CS370 Operating Systems

Colorado State University Yashwant K Malaiya Fall 2021 Lecture 2



Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

Notes

Logistics:

- Help Sessions: material not covered in lectures
 - Required: attend or watch video.
 - Coming Tues/Wed (5:30-6 PM) : HW1 inc C pointers, dynamic memory allocation, makefiles, Valgrind
- On-line quizzes
 - Released Fri evening, due Monday evening 11 PM.
 - Allow enough time. Some may take 30-40 minutes or more.
 - No collaboration of any type among the students is allowed.
- IC Quizzes
 - In-class, iClicker
 - Almost everyday 1-2 times.
 - Distance students evaluated differently



FAQ

Assignments & Quizzes:

- You must work individually. No collaboration is permitted.
 - TAs will check to ensure there was no collaboration.
 - Automated/manual/data-based approches
- HW Requirements (C/Java/Python):
 - submissions must compile and run on the machines in the <u>CSB-120</u> <u>Linux lab</u>.
 - C and Java: You will provide your own makefile
 - the TAs will test them on department machines.
 - More details in assignment documents
 - HW1 will be available today



Expectations

- Teams: Should be checked daily.
 - Updates will be shared on Teams.
 - You can have discussions with your peers and GTAs.
 - But note
 - No code can be exchanged under any circumstances
 - Appropriate use expected
 - Private posts: use email
- Lectures: You are expected to attend all lectures (except distance students)
 - Recordings on Teams



From Operator to Operating System



5

What is an Operating System?



Introductions

- In Teams
- Class-room:
 - We will finish that in 2-3 lectures
 - When I call your name,
 - Give your compete name,
 - where you are from (city, country)
 - Degree you are working for, area of interest



What is an Operating System?



- Referee
 - Manage sharing of resources, Protection, Isolation
 - Resource allocation, isolation, communication
 - Isolation among threads, processes, users, virtual machines/containers
- Illusionist
 - Provide clean, easy to use abstractions of physical resources
 - Infinite memory, dedicated machine
 - Higher level objects: files, users, messages
 - Masking limitations, virtualization

Glue

- Common services
 - Storage, Window system, Networking
 - Sharing, Authorization
 - Look and feel



Today

- History and major developments
- Input/output
 - Interrupts
 - DMA
- Multiprocessor, Multiprogramming, Multitasking
- Memory
- Storage



Short History of Operating Systems

- One application at a time
 - Had complete control of hardware
- Batch systems
 - Keep CPU busy by having a queue of jobs
 - OS would load next job while current one runs
- Multiple programs on computer at same tin

1960s 80286 (1984)

Dua

core

2004

Vt-x

2005

Colorado State Un

- Multiprogramming: run multiple programs at seemingly at the "same time"
- Multiple programs by multiple or single user
- Multiple processors in the same computer
- Multiple OSs on the same computer.

One Processor One program View

Early processors (LC-3 is an example, simplified ARM)

- Instructions and data fetched from Main Memory using a program counter (PC)
- Traps and Subroutines
 - Obtaining address to branch to, and coming back
 - Using Stack Frames for holding
 - Prior PC, FP
 - Arguments and local variables
- Dynamic memory allocation and heap
- Global data



One Processor One program View

- External devices: disk, network, screen, keyboard etc.
- Device interface: Status and data registers
- User and Supervisor modes for processor
 - User mode (for user programs)
 - Some resources cannot be used directly by a user program
 - I/O can be done only using *system calls* (traps)
 - Supervisor (or Kernel, priveiledged) mode
 - Access to all resources
 - Input/output operations are done in kernel mode, hence require system calls.
- 1/0
 - Device drivers can use polling or interrupt
 - Interrupts need context switch ____
 - I/O done in supervisor mode
 - System calls invoke devise drivers

Enough info to resume (registers, process state etc)



What a simple view don't include

- Cache between CPU and main memory
 - Makes the main memory appear much faster



- Direct memory access (DMA) between Main Memory and Disk (or network etc)
 - Transfer by blocks at a time
- Neglecting the fact that memory access slower than register access
- Letting program run *concurrently* (Multiprogramming) or with many threads
- Multiple processors in the system (like in Multicore)



Information transfer in a system

- CPU Registers (Caches) Memory
 - CPU addresses memory locations
 - Bytes/words at a time
 - We will see some details
- Memory (Controllers hw/sw) external devices
 - Chunks of data
 - External devices have their own timing
 - DMA with interrupts
 - Disk is external!



System I/O (Chap 1, 12 SGG 10the)



A Modern processor: SandyBridge



- Package: LGA 1155
 - 1155 pins
 - 95W design envelope
- Cache:
 - L1: 32K Inst, 32K Data (3 clock access)
 - L2: 256K (8 clock access)
 - Shared L3: 3MB 20MB

- Transistor count:
 - 504 Million (2 cores, 3MB L3)
 - 2.27 Billion (8 cores, 20MB L3)



Functionality comes with great complexity!



I/O Hardware (Cont.)

- I/O Devices have associated registers where device driver places commands, addresses, and data
 - Data-in register, data-out register
 - status register, control register
 - Typically, 1-4 bytes, or FIFO buffer
- Devices have associated addresses, used by
 - Direct I/O instructions
 - Memory-mapped I/O
 - Device data and command registers mapped to processor address space



I/O Transfer rates MB/sec



Polling vs Interrupt

- Polling: IO initiated by software (P&P, ch 8)
 - CPU monitors readiness
 - Keeps checking a bit to see if it is time for an IO operation,
 - not efficient
- Interrupts: IO is initiated by hardware (P&P ch 10.2, YZ ch. 10)
 - CPU is informed when the external device is ready for an IO
 - CPU does something else until interrupted

Patt & Patel (CS), Yifeng Zhu (ECE)



Interrupts

- Polling is slow
- Interrupts used in practice
- CPU Interrupt-request line triggered by I/O device
 - Checked by processor after each instruction
- Interrupt handler receives interrupts
 - Maskable to ignore or delay some interrupts
- Interrupt vector to dispatch interrupt to correct handler
 - Context switch at start and end
 - Based on priority
 - Some interrupts maybe nonmaskable
 - Interrupt chaining if more than one device at same interrupt number



Exact details cpu

specific

Interrupts (Cont.)

- Interrupt mechanism also used for exceptions, which include
 - Terminate process, crash system due to hardware error
 - Page fault executes when memory access error
 - OS causes switch to another process
 - System call executes via trap to trigger kernel to execute request



Direct Memory Access (DMA)

- for movement of a block of data
 - To/from disk, network etc.
- Requires DMA controller
- Bypasses CPU to transfer data directly between I/O device and memory
- OS writes DMA command block into memory
 - Source and destination addresses
 - Read or write mode
 - Count of bytes
 - Writes location of command block to DMA controller
 - Bus mastering of DMA controller grabs bus from CPU
 - Or Cycle stealing from CPU but still much more efficient
 - When done, interrupts to signal completion



Interrupt-Driven I/O Cycle



Six Step Process to Perform DMA Transfer



Direct Memory Access Structure

- high-speed I/O devices
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block





I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
 - Memory management of I/O including
 - **buffering** (storing data temporarily while it is being transferred),
 - **caching** (storing parts of data in faster storage for performance),
 - spooling (the overlapping of output of one job with input of other jobs) like printer queue
 - General device-driver interface
 - Drivers for specific hardware devices



Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- New devices talking already-implemented protocols need no extra work
- Each OS has its own I/O subsystem structures and device driver frameworks
- Devices vary in many attributes
 - Character-stream or block
 - Sequential or random-access
 - Synchronous or asynchronous (or both)
 - Sharable or dedicated
 - Speed of operation
 - read-write, read only, or write only



A Kernel I/O Structure







Storage Structure

- Main memory only large storage media that the CPU can access directly
 - Random access
 - Typically volatile (except for ROM)

Disk for short

Memory for short

- Secondary storage extension of main memory that provides large nonvolatile storage capacity
 - Hard disks (HDD) rigid platters covered with magnetic recording material
 - Disk surface divided into tracks, which are subdivided into sectors
 - The disk controller transfers between the device and the processor
 - Solid-state disks (SSD) faster than hard disks, lower power consumption
 - More expensive, but becoming more popular
- Tertiary/removable storage
 - External disk, thumb drives, cloud backup etc.



Storage Hierarchy

- Storage systems organized in hierarchy
 - Speed
 - Cost
 - Volatility
- Caching copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- Device Driver for each device controller to manage I/O
 - Provides uniform interface between controller and kernel



Storage-Device Hierarchy





Performance of Various Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Movement between levels of storage hierarchy can be explicit or implicit

- Cache managed by hardware. Makes main memory appear much faster.
- Disks are several orders of magnitude slower.



General Concept: Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
 - If it is, information used directly from the cache (fast)
 - If not, data copied to cache and used there
- Cache smaller than storage being cached
 - Cache management important design problem
 - Cache size and replacement policy
- Examples: "cache", browser cache ..



Cache la

Poudre?

Multilevel Caches

- Cache: between registers and main memory
 - Cache is faster and smaller than main memory
 - Makes main memory appear to be much faster, if the stuff is found in the cache much of the time
 - Hardware managed because of speed requirements
- Multilevel caches
 - L1: smallest and fastest of the three (about 4 cycles, 32 КВ)
 - L2: bigger and slower than L1 (about 10 cycles, 256KB)
 - L3: bigger and slower than L2 (about 50 cycles, 8MB)
 - Main memory: bigger and slower than L3 (about 150 cycles, 8GB)
- You can mathematically show that multi-level caches improve performance with usual high hit rates.

Multiprocessors



Multiprocessors

- Past systems used a single general-purpose processor
 - Most systems have special-purpose processors as well
- Multiprocessor systems were once special, now are common
 - Advantages include:
 - 1. Increased throughput
 - 2. Economy of scale
 - **3.** Increased reliability graceful degradation or fault tolerance
 - Two types:
 - Asymmetric Multiprocessing each processor is assigned a specific task. (older systems)
 - 2. Symmetric Multiprocessing each processor performs all tasks

Symmetric Multiprocessing Architecture





Multiprocessor

Multi-chip and multicore

- Multi-chip: Systems containing all chips
 - Chassis containing multiple separate systems
- Multi-core





Multiprogramming and multitasking

- Multiprogramming needed for efficiency
 - Single user cannot keep CPU and I/O devices busy at all times
 - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
 - A subset of total jobs in system is kept in memory
 - One job selected and run via job scheduling
 - When it has to wait (for I/O for example), OS switches to another job
- Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
 - Response time should be < 1 second
 - − Each user has at least one program executing in memory ⇒ process
 - − If several jobs ready to run at the same time ⇒ CPU scheduling
 - If processes don't fit in memory, swapping moves them in and out to run
 - Virtual memory allows execution of processes not completely in memory