Test 1 Revision

Edward Zhang

SOFTENG 370 T4

2011 SE370/CS340 Test

Question

Very early computers did not provide interrupt handling capabilities. What advantage does an interrupt handling system provide?

2011 SE370/CS340 Test

Question

Very early computers did not provide interrupt handling capabilities. What advantage does an interrupt handling system provide?

SPOOLing: Interrupt driven I/O, removing the need to wait for our computers to wait for I/O. Now we can do other stuff while waiting for our punch card input or printer output. Also allows for preemptive multitasking since a clock can interrupt.

2011 SE370/CS340 Test

Question

The first personal computer operating systems (including the Mac) were effectively resident monitors. Describe one way in which early PC operating systems were like resident monitors.

2011 SE370/CS340 Test

Question

The first personal computer operating systems (including the Mac) were effectively resident monitors. Describe one way in which early PC operating systems were like resident monitors.

- Single task at a time
- No memory protection
- Standard IO routines in memory (as opposed to drivers and syscalls)

Derived from 2018 SE370 Test

Question

What is an issue with using "trap and emulate" virtualization on x86 (prior to virtualization extensions such as Intel VT and AMD-V)?

Derived from 2018 SE370 Test

Question

What is an issue with using "trap and emulate" virtualization on x86 (prior to virtualization extensions such as Intel VT and AMD-V)?

- Instructions exist that can run in both user and kernel mode, give different output (such as `POPF`).
- Instructions also exist to allow a program to determine whether it was in privileged.
- These instructions don't throw an exception (trap), and thus cannot be emulated by the VMM.

Question

. . . and how do hardware virtualization extensions help resolve this?

Question

. . . and how do hardware virtualization extensions help resolve this?

Additional instructions allow the VMM to enter a special privledged mode (some call this "ring -1", although it's not a real protection ring), which allows it to host different guest kernels, all of which believe they have ring 0 (kernel mode) access to the system.

2018 SE370 Test

Question

The Windows subsystem for Linux is NOT a virtual machine. Explain how it is different from a virtual machine.

2018 SE370 Test

Question

The Windows subsystem for Linux is NOT a virtual machine. Explain how it is different from a virtual machine.

- No seperate kernel - instead, Unix syscalls are mapped into NT ones through a kernel module (LXCore)
- Somewhat similar to Application virtualization

Trivia

WSL2 uses a real Linux kernel running under Hyper-V, due to performance and feasibility issues implementing all syscalls in LXCore/on top of NT.

2011 SE370/CS340 Test

Question

Give reasons why a language such as Java is seldom used to implement operating systems?

2011 SE370/CS340 Test

Question

Give reasons why a language such as Java is seldom used to implement operating systems?

- No direct access to memory*
- Insufficient control over memory allocation*
- Runs inside JVM rather than directly on hardware
- ~~Performance~~ - not entirely true

Trivia

*One can technically manually allocate off-heap memory or access memory-mapped devices using `sun.misc.unsafe`

2011, 2018 Test

Question

Explain the difference between a thread and a process?

2011, 2018 Test

Question

Explain the difference between a thread and a process?

Processes have their own memory space and connections to files/devices (file descriptors), whereas threads typically share memory space within a given process (but have their own stack so they can be executing different code).

2018 SE370 Test

Question

What is the most important difference between system level and user level threads, and what is a consequence of the difference?

2018 SE370 Test

Question

What is the most important difference between system level and user level threads, and what is a consequence of the difference?

- With user-level threads, the OS only sees one thread per process, whereas with system-level threads the OS is aware of multiple threads per process.
- On a multiprocessor, different system-level threads can be scheduled on different processors, since the OS can schedule them on different processors.

2018 SE370 Test

Question

In general a small time-slice makes computers responsive. Explain why this is so and what penalty is paid as the time-slice gets smaller.

2018 SE370 Test

Question

In general a small time-slice makes computers responsive. Explain why this is so and what penalty is paid as the time-slice gets smaller.

- A context switch between two tasks takes a certain amount of time, as registers, stack pointers, etc. need to be switched out.
- As a result, smaller time slices means more context switches and thus lower throughput (but lower latency!).

2011 SE370/CS340 Test

Question

What error do you commonly get if you try to access a pointer which has not been initialised correctly in Linux?

2011 SE370/CS340 Test

Question

What error do you commonly get if you try to access a pointer which has not been initialised correctly in Linux?

Segmentation Fault (a memory access violation)

Step 1

- Why are we segfaulting?
- Stack vs. Heap memory allocation
- Expanding the size of the stack

Step 2

- What performance speedup did you get?
- Recall what `pthread_create` and `pthread_join` do

Step 2

- What performance speedup did you get?
- Recall what `pthread_create` and `pthread_join` do

Performance gain should've been nearly 2x

Step 3

Why is this a dumb idea?

Step 3

Why is this a dumb idea?
Consider how many times do we have to copy the arary. Also context switch penalities when number of active threads dramatically exceeds the number of h/w threads.

Step 4

What speedup did you get? With n cores, did you get near n times speedup?

Step 4

What speedup did you get? With n cores, did you get near n times speedup?
If you have a locked counter for number of threads, there can be massive contention for this, thus decreasing performance. However a different algorithm to more effectively divide up tasks without a global counter could result in a performance improvement (although the question asked for a counter).

Step 5

Consider how spinlocks can be more performant than mutexes.

Step 5

Consider how spinlocks can be more performant than mutexes. Spinlocks don't put a thread to sleep - thus, if the wait time is very short then spinlocks are more performant as we don't need to switch threads away and back again.

Trivia

Some implementations of mutex are actually hybrid-mutexes. This means, assuming there's more than 1 CPU core, a mutex will actually spinlock for a certain period of time before the thread is put to sleep, thus improving performance in some time.

Step 6

Processes vs. Threads (see question from before). Performance of pipes.

Step 6

Processes vs. Threads (see question from before). Performance of pipes.
Again pipe performance is implementation dependent. However, pipes are generally slower than shared memory, so we should see this be slower than Step 2.
Also consider creating a process is slightly more expensive than a thread, although this doesn't really matter if we're just creating one extra process.

Step 7

It's the same thing but more overhead concerns because more processes being created, and more pipes communicating.
Also depending on how you deal with contention for the core/process counter - do you pass one value around all the processes, or try something smarter (maybe use the pipe as a semaphore)

Step 8/9

Very similar to threads