SOLUTIONS TO THE FIRST 3360/6310 QUIZ

Jehan-François Pâris
Summer 2016
First question

- Why would a process *interrupt itself*?
First question

- Why would a process **interrupt itself**?
  - When it has to do a system call.
First question

- Why is *memory protection* always implemented in hardware?
First question

- Why is memory protection always implemented in hardware?
  - Because any software solution would be too slow.
  - Because it must be done for every memory reference.
First question

- How does a processor switch its mode from *user mode* to *privileged mode*?
First question

- How does a processor switch its mode from *user mode* to *privileged mode*?

  - Through the interrupt mechanism:
    - The interrupt handler will run in privileged mode.
First question

- In which **state** is a process that is performing an I/O?
First question

- In which **state** is a process that is performing an I/O?

  - In the **BLOCKED** state.

- Other good answers are **BLOCKED** and **SLEEPING**.
First question

- Why is `fork()` a very expensive system call?
First question

- Why is `fork()` a very expensive system call?
  - Because it has to allocate—and populate—a new address space.
First question

- Why is it easier to write programs using kernel-supported threads than user-level threads?
First question

Why is it **easier** to write programs using kernel-supported threads than user-level threads?

- Because kernel-supported threads let you use blocking system calls.
Second question

- What is the main advantage of all lightweight processes over regular processes?
Second question

What is the main advantage of *all lightweight processes* over *regular processes*?

- Creating them is much cheaper than creating a regular—"heavyweight"—process.
- Does not require allocating a new address space.
Second question

- What is the main disadvantage of *microkernels*?
Second question

■ What is the main disadvantage of *microkernels*?

□ They are *slower*:

■ Each system request handled by a user-level server requires *two additional context switches*. 
Second question

- What is the main disadvantage of *delayed writes*?
Second question

- What is the main disadvantage of delayed writes?

  - Updates will be lost if a crash happens before the data are written to disk.
Second question

- What is the main disadvantage of a computer system that \textit{lacks memory protection}?
Second question

- What is the main disadvantage of a computer system that lacks memory protection?
  - Malicious programs can tamper with the kernel.
  - Bad programs can crash the system.
Second question

What is the main advantage of modular kernels?
Second question

- What is the main advantage of modular kernels?

  - They allow users/administrators to add functionality to the kernel without having to recompile it.
Second question

- What is the main disadvantage of these *modular kernels*?
Second question

- What is the main disadvantage of these modular kernels?
Second question

- What is the main disadvantage of these modular kernels?

  - Modules are often less reliable than the rest of the kernel.
Third question

How many lines will this program print?

```c
int main(){
    if (fork() == 0) {
        printf("Hello World!\n");
    }
    printf("Goodbye!\n")
}```
Third question

- How many lines will this program print?
  - `int main(){
      if (fork() == 0) {
        printf("Hello World!\n");
      }
      printf("Goodbye!\n")
  }

- **Three lines:**
  - Hello World!
  - Goodbye!
  - Goodbye!
Fourth question

- What should the OS do when there is *not enough free memory*?

- Which processes are the *best candidates* for this action?

- And the *worst candidates*?
Fourth question

- What should the OS do when there is *not enough free memory*?
  - Swap out some processes

- Which processes are the *best candidates* for this action?

- And the *worst candidates*?
Fourth question

- What should the OS do when there is *not enough free memory*?
  - Swap out some processes.

- Which processes are the *best candidates* for this action?
  - Very low priority processes,
  - Processes that have been in the BLOCKED state for a long time.

- And the *worst candidates*?
Fourth question

- What should the OS do when there is not enough free memory?
  - Swap out some processes.

- Which processes are the best candidates for this action?
  - Very low priority processes,
  - Processes that have been in the BLOCKED state for a long time.

- And the worst candidates?
  - Processes in the ready queue.
Fifth question

- What is the meaning of the zero value in _exit(0)?
Fifth question

- What is the meaning of the *zero value* in `exit(0)`?

  - It indicates a normal termination.
  - *Nothing to report*
Sixth question

- What is the default action a Unix process takes when it receives a signal?

- How can it specify a different action?

- Is it always possible to do so?
Sixth question

- What is the *default action* a Unix process takes when it receives a signal?
  - The processes *terminates*.

- How can it specify a *different action*?

- Is it always possible to do so?
Sixth question

- What is the \textit{default action} a Unix process takes when it receives a signal?
  - The processes \textit{terminates}.

- How can it specify a \textit{different action}?
  - It can use \texttt{signal(…)} to \texttt{catch} the signal.

- Is it always possible to do so?
Sixth question

- What is the **default action** a Unix process takes when it receives a signal?
  - The processes **terminates**.

- How can it specify a **different action**?
  - It can use `signal(...)` to **catch** the signal.

- Is it always possible to do so?
  - No, the **SIGKIL** signal cannot be caught.
    - Also known as signal number 9.