SOLUTIONS FOR THE SECOND 3360/6310 QUIZ

Jehan-François Pâris
Summer 2019
First question

- Consider the function
  
  ```c
  void doubledouble(int *pa, int *pb) {
    *pa = (*pa)*2;
    *pb = (*pb)*2;
  } // doubledouble
  ```

- and assume the following calling sequence:
  
  ```c
  alpha = 1;
  doubledouble (&alpha, &alpha);
  ```
Passing by reference

**Caller:**

```
... 
... 
```

**Procedure will decrement twice variable alpha**

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

Caller:

\[
\begin{align*}
\text{...} \\
\text{...} \\
\text{alpha}
\end{align*}
\]

Procedure will decrement once each input values

1, 1

\[
\begin{align*}
*p_{a} &= (*p_{a}) \times 2; \\
*p_{b} &= (*p_{b}) \times 2;
\end{align*}
\]
First question

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

- A *conventional procedure call*?
  - $\text{alpha} = 4$
- A *remote procedure call*?
  - $\text{alpha} = 2$
Second question

- What should a System V Release 4 do
  - When a process returns to the ready queue after having exhausted its CPU time slice?
  - When a process returns to the ready queue after having performed a system call?
Second question

- What should a System V Release 4 do
  - When a process returns to the ready queue after having exhausted its CPU time slice?
    - Decrease its priority
  - When a process returns to the ready queue after having performed a system call?
Second question

- What should a System V Release 4 do
  - When a process returns to the ready queue after having exhausted its CPU time slice?
    - Decrease its priority
  - When a process returns to the ready queue after having performed a system call?
    - Increase its priority
Third question

- Consider a stride scheduler that manages a ready queue with two processes A and B, such that
  - The stride of process A is equal to 4,
  - The stride of process B is equal to 10.
Third question

- When the scheduler starts a new epoch,
  - Which process will the scheduler select \textit{first}?
    - A
      - Original pass values are equal to stride
        \[ P(A) = 4, \ P(B) = 10 \]
      - Pick the process with lower pass value
Third question

When the scheduler starts a new epoch,

- Which process will the scheduler select next?
  - A

- New pass values are
  \[ P(A) = 4 + 4 = 8, \quad P(B) = 10 \]

- Pick the process with lower pass value
Fourth question

- What is the *main purpose* of the Intel `exch register, lockvar` instruction?
Fourth question

- What is the main purpose of the Intel `exch register, lockvar` instruction?
  - To implement spinlocks/buy waits
Fourth question

- What is the main advantage of idempotent remote procedures?
Fourth question

- What is the **main advantage** of **idempotent remote procedures**?
  
  - *We do not have to worry about multiple executions of the procedure by a remote server*
    
    - The outcome would be identical
Fourth question

- Which is typically the least restrictive way to prevent deadlocks?
Fourth question

Which is typically the least restrictive way to prevent deadlocks?

- Denying the circular wait condition
Fourth question

- What is the difference between blocking and non-blocking receive primitives?
Fourth question

What is the difference between **blocking** and **non-blocking** receive primitives?

- A blocking receive wait until it can retrieve a message from the process mailbox
- A non-blocking primitive does not
Fourth question

- What is the main advantage of datagrams?
Fourth question

- What is the main advantage of datagrams?
  - They do not require setting up any kind of connection *before* sending data
  - Best for short interactions
Fourth question

- When is it a good idea to use *busy waits*?
Fourth question

- When is it a good idea to use *busy waits*?
  - For short waits on multicore architectures
Fifth question

What happens when a monitor procedure *signals* a condition X and

- A monitor procedure is waiting for that signal?
- No monitor procedure is waiting for that signal?
Fifth question

What happens when a monitor procedure *signals* a condition X and

- A monitor procedure is waiting for that signal?
  - The calling procedure releases the monitor
  - One of the waiting procedures is allowed to proceed
Fifth question

- What happens when a monitor procedure \textit{signals} a condition X and
  - No monitor procedure is waiting for that signal?
    - \textbf{The signal has no effect}
Sixth question

- What is the sole difference between streams and virtual circuits?
Sixth question

- What is the sole difference between *streams* and *virtual circuits*?

- Streams do *not* preserve message boundaries
Seventh question

- An interstate bus that can carry 40 passengers, has a single door that let one passenger get in or out at any time.
- Add semaphores to the following two functions to ensure that
  - The bus will never be overloaded and
  - Passengers will not collide with each other when embarking or debarking the bus.
Declarations

- semaphore freeSeats = 40;

  semaphore door = _____;
Get_on() function

get_on(){
    __________________________;
    __________________________;
    __________________________;
    go_through_door();
    __________________________;
    __________________________;
    __________________________;
} // get_on
Get_out() function

get_out(){
  _______________________;  _______________________;  go_through_door();  _______________________;  _______________________;  _______________________;  _______________________;  _______________________;  }
  // get_out
Declarations

- semaphore freeSeats = 40;

  semaphore door = 1; // mutex
Get_on() function

```c
get_on()
{
    P(&freeSeats); // must be first
    P(&door);
    go_through_door();
    V(&door);
    _____________________________;
} // get_on
```
Get_out() function

```c
void get_out()
{
    ________________;
    P(&door);
    go_through_door();
    V(&door);
    V(&door);
    V(&freeSeats);
}
```