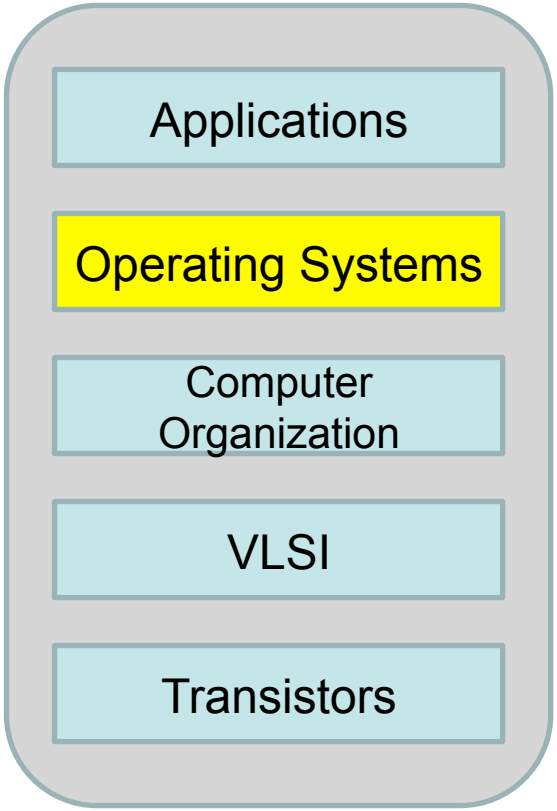
Operating Systems Introduction

Chester Rebeiro

IIT Madras

Webpage : http://www.cse.iitm.ac.in/~chester/courses/16o_os/index.html
Moodle : Institute moodle
Google group : https://groups.google.com/forum/#!forum/osiitm_2016

The Layers in Systems



OS usage

- Hardware Abstraction

  turns hardware into something that applications can use

- Resource Management

  manage system's resources

A Simple Program

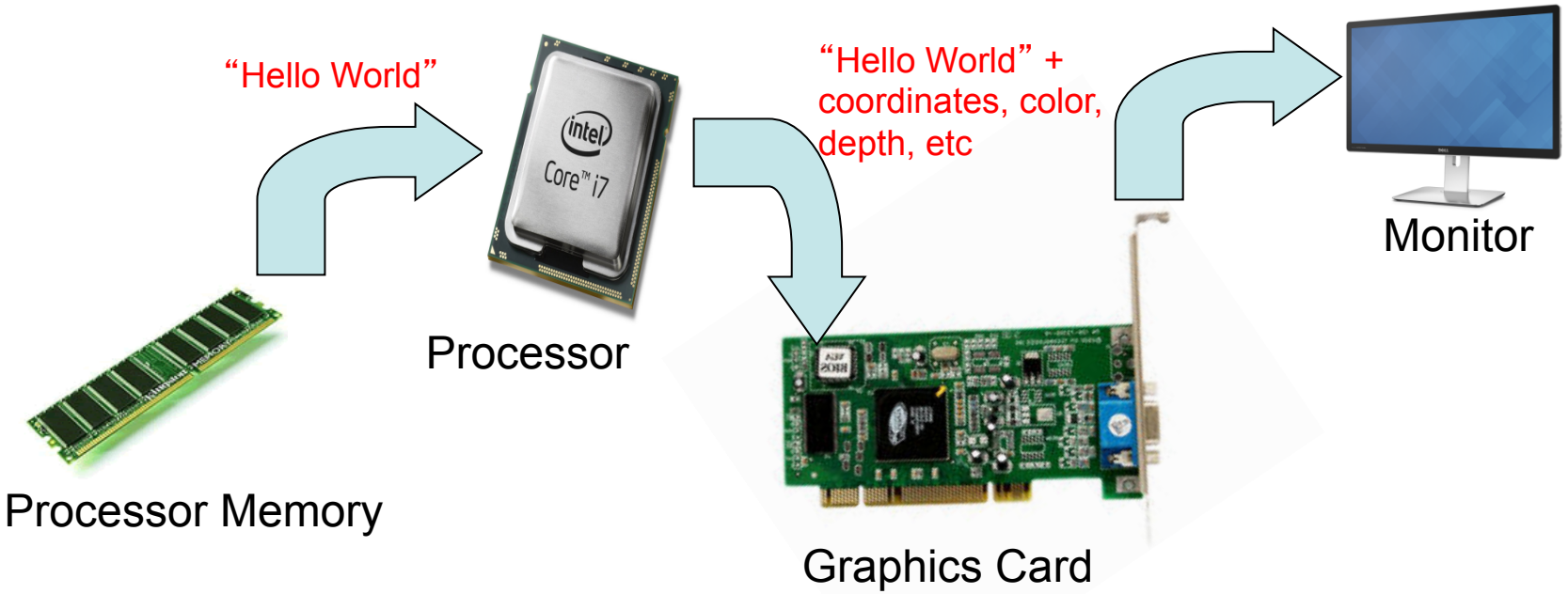
What is the output of the following program?

```c
#include <stdio.h>

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

How is the string displayed on the screen?

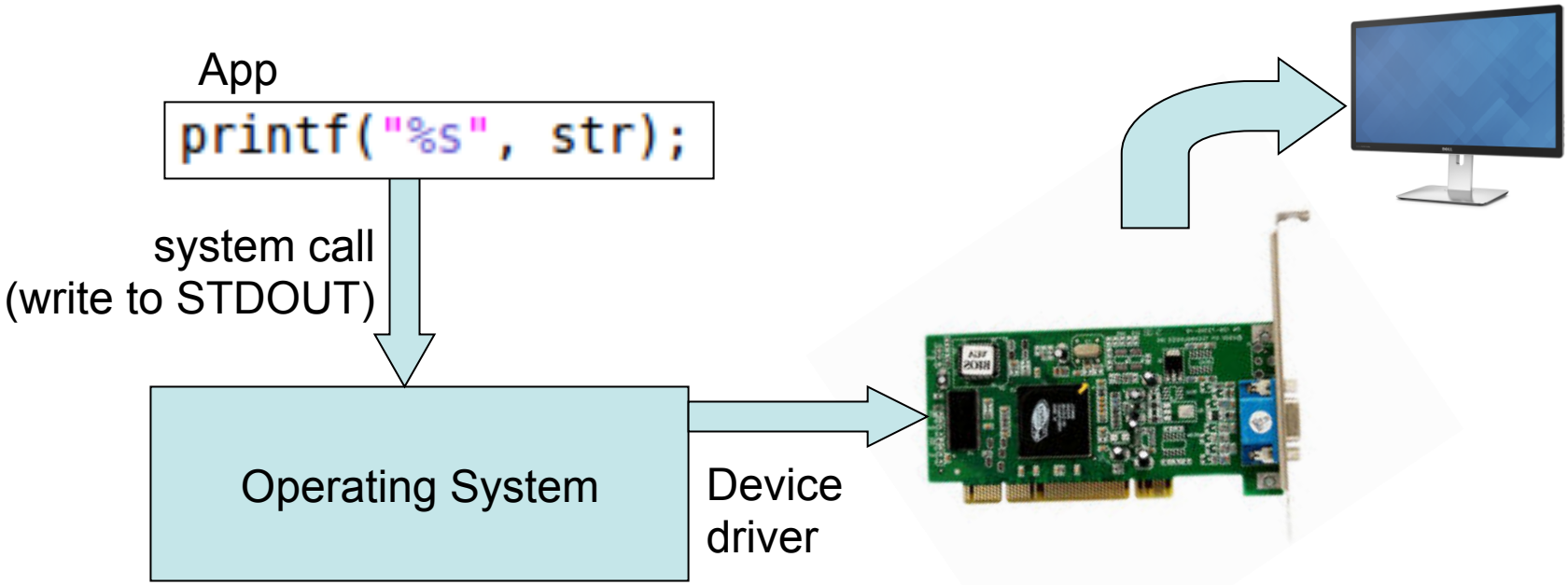
Displaying on the Screen



- Can be complex and tedious
- Hardware dependent

Without an OS, all programs need to take care of every nitty gritty detail

Operating Systems Provide Abstraction

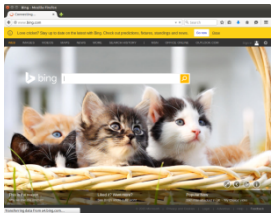


- **Easy to program** apps
  - No more nitty gritty details for programmers
- **Reusable functionality**
  - Apps can reuse the OS functionality
- **Portable**
  - OS interfaces are consistent. The app does not change when hardware changes

OS as a Resource Manager

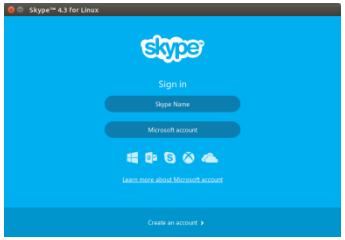
- Multiple apps but limited hardware

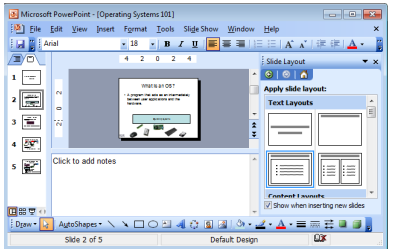
Apps



```
#include <stdio.h>

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





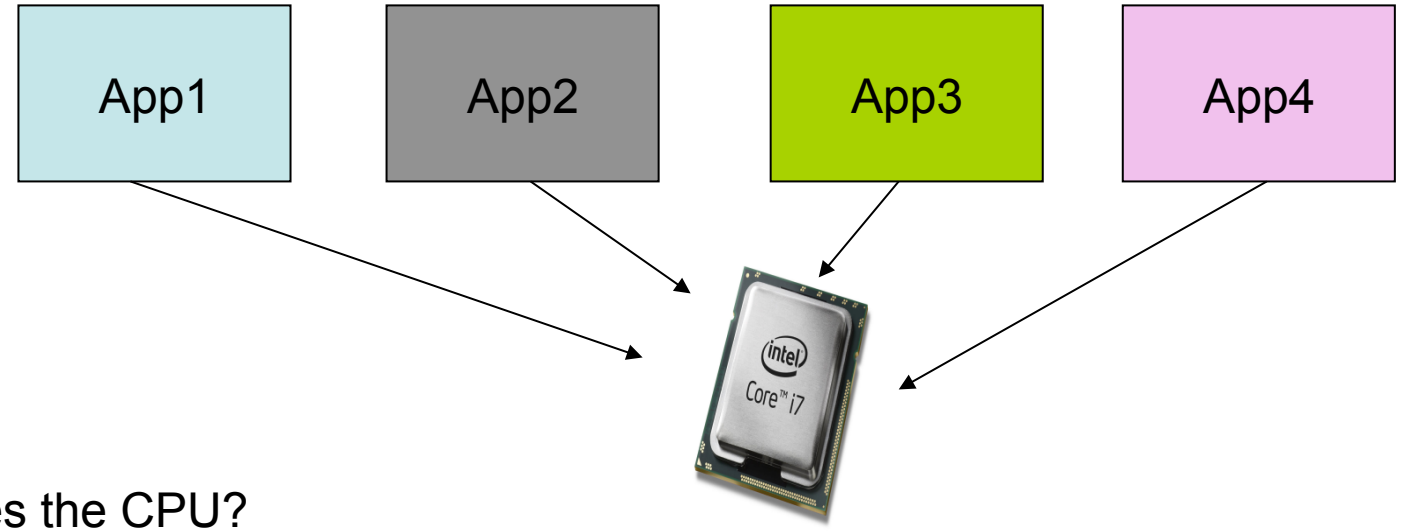
Operating Systems

A few processors

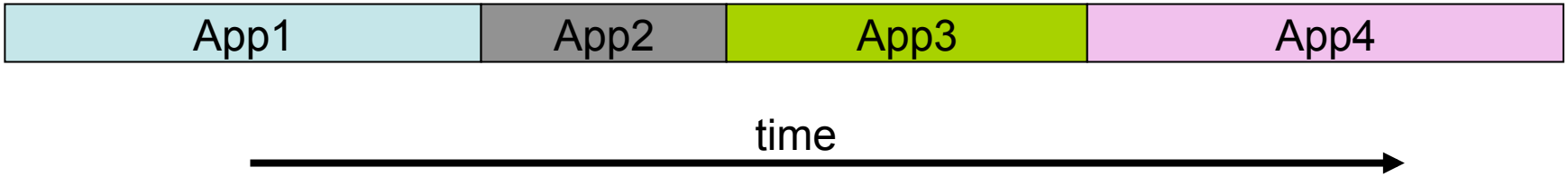
OS as Resource Manager

- OS must manage CPU, memory, network, disk etc…
- Resource management
  - allows multiple apps to share resources
  - protects apps from each other
  - Improves performance by efficient utilization of resources

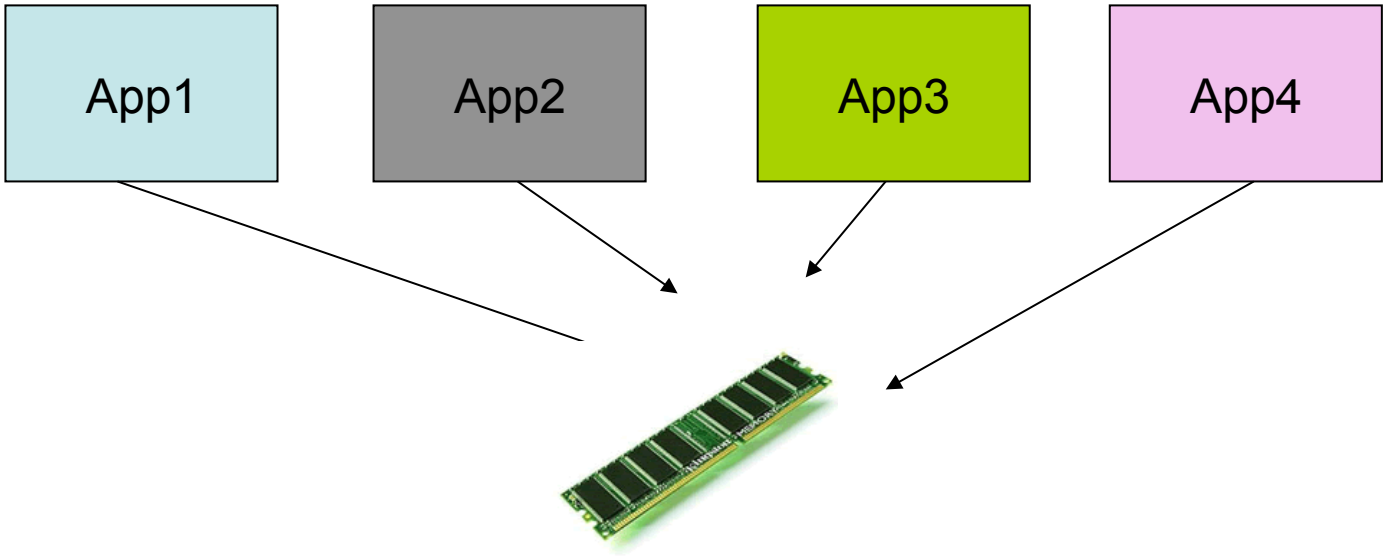
Sharing the CPU



Who uses the CPU?



Sharing the RAM



Operating Systems Types

- Application Specific
  - Embedded OS
    - eg. Contiki OS, for extremely memory constraint environments
  - Mobile OS
    - Android, iOS, Ubuntu Touch, Windows Touch
  - RTOS
    - QNX, VxWorks, RTLinux
  - Secure Environments
    - SeLinux, SeL4
  - For Servers
    - Redhat, Ubuntu, Windows Server
  - Desktops
    - Mac OS, Windows, Ubuntu

JOS and xv6

- Designed for pedagogical reasons
- Unix like (version 6)
  - Looks very similar to modern Linux operating systems
- Theory classes : xv6
  - Well documented, easy to understand
- Lab : JOS
  - Build your own operating system from the skeleton

L2_1 : Choose the OS?

- A company WishWash decides to make a washing machine with the following features.
  - It is a fully automatic washing machine, and has features such
    (1) smart wash (runs a learning algorithm to determine how long clothes must be washed),
    (2) Wifi connect
    (to see if your clothes are done via a mobile app),
    (3) emergency stop button
    (4) Lowest cost in the market
  - **What is the type of OS you would suggest WishWash use?**
    - **What are the devices the OS should abstract?**
    - **Are there any special features for the OS?**
    - **Would security features be needed in the OS. If Yes, then why?**
    - **What applications would run on the OS. Comment on the priorities of the applications.**

Course Structure

- Syllabus
  - Overview of Operating Systems
  - PC Hardware
  - Memory Management
  - Interrupts
  - Context Switching
  - Processes
  - Scheduling
  - Cooperating Processes
  - Synchronization
  - File Systems
  - Security

Textbooks / References

- **"xv6: a simple, Unix-like teaching operating system"**, Revision 8, by Russ Cox, Frans Kaashoek, Robert Morris
- **"Operating System Concepts"**, 8th edition, by Adraham Silberschatz, Pert B. Galvin, and Greg Gagne, Wiley-India edition
- The xv6 source code booklet (revision 8)

http://www.cse.iitm.ac.in/~chester/courses/16o_os/index.html

Logistics

- Theory Classes (CS24, Slot F)
  - Wednesdays : 11:00 - 11:50 AM
  - Thursdays : 9:00 - 9:50 AM
  - Fridays : 8:00 - 8:50 AM
- Lab (System's Lab, Slot P)
  - Monday's : 2:00 – 5:00 PMs

Exams

- Quiz 1 : 20%
- Quiz 2 : 20%
- Final : 40%
- Assignments / class assignments / group participation : 20%

Google Group

https://groups.google.com/forum/#!managemembers/osiitm_2016

Operating Systems

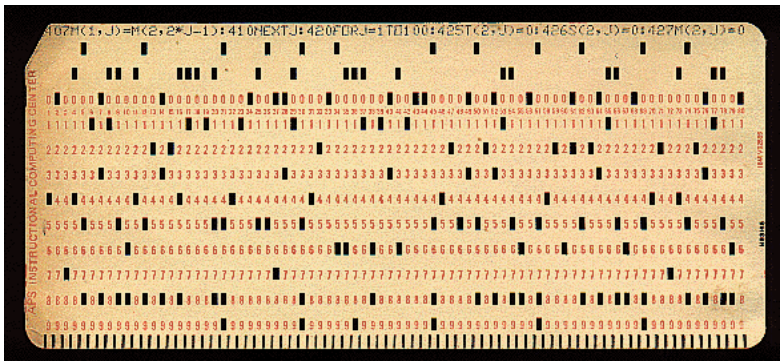
(How did it all start?)

OS Evolution

- **Evolution driven by Hardware improvements + User needs**
  - eg. low power requirements, Increased / reduced security, lower latency
  - Evolution by
    - New/better abstractions
    - New/better resource management

Gen 1: Vacuum Tubes

- Hardware
  - Vacuum tubes and IO with punchcards
  - Expensive and slow
- User Apps
  - Generally straightforward numeric computations done in machine language



IBM Punch card

ENIAC

Gen 1 : OS

- OS: Unheard of
- Human feeds program and prints output

George Ryckman, on IBM's first computer

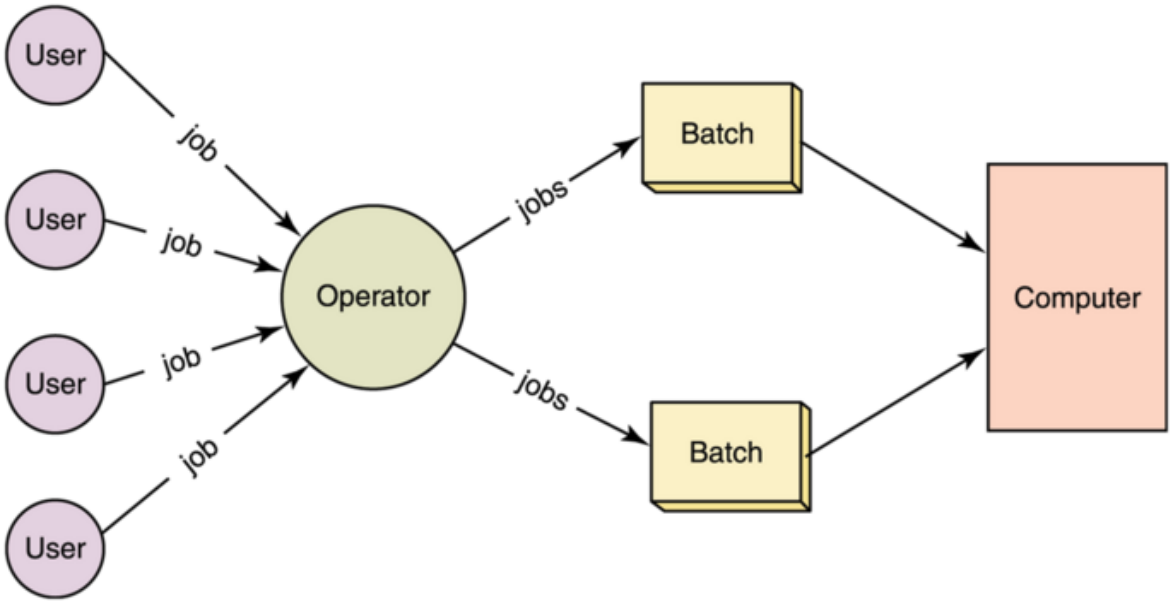
> Each user was allocated a minimum 15-minute slot, of which time he usually spent 10 minutes in setting up the equipment to do his computation ... By the time he got his calculation going, he may have had only 5 minutes or less of actual computation completed—wasting two thirds of his time slot.

The cost of wastage was $146,000 per month (in 1954 US Dollars)

Gen 2 : Mainframes

- Hardware
  - transistors
- User Programs
  - Assembly or Fortran entered using punch cards
- OS : Batch systems
  - Possibly greatest invention in OS
    - Computers may be able to schedule their own workload by means of software

Batch Systems



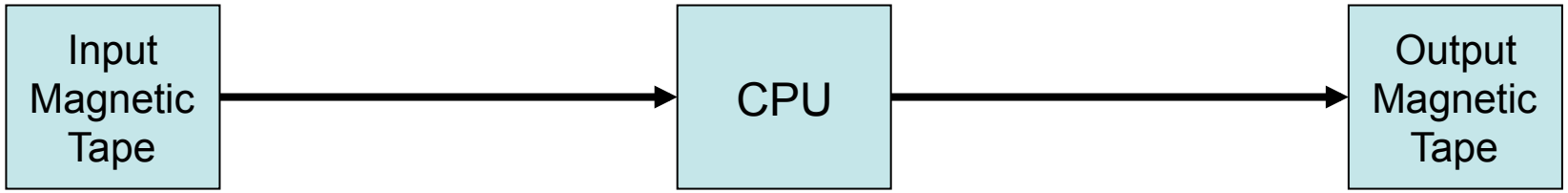
- Operator collects jobs (through punch cards) and feeds it into a magnetic tape drive
- Special Program reads a job from input tape drive and on completion writes result to output tape drive
- The next program is then read and executed
- Printing was done offline

Batch Systems (pros.)

- Pros
  - Better utilization of machine

Batch Systems (cons.)

- In Batch Systems execute time includes reading from input and writing to output.
- I/O considerably slower than execution
  - Magnetic tapes were best read sequentially
- Therefore programmer must wait for long time

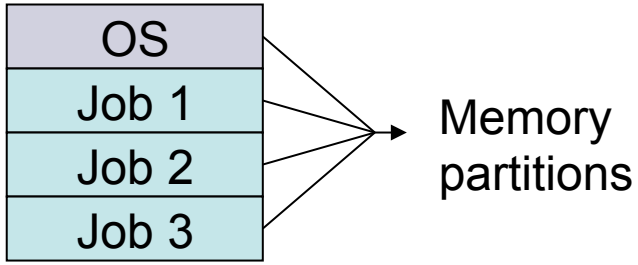


Gen 3 : Mini computers

- Hardware
  - SSI/MSI/LSI ICs
  - Random access memories
  - Interrupts (used to simulate concurrent execution)
- User Programs
  - High level languages (Fotran, COBOL, C, …)
- Operating Systems
  - Multiprogramming
  - Spooling
  - Time sharing

Multiprogramming

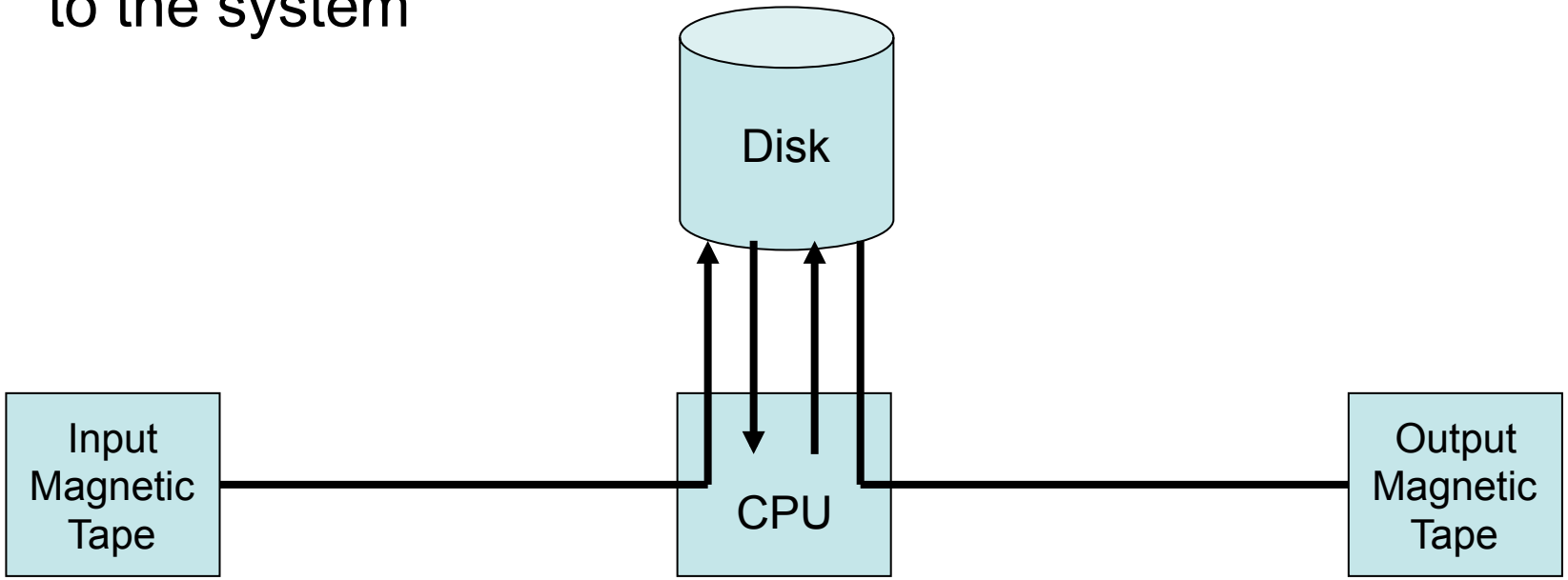
- Multiple jobs in memory
  - When one waits for I/O the next job executes
- OS controls
  - scheduling of jobs
  - Protection between jobs



Multiprogramming with
3 jobs in memory

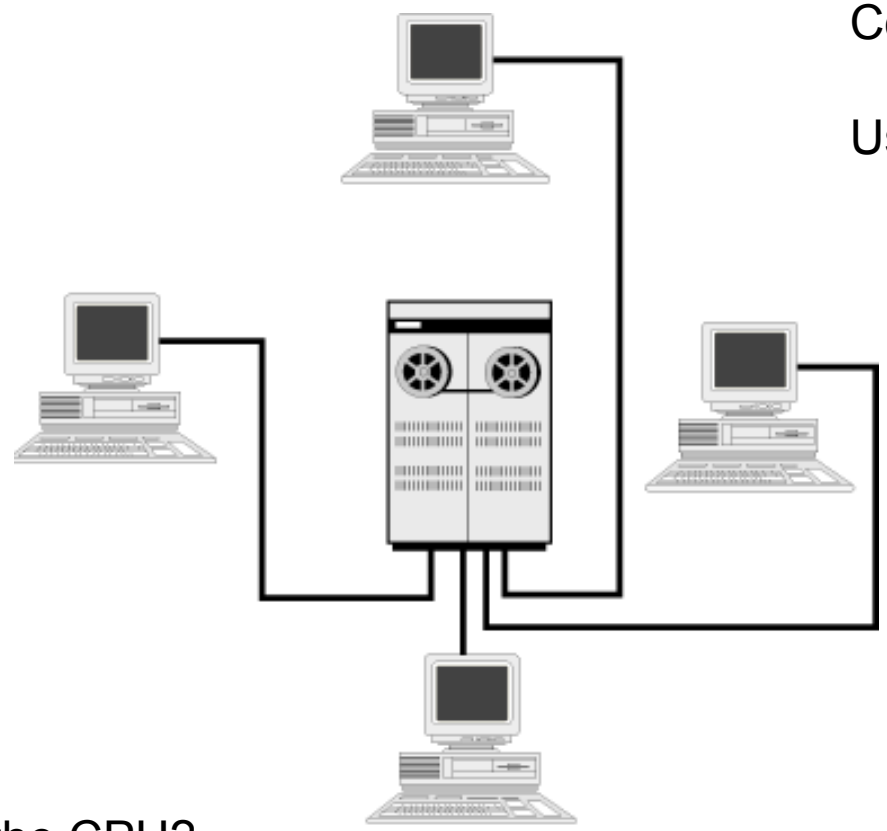
Spooling

- Uses buffers to continuously stream inputs and outputs to the system



- Pros : better utilization / throughput
- Cons : still not interactive

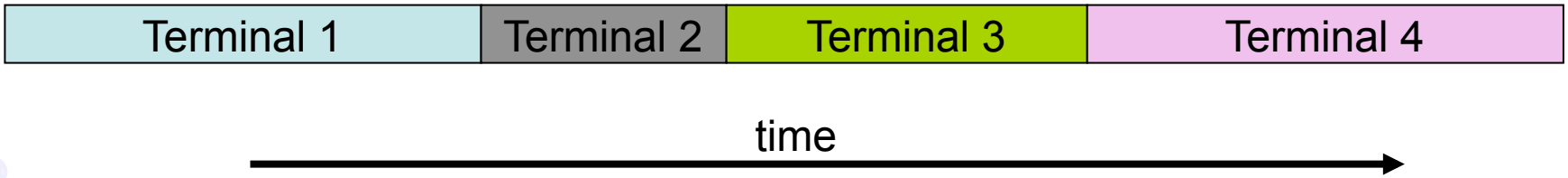
Timesharing



Computers much faster than terminals

Uses interrupts

Who uses the CPU?



Timesharing

John McCarthy, 1962

By a time-sharing computer system I shall mean one that interacts with many simultaneous users through a number of remote consoles. Such a system will look to each user like a large private computer...When the user wants service, he simply starts typing in a message requesting the service. The computer is always ready to pay attention to any key that he may strike.

Because programs may ... do only relatively short pieces of work between human interactions, it is uneconomical to have to shuttle them back and forth continually to and from secondary storage. Therefore, there is a requirement for a large primary memory ... The final requirement is for secondary storage large enough to maintain the users' files so that users need not have separate card or tape input-output units.

Multics, 1964

- Multiplexed Information and Computing Service
- Ambitious project started in MIT
- Introduced several new OS features but was not successful by itself
  - Segmented and Virtual memory
  - High level language support
  - Multi language support
  - **Security**
  - File system hierarchies
  - Relational databases
  - Shared memory multiprocessor

Gen 4 : Personal Computers

- Hardware
  - VLSI ICs
- User Programs
  - High level languages
- Operating Systems
  - Multi tasking
  - More complex memory management and scheduling
  - Synchronization
  - Examples : Windows, Linux, etc

Unix

- Dennis Ritchie and Ken Thomson tried to find an alternative for Multics
- Appeared at the right time

Slater, 1987

> The advent of the smaller computers, the minis—especially the PDP-11—had spawned a whole new group of computer users who were disappointed with existing operating software. They were ready for Unix. Most of Unix was not new, but rather what Ritchie calls "a good engineering application of ideas that had been around in some form and [were now] made convenient to use."

Unix adopted

- Spread and soon became widely adopted

Aho, 1984

> In the commercial world there are 100,000 Unix systems in operation ... Virtually every major university throughout the world now uses the Unix system.

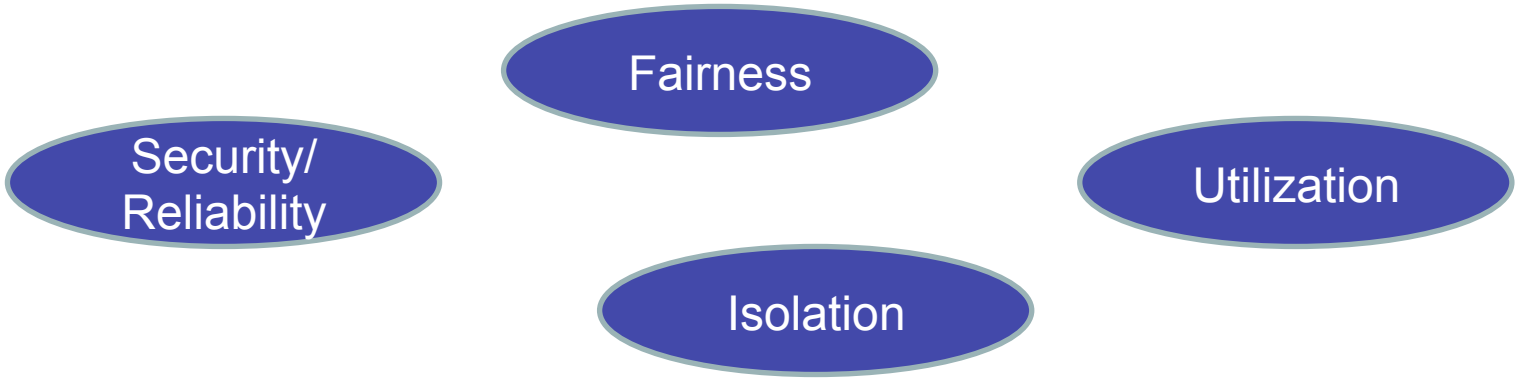
- Wide spread adoption arguably a hindrance to research?

Smartphones & Tablets

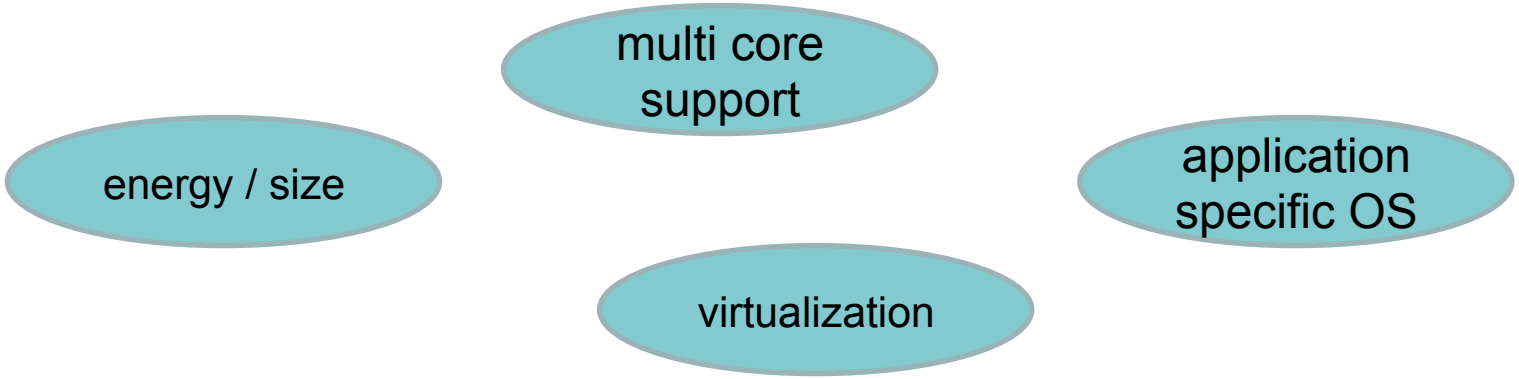
- Hardware
  - VLSI ICs, low power requirements & high compute power
- Operating Systems
  - User friendly
  - Power awareness
  - Always connected
  - Offload to cloud
  - Better protection, Virtual machines
  - Examples : Android, iOS

OS Buzzwords

- Buzzwords that have been around



- Contemporary buzzwords



OS Research Trends