has single level page table?

  $16 \times 2K = 32$ Kbytes
- (ii) What is the maximum address space with 2-level hierarchical page table?

  $16 \times 16 \times 2$ Kbytes $= 512$ Kbytes