Solutions to the second midterm

COSC 4330/6310
Summer 2013
First question

Consider the following set of parameters for a System V Release 4 scheduler:

<table>
<thead>
<tr>
<th>ts_quantum</th>
<th>ts_tqexp</th>
<th>ts_slpret</th>
<th>ts_maxwait</th>
<th>ts_lwait</th>
<th>LVL</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>1</td>
<td># 0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>500</td>
<td>2</td>
<td># 1</td>
</tr>
<tr>
<td>200</td>
<td>X</td>
<td>Y</td>
<td>200</td>
<td>Z</td>
<td># 2</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>200</td>
<td>3</td>
<td># 3</td>
</tr>
</tbody>
</table>
Which priority level has the highest priority? (5 points)
### First question

<table>
<thead>
<tr>
<th>ts_quantum</th>
<th>ts_tqexp</th>
<th>ts_slpret</th>
<th>ts_maxwait</th>
<th>ts_lwait</th>
<th>LVL</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>1</td>
<td># 0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>500</td>
<td>2</td>
<td># 1</td>
</tr>
<tr>
<td>200</td>
<td>X</td>
<td>Y</td>
<td>200</td>
<td>Z</td>
<td># 2</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>200</td>
<td>3</td>
<td># 3</td>
</tr>
</tbody>
</table>

> Which priority level has the highest priority? (5 points)

- **Level 3**
What are the most reasonable values for X, Y and Z? (3×5 points)
What are the most reasonable values for X, Y and Z? (3×5 points)

- $X = 1$
- $Y = 3$
- $Z = 3$
Second question

- Consider the function
- `void doublesquare(int *one, int *two) {
  *one = *one * *one;
  *two = *two * *two;
} // doublesquare`
- and assume the following calling sequence:
- `alpha = 2;
  doublesquare (&alpha, &alpha);`
Second question

➢ What will be the value of alpha after the call assuming that the call was

• A conventional procedure call? (5 points)

• A remote procedure call? (5 points)
Second question

- What will be the value of alpha *after the call* assuming that the call was
  - A *conventional procedure call*? (5 points)
    - alpha = 16
  - A *remote procedure call*? (5 points)
Second question

- What will be the value of alpha after the call assuming that the call was
  - A *conventional procedure call*? (5 points)
    - alpha = 16
  - A *remote procedure call*? (5 points)
    - alpha = 4
Third question

- What is the main difference between virtual circuits and datagrams? (5 points)
Third question

- What is the main difference between virtual circuits and datagrams? (5 points)
  - Even reliable datagrams cannot guarantee that some messages will be not duplicated or arrive out of order
Third question

- What is the main difference between virtual circuits and streams? (5 points)
What is the main difference between virtual circuits and streams? (5 points)

- Streams do not preserve message boundaries
Consider the following solution to the mutual exclusion problem:

shared int reserved[2] = {0, 0};

void enter_region(int pid) {
    reserved[pid] = 1;
    while (reserved[1 - pid] == 1);
} // enter_region

void leave_region(int pid) {
    reserved[pid] = 0;
} // leave_region
Fourth question

- What is wrong with it? (5 points)
- When does this problem happen? (5 points)
Fourth question

- **What is wrong with it? (5 points)**
  - A deadlock will occur

- **When does this problem happen? (5 points)**
  - When two processes attempt to enter the critical section in lockstep
Fifth question: True or False

For each of the statements below, indicate in one sentence whether the statement is true or false (2 points), and why (3 points).
Fifth question: True or False

- A *mutex semaphore* can only have two correct values
Fifth question: True or False

- A `mutex semaphore` can only have two correct values
  - **TRUE**, these values are zero and one
Fifth question: True or False

- A client-server pair cannot be **deadlock-free**.
Fifth question: True or False

- A client-server pair cannot be deadlock-free.
  - TRUE, we cannot deny any of the four Haberman's necessary conditions for deadlocks.
Fifth question: True or False

- You cannot pass a list to a remote procedure
Fifth question: True or False

- You cannot pass a list to a remote procedure
  - False, you can send it as an array with instructions on how to reconstruct it
Fifth question: True or False

- Atomic transactions implement the at most once semantics.
Fifth question: True or False

Atomic transactions implement the **at most once semantics.**

- **FALSE**, they implement the **all or nothing semantics**
  - *Each transaction is executed exactly once or not at all*
Sixth Question

A laundromat has sixteen washing machines and eight dryers. Assuming that all customers use one washing machine to wash their clothes then one dryer to dry them, add the necessary semaphore calls to the following program segment:
Sixth Question

- semaphore ___________ = ___; (2 points)
- semaphore ___________ = ___; (2 points)

- customer (int who) {
  __________________________; (4 points)
  wash_clothes( );
  __________________________; (4 points)
  __________________________; (4 points)
  __________________________; (4 points)
  dry_clothes( );
  __________________________; (4 points)
Sixth Question

- semaphore `washers = 16;` (2 points)
- semaphore `dryers = 8;` (2 points)

- `customer (int who) {`
  - `________________________;` (4 points)
  - `wash_clothes( );` (4 points)
  - `________________________;` (4 points)
  - `________________________;` (4 points)
  - `dry_clothes( );` (4 points)
  - `________________________;` (4 points)
Sixth Question

- semaphore **washers** = 16; (2 points)
- semaphore **dryers** = 8; (2 points)

- **customer** (int who) {
  - P(&washers); (4 points)
  - wash_clothes( );
  - __________________________; (4 points)
  - __________________________; (4 points)
  - __________________________; (4 points)
  - dry_clothes( );
  - __________________________; (4 points)
Sixth Question

- semaphore **washers** = 16; (2 points)
- semaphore **dryers** = 8; (2 points)

- **customer (int who) {**
  - `P(&washers);` (4 points)
  - `wash_clothes( );` (4 points)
  - `V(&washers);` (4 points)
  - `______________________;` (4 points)
  - `dry_clothes( );` (4 points)
  - `______________________;` (4 points)
Sixth Question

- semaphore washers = 16; (2 points)
  semaphore dryers = 8; (2 points)
- customer (int who) {
  P(&washers);  (4 points)
  wash_clothes( );
  V(&washers);  (4 points)
  P(&dryers);  (4 points)
  dry_clothes( );
  __________________________;  (4 points)
Sixth Question

- semaphore **washers** = 16; (2 points)
  semaphore **dryers** = 8; (2 points)

- **customer** (int who) {
  P(&washers);   (4 points)
  wash_clothes();
  V(&washers);   (4 points)
  P(&dryers);    (4 points)
  dry_clothes();
  V(&dryers);    (4 points)
Seventh Question

Answer in *one or two sentences* to each of the following questions (2×5 points)

- What is the major advantage of *non-blocking sends*?
- How can scheduling algorithms prevent *starvation*?
Seventh Question

What is the major advantage of non-blocking sends?
Seventh Question

- What is the major advantage of *non-blocking sends*?
  - The sender process does not have to wait the message it has sent is received by the receiver process.
Seventh Question

- How can scheduling algorithms prevent *starvation*?
Seventh Question

- How can scheduling algorithms prevent *starvation*?
  
  - By increasing the priorities of processes that have remained for too long in the ready queue.