COSC 3360/6310
FIRST QUIZ ANSWERS

Jehan-François Pâris
jfparis@uh.edu
Summer 2019
First question

- How can we prevent a process from writing into the address space of another process?
First question

- How can we prevent a process from writing into the address space of another process?

- **Memory protection**
  - It checks every reference issued by a core
First question

- What made UNIX *more portable* than its predecessors?
First question

- What made UNIX *more portable* than its predecessors?

  - *It was written in a high-level language.*
  
  - *Its source code was available.*
First question

- How do processors switch from *user mode* to *privileged mode*?
First question

- How do processors switch from user mode to privileged mode?
  - Through the interrupt mechanism
  - By doing a system call (still full credit)
First question

Why do most operating systems on the market continue to use *monolithic kernels*?
First question

- Why do most operating systems on the market continue to use *monolithic kernels*?
  - Because they are faster.
First question

- Why is `fork()` one of the most expensive system calls?
First question

- Why is \texttt{fork()} one of the most \textit{expensive} system calls?

  - \textbf{Because it has to create a new address space and fill it with an exact copy of the parent address space.}
First question

- What happens when a process does a `wait(...)` on a child process that has already terminated?
First question

- What happens when a process does a `wait(...)` on a child process that has *already terminated*?

  - It keeps going without further delay.
Second question

- *Main advantage of timer interrupts:*
Second question

- **Main advantage of timer interrupts:**
  - They prevent processes from monopolizing a core.
Second question

- *Main advantage of microkernels:*
Second question

- Main advantage of microkernels:
  - They provide a safe mechanism for adding new features to the kernel.
  - They are both safe and extensible
Second question

- **Main advantage of dual-mode CPUs:**
Second question

- **Main advantage of dual-mode CPUs:**
  - They allow us to prevent user processes from directly accessing hard drives and other peripherals.
Second question

- *Main disadvantage of user-level threads:*
Second question

Main disadvantage of user-level threads:

- The kernel will block all threads of a process each time any of them does a blocking system call.
Second question

- *Main disadvantage of non-preemptive scheduling policies:*
Second question

- **Main disadvantage** of non-preemptive scheduling policies:
  - They let CPU-bound processes monopolize cores.
Second question

- Main disadvantage of the round-robin scheduling policy:
Second question

- **Main disadvantage of the round-robin scheduling policy:**
  
  - It performs poorly when the system is heavily loaded.
  - Too many context switches
Third question

What will the following program print out?

```c
main() {
    fork();
    if (fork() == 0) {
        cout << "Hello!\n";
        _exit(0);
    }
    cout << "Goodbye!\n"
}
```
What will happen (II)
What will happen (III)
Third question

- The program will print out **two** “Hello!” and **two** “Goodbye!”
Fourth question

Complete the following sentences:

- A running process will return to the ready state if either
  
  ________________________________

  or

  ________________________________

- It will go to the blocked state if
  
  ________________________________
Fourth question

Complete the following sentences:

- A running process will return to the ready state if either a timer interrupt occurs or a higher priority process arrives.
- It will go to the blocked state if ________________________.
Fourth question

Complete the following sentences:

- A running process will return to the ready state if either a timer interrupt occurs or a higher priority process arrives.
- It will go to the blocked state if it performs a system call.
Fourth question

- Complete the following sentences:
  - We can safely swap out processes that have remained a long time in the **blocked** state.
  - When a process does a blocking system call, it remains in the **blocked** state until the call is completed then goes to the **ready** state.
Fifth question

- Why is it important to have a \textit{signal} that \textit{no process can catch}?
Fifth question

- Why is it important to have a signal that no process can catch?

  We should be able to terminate all processes.
Sixth question

- When should we *suspend processes*?

- Which processes are the *best candidates* for being suspended?
Sixth question

- When should we **suspend processes**?
  - **When we need to make space in main memory.**

- Which processes are the **best candidates** for being suspended?
Sixth question

- When should we *suspend processes*?
  - *When we need to make space in main memory.*

- Which processes are the *best candidates* for being suspended?
  - *Processes that have been in the BLOCKED state for a long time.*