COMP 252 Principles of Systems Software

Fall Semester 2001

Midterm Examination Solution

Date: November 3, 2001 (Saturday) **Time: 15:00- 16:30**

Name:_____ **Student ID:**_____ **Email:**_____

Instructions:

1. This examination paper consists of 6 pages (including this page) and 5 questions.
2. This is a **CLOSED** book exam!
3. Please write your name, student ID on this page.
4. You have **90 minutes** to complete this exam. Don't spend too much with one problem. Be smart, get to the questions that are relatively easier for you first.
5. This is not an essay contest. Don't waste your time writing irrelevant materials; it **DOES NOT** add any points. Be concise; In most case, only **one** or **two sentences** will do!
6. Please answer all the questions within the space provided on the examination paper. You may use the back of the pages for your rough work.
7. Please read *each question very carefully* and answer the question clearly and to the point. Make sure that your answers are neatly written, readable and legible.
8. Show all the steps you use in deriving your answer, wherever appropriate.
9. For each of the questions assume that the concepts are known to the graders. Concentrate on answering to the point what is asked. Do not define or describe the concepts.

Question	Points	Score
1	20	
2	10	
3	20	
4	30	
5	20	
TOTAL	100	

1. Questions and Answers [20 points]

(a) What are the *two* main objectives of an Operating System (OS)? [6 points]

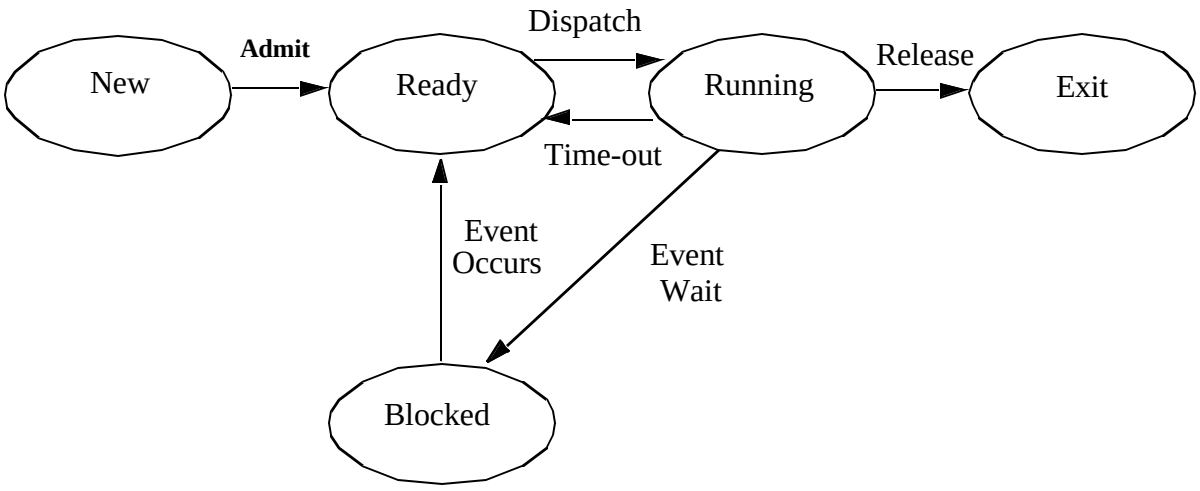
Answers:
1) to provide the convenience for users(3 points), and
2) to manage the system resources (3 points)

(b) The speed mis-match between a fast CPU (processor) and slow I/O devices is one of the major issues that need to be addressed. Please name two techniques commonly used to deal with this problem? What is the main difference between these two techniques? [14 points]

Answer:
Interrupt (4 points) versu Direct Memory Access (DMA) (4 points), the difference is that in DMA during the I/O operation, there is no CPU intervention (6 points).

2. Consider the following ***five state process model***, please fill out the events (in case there are multiple events, please name one) that trigger the transitions between the states. [10 points]

Answer:



Each arrow 2 points

3. Thread [20 points]

(a) Please name *two* items that all threads within the same process share? Why does each thread has its own register set? [10 points]

Answer:
Process control block and user address space. Or this can be answered as the state of the process, and the memory/resources of its process (3 points each)

Each thread needs its register set in order to save its context when not running (4 points)

(b) Consider a process with multiple threads, can we implement them by using multiple processes? what are the main advantages of using the thread approach (name two). [10 points]

Answer:
This certainly can be implemented by multiple processes, with redundant information (data and memory) (4 points) . The key benefits of using thread are: 1) It takes far less time to create a new thread in an existing process than to create a new process; 2) It takes less time to terminate a thread than a process; 3) It takes less time to switch between two threads within the same process than switching between two prccesses; 4) threads enable more efficient communications (3 points each).

4. Concurrency and mutual exclusion [30 points]

(a) Consider the following the solution for infinite-buffer *Producer/Consumer problem*. Is this program correct? Please briefly justify your answer [10 points].

```
---------------------------------------------------------------------------------------------
void producer()                    void consumer()
{                                  {
While (true) {                     While (true) {
    produce();                         P(s);
    P(s);                              P(n);
    append();                          take();
    V(s);                              V(s);
    V(n);                              consume();
    }                                  }
}                                  }
---------------------------------------------------------------------------------------------
```

Answers:
This program is NOT correct (1 points), as it is can potentially lead to a deadlock (3 points).

If consumer() does **P**(s) and **P**(n), if the buffer is empty, it will wait on *semaphone* n. Then producer() does **P**(s); since this was "locked" by consumer(), so a deadlock happens (4); The correct program should be that consumer() does **P**(n) first before does **P**(s) (2 points).

(b) Explain what is achieved with the following mutual-exclusion scheme. The semaphore *mutex* is initialized to 1, and semaphore *synch* is initialized to 0. [10 Points]

Process0	Process1
While (true) {	**While** (true) {
P(*synch***);**	**P(***mutex***);**
P(*mutex*);	
critical section	critical section
V(*mutex*);	**V**(*mutex*);
	V(*synch*);
remainder section	remainder section
}	}

Answer:

1) This gurantee the mutaul exclusion (2 points)

2) Since each time process0 executes, it first does **P**(*synch*) operation; while each time Process1 executes, it does the **V**(*synch*) operation at the end, so this guarantees that at least one Process1 execution has to be prior to a Process0's execution (4 points). This makes sure that the total number of executions of Process0 is always smaller than (or at most equaled to) the total number of Process1 execution (4 points)

(c) Suppose we have two processes, *Process0* and *Process1*, both need to execute a critical section. We would like to enforce a *strict alternation* on the execution of the critical section by the two processes. Please write the code for Process0 and Process1 using *wait P()* and *V()* operations to enforce the proper synchronization. (Hint: you are allowed to use multiple semaphores, please make sure the values are properly initialized) [10 points]

Answer:

Process0	Process1
While (true) {	**While** (true) {
P(*synch1***);**	**P(***synch2***);**
P(*mutex*);	**P(***mutex***);**
critical section	critical section
V(*mutex*);	**V**(*mutex*);
V(*synch*2);	**V**(*synch*1);
remainder section	remainder section
}	}

(7 points)

Initialization *synch1*=*synch2* =1, or *synch1*=1, *synch2*=0, or *synch1*=0, *synch2*=1 (3 points)

5. Deadlocks [10 points]

(a) Recall there are *three* necessary conditions for a deadlock to happen, namely: *mutual exclusion*, *hold-and-wait* and *no preemption*. One way to ensure no deadlock is to break one of the conditions. For example, we can design a OS that requires a process either to acquire all its needed resources before execution, or wait until all its required resources to become available. What are the main disadvantages of this approach (provide two)? [5 points]

Answer:
1) The process might have to wait a long time before it can acquire all resources
2) The process does not need to use all the resources it holds at the same time, thus resulting in a waste and other process(es)can not use them
3) The proecss might not know prior that all the resources it needs before its execution

(any two of the above get 5 points, if only one is correct, 3 points are given)

(b) Consider the following resource allocation, please calculate the **Resource**() vector. Is the state safe? why? [15 points]

	Allocation			**Claim**		
	R1	R2	R3	R1	R2	R3
P0	0	1	0	7	5	3
P1	2	0	0	3	2	2
P2	3	0	2	9	0	2
P3	2	1	1	2	2	2
P4	0	0	2	4	3	3

Available = (3, 3, 2)

Resources = (10, 5, 7)

Answer:
The the state is safe (5 points). As the sequence <P1, P3, P4, P2, P0> satisfies the safety criterion. In another word, this sequence can run to completion (10 points).