Operating Systems and Networks

Lecture 01:

Introducing the course

Behzad Bordbar

Introducing the course

❑Why we created this new module

- Personal experience and trends in technology
- HoS

Important

❑First time

❑Both practical and theory

❑Requires lots of work and time… exam HARD ☹

But

❑Does not require lots of background

Introducing the course (continue)

1. Demonstrate an understanding of the **fundamental** concepts and issues involved in OS and networking of IP-based systems Interact and manage an Operating System
2. Demonstrate an understanding of the challenges involved in the design of Distributed Systems in general and main methods of addressing them
3. Explain Transport Layer protocols and their differences
4. Understand the basics and practical issues and architectures involving in important Application Layer protocols
5. Demonstrate **practical understanding** of the theoretical foundations of Operating Systems and Distributed Systems

Introducing the course (continue)

- Sessional: 1.5 hr examination (80%), continuous assessment (20%).
- Supplementary (where allowed): 1.5 hr examination only (100%).
- You will have lab hours to ask your questions from a teaching assistant (time will be announced later) .

What am I going to learn?

Mixture of theory AND practice of:

- OS Fundamentals and architecture
- Shell programming
- OS networking
- OS support for Distributed Systems
- Distributed object, RMI and RPC
- Virtualisation, Xen, KVM *
- cryptographic algorithms and Security
- P2P (*)
- Wireless protocols
- Web servers: Apache server and nginx
- subversion and git
- maven (*)
- Distributed file systems ()
- HDFS
- Web Services (*)
- NoSQL and OS (*)

But we don't teach kernel programming

Exercise 0: learn Linux

- **This course is not for you if you don't want to learn Linux**
- Mac people are OK, as have access to shell.
- Cygwin and MS powershell wont do ☹
- Is Linux a new OS to you or need to brush up?
  - SoCS machines
  - Dual boots system
  - Virtualised environment **(**install Ubuntu on **Vmware player** or **Virtual box)**

Exercise 0: Learn Basic Linux commands, for example from

http://www.debianhelp.co.uk/commands.htm

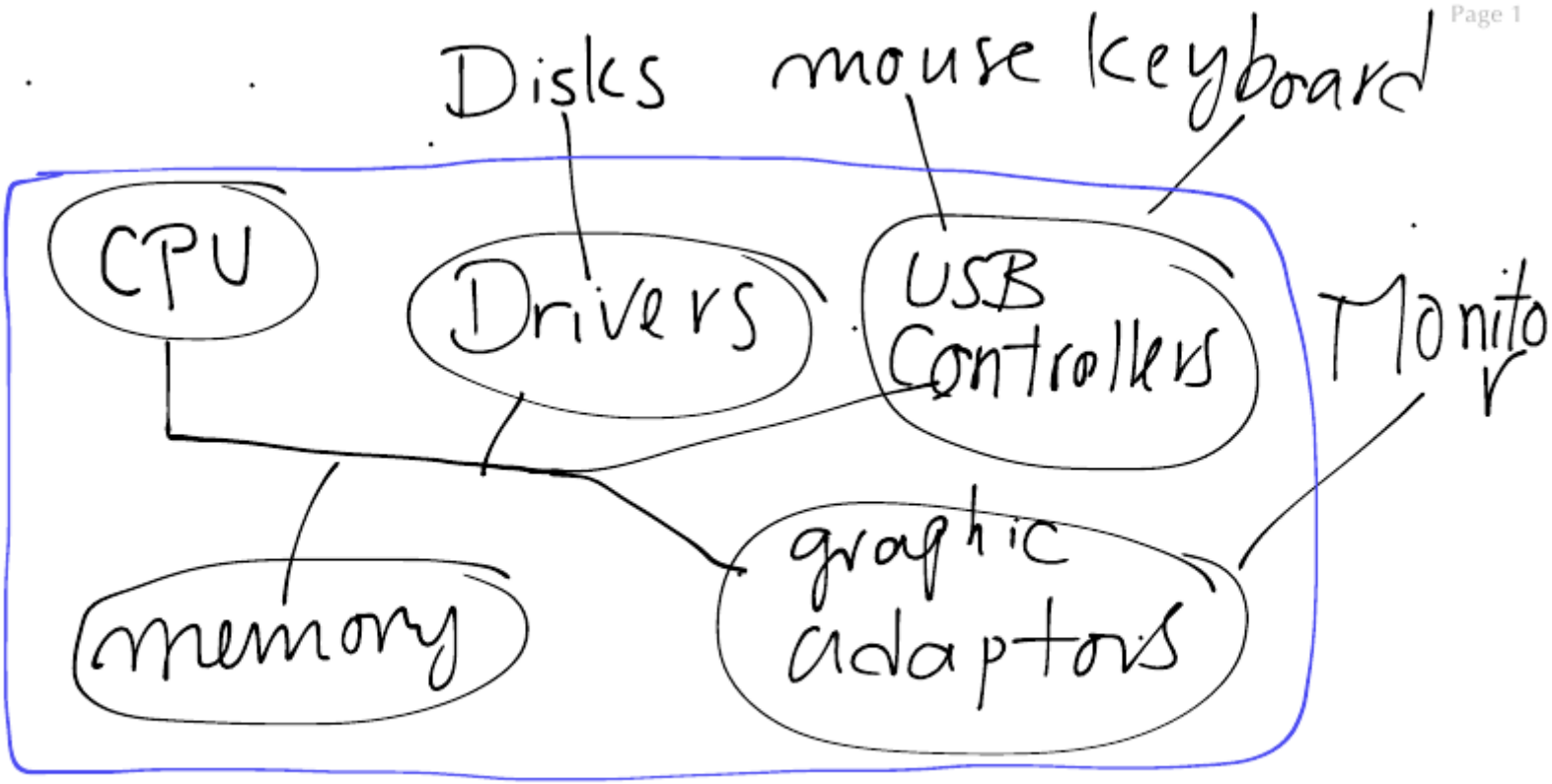
What reading material can help me?

- excellent library with lots of books on OS and Networking and Distributed systems
- Lots of online books

- Modern Operating Systems by Andrew S. Tanenbaum
- Operating System Concepts by Abraham Silberschatz, Peter B. Galvin and Greg Gagne
- Distributed Systems: Concepts and Design by George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair
- Data Communications and Networking by Behrouz Forouzan

Operating system: preliminaries

- What is in your computer?



- We want to learn these? But what is OS?

central processing unit (CPU)

- "brain" of the computer

CPU

Semiconductor companies:

- **Intel**
- PC: core i7, i5,....
- Sever and Workstation processor: Intel Xeon* processor E7, E5, ...
- Mac: Hawell, Ivy Bridge, Sandy Bridge,...

AMD (Advanced Micro Devices Ltd)

- Boldddozer (Vishera, Zambezi,...), K10 series (Athlon, Operon, ...)

All above are based on x86: **backward compatible (?)** set architecture based on Intel 8086 cpu.

AMD also produces CPU based on a blueprint developed by ARM

- Majority of mobile phones use ARM

CPU(continue)

How does it work? *Draw picture!*

- fetch the first instruction from memory,
- decode it to determine its type and operands
- execute it, and
- then fetch, decode, and execute subsequent instructions.

Instructions are CPU specific

- i7* core can not execute ARM instructions!

CPU(continue)

Accessing memory to get an instruction or data takes much longer than executing an instruction:

- CPUs contain registers inside to hold variables and temporary results.

What is instruction set?

- Load/store data from/to memory into a register
- combine two operands from registers and store result, for example adding
- …

Register

- 8-bit or 32 –bit storage. Examples of registers are:
- **General purpose** reg.: temporary data & results
- **program counter**: memory address of the next instruction to be fetched and then program counter is updated to point to its successor.
- **Stack pointer**: points to the top of the current stack in memory

Stack?

- Stack contains one frame for each procedure to be entered
- stack frame holds those input parameters, local variables, and temporary variables that are not kept in registers.

CPU modes

- At least two modes, **kernel mode** and **user mode**
- kernel mode: CPU can execute every instruction in its instruction set and use every feature of the hardware (complete access to hardware).
- When CPU in kernel mode we say OS in kernel mode.
- User programs run in user mode, which permits only a subset of the instructions to be executed and a subset of the features to be accessed. Generally, all instructions involving I/O and memory protection are disallowed in user mode.
- Program Status Word (PSW) is a register that (among other things) keeps the mode of the CPU

System call and Trap

So how do a user program do for example I/O?

- Ask OS: a user program must make **a system call**, which **traps** into the kernel and invokes the operating system.
- The TRAP instruction switches from user mode to kernel mode and starts the operating system.
- When the work has been completed, control is returned to the user program at the instruction following the system call.

We will look at this in details later! Jut one small point

Not every trap is caused by system calls!

- Some traps are caused by the hardware to warn of an exceptional situation such as an attempt to divide by 0 or an arithmetic underflow.
- Trap causes operating system gets control and must decide what to do. Example:
  - OS terminates program when error
  - error can be ignored and underflowed number set to 0
- Do you know about exception handling: that is when control is handled to program.

GPU =/=CPU

- Graphic Processing Unit (GPU) coined by Nvidia
- Called VPU (Visual Processing Unit) by ATI (a competitor of Nvidia)
- GPU: electoronic circuit specialised for graphic processing
- Presented as a video card in a PC for processing graphics
- Originally designed and used for image manipulation. Turned out to be suitable for parallel computing (most notably CUDA)

How to say what CPU you have

$ cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel(R) Core(TM) i7-3667U CPU @ 2.00GHz
stepping ….

- You see a number i7-3667U, what does it mean?

See intel page

- or run hardware lister

$ sudo lshw
$ sudo lshw |grep -i cpu

What happens when your computer starts?

Bootstrap program (bootloader, GRUB) runs first :

- It initializes all aspects of the system, from **CPU registers** to **device controllers** to **memory contents.**
- Load operating system and kernel to memory and start it.

Where is it saved if no CPU powered up?

- firmware: hardware in **read-only memory (ROM)** or **electrically erasable programmable read-only memory (EEPROM)**,
- Why in firmware? Can not be infected easily with virus

After your computer started:

- program at boot time become System Daemons/processes (in unix “init”)
- Event occur (mouse click, a program want to access a file…) we refer to these as **interrupts** from either the hardware or the software.
- Hardware may trigger an interrupt at any time by sending a signal to the CPU.
- Software may trigger an interrupt by executing a special operation called a **system call… (**we see what happens with this, how about hardware**)**

When CPU interrupted:

- it stops what it is doing and immediately transfers execution to a fixed location.
- The fixed location usually contains the starting address where the service routine for the interrupt is located.
- The interrupt service routine executes
- on completion, the CPU resumes the interrupted computation.

Memory

An array of bytes.

- read-only memory (ROM) and EEPROM
- ROM can not be modified: suitable for bootstrap program or game cartridges
- EEPROM can be modified but not frequently: smartphones have EEPROM to store their factory-installed programs.
- CPU needs read **and write**: **main memory** (also called random-access memory, or RAM).
- Main memory commonly is implemented in Dynamic Random-Access Memory (DRAM) technology.

Memory (continue)

- Register is another type of memory.

Main memory goes away when machined turned off:

- Secondary storage: magnetic disk, optical disk, tapes

Cache

- The data that has been used a lot is cached in a faster storage system.
- So if CPU looking for info, first check cache then main memoery. …

[cache is a concept more general than operating system, your browser has a cache, your dbms has a cache]

Speed of access to memory

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

Picture from Dinosaur book

Summary

- Description of the module

Started by preliminaries of OS

- CPU
- Register
- System call trap and interrup
- What happens when computer starts?
- Different types of Memory and their speed

Will continue with preliminaries of OS