

CS 370: OPERATING SYSTEMS [INTRODUCTION]

Shrideep Pallickara
Computer Science
Colorado State University

August 23, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.1

Frequently asked questions from the previous class survey

- Lecture slides: where are they? <http://www.cs.colostate.edu/~cs370>
 - Where is the schedule of topics?
- Term project: Complexity, requirements, etc
- Kernel vs the OS
 - What is a kernel? What does it do? What does the rest of the OS do?
- How does the kernel create this illusion of multitasking?
- User-mode/kernel mode
- Can you go deeper into memory management?
- Fault isolation?
- Is Unix/Linux better or Windows?

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.2

Frequently asked questions from the previous class survey

- Tests
 - Will the tests have concepts not taught in class ... e.g. from textbook? NO!
 - How do I study for the quizzes/exams?

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.3

Topics covered in this lecture

- Caches and main memory
- Secondary storage
- Relative speeds of the memory hierarchy
- The Kernel Abstraction
- Buses

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.4

Memory hierarchy: Cache memory

- Mostly *controlled by hardware*
- Main memory divided up into **cache lines**
 - Usually 64 bytes
 - Addresses 0-63 in cache line 1, 64-127 in cache line 2, and so on
- Most heavily used cache lines are stored in high-speed cache **close** to the CPU

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.5

When a program needs to read a memory word

- Cache hardware checks if the needed line is in the cache
- If it is, that's a **cache hit**
 - Request satisfied from cache in about **2 clock cycles**
 - No memory access needed
- If needed line is not present in cache
 - **Cache miss**, and must access memory
 - **Substantial** time penalty

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.6

Caching is a powerful concept used elsewhere too.
Let's see when ...

- ① Large resource *can be divided* into pieces
 - ② Some pieces *used more heavily* than others
- OS caching examples:
 - ▣ Pieces of heavily used files in main memory
 - Reduce disk accesses
 - ▣ Conversion of file names to disk addresses
 - ▣ Addresses of Web pages (URLs) as hosts

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.7

CPU's usually have a couple of caches

- **L1 cache** is inside the CPU
 - ▣ Typically in the order of 16 KB
 - ▣ No access delay
- **L2 cache** holds several MB of data
 - ▣ Access delay of 1-2 clock cycles

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.8

Main Memory

- Usually called **RAM** (Random Access Memory)
- Cache misses go to the main memory
- **Volatile**
 - ▣ Contents lost when power is turned off
- Memory size is of the order of several GB in most modern desktops

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.9

Computers run most of their programs from
(rewriteable) main memory

- Typically implemented in a technology called DRAM (dynamic random access memory)
- Ideal Scenario: Programs and data reside permanently in main memory. BUT ...
 - ▣ Space is *limited*
 - ▣ Main memory is *volatile* storage

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.10

Secondary storage is needed to hold large quantities
of data permanently

- Programs use the disk as the source and destination of processing
- Seek time 7 ms
- SPIN: 7200 – 15000 RPM
- Transfer rate
 - ▣ Disk-to-buffer: 70 MB/sec (SATA)
 - ▣ Buffer-to-Computer: 300 MB/sec
- Mean time between failures
 - ▣ 600,000 hours
- 1 TB capacity for less than \$100

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.11

Improvements in hard disk capacity

- 1980 - 5 MB
- 1991 - 100 MB
- 1995 - 2 GB
- 1997 - 10 GB

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.12

Improvements in hard disk capacity

- 2002 - 128 GB addressing space barrier [28 bits]
 - ▢ Old IDE/ATA interface: 28-bit addressing
 - ▢ $2^{28} \times 512 = 2^{28} \times 2^9 = 2^{37} = 128 \text{ GB} = 137,438,953,472 \text{ bytes}$
- 2003 – Serial ATA introduced
 - ▢ Bus interface providing connections to mass storage devices
- 2005 - 500 GB hard drives
- 2008 - 1 TB hard drives

August 23, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.13

Characteristics of peripheral devices & their speed relative to the CPU

Item	time	Scaled time in human terms (2 billion times slower)
Processor cycle	0.5 ns (2 GHz)	1 second
Cache access	1 ns (1 GHz)	2 seconds
Memory access	70 ns	140 seconds
Context switch	5,000 ns (5 μ s)	167 minutes
Disk access	7,000,000 ns (7 ms)	162 days
Quantum	100,000,000 ns (100 ms)	6.3 years

August 23, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.14

Mechanical nature of disks limits their performance

- Disk access times *have not* decreased exponentially
 - ▢ Processor speeds are growing *exponentially*
- Disparity between processor and disk access times continues to grow
 - ▢ 1:14,000,000

August 23, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.15

RELATIVE SPEEDS OF THE MEMORY HIERARCHY

August 23, 2018

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.16

Since caches have limited size, cache management is critical

Level	1	2	3	4
Name	registers	cache	Main memory	Disk Storage
Typical Size	< 1 KB	< 16 MB	< 64 GB	> 100 GB
Implementation Technology	Custom memory, CMOS	On/off chip CMOS SRAM	CMOS DRAM	Magnetic disk
Access times	0.25 ns	0.5-25 ns	80-250 ns	> 5 ms
Bandwidth (MB/sec)	20,000 – 100,000	5000-10,000	1000-5000	80-300
Managed by	compiler	hardware	OS	OS
Backed by	cache	Main memory	Disk	CD/Tape

August 23, 2018
 Professor: SHRIDEEP PALICKARA

L2.17

ONTOGENY RECAPITULATES PHYLOGENY

August 23, 2018

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.18

After Charles Darwin's book ON THE ORIGIN OF SPECIES was published

- German zoologist Ernst Haeckl stated



- i.e. human egg goes through stages of being a fish, ... , before becoming human baby
- Modern biologists think this is a gross simplification!

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.19

Something vaguely similar has happened in the computer industry

- Each new species (*type of computer*) goes through the development its ancestors did
 - Both in hardware and software
 - Mainframe, mini computers, PC, handheld, etc

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.20

Much of what happens in computing and other fields is technology driven

- Ancient Romans lacked cars not because they liked walking
 - It is because they didn't know to build cars
- PCs exist not because people have a centuries-old pent-up desire to own one
 - It is now possible to manufacture them cheaply

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.21

Technology affects our view of systems

- A **change** in technology renders some idea **obsolete**
 - Another change could **revive** it
- Especially true when change has to do with **relative performance**
 - Of different parts of the system

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.22

Let's look at this relative performance

- When CPUs become faster than memories?
 - Caches become important to speed-up slow memory
- If new memory technology makes memories much faster than CPUs?
 - Caches will vanish!
- In biology extinction is forever
 - In computer science, it is sometimes only for a few years

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.23

Historical developments Large Memories

- IBM 7090/7094 1959-1964
 - 128 KB of memory
 - Programmed in assembly language (even the OS)
 - With time FORTRAN/COBOL and assembly was dead
- PDP-1 had only 4096 18-bit words of memory
 - Assembly is back!
 - Over time memory increases, assembly is out
- Microcomputers in 1980s
 - 4 KB memory and assembly is back again

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.24

Other places where such a cycle has gone on?

- Protection hardware
- Disks
- Virtual memory
- What may seem dated ideas on PCs
 - May soon come back on embedded computers

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.25

PERFORMANCE

August 23, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.26

There are two approaches to improving performance

- Determine component **bottlenecks**
 - Replicate
 - Improve

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.27

To replicate or improve?

"If one ox could not do the job, they [pioneers] did not grow a bigger ox, but used two oxen."

-- Admiral Grace Murray Hopper
Computer Software pioneer

"If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?"

-- Seymour Cray
Computer Hardware pioneer

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.28

Multiprocessor systems have 2-or-more processors in close communications

- The processors **share** the bus, and *may* share clock, memory and peripheral devices
- Advantages:
 - Increased throughput
 - Reliability

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.29

Multiprocessor systems fall in two categories based on control

- Asymmetric multiprocessing:
 - Controller processor manages the system
 - Workers **rely on controller** for instructions
- Symmetric multiprocessing
 - Processors are **peers** and perform all OS tasks
 - Have own set of registers and local cache
 - Share physical memory
 - **Supported by virtually all modern OS**

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.30

Recent trend has been towards adding multiple cores

- *Raison d'être*
 - On chip communications are much faster
 - Uses less power than multiple single-core chips
 - Cope with heat dissipations
 - Improve Thread level parallelism
- Number of cores **doubling** every year
 - Each core also gets more execution pipelines
 - Gartner Projection: 1024 cores soon!
- Challenge: Re-engineering programs daunting

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.31

Good fences make good neighbors
17th century proverb

THE KERNEL ABSTRACTION

August 23, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.32

A central role of the OS is **protection** — the isolation of potentially misbehaving applications and users

- Implementing protection is the job of the OS **kernel**
- The kernel has full access to all of the machine hardware
 - The lowest level of software running on the system
 - Necessarily trusted to do anything with the hardware
- Everything other than the kernel — that is, the untrusted software running on the system — is run in a restricted environment
 - Less than complete access to the full power of the hardware

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.33

What hardware is needed to let the kernel provide isolation?

- At a minimum, the hardware must support **three** things:
- **Privileged Instructions:** All potentially unsafe instructions are prohibited when executing in user mode
- **Memory Protection:** All memory accesses outside of a process's valid memory region are prohibited when executing in user mode
- **Timer Interrupts:** Regardless of what the process does, the kernel must have a way to periodically regain control from the current process

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.34

Conceptually, the kernel/user mode is a one-bit register

- When set to **1**, the processor is in kernel mode and can do anything
- When set to **0**, the processor is in user mode and is restricted

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.35

Privileged Instructions

- Instructions available in kernel mode, but not in user mode, are called **privileged instructions**
- To do its work, the kernel must be able to execute these instructions
 - Change privilege levels, adjust memory access, and disable and enable interrupts, set/reset timers
- If these instructions were available to applications?
 - A rogue application would in effect have the power of the kernel

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.36

Making memory sharing safe

- The kernel must be able to configure the hardware so that each application process can read and write **only its own memory**
 - Not the memory of the operating system or any other application
- While it might seem that read-only access to memory is harmless, the OS needs to provide both security and privacy.
 - For example, user passwords may be stored in kernel memory while they are being verified

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.37

Hardware timers

- Timers can be set to **interrupt** the processor after a specified delay
 - Either in time or after some number of instructions have been executed
- Each timer interrupts one processor ... separate timer for each CPU
 - The kernel might set each timer to expire every few milliseconds
- Resetting the timer is a privileged operation
 - User-level process cannot inadvertently or maliciously disable the timer
- How does the kernel know if an application is in an infinite loop?
 - It doesn't!

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.38

Mode transitions

- The kernel places a user process in a carefully constructed sandbox
 - The next question is how to safely transition from executing a user process to executing the kernel, and vice versa
- These transitions are **not rare events**
 - E.g.: A web server might switch between user mode and kernel mode thousands of times per second
- Transitions must be both fast and safe

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.39

There are three reasons for the kernel to take control from a user process

- Reasons: interrupts, processor exceptions, and system calls
- Asynchronous events
 - **Interrupts** are triggered by an **external event** and can cause a transfer to kernel mode after any user-mode instruction
- Synchronous events
 - **Processor exceptions** and **system calls** are triggered by **process execution**
 - The term **trap** refers to any synchronous transfer of control from user mode to the kernel

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.40

Interrupts are also used to inform the kernel of the completion of I/O requests

- Mouse device hardware triggers an interrupt every time the user moves or clicks on the mouse
 - The kernel, in turn, notifies the appropriate user process — the one the user was "mousing" across
- Virtually **every I/O device generates an interrupt** whenever some input arrives for the processor and whenever a request completes
 - E.g.: the Ethernet, WiFi, hard disk, thumb drive, keyboard, mouse, etc.

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.41

As the processor executes instructions, it checks for whether an interrupt has arrived

- If so, it completes or stalls any instructions that are in progress
 - Instead of fetching the next instruction, the processor hardware saves the current execution state
 - Starts executing at a specially designated interrupt handler in the kernel

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.42

Processor exceptions

- A processor exception is a **hardware event** caused by user program behavior that causes a transfer of control to the kernel
- A processor exception occurs whenever a process
 - Attempts to perform a privileged instruction
 - Accesses memory outside of its own memory region
 - Causes an arithmetic overflow. E.g. divide-by-zero
 - Accesses a word of memory with a non-aligned address
 - Attempts to write to read-only memory

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.43

User processes can also transition into the kernel voluntarily

- To request that the kernel perform an operation on the user's behalf
- A **system call** is any procedure provided by the kernel that can be called from user level
 - Examples include system calls to establish a connection to a web server, to send or receive packets over the network, to create or delete files, to read or write data into files, and to create a new user process.

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.44

To protect the kernel from misbehaving user programs

- It is key that the hardware transfers control on a system call to a pre-defined address
 - User processes cannot be allowed to jump to arbitrary places in the kernel
- The kernel handles the details of:
 - Checking and copying arguments
 - Performing the operation, and
 - Copying return values back into the process's memory

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.45

System calls provide the illusion that the kernel is simply a set of library routines available to users

- Implementing system calls requires the operating system to define a **calling convention**
- Once the arguments are in the correct format, the user-level program can issue a system call by executing the trap instruction to transfer control to the kernel
- The kernel implement its system calls in a way that protects itself from all errors and attacks that might be launched
 - Extreme version of defensive programming: always assume that system call parameters are intentionally designed to be as malicious as possible.

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.46

When an interrupt, processor exception or system call trap occurs ...

- How does the processor know what code to run?
- The processor has a special register that points to an area of kernel memory called the **interrupt vector table**
- The hardware determines which device caused the interrupt, if the trap instruction was executed, or what exception condition occurred
 - Thus, the hardware can select the right entry from the interrupt vector table and invoke the appropriate handler
- The format of the interrupt vector table is processor-specific

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.47

The interrupt vector table on the x86

- Entries 0 – 31: are for different types of processor exceptions
 - E.g: divide-by-zero anything related to arithmetic overflow
- Entries 32 – 255 are for different types of interrupts
 - Timer, keyboard, etc.
 - By convention, entry 64 points to the system call trap handler

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.48

What about kernel to user mode transitions?

- New process
- Resume after an interrupt, processor exception, or system call
- Switch to a different process
- User-level upcall
 - Most OS provide user programs with the ability to receive asynchronous notification of events

August 23, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.49

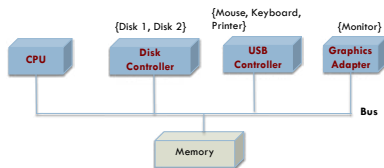
BUSES

August 23, 2018

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.50

A simple bus-based structure



August 23, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.51

Limitations of the bus structure from the earlier slide

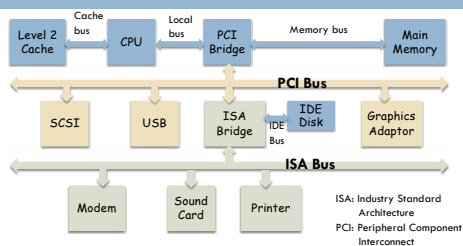
- As processors and memories got faster
 - Ability of a single bus to handle *all traffic* strained considerably
- Result?
 - Additional buses were added
 - For faster I/O devices and CPU-memory traffic

August 23, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.52

What a modern bus architecture looks like



August 23, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.53

There are two main BUS standards

- Original IBM PC ISA (Industry Standard Architecture)
- PCI (Peripheral Component Interconnect)
 - From Intel

August 23, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L2.54

The IBM PC ISA bus

- Runs at 8.33 MHz
- Transfers 2 bytes at once
- Maximum speed = 16.67 MB/sec
- Included for backward compatibility
 - ▢ Older and slower I/O cards

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.55

The PCI bus

- Can run at 66 MHz
- Transfer 8 bytes at once
- Data transfer rate: 528 MB/sec
- Most high-speed I/O devices use PCI
- Newer computers have an updated version of PCI
 - ▢ **PCI Express**

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.56

Other specialized buses: IDE (Integrated Drive Electronics) bus

- For attaching peripheral devices
 - ▢ CD-ROMs and Disks
- Grew out of the disk controller interface

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.57

Other specialized buses: USB (Universal Serial Bus)

- Attach **slow** I/O devices to the computer
 - ▢ Keyboard, mouse etc
- Uses a small **4-wire** connector
 - ▢ **Two** supply electrical power to the USB devices
- Centralized bus
 - ▢ Root device polls I/O devices every millisecond
 - Check if they have any traffic

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.58

Some more information about USB

- All USB devices share a **single** USB device driver
 - ▢ *No need to install* a driver for each device
 - ▢ Can be added to computer *without need to reboot*
- USB 1.0 has a transfer rate of 1.5 MB/sec
- USB 2.0 goes up to 60 MB/sec
- USB 3.0
 - ▢ Specification ready on 17 November 2008
 - ▢ Theoretical signaling rate: 600 MB/sec (4.8 Gbps)
 - ▢ USB 3.1: Jan 2013 goes to 10 Gbps
 - ▢ US 3.2 released in September 2017 transfer modes 10 and 20 Gbps

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.59

Other buses

- SCSI (Small Computer System Interface)
 - ▢ High performance bus
 - ▢ For devices that need high bandwidth
 - Fast disks, scanners
 - ▢ Up to 320 MB/sec
- IEEE 1394
 - ▢ Sometimes called FireWire (used by Apple)
 - ▢ Transfer speeds of up to 100 MB/sec
 - Camcorders and similar multimedia devices
 - ▢ No need for a central controller (unlike USB)

August 23, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.60

The contents of this slide-set are based on the following references

- Thomas Anderson and Michael Dahlin, *Operating Systems: Principles and Practice*, 2nd Edition, Recursive Books. ISBN: 0985673524/978-0985673529. [Chapter 2]
- Andrew S Tanenbaum and Herbert Bos, *Modern Operating Systems*, 4th Edition, 2014, Prentice Hall. ISBN: 013359162X/ 978-0133591620. [Chapter 1]
- Avi Silberschatz, Peter Galvin, Greg Gagne, *Operating Systems Concepts*, 9th edition, John Wiley & Sons, Inc. ISBN-13: 978-1118063330. [Chapter 1, 2]
- Kay Robbins & Steve Robbins, *Unix Systems Programming*, 2nd edition, Prentice Hall ISBN-13: 978-0-13-042411-2. [Chapter 1]

August 23, 2018
Professor SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L2.11