Lecture Questions

From the first few lectures

Lecture Questions - Architecture

What is the basic control flow of the system?

- The CPU switches between running OS code and application code

Why do transitions from user code to the OS take place?

- Interrupts – some IO device (typically) needs attention
- Exceptions – the CPU has detected something problematic in executing an instruction
- Trap – the purpose of the instruction being executed is to transition into the OS (syscall)

Since they run on the same CPU, why can't applications do everything the OS can do?

- The hardware has two or more privilege levels
- Some instructions are privileged –require a sufficiently high privilege level – for the CPU to be willing to execute them

What happens on a transition from user code into the OS?

- Some registers that execution of the OS is about wipe out are saved by hardware, e.g., the user code PC at which the switch is occurring
- The PC is set to the address previously set by the OS. (a safe entry point into the OS)
- The privilege level of the CPU is elevated so that it can execute privileged instructions
- The hw or sw saves all registers so that execution of the user code can eventually be resumed

On a transition from the OS to user code?

- The previously saved registers (including the PC) are restored on the CPU
- The privilege level is lowered to user level

Lecture Questions - Architecture (2)

What mechanisms does the hardware provide to help the OS keep control of the system?

- CPU privilege level + privileged instructions
- memory access limitations, e.g., virtual memory
- The exception mechanism – detecting when something that needs OS attention has happened and causing a switch into the OS

When the OS is running, what stack is it using (in xk)?

- A per-process kernel stack

How does xk use the segmented memory system provided by x86_64?

- It basically renders it moot by mapping every hardware segment to the full linear address space (i.e., base 0 and length 4GB)

How is memory protected?

- On modern system, virtual memory
- The OS sets a privileged CPU register to point to address mapping structures for the address space the CPU should be using (e.g., when it dispatches a user process)

How are IO devices protected?

- Depending on the architecture or even system, it could be privileged instructions are required to communicate with the IO devices, or it could be that protected addresses must be read/written to communicate with them

Lecture Questions - Structure

What are some important components of an OS?

- process/thread management; memory/address space management; device IO; file systems; syscall interface / shells / windowing system; networking;

“Program” vs. “Process” vs. “Thread”

- Program is source code; process is running code; thread is an independent sequence of instruction execution within a process

What are the (execution) states of a process?

- Running, runnable, and blocked

What does it mean to be “blocked”?

- That not only are you not running, you’re not eligible to run until something more happens

What is the relationship between processes and address spaces?

- There is usually a 1-1 relationship between processes and address spaces

What’s special about device drivers?

- They’re written by third parties, but execute in privileged mode

Lecture Questions - Structure (2)

What's the difference between a disk and a filesystem?

- A disk is an array of blocks; a filesystem abstracts that to directories, files, and the like

What is the "scope" of a namespace and why does it matter?

- The region over which a single text string name will refer to the same object
- When my code communicates with some other code, we need a way to name objects. The scope of a name limits with who I can communicate using it.

If there were no adversaries would I need the OS to provide protection?

- Yes, because my code has bugs and I want to protect myself from myself

What does "OS structure" mean and why do we care?

- It's just the universal issue properly structuring and maintaining a large body of code while achieving efficiency
- Because the OS is used by every application, it's correctness and performance permeate everything.

What alternatives to monolithic have been tried?

- Layered; microkernel; variations on systems that export hardware interface or otherwise achieve similar goals

Why would you build a software layer that imitates a hardware layer? What's the real goal?

- You might want to provide virtual machines in this way
- The real goal: assured isolation and containment. There might be some cheaper

Lecture Questions - Processes

What is a process?

What lower level resources are hidden/simplified by the process abstraction?

Which aspects of process semantics most commonly important to software design issues? to system management issues?

Why is process creation broken into fork() and then exec()?

How do you make fork() less expensive?

What does a shell do?

How can processes communicate with each other? why would you want them to?

Why might you want additional abstraction built above processes?

What's the relationship of "user" to "process"?

Is "user" the fundamental abstraction?

Lecture Questions - Threads

What is a thread?

What's wrong with just using processes?

Can processes see each other's virtual address space? Can threads within the same process?

What are the advantages to using threads?

Kernel threads vs user-level threads

Lecture Questions - Synchronization

What are critical sections?

What is mutual exclusion?

What is the difference between spinning and blocking?

Lab Questions

From labs 1 and 2

Lab 1 Overview

What are the functions implemented ?

- Major filesystem operations including file_open, file_read, file_write, file_dup, file_stat and file_close

What were the major additions/changes to xk to get the functions to work ?

- Addition of a structure for bookkeeping information related to each specific file (file_info struct)
- Addition of a global file table that contains the file_info structs of all the open files in the system
- Addition of a process file table that contains references to the file_info structs in the global file table for all the open files in the current process

Why do we have a system call which then calls the specified function ?

- A system calls parses all the arguments given by the user and makes sure that they are valid before calling the specified function with those arguments. Making sure that the user is following the conventions expected.

Lab 2 Overview

What is the difference between a spinlock and a sleeplock ?

- A spinlock disables CPU interrupts and only releases the lock on the resource when release is called. A sleeplock releases the lock when either releasesleep is called or when the CPU's timer goes off.

When to use a spinlock vs. sleeplock?

What is the difference between fork and exec ?

- Fork creates a new process, and copies over all the information (like vspace, file table) from parent to child process. Exec replaces the stack of an existing process to set it up execute a given program.

What is the difference between wait and exit ?

- Exit means to kill the current process while wait looks for a killed child process of the calling parent process.
- Interesting to think of when we initially want to kill a process, vs when it is actually killed.

Lab 2 Overview (2)

Why do we need a pipe for inter-process communication ?

- A pipe serves as an optimization as processes only need to go to the allocated memory to read data from the pipe's buffer whereas processes need to go to disk when data is written to inodes, which is much slower.
- Also, the processes holding the read and write ends can be running concurrently in a multi-core environment.