Solutions to the first midterm

COSC 4330/6310
Summer 2012
First question: True or false

- Processes waiting for the CPU are in the **waiting** state.
First question: True or false

- Processes waiting for the CPU are in the **waiting** state.

- **FALSE**, they are in the **ready state**
First question: True or false

UNIX was the first system to be written in C.
First question: True or false

- UNIX was the first system to be written in C.
- **TRUE**, it was designed to be portable (and C was specifically written for UNIX)
First question: True or false

- Memory protection is always implemented in hardware.
First question: True or false

- Memory protection is always implemented in hardware.

- TRUE, any other solution would be too slow
First question: True or false

- `execve()` system calls are often followed by a `fork()` system call.
First question: True or false

- **execve()** system calls are often followed by a **fork()** system call.

- FALSE, it is the other way around:

  **fork()** system calls are often followed by an **execve()** system call.
First question: True or false

- In a *multiprogramming system*, there can be *many programs* in the system but only *one process*. 
First question: True or false

- In a multiprogramming system, there can be many programs in the system but only one process.

- FALSE, there are many processes competing for one of the CPU cores
First question: True or false

Most modern operating systems have a *microkernel*.
First question: True or false

- Most modern operating systems have a microkernel.
- FALSE, microkernels are too slow
Second question: Advantages and disadvantages

- What is the major disadvantage of modular kernels over monolithic kernels?
Second question: Advantages and disadvantages

- What is the major disadvantage of modular kernels over monolithic kernels?
  - They make the kernel less robust.
Second question:
Advantages and disadvantages

- What is the major advantage of *modular kernels* over *monolithic kernels*?
Second question: Advantages and disadvantages

- What is the major advantage of modular kernels over monolithic kernels?

- They let users add functionality to the kernels like new file systems or device drivers for new devices.
Second question: Advantages and disadvantages

- What is the major disadvantage of CPUs that do not have a privileged mode?
Second question: Advantages and disadvantages

- What is the major disadvantage of CPUs that do not have a privileged mode?

- They cannot prevent user processes from executing I/O instructions.
What is the major disadvantage of not having memory protection?
Second question:
Advantages and disadvantages

- *What is the major disadvantage of not having memory protection?*

- We cannot prevent user processes from tampering with the kernel (and other user processes)
Second question: Advantages and disadvantages

- What is the major advantage of *user-level threads*?
Second question: Advantages and disadvantages

- What is the major advantage of user-level threads?

- They are portable and can run on kernels that do not support threads.
Third question: I/O redirection

Complete the following fragment of code to ensure that the *standard input* of the process is redirected to the pipe `mypipe`.

```
int fd, mypipe[2];

// Your code here

close(mypipe[0]); close(mypipe[1]);
```
Third question: I/O redirection

- Complete the following fragment of code to ensure that the **standard input** of the process is redirected to the pipe `mypipe`.

```c
int fd, mypipe[2];
pipe(mypipe);
close(0); dup(mypipe[0];
close(mypipe[0]); close(mypipe[1]);
```
Third question: I/O redirection

- Complete the following fragment of code to ensure that the **standard output** of the process is redirected to the pipe `mypipe`.

```
int fd, mypipe[2];

close(mypipe[0]); close(mypipe[1]);
```
Third question: I/O redirection

- Complete the following fragment of code to ensure that the *standard output* of the process is redirected to the pipe `mypipe`.

```c
int fd, mypipe[2];
pipe(mypipe);
close(1); dup(mypipe[1]);
close(mypipe[0]); close(mypipe[1]);
```
Fourth question

How many lines of output will the following program print? (5 points)

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

__________ lines
Fourth question

How many lines of output will the following program print? (5 points)

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

Three lines
Fifth question

- Give an example of a real time process with \textit{soft deadlines}.
Fifth question

• Give an example of a real time process with *soft deadlines*.

• A DVD player
Sixth question

- What is happening when a UNIX process issues a wait() system call and all its child processes have already terminated?
- Will the process wait forever?
Sixth question

- What is happening when a UNIX process issues a `wait()` system call and all its child processes have already terminated?
- Will the process wait forever?

- NO, processes that have terminated but have not yet been waited for by their parents remain in the process table in the ZOMBIE state.
- The waiting process returns immediately.
Seventh question

Which UNIX system call can we use to catch signals? What does it mean? Is it always possible?
Seventh question

- Which UNIX system call can we use to catch signals? What does it mean? Is it always possible?

- `signal(…)`
  - `signal` specifies what the process should do when it receives a specific signal
  - No signal number 9 (SIGKIL) cannot be caught
Eighth question

- Why should we *prevent* users of a multi-user system from *rebooting* the OS from their own CD-ROM?
Eighth question

- Why should we *prevent* users of a multi-user system from *rebooting* the OS from their own CD-ROM?

- User could reboot the system with an OS that will let do things they are not authorized to do.