CP 226: Operating Systems

Lecture One

Reference Book :Chapter One and Two

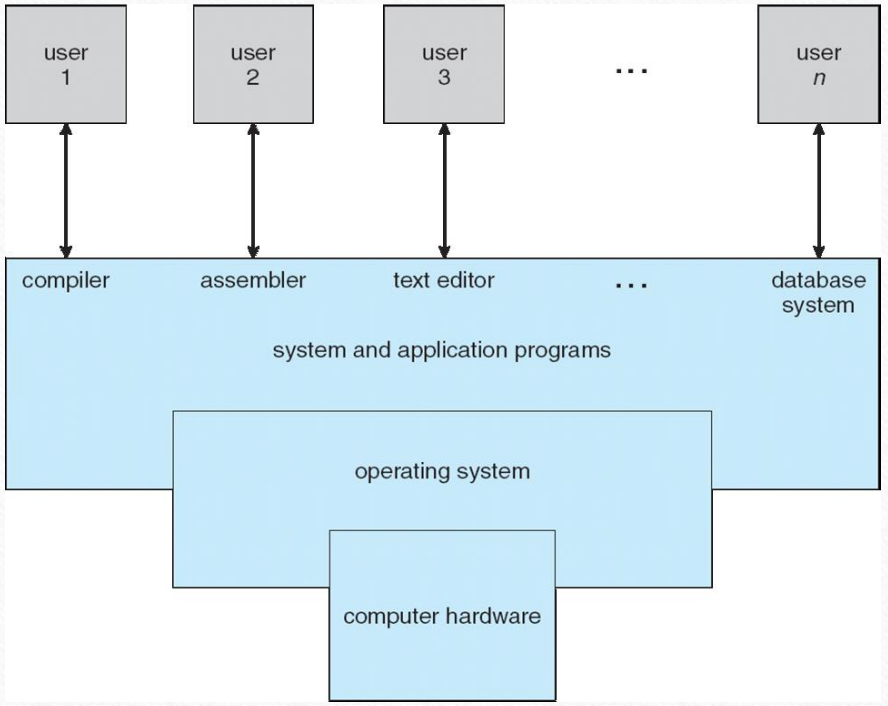
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs
  - solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Four Components of a Computer System



OS Usage/Definition

- **Hardware Abstraction**
  - Turn hardware into something that application can use
- OS is a **resource allocator/Manager**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

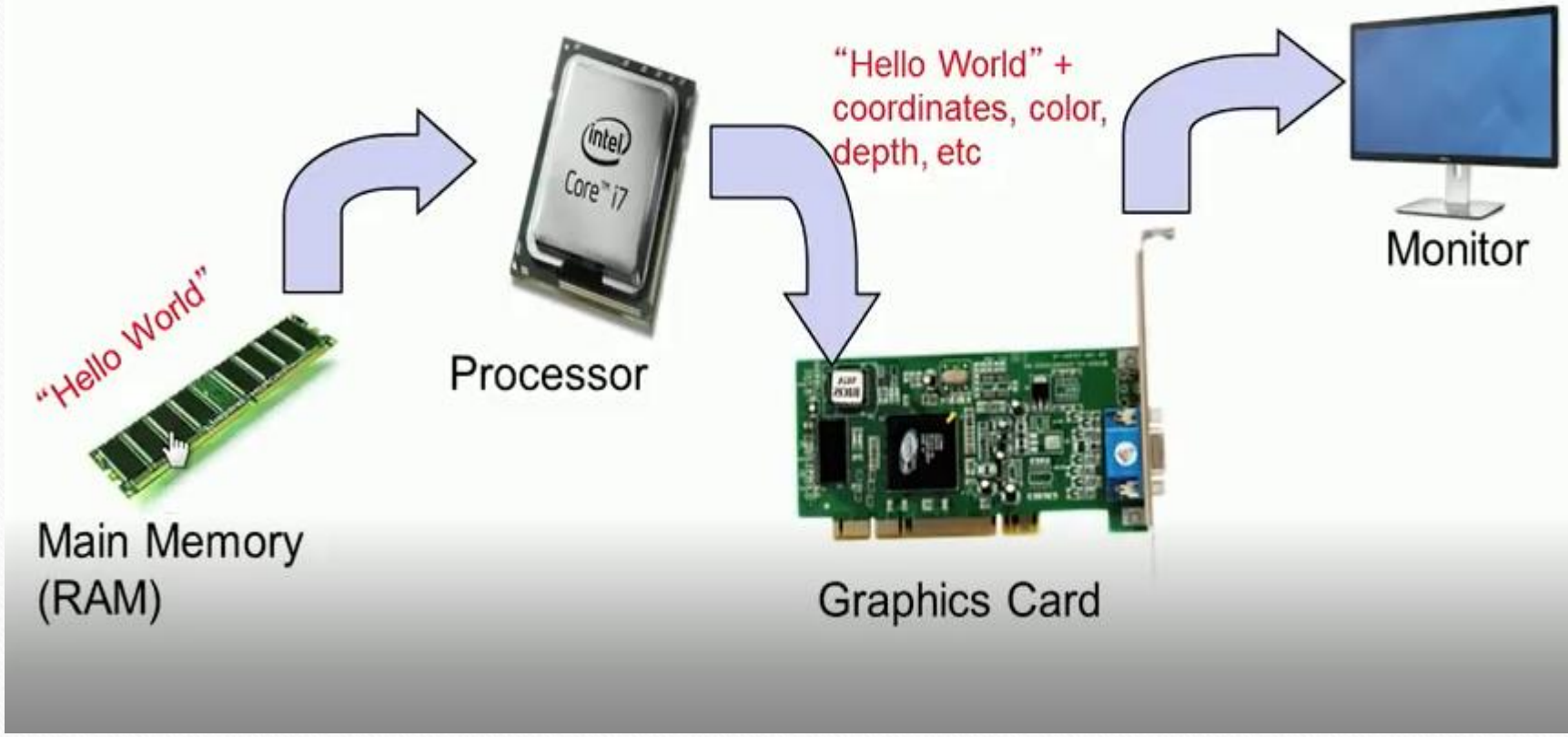
Example

- What is the output of the following program?

```c
#include <stdio.h>

int main(){
  char str[] = "Hello World\n";
  printf("%s", str);
}
```

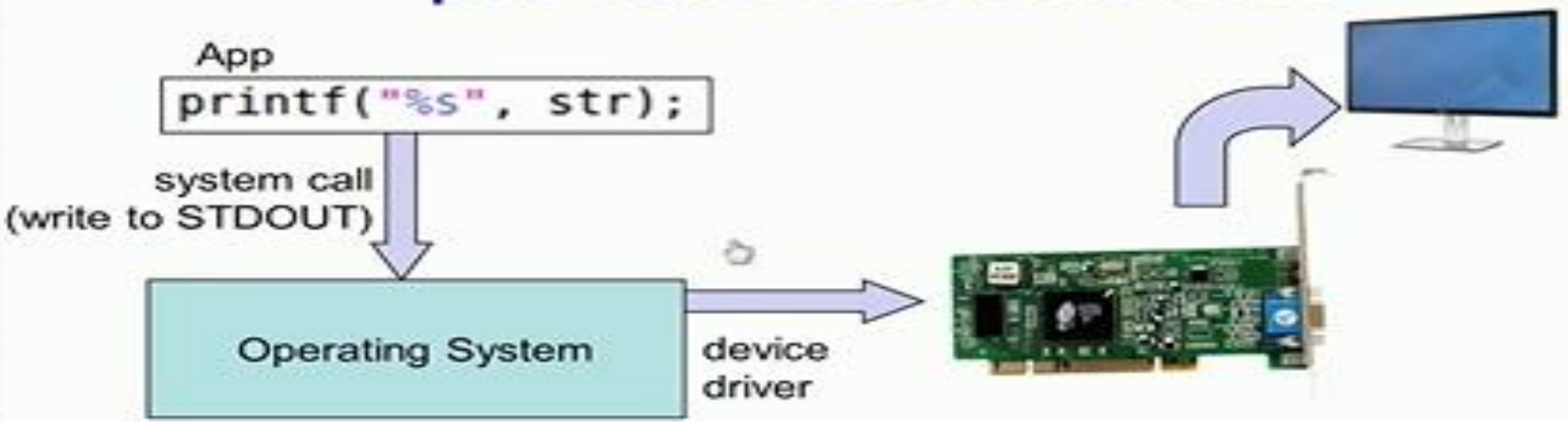
Displaying on the Screen



Example

- Without OS, all programmers need to take care of every nitty gritty detail of HW
- Execution of program can be complex and tedious
- Program become hardware dependent
- Thus**, Abstraction**

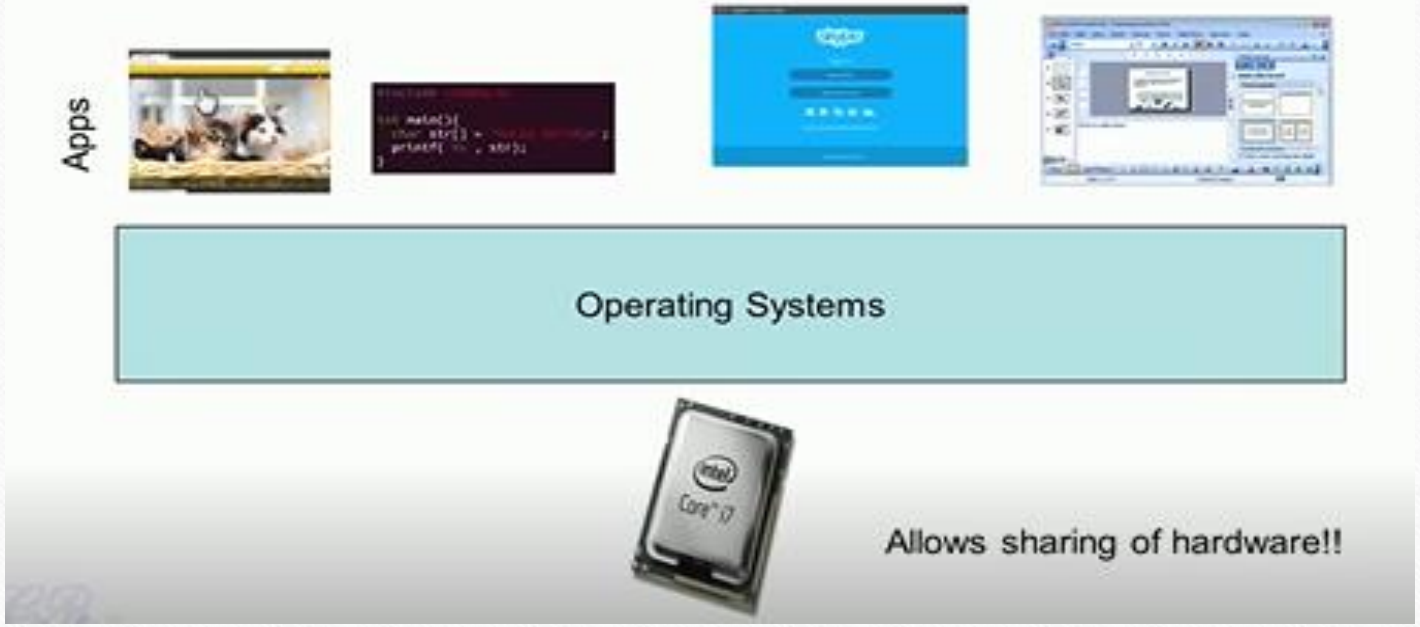
Operating Systems provide **Abstraction**



- **Easy to program apps**
  - No more nitty gritty details for programmers
- **Reusable functionality**
  - Apps can reuse the OS functionality
- **Portable**
  - OS interfaces are consistent. The app does not change when hardware changes

OS as Resource Manager

- Multiple apps but limited hardware



OS as Resource Manager …

- OS must manage CPU, Memory, network, secondary Storage (HD) etc.
- Resource management
  - Allow multiple apps to share resources
  - Protects apps from each other
  - Improves performance by efficient utilization of resources
- While sharing, the program should be isolated from each other

Operating System Definition (Cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is a good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program.

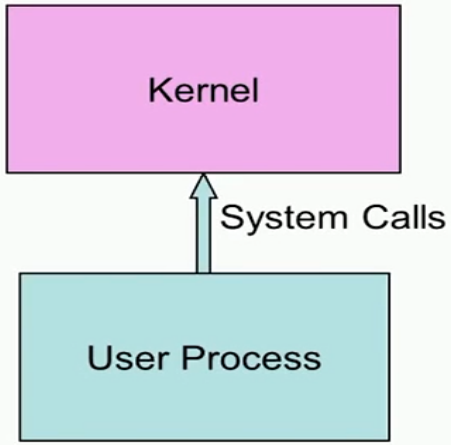
Computer Startup

- A computer to start running to get an instance when it is powered up or rebooted it need to have an initial program to run
- The initial program is known as **bootstrap** need to be simple
- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

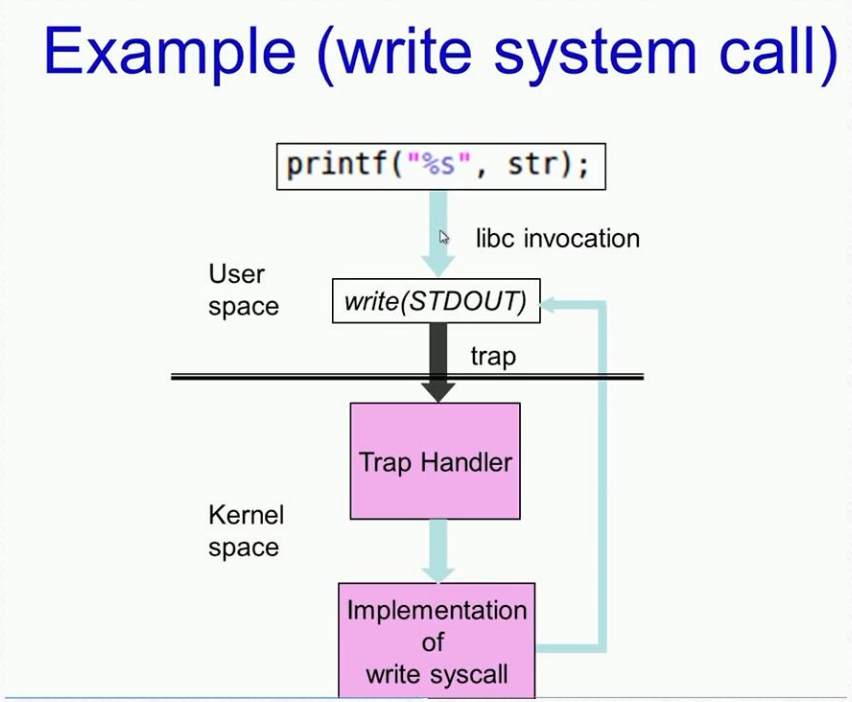
System Calls

Communicating with the OS (System Calls)

- System call invokes a function in the kernel using a Trap
- This causes
  - Processor to shift from user mode to privileged mode
- On completion of the system call, execution gets transferred back to the user process



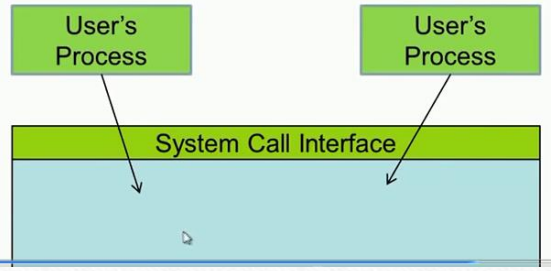
Example of System Calls



System Call Interfaces

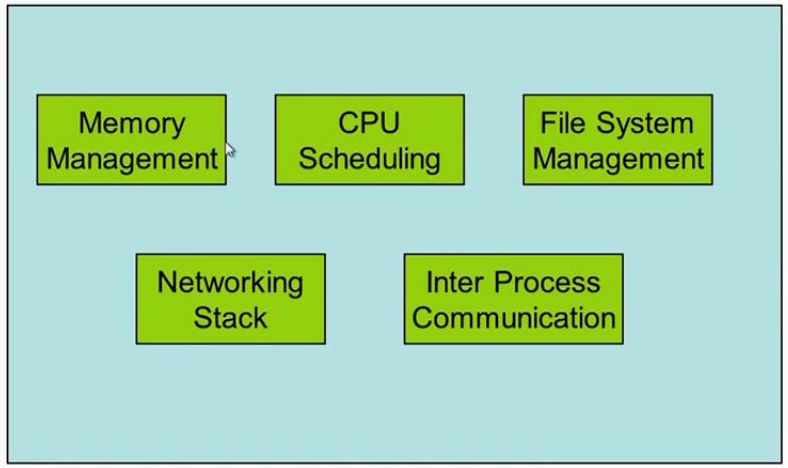
System Call Interfaces

- System calls provide users with interfaces into the OS.
- What set of system calls should an OS support?
  - Offer sophisticated features
  - But yet be simple and abstract whatever is necessary
  - General design goal : rely on a few mechanisms that can be combined to provide generality



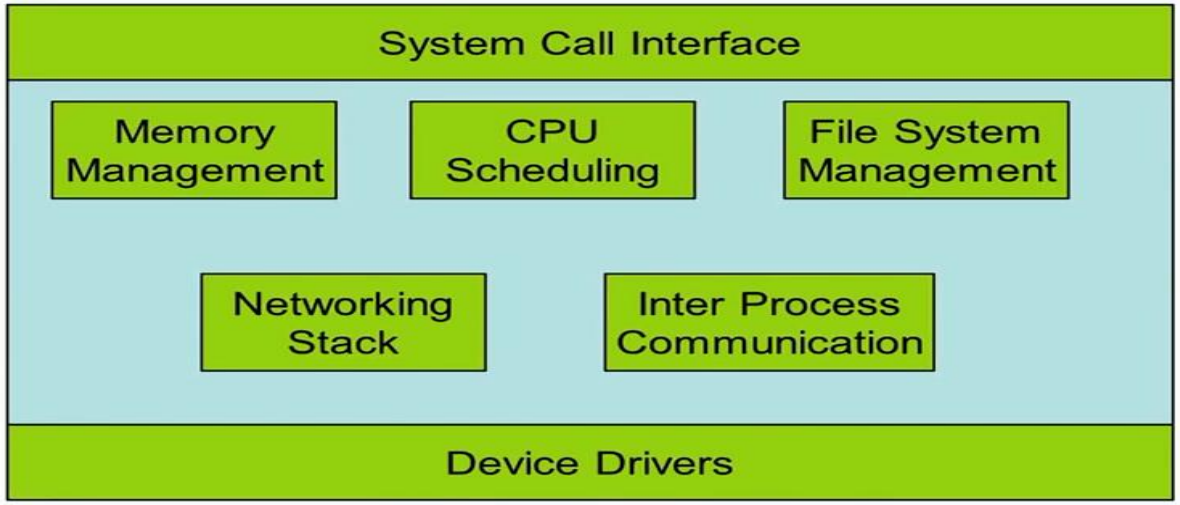
Components of OS

What goes into an OS?



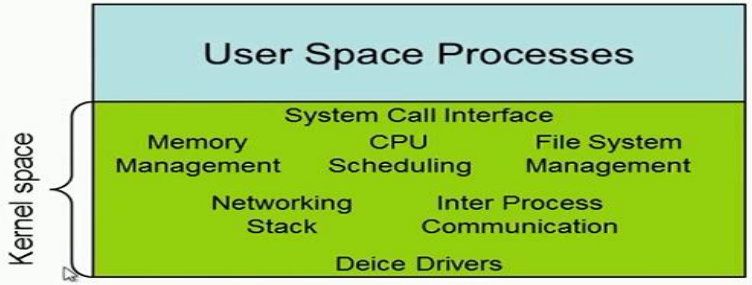
Components of OS (Cont..)

What goes into an OS?



OS Structure

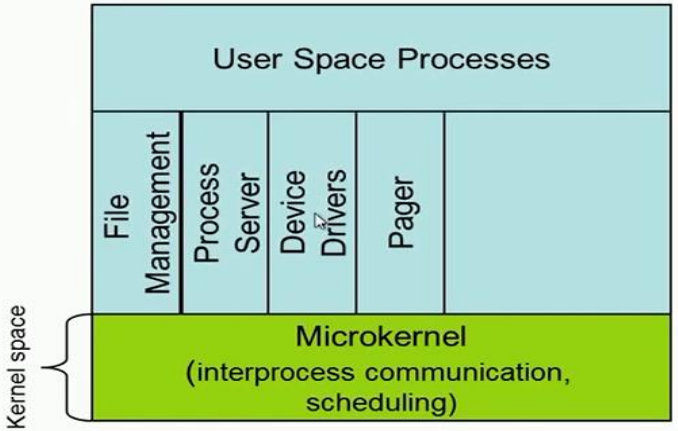
OS Structure : Monolithic Structure



- Linux, MS-DOS, xv6
- All components of OS in kernel space
- Cons : Large size, difficult to maintain, likely to have more bugs, difficult to verify
- Pros : direct communication between modules in the kernel by procedure calls

OS Structure (Cont..)

OS Structure : Microkernel



Eg. QNX and L4

- Highly modular.
  - Every component has its own space.
  - Interactions between components strictly through well defined interfaces (no backdoors)
- Kernel has basic inter process communication and scheduling
  - Everything else in user space.
  - Ideally kernel is so small that it fits in the first level cache

Hybrid Kernel

- Read

Discussion Questions

- How OS provide HW abstraction to USER and application program?
- User space and Kernel space (discus)
- How does OS ensures isolation property among different processes?
- Why a programmer dos not need to care about the architecture of computer system when writing a program?
- Monolithic , Microkernel and Hybrid kernel (discus)
- Why is it possible to have OS for mobile devices which has limited memory size?

Reading

- Read Chapter 1 and 2 from main reference book
- Attempt questions at the end of each chapter

- END of Part 1