SOLUTIONS FOR THE FIRST 4330 QUIZ

Jehan-François Pâris
Spring 2015
Advantages and disadvantages

- Match each of the following advantages or disadvantages with the *single sentence* that describes it best:

- *Hint: Several of the choices offered are plain wrong*
Advantages and disadvantages

- Main disadvantage of delayed writes
Advantages and disadvantages

- Main disadvantage of delayed writes
  - *Will result in a data loss if the system crashes at the wrong time.*
Advantages and disadvantages

- Main advantage of dual-mode CPUs
Advantages and disadvantages

- Main advantage of dual-mode CPUs

  - Prevent user processes from directly accessing disk drives and other peripherals
Advantages and disadvantages

- Main advantage of timer interrupts
Advantages and disadvantages

- Main advantage of timer interrupts

  - Prevent user processes from monopolizing a CPU core
Advantages and disadvantages

- Main advantage of DMA controllers
Advantages and disadvantages

- Main advantage of DMA controllers
  - *Speed up data transfers between the disk drive and the main memory*
  - On the other hand, they do *not* provide user processes with direct access to disk drives and other peripherals
    - Would be very bad!
Advantages and disadvantages

- Main advantage of lightweight processes
Advantages and disadvantages

- Main advantage of lightweight processes

  - *Are much cheaper to create than conventional processes*
Advantages and disadvantages

- Main disadvantage of microkernels
Advantages and disadvantages

- Main disadvantage of microkernels

- *Introduce additional context switch delays in the processing of requests*
Advantages and disadvantages

- Main advantage of modular kernels
Advantages and disadvantages

- Main advantage of modular kernels
  
  - *Let users add new features to the kernel*
Advantages and disadvantages

- Main disadvantage of modular kernels
Advantages and disadvantages

- Main disadvantage of modular kernels
  - *Increase the risk of system crashes*
Advantages and disadvantages

- Main advantage of time sharing
Advantages and disadvantages

- Main advantage of time sharing
  - *Allows multiple interactive users to share the same computer*
Advantages and disadvantages

- Main advantage of multi-threaded servers
Advantages and disadvantages

- Main advantage of multi-threaded servers
  - Can process multiple client requests in parallel
Questions with short answers

- How can we prevent processes from accessing the *address spaces of other processes*?

- How can we prevent user processes from *tampering with the kernel*?
Questions with short answers

- How can we prevent processes from accessing the *address spaces of other processes*?

- How can we prevent user processes from *tampering with the kernel*?
  
  - *By adding memory protection.*
Questions with short answers

- In a dual-mode CPU, how can the CPU switch from *user mode* to *privileged mode*?
Questions with short answers

- In a dual-mode CPU, how can the CPU switch from *user mode* to *privileged mode*?

  - When it processes an interrupt (as the interrupt will leave the program counter in a safe location INSIDE the kernel.)
Questions with short answers

- What is the main difference between real-time applications with **hard** and **soft deadlines**?
Questions with short answers

- What is the main difference between real-time applications with **hard** and **soft deadlines**?

  - **Missing a hard deadline can have catastrophic consequences while missing a soft deadline is a mere inconvenience**
Questions with short answers

- What would have happened if Unix had remained written in assembly language?
Questions with short answers

- What would have happened if Unix had remained written in assembly language?

  - *It would not have been ported to other architectures and would NOT have had the same impact*
Questions with short answers

- Why is fork() one of the costliest system calls?
Questions with short answers

- Why is fork() one of the costliest system calls?
  - Because it requires making a copy of the address space of the forking process
Questions with short answers

- In which *state* is a process *performing a disk I/O*?
Questions with short answers

- In which state is a process performing a disk I/O?
  - In the WAITING STATE
Questions with short answers

- In which **state** is a process **waiting for a core**?

  - *In the READY STATE*
I/O Redirection

- How would you let a program read its standard input from the file input.txt?

  ```python
  fh = open("data.txt", O_RDONLY);
  ______
  ______
  close(fh);
  ```
I/O Redirection

- How would you let a program read its standard input from the file input.txt?

  - `fh = open("data.txt", O_RDONLY);`
  - `close(0)    // Close stdio`
  - `________`
  - `close(fh);`
I/O Redirection

- How would you let a program read its standard input from the file input.txt?

  ```c
  fh = open("data.txt", O_RDONLY);
  close(0) // Close stdio
  dup(fh)   // Duplicate fh into stdio
  close(fh);
  ```
Parent and child processes

- Add the two system calls that will ensure that the program will print exactly once Hello World! and Goodbye! *in that order*. (2×5 points)
Parent and child processes

int main()
{
    if (fork() == 0) {
        printf("Hello World!\n");

        ______
    }

    ______
    printf("Goodbye!\n")
} // main
Parent and child processes

```c
int main(){
    if (fork() == 0) {
        printf("Hello World!\n");
        _exit(0);  // Terminate child process
    }

    printf("Goodbye!\n")
} // main
```
Parent and child processes

```c
int main()
{
    if (fork() == 0) {
        printf("Hello World!\n");
        _exit(0); // Terminate child process
    }
    wait(0); // Forces parent to wait
    printf("Goodbye!\n")
} // main
```
Unix signals

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

- What can it do to prevent that from happening?

- Is this always possible?
Unix signals

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

- What can it do to prevent that from happening?

- Is this always possible?
Unix signals

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

- What can it do to prevent that from happening?  
  - The process can catch the signal

- Is this always possible?
Unix signals

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

- What can it do to prevent that from happening?
  - *The process can catch the signal*

- Is this always possible?
  - *NO, the SIGKIL signal cannot be caught*
  - *NO, signal number nine cannot be caught*
Process state transitions

- Which events will bring a RUNNING process into the WAITING state?
Process state transitions

- Which events will bring a RUNNING process into the WAITING state?

  - The process issues a (blocking) system request
Process state transitions

- Which events will bring a WAITING process into the READY queue?
Process state transitions

- Which events will bring a WAITING process into the READY queue?

  - The completion of a pending system request