SOLUTIONS FOR THE SECOND 3360/6310 QUIZ

Jehan-François Pâris
Summer 2017
First question

- Consider the function
  ```
  void doubledecrement(int *pa, int *pb){
      *pa = (*pa)--;  
      *pb = (*pb)--;  
  }  // doubledecrement
  ```

- and assume the following calling sequence:
  ```
  alpha = 2;    
  doubledecrement (&alpha, &alpha);
  ```
Passing by reference

**Caller:**

... 

... 

alpha

**Procedure will decrement twice variable alpha**

*pa = (*pa)--;
*pb = (*pb)--;
Passing by value and result

**Caller:**

\[ \ldots \]
\[ \ldots \]
\[ \text{alpha} \]

Procedure will decrement once each input values

2, 2

*pa = (*pa)--;

1, 1

*pb = (*pb)--;
First question

What will be the value of \textit{alpha} \textit{after the call} assuming that the call was:

- A \textit{conventional procedure call}?
  - \textit{alpha} = 0
- A \textit{remote procedure call}?
  - \textit{alpha} = 1
Consider the following system V.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>X</td>
<td>1</td>
<td>16000</td>
<td>0</td>
<td>#0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>8000</td>
<td>2</td>
<td>#1</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>4000</td>
<td>3</td>
<td>#2</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>3</td>
<td>#3</td>
</tr>
</tbody>
</table>

Which level corresponds to the highest process priority?
Second question

Consider the following system V.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>X</td>
<td>1</td>
<td>16000</td>
<td>0</td>
<td>#</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>8000</td>
<td>2</td>
<td>#</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>4000</td>
<td>3</td>
<td>#</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>3</td>
<td>#</td>
</tr>
</tbody>
</table>

Which level corresponds to the highest process priority?

- Level 3
Second question

Consider the following system V.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>X</td>
<td>1</td>
<td>16000</td>
<td>0</td>
<td>#</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>8000</td>
<td>2</td>
<td>#</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>4000</td>
<td>3</td>
<td>#</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>3</td>
<td>#</td>
</tr>
</tbody>
</table>

What brings a process in the ts_tqexp column?
Second question

- Consider the following system V.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>X</td>
<td>1</td>
<td>16000</td>
<td>0</td>
<td>#</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>8000</td>
<td>2</td>
<td>#</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>4000</td>
<td>3</td>
<td>#</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>3</td>
<td>#</td>
</tr>
</tbody>
</table>

- What brings a process in the `ts_tqexp` column?

  - The process returned to the ready queue after having exhausted its time slice.
Second question

Consider the following system V.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>X</td>
<td>1</td>
<td>16000</td>
<td>0</td>
<td># 0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>8000</td>
<td>2</td>
<td># 1</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>4000</td>
<td>3</td>
<td># 2</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>3</td>
<td># 3</td>
</tr>
</tbody>
</table>

Which value in the table is **not correct**?
Second question

Consider the following system V.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>X</td>
<td>1</td>
<td>16000</td>
<td>0</td>
<td># 0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>8000</td>
<td>2</td>
<td># 1</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>4000</td>
<td>3</td>
<td># 2</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>3</td>
<td># 3</td>
</tr>
</tbody>
</table>

Which value in the table is not correct?

- The value for `ts_lwait` at level 0
Second question

- Consider the following system V.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>X</td>
<td>1</td>
<td>16000</td>
<td>0</td>
<td>#</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>8000</td>
<td>2</td>
<td>#</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>4000</td>
<td>3</td>
<td>#</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>3</td>
<td>#</td>
</tr>
</tbody>
</table>

- Why?
Second question

- Consider the following system V.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>X</td>
<td>1</td>
<td>16000</td>
<td>0</td>
<td>#0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>2</td>
<td>8000</td>
<td>2</td>
<td>#1</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>4000</td>
<td>3</td>
<td>#2</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>3</td>
<td>#3</td>
</tr>
</tbody>
</table>

- Why?

  - It does not prevent starvation for level 0 processes
Third question

- What is the *main drawback* of *virtual circuits and streams*?
Third question

- What is the **main drawback** of **virtual circuits and streams**?

  - The high setup cost of the connection.
Third question

What is the *sole disadvantage* of *atomic transactions*?
Third question

- What is the *sole disadvantage* of *atomic transactions*?
  - Their high cost.
Third question

- How can we prevent deadlocks by **denying** the **circular wait** condition?
Third question

How can we prevent deadlocks by denying the circular wait condition?

- By requiring them to acquire all their resources in the same linear order.
Third question

- What is the difference between *blocking sends* and *non-blocking sends*?
Third question

- What is the difference between **blocking sends** and **non-blocking sends**?

  - A blocking send waits until its message has been delivered to its recipient.
  - A non-blocking send does not.
Third question

What is the main disadvantage of lottery scheduling?
Third question

- What is the *main disadvantage* of *lottery scheduling*?

  - *It does not guarantee that processes that received a lot of tickets will always execute ahead of the others.*
Third question

- What is the **safest place** to put your `signal()` call in your monitor procedures?
Third question

- What is the **safest place** to put your `signal()` call in your monitor procedures?

  - At the end of these procedures.
Fourth question

- A small parking lot has space for 30 cars and a single entry/exit point that can only accommodate one car at a time.
- Complete the following solution in a way that avoids deadlocks.
Declarations

- semaphore spaces = ______;

- semaphore green = ______;
leave_lot() function

leave_lot()
{
    P(&green);
    get_out();
    V(&green);
    V(&green);
    V(&spaces);
}
    //leave_lot
enter_lot() function

```c
enter_lot()
{
    /* code goes here */
    get_in();
    /* code goes here */
} // enter_lot
```
Declarations

```plaintext
- semaphore spaces = 30;
  semaphore green = 1; // our mutex
```
enter_lot() function

```c
enter_lot()
{
    P(&spaces);

    ______________________________;

    get_in();

    ______________________________;

} // enter_lot
```
enter_lot() function

```c
    enter_lot()
    {
        P(&spaces);
        P(&green);
        get_in();
        V(&green);
    } // enter_lot
```
Fifth question

- What is the major disadvantage of *busy waits*?

- What can we do to eliminate them?

- Are there cases where it is better to keep them?
Answers

- Busy waits waste CPU cycles and cause unnecessary context switches.
- We should use instead kernel-supported solutions that move the waiting processes to the blocked state.
- For short waits in multicore architectures.