CSci 5103
Operating Systems

Jon Weissman

The Landscape at 50K feet
OSPP Chap. 2

Announcements

- Kindle version of the book
  - $32 approximately

- UNITE is providing audio podcasts of the lectures (obviously merge with ppt but will not get you any board work)

Referee

- Share or multiplex resources
- Space share
  - Parallel computer containing 64 processors
  - 4 jobs each using 16 processors can run together
- Time share
  - Single CPU can be time-multiplexed to run slices of multiple jobs in time

A First Look at Some Key Concepts: #1

- **kernel**
  - The software component that controls the hardware directly, and implements the core privileged OS functions.
  - Modern hardware has features that allow the OS kernel to protect itself from untrusted user code.
  - User code can invoke the kernel only at well-defined entry points – what are those?

Kernel

- Different OS organizations
- Microkernel
  - Small kernel, rest of OS possibly in user-space
  - Mostly research systems: Mach, Amoeba, Minix
  - Some mobile OS: symbian, blackberry
- Monolithic
  - Everything is in OS domain
  - Linux, Windows
  - Many try to isolate a “kernel” to be the machine-dependent interface code

Key Concept #2

thread

- An executing stream of instructions and its CPU register context.
- Hardware may directly support threads – *hyper-threading* (each core has two separate architectural contexts), x86 has this mode
- Generally, hardware is unaware of threads, and the OS/user libraries must provide it

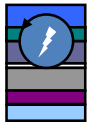
More on threads

- A thread is *schedulable*
  - it runs on a CPU core
  - defined by CPU register values (PC, SP)
  - *suspend*: save register values in memory
  - *resume*: restore registers from memory
- Multiple threads can execute independently
  - They can run in parallel on multiple CPUs...
    - **physical concurrency**
  - …or arbitrarily interleaved on a single CPU
    - **logical concurrency**
- Each thread must have its own stack

Key Concepts #3 and #4

virtual address space

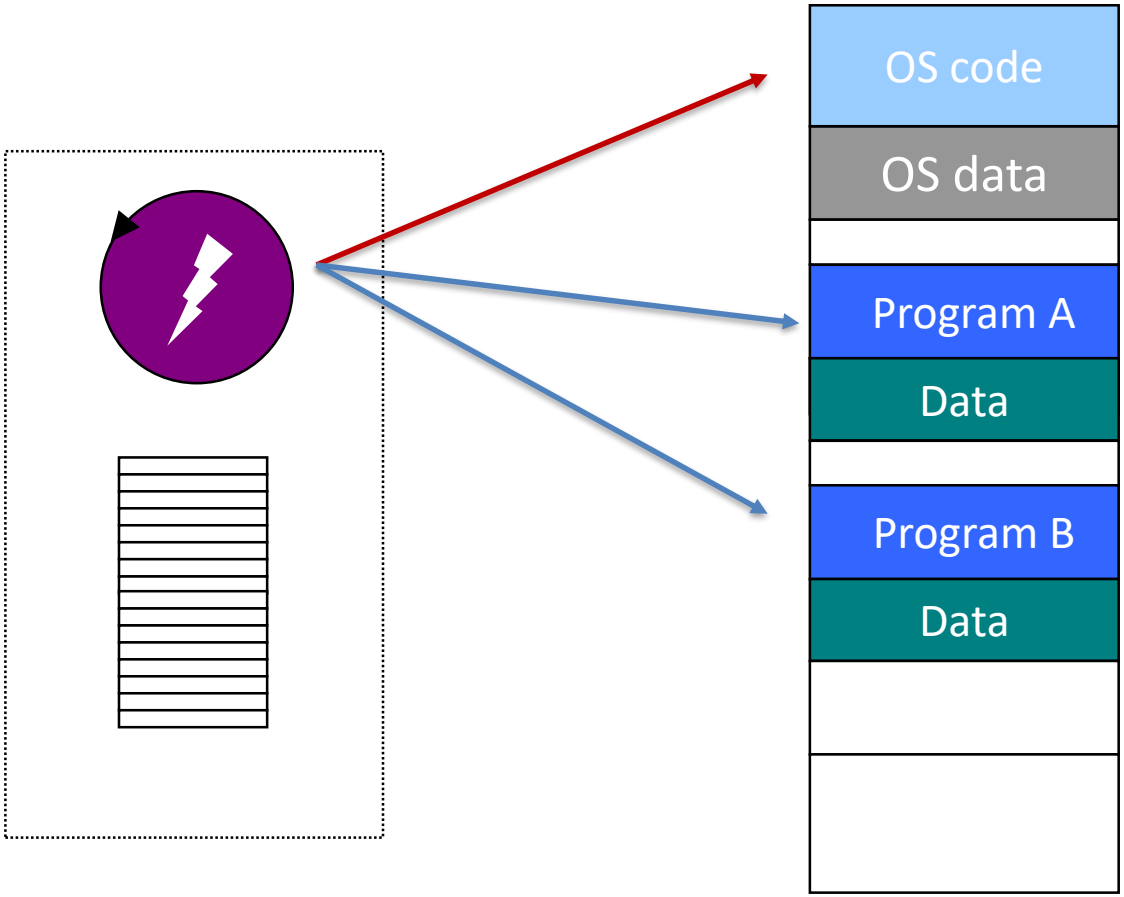
- An execution context for threads/processes that provides an independent name space for addressing code and data



process

- An execution of a program, consisting of a virtual address space, one or more threads, other resources, some OS kernel state. **Unit of isolation!**

Memory and the CPU

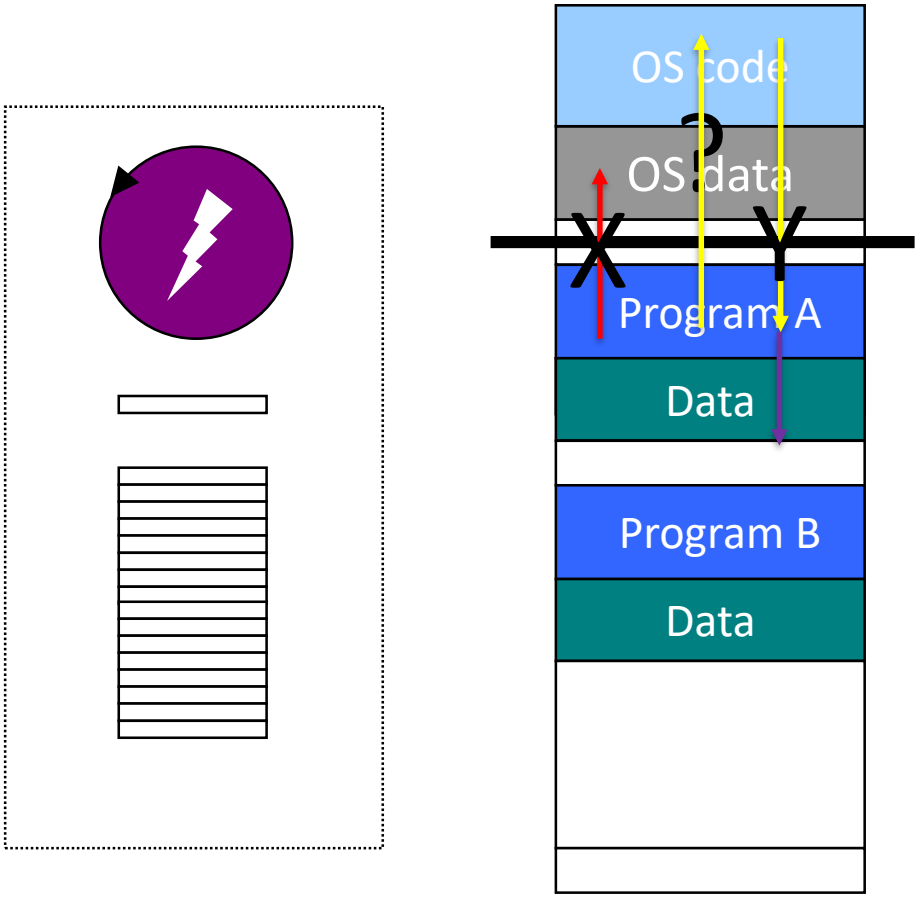


What is different between red and blue?

The Kernel

- The kernel code is “shared” by all user programs, **but** *the kernel is protected*:
  - User code cannot access internal kernel data structures directly
  - Think: object-oriented methods
    - Cannot access private variables and methods, only public ones
  - Hardware maintains mode bits to track whether kernel or user code is executing

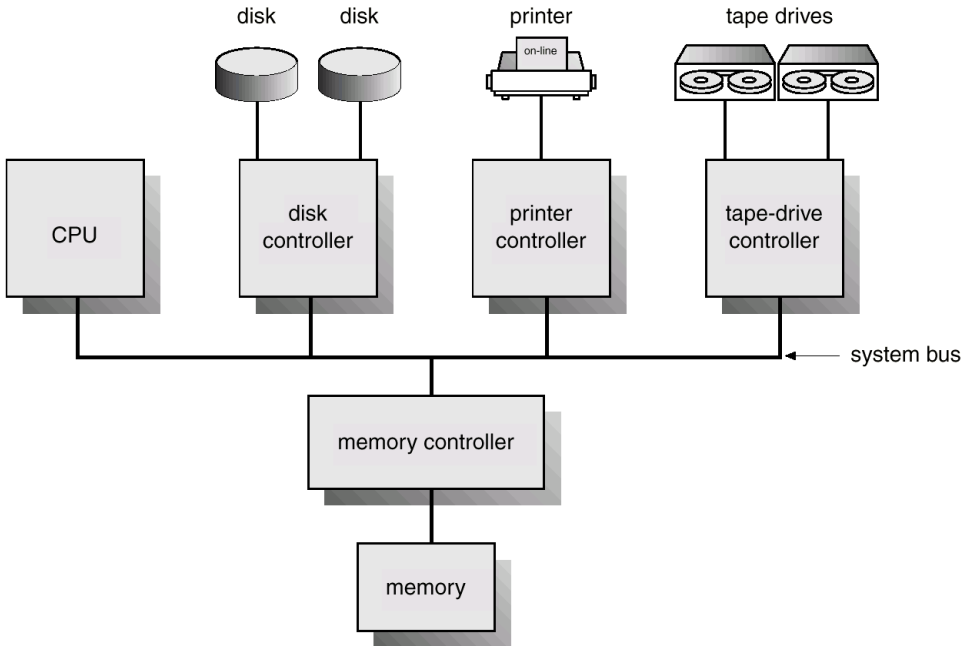
A Protected Kernel



What about program A -> Program B or B's data?

Turning to Hardware (Briefly)

- How does the OS interact with the external devices?
  - I/O Structure
  - Storage Structure



  - Each device controller is in charge of a particular device type
  - OS has special code to communicate with controllers
  - ?

Device Drivers

- Device drivers … (i.e. glue)
  - Most of the OS code is device drivers
  - Assembly or mix of assembly and C generally
  - Contains special I/O instructions

- Today, dynamically load device drivers into the OS
  - Why is this critical?
  - What is the problem with device drivers?

I/O

- User code cannot issue I/O instructions directly
  - Why?
- System call – the method used by a program to request action by the operating system
- Usually takes the form of a trap to a specific location in the kernel code

I/O Operation

- I/O devices and the CPU can execute concurrently
- How does device controller inform CPU that it has finished?
  - an interrupt
- CPU moves data from/to RAM to the device
  - With DMA, CPU just initiates, DMA controller can transfer between RAM <-> device

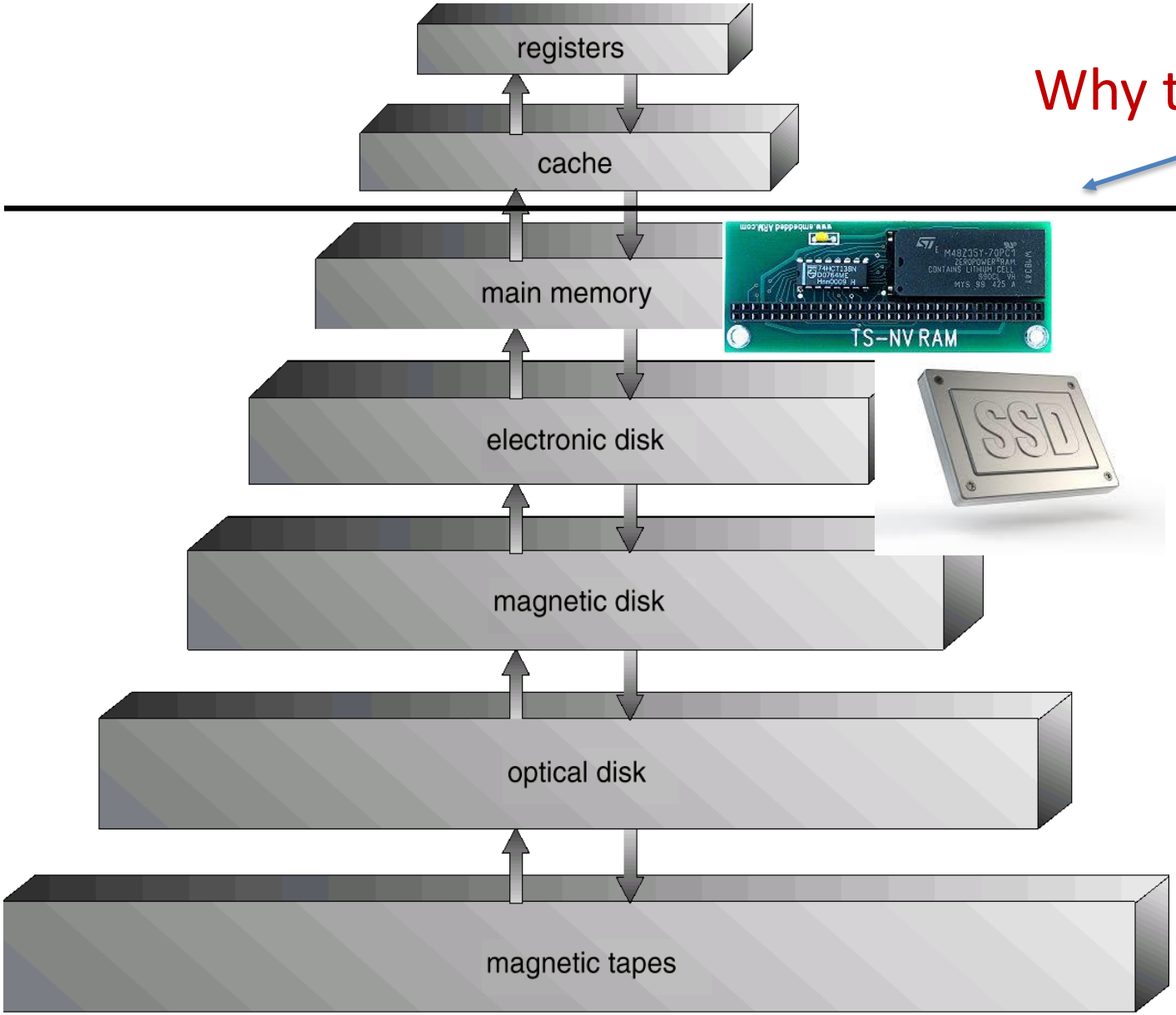
Interrupts: Key Ideas

- Interrupts transfer control to an interrupt service routine in the kernel
- A trap is a software-generated "interrupt" caused either by an error or a user request
- Q: What is meant by a user request?
- An operating system is *interrupt* driven.
  - Why? What is the alternative (suppose devices didn't raise interrupts)?

I/O Structure

- I/O types
  - Asynchronous
    - After I/O starts, control returns to kernel without waiting for I/O completion
    - Get an interrupt or notification when finished
  - Synchronous
    - CPU idles until I/O is ready (one I/O at a time)
  - API: synchronous I/O (built on asynchronous kernel I/O)
  - API: asynchronous I/O (ditto)

Storage-Device Hierarchy



Storage Issues

- Latency
  - Crossing the bus
  - Controller logic
  - Mechanical operations (HDD): very high
- Throughput
  - Sustained performance

Storage

- Memory is a large array of bytes, each with its own address. It is contains rapidly accessible data shared by the CPU and I/O devices.

- Main memory is a volatile storage device. It loses its contents in the case of system failure, power-down. Though this may be changing …

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to "back up" main memory.

Common OS System Components – 50K feet

- Process Management
- Main Memory Management
- Secondary-Storage Management
- I/O System Management
- File Management

Process Management

- A *process* is a program in execution. A process needs certain resources, including:
  - CPU, memory, files, access to I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion
  - Process suspension and resumption
  - Process communication and synchronization
  - Process scheduling
  - Bookkeeping: accounting

Main-Memory Management

- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom
  - Keep track of free memory
  - Allocate and deallocate memory space as needed

Secondary-Storage Management

- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling - which requests to serve in which order

    Q: given our brief discussion of I/O, why is this an issue?

I/O and File System Management

- The I/O system consists of:
  - Device-drivers
  - A buffer-caching system
- A *file* is a collection of related information defined by its creator. Commonly, files represent programs and data.
- The operating system is responsible for the following activities in connections with file management:
  - File/Directory creation, deletion, access, protection

Next Week

- The Kernel
- Read Chap. 2 OSPP, 3 OSPP (skim – refresh)
- HW #1 out on Thursday

Have a great weekend!