# CS 460

Mid Term Exam
(Partial Solutions)

Feb. 2013, Dr. T.L. Yu

Answer all questions. For each of the multiple-choice questions, always choose the **best** answer.

1. ( 20 points )
   Five batch jobs **A** through **E** arrive at a computer center in the order **A** to **E** at almost the same time. They have estimated running times of 6, 4, 1, 3, and 7 minutes. Their ( externally determined ) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead.

   a) Round Robin ( assume quantum = 1 )
   b) Priority scheduling
   c) First-come first-served
   d) Shortest job first

   For a), assume that the system is multitasking, and that each job gets its fair share of the CPU. For b) through d) assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound.

   **Solutions**:

   a) RR with quantum = 1
   $T_{around}$ = ( 19 + 15 + 3 + 12 + 21 ) / 5 = 70/5 = 14 ( minutes )

   b) Priority:
   $T_{around}$ = ( 4 + 11 + 17 + 18 + 21 ) / 5 = 71 / 5 = 14.2 ( minutes )

   c) FCFS:
   $T_{around}$ = ( 6 + 10 + 11 + 14 + 21 ) / 5 = 62 / 5 = 12.4 ( minutes )

   d) SJF:
   $T_{around}$ = ( 1 + 4 + 8 + 14 + 21 ) / 5 = 48 / 5 = 9.6 ( minutes )

2. ( 15 points ) a) In general, a semaphore has two functions, **down()** which decreases the semaphore value and **up()**, which increases the semaphore value. What is special about these functions? Give an alternate name that people commonly used for each of the two functions.

   b) and c) Two threads, **T1()** and **T2()** are accessing the same critical section ( CS ). They have to wait on two semaphores S1, and S2 before they can enter the CS. Consider the following piece of code for T1 and T2:

```
T1:                          T2:
    down ( S1 );                 down ( S2 );
    down ( S2 );                 down ( S1 );
    CS();                        CS();
    up ( S2 );                   up ( S1 );
    up ( S1 );                   up ( S2 );
```

   b) Does the code achieve the purpose of mutual exclusion? Why?

   c) Is deadlock possible for T1 and T2? Why? If your answer is yes, modify the code so that the two threads are deadlock free. If your answer is no, prove your claim.

   **Solutions:**

   a) A counting semaphore is a semaphore with an integer value that can be larger than 1. A semaphore is an integer variable that, apart from initialization, is accessed only through two standard operations: **down()** and **up()**, which differ from usual functions in the way that they are **atomic**. That is, they must be executed indivisibly.
   A semaphore is a tool used for **synchronization** or to achieve mutual exclusion between processes or threads when accessing a critical section.

   Alternate names:

   down() ~ wait(), P()
   up() ~ signal, V(), notify()

   b)Yes, it is because a thread can enter the critical section only if it has acquired the two semaphores ( "keys" ). If one thread

holds a semaphore, the other cannot have it.

c)Yes. Because threads run concurrently, it could happen that T1 has executed **down ( S1 )** and T2 has executed **down ( S2 )** . So now T1 is waiting for T2 to release ( UP ) S2 and T2 is waiting for T1 to release S1 and thus they are in deadlock.
To prevent deadlock, we can simply require both T1 and T2 to 'lock' ( down ) semphores in the same order S1, S2 so that whoever has first locked S1 can proceed to lock S2 and the other thread has to wait on S1. The following is the modified code that is deadlock free:

```
T1:
  down ( S1 );
  down ( S2 );
  CS();
  up ( S2 );
  up ( S1 );
```

```
T2:
  down ( S1 );
  down ( S2 );
  CS();
  up ( S2 );
  up ( S1 );
```

3. (15 points )
Suppose a system has only one type of resource and uses the Banker's Algorithm to avoid deadlock. Initially, the system has only 11 units of resource and three processes, A, B, and C, which are estimated that at most they need the following amount of units to run to the end:

| Process | max. units needed |
|---|---|
| A | 8 |
| B | 5 |
| C | 3 |

At a certain time, the system has granted the following amount of units to the processes.

| Process | Outstanding units |
|---|---|
| A | 4 |
| B | 2 |
| C | 2 |

a) Suppose Process B requests for another 1 unit, should the system grant the unit to B? Why? ( Show your steps clearly. )
b) If instead Process C requests for another unit, should the system the unit to C? Why?

**Solutions:**

a) **Yes**, the system should grant the unit to B. This is because after granting the unit, the number of resource units available in the system is
*available* = 11 - ( 4 + (2 + 1) + 2) = 11 - 9 = 2
The maximum possible requests of A, B, C are 4, 2, 1. The system has enough units to terminate B, which then returns all units; after B termination, *available* becomes 5; the system can then terminate A and B. So the system is in a safe state after granting the 1 unit. Thus it should grant it to B. The following table shows the 'terminating' steps:

| Process | Units Allocated | max. need | may still need | Avail ≥ still need? | terminate order | new avail after terminate |
|---|---|---|---|---|---|---|
| A | 4 | 8 | 4 | Yes( 7 ≥ 4 ?) | 3 | 11 |
| B | 3 | 5 | 2 | Yes ( 2 ≥ 2 ?) | 1 | 5 |
| C | 2 | 3 | 1 | Yes ( 5 ≥ 1 ) | 2 | 7 |

b)
Yes, if 1 unit is granted to C, the system has *avail* = 2. It can terminate all processes as shown below implying that the new state is a **safe** state. Thus the system should grant the 1 unit to C.

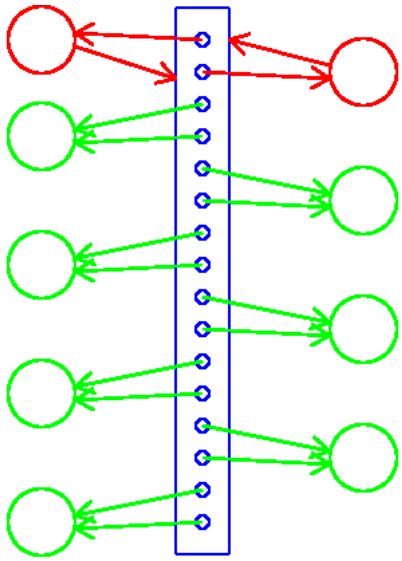
| Process | Units Allocated | max. need | may still need | Avail ≥ still need? | terminate order | new avail after terminate |
|---|---|---|---|---|---|---|
| A | 4 | 8 | 4 | Yes( 7 ≥ 4 ?) | 3 | 11 |
| B | 2 | 5 | 3 | Yes ( 5 ≥ 3 ?) | 2 | 7 |
| C | 3 | 3 | 0 | Yes ( 2 ≥ 0 ) | 1 | 5 |

4. ( 10 points )
A computer has sixteen drives, with 9 processes competing for them. Each process may need two drives. Is the system deadlock free? Give your reasons for the answer with the help of **resource graph diagrams**.

**Solutions:**

Yes, the system is deadlock **free**. This is because in the worst case, all 9 processes request two drives. As we have 16 drives, some of the processes can get two drives to finish their tasks and return the the drives. Then some of the remaining proceses can get two drives, finish and return and so on. So eventually, all processes can finish and return all drives.

The figure on the right illustrates the worst case situation. The processes colored in red are competing for a drive but when any green process has finished using and returned the two drives, they can be allocated to the red processes.



The answers to the following are in bold and green.

5. ( 5 points ) This question relates to Labs that you have done. Which of the following software application allows you to establish high-availability in a Linux Virtual Server cluster? ( Choose the **best** answer. )
   a) ipvsadm
   **b) heartbeat**
   c) ifconfig
   d) ip
   e) arp
6. ( 5 points ) This question relates to the labs you have done. The function **sem_open**() is a POSIX function that creates a named semaphore. What is the POSIX semaphore function that perform a DOWN operation? What is the POSIX semaphore function that performs an UP operation?

   **DOWN ~ sem_wait()**
   **UP ~ sem_post()**

7. ( 5 points ) This question relates to the labs you have done. Which of the following is not a signal function (a function that is used by a process to send a signal to another process or to catch a signal)?
   a) kill
   b) raise
   c) alarm
   **d) ding**
   e) signal

8. ( 5 points ) Which of the following regarding an operating system ( OS ) is **not** true?
   a) It allocates resources.
   b)It provides the run-time environment.
   **c)Its primary goal is the efficiency of operation and secondary goal is the convenience of operation.**
   d)It hides heterogeneous hardware from users.
   e) An OS may be regarded as a virtual machine providing higher level abstraction to application programs.

9. ( 5 points ) Which of the following regarding deadlock and its resource allocation graph is **not** true?

   a) if the graph contains no cycle, no process is in deadlock
   **b)if the graph contains a cycle, some processes are in deadlock**
   c) if the graph can be reduced, no process is in deadlock
   d) if the graph cannot be reduced by all the processes, the irreducible processes constitute the set of deadlock processes in the graph e) none of the above

10. ( 5 points ) Which of the following is **not** true ?

    a) Priority inversion may occur in real-time scheduling.
    b) A thread shares with its peers the data section.
    **c) A process shares with its peers the data section.**
    d)Each processor in symmetric multiprocessing performs self-scheduling.
    e) Shortest Job First scheduling is optimal.

11. ( 5 points ) Which of the following is **true**?

**a) A thread can be in only one semaphore's waiting queue at a time.**
b) Nonpreemptible resources must be hardware.
c) Processes may deadlock as a result of contending for the processor.
d) An unsafe state is a deadlocked state.
e) Two processes, A and B may deadlock if both of them request three records, 1, 2, and 3 in a database in the order 2, 1, 3.

12. ( 5 points ) How many processes does the following piece of code create? Why?

```
int main()
{
  fork();
  fork();
  return 0;
}
```

**The code creates four processes. When executed, a process is created. The first fork() creates a child process. Then both the parent and the child will execute the last fork; each creates a process. Therefore, a total of four processes will be created.**