First question

- What is the difference between *interactive applications* and *real-time applications*?
First question

- What is the difference between *interactive applications* and *real-time applications*?

- Interactive applications do not have strict deadlines.
First question

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

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

  - Because it needs to be done very fast.
First question

- Why is it impossible to hijack *interrupts*?
First question

- Why is it impossible to hijack *interrupts*?

  - Because their destination addresses are hard-wired.
First question

- What is the purpose of the `argv` parameter in the `execv()` system call?
First question

- What is the purpose of the `argv` parameter in the `execv()` system call?

  - It points to a list of parameters to be passed to the program fetched by the `execv()` system call.
First question

- Why are most servers *multithreaded*?
First question

- Why are most servers *multithreaded*?
  - In order to be able to handle multiple requests in parallel.
First question

- Give an example of a **soft real-time** application.
First question

- Give an example of a \textit{soft real-time} application.

  - Watching a video.
Second question

- Main advantage of modular kernels:
Second question

- **Main advantage of modular kernels:**
  - They let users add functionality to the kernel without having to recompile it (and without having to endure additional delays).
Second question

- *Main disadvantage of delayed writes:*
Second question

- Main disadvantage of delayed writes:
  - They do not guarantee the durability of writes.
Second question

- Main advantage of monolithic kernels:
Second question

- **Main advantage of monolithic kernels:**
  - They are fast.
Second question

- Main advantage of symmetric organization for multiprocessor OSes:
Second question

Main advantage of symmetric organization for multiprocessor OSes:

- They avoid potential bottlenecks (by letting the kernel to run on each processor).
Second question

- Main advantage of lightweight processes over conventional processes:
Second question

- Main advantage of lightweight processes over conventional processes:
  
  - They can be created much more cheaply than conventional processes because they share the address space of their parent.
Second question

- Main disadvantage of hard drives:
Second question

- **Main disadvantage of hard drives**

  - They are by far the slowest component of any computer system.
Third question

What will the following program print out?

```cpp
#include <iostream>

int main() {
    std::cout << "Hello!\n";
    std::cout << "Goodbye!\n";
    return 0;
}
```
What will happen (II)
What will happen (II)

Program will print two times "Hello!" and four time "Goodbye!"
Third question

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

- Complete the following sentences:

  - We can safely swap out processes that have remained a long time in the ________ state.

  - When a process does a blocking system call, it remains in the ________ state until the call is completed then goes to the ________ state.
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 ____________ state until the call is completed then goes to the ____________ state.
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 _____________ state.
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

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

- How can a process specify a different action?

- Is it always possible?
Fifth question

- What is the default action a Linux process takes when it receives a **signal**?
  - It terminates.
- How can a process specify a **different action**?
- Is it **always possible**?
Fifth question

What is the default action a Linux process takes when it receives a signal?
- It terminates.

How can a process specify a different action?
- It can catch the signal using signal().

Is it always possible?
Fifth question

- What is the default action a Linux process takes when it receives a *signal*?
  - *It terminates.*

- How can a process specify a *different action*?
  - *It can catch the signal using signal().*

- Is it *always possible*?
  - *No, the SIGKIL signal cannot be caught.*