CS341: Operating System

Introduction to Operating System

Lect02 : 31th July 2014

Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

1

Outline

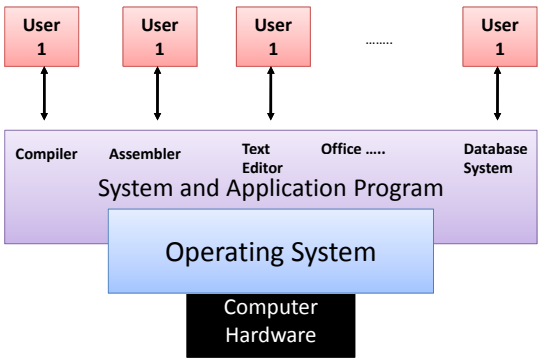
- What are the basic organization of computing systems?
- What is *"Operating System"* ? What an OS do? OS types?
- What do we study in this course and why to study ?
  - Motivation and Objective
- What are the major components of operating systems?
- What are the types of computing environments?
- Is there possibility of exploring open-source operating systems?
- How is the course structured?

2

Computer System Structure: Four Components

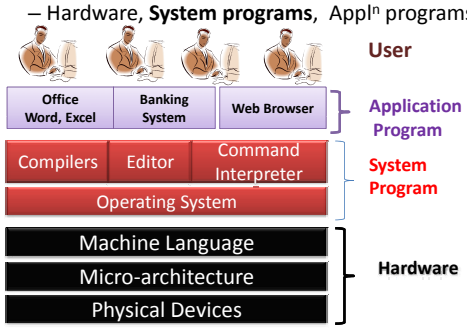
- Hardware : Provides basic computing resources
  - CPU, memory, I/O devices
- Operating system:
  - Controls and coordinates use of hardware among various applications and users
- Applⁿ programs
  - Define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, ***compilers, assembler,*** web browsers, database systems, video games
- Users : People, machines, other computers

Four Components of a Computer System



Introduction to Computer System

- A computer system consists of
  - Hardware, **System programs**, Applⁿ programs, Users



5

OS : Laymen Definition

An operating system manages all:

- **Input**
  - getting information into the computer from an external sources
  - keyboard, a mouse, a scanner, or a disk.
- **Processing**
  - After receiving input: manipulates or alters the data
- **Output**
  - Once the input has been processed
  - Result output to a monitor, printer, disk or sent via email or the Web

6

OS: Another Definition

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

7

OS Examples

- **PC OS**
  - Microsoft : Window 95, 98, XP, Vista, 7, 8, NT
  - Apple : Machintos, IBM : OS 2, OS 360/390
  - Unix, Linux, Ubuntu, Fedora, BSD Unix, Solaris
- **Embedded OS**
  - Android, iOS, Window CE/Phone 8.1, Bada OS, QNX, MeeGo, BlackBarry, uLinux, TinyOS
- **Web Browser OS :** Crome OS, EyeOS, YouOS
- **Router OS:** CSIR ONET, Netware, Cisco IOS, SAN-OS

8

OS Types

- **Mainframe OS, Server OS**
- **Multiprocessor operating systems**
- **Personal computer operating systems**
- **Real-time operating systems**
  - **Air craft, Radar Detection, Naval/Space Machine**
- **Embedded operating systems**
  - **Mobile, Printer, Scanner, Projector, Camera, Washing Machine**
- **Smart card operating systems**
  - **Ecos, TinyOS, SensorOS**

9

What Operating Systems Do

- Depends on the point of view
- PC
  - Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- Mainframe
  - Shared computer to keep all users happy and make proper and fair use of resource

What Operating Systems Do

- Users of dedicate systems: **workstations**
  - Have dedicated resources but frequently use shared resources from **servers**
- Handheld computers (Mobile, Laptop,..)
  - Resource poor, optimized for usability and battery life
- Embedded Computer
  - Some computers have little or no user interface
  - Embedded computers in devices and automobiles

What is an Operating System

- **OS is an extended machine**
  - Hides the messy details which must be performed
  - Presents user with a virtual machine, easier to use
- **OS is a resource manager**
  - Each program gets time with the resource
  - Each program gets space on the resource

12

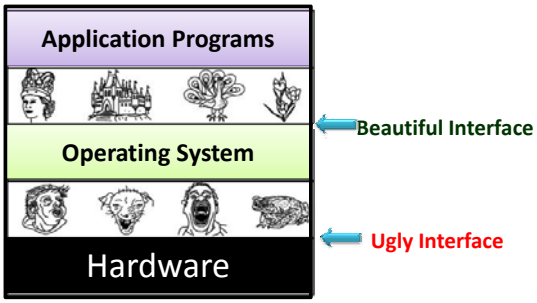
What is an Operating System

- 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

13

OS as an Extended Machine

Operating systems turn ugly hardware into beautiful abstractions.



OS as a Resource Manager

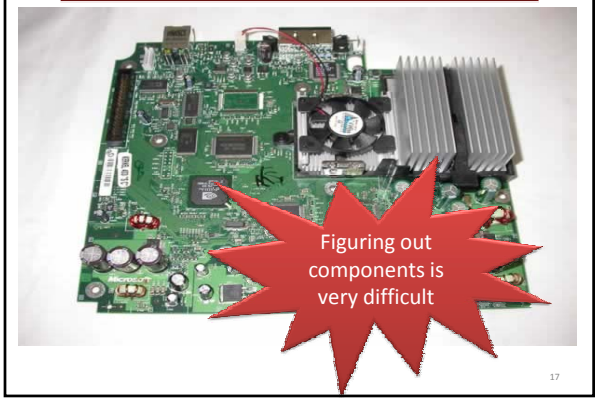
- Allow multiple programs to run at the same time
- Manage and protect memory, I/O devices, and other resources
- Includes multiplexing (sharing) resources in two different ways:
  - In time
  - In space

OS as Abstraction

- Delving into the depths reveals more information
- An abstraction omits unneeded detail, helps us cope with complexity

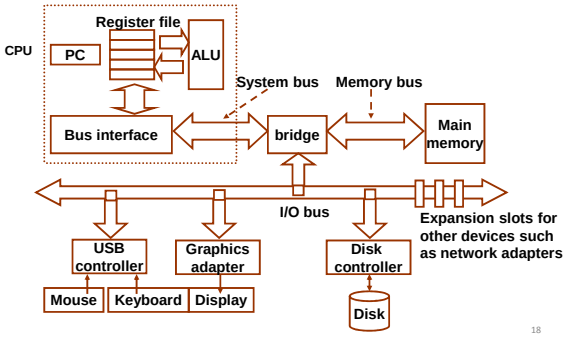
16

Example: Hardware Abstraction



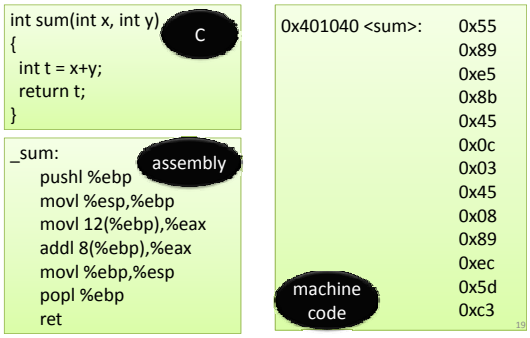
17

Example: Hardware Abstraction



18

Example: Software Abstraction



19

Example: OS level Abstraction

- Manage physical resources
- Provide virtual resources
- Device driver takes cares of all the internal details
- Writing to HDD disk, SDD, Network Buffer
  - Provide the path, use same API every where
  - Open, Read , Write

20

Example: OS level Abstraction

- A well-designed interface is beautiful thing
  - It is even more beautiful to fully appreciate to transform a nasty, low-level interface
  - Ability to put a collection of useful abstractions
  - Sockets, file systems, and address spaces into a single convenient package
- **Probably one of the best contributions of computer science.**

21

Operating System

- Bridge the gap between hardware and software
- Establish a foundation for building higher-level programs
  - How to optimize programs?
  - How to debug large systems
  - How to deal with complexity

22

Operating System

- Provides a virtual execution environment on top of hardware
  - That is more convenient than the raw hardware interface
- "All of the code you did not write"
  - **Printf/scanf/open/close/read/write library are system specific…no one write..we simply use**
- More Simple, More reliable, More secure, More portable, More efficient……

23

Operating System Definition

- **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.

OS : When you turn on Computer, it Startup

- **When you turn on a PC, it should give you a place/environment to work**
- **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

What do OSes do?

- Manage physical resources
- Provide virtual resources
- Implement mechanisms and enforce policies for the control and use of resources
- Mediate the interaction of mutually distrusting applications

26

What Physical Resources Do OSes Control?

- CPU, Memory
- Storage Devices, Networks
- Input Devices (keyboard, mice, cameras)
- Output Devices (printers, displays, speakers)
- And many virtual resources

27

Issues In OS Design

- **Structure:** how is an OS organized?
- **Concurrency**: how are parallel activities created and controlled?
  - All hardware are run in parallel CPU, HDD, N/W
  - User want many application to run in parallel…
- **Sharing**: how are resources shared?
- **Naming**: how are resources named by users?

28

Issues In OS Design

- **Protection:** how are distrusting parties protected from each other?
- **Security:** how to authenticate, authorize and ensure privacy?
- **Performance:** why is it so slow?

29

More OS Issues

- **Reliability:** how do we deal with failures?
- **Extensibility:** how do we add new features?
- **Communication:** how do we exchange information?
- **Scale:** what happens as demands increase?
- **Persistence:** how do we make information outlast the processes that created it?
  - Don't start a game every time from the start..
- **Accounting:** who pays the bills and how do we control resource usage?

30

Why different Operating System?

- What is the difference between OSes desktops, laptops, mobile phones, washing machines etc.?
  - Performance /speed, Power consumption, Cost
  - General purpose /special purpose
  - Domain Specific: Network, DSP, Image. Crypto

31

Why different Operating System?

- **Different OS optimization parameter lead to different problem hence the different algorithm/heuristics**
- Is Android OS is same as Ubuntu ? === > No
  - **Power and memory consumption is very crucial in Phone**
  - **Battery level input need to monitor**

32

Why to Learn Operating Systems?

- Provides an understanding from the bottom up
- Even if few people build OSes, understanding how OSes work is crucial for building working systems
- Who knows..
  - You got places in Microsoft, Google, HP, Amazon..
  - And need to work in OS, device drive, multicore….

33

Why to Learn Operating Systems?

- This course will go far beyond OS design to cover all aspects of computer system, including
  - **Concurrency,**
  - **Synchronization,**
  - **Input/output, filesystems,**
  - **Networking, routing,**
  - **Distributed systems and** so forth
- Engineering pride alone requires full understanding

34

Fact

- There has never been as exciting a time to work on systems hardware and software as now!!!
- The world is increasingly dependent on computer systems
  - **Connected, networked, interlinked**
- People just **do not know how to build** good systems…
  - Still evolving…..**heuristics and heuristics**….

35

CS431 Course Objectives

To learn –

- Theory behind resource management
- Theory behind concurrency and related issues
- How to adopt to new hardware resources and standardization
- Issues affecting modern operating systems (Android, VxWork, Distributed OS, Cloud OS, Cluster OS)

36

CS431 Course Objectives

To learn –

- Advanced Power Management Support
- Multi-core support in OS
- How to ***device driver*** and how all the interface works
- How the system call works..

37

OS Motivation: Resource Management

- At the hardware level resources are typically dedicated.
- An OS provides versions of these resources that are
  - Either virtualized (each user gets the illusion of having a copy of the resource)
  - Or arbitrated (one user at a time, but with queuing handled by the OS).

38

OS Motivation: Resource Management

- Strategies used to give multiple users access to a dedicated physical resource
  - Also used in many user-level programs.
- By studying these issues explicitly
  - You will learn patterns that can be reused in many other contexts.

39

OS Motivation: Concurrency

- Writing concurrent code is not easy
  - Especially using threads with shared memory and locks.
  - A large number of current CS students will do this at some point in their careers.
- There is a growing trend to address concurrency
  - Introduction to threads, races, deadlocks

40

OS Motivation: Concurrency

- The material is hard
  - **Actually the material is easy, but applying it is hard**
  - **Useful to see it several times before graduating.**
- A solid introduction to concurrent programming
  - A major benefit of taking an OS course

41

OS Motivation: Performance Analysis and Contention Resolution

- When resources are shared, contention typically follows.
- Contention can be resolved in many ways
  - Using queuing, fair sharing, or prioritization.
- In some cases, such as CPU scheduling
  - No single technique suffices and the best known solutions are bizarre hybrids.

42

OS Motivation: Performance Analysis and Contention Resolution

- Often, the most interesting problem
  - Figuring out what kind of contention is the root cause of some observable problem.
- An OS is a perfect context for learning these ideas, whose applicability is much broader than computer science
- ***Critical Section, Deadlock and Locking***
  - ***Theoretical Analysis and Proof of Algorithms***

43

OS Motivation: Interfaces and Hiding Complexity

- A well-designed interface is beautiful thing
  - It is even more beautiful to fully appreciate to transform a nasty, low-level interface
  - Ability to put a collection of useful abstractions
  - Sockets, file systems, and address spaces into a single convenient package
  - **Probably one of the best contributions of computer science.**
- This is so common place that it's easy to overlook the real awesomeness.

44

OS Motivation: No Magic Here

- It's easy to view the OS as a magical force
  - **Good : Giving us smooth multitasking, efficient storage management, etc**
  - **Evil : giving us blue screens, thrashing, security problems, and scheduling anomalies.**
- Public and You
  - This view is fine for the general public.
  - On the other hand, if you plan to tell people you're a computer scientist, you need to have peeked behind this curtain.

45

OS Motivation: No Magic Here

- What will you find there?
- Too often it seems like sort of a sad collection of
  - Dull linked lists,
  - Dodgy resource heuristics,
  - And ill-maintained device drivers.

46

OS Motivation: No Magic Here

- A good OS course should show students ➔
  - There is great code in the kernel
  - You just have to know where to look.
- When you first see it
  - You will not understand
  - But when you understand a bit, feel easier……
- Mostly, kernel code is perfectly earthly world
  - Anyone can write it,
- A bit more care and attention to detail than
  - User-mode code because the consequences of a bug are greater.

47

OS Motivation: Dealing with Big Software

- OS code : multi-million line codes and base is a nightmare
  - Documentation is incorrect and scattered
  - Interfaces are clunky (makes noises), wide
  - Interactions are subtle (so precise as to be difficult to analyze),
  - And error messages inscrutable (impossible to understand or interpret)
- But welcome to real life
  - We can't usually just start over due to problems like these.

48

OS Motivation: Dealing with Big Software

- As a student: if you can begin to develop
  - A systematic approach to learning the relevant parts of a big piece of software that your code has to fit in
  - Your life will be a lot easier later on.
- You can hate the Linux kernel
  - But it's far better software than other stuff you'll run into during your career.

49

OS Motivation: System Design

- Designing any engineered system, including a software system
  - Is an exercise in compromise.
- How much emphasis is placed on
  - Reliability? Performance? Cost? Maintainability?
- Since operating systems are large
  - Performance-critical programs that tend to last for decades
  - They are a great place to learn about these kinds of tradeoffs.

50

OS Motivation: System Design

- Students who develop a sharp eye
  - For finding an appropriate design point are incredibly useful in industry
- This stuff is more of an art than a science
  - You need to look at a lot of code
  - Understand the issues,
  - And learn to think about this stuff for yourself.

51

Course Structure
1st Half (Algorithmic Prospects)

- Process Management
  - Process and thread, scheduling examples
  - ***Scheduling Algorithms: Theoretical prospects***
- Concurrency
  - Mutual exclusion, synch., semaphores, deadlocks
  - Atomic Instructions, ***design and proof of Synchronization algorithms and polices***
- Memory Management
  - Allocation, protection, hardware support, paging, segmentation, virtual memory, demand paging, allocation, replacement, TLBs
  - ***Algorithmic treatment: Memory management***

52

Course Structure
2nd Half (System Prospective)

- File Management
  - Naming, file operations and their implementation;
- File Systems
  - Allocation, free space management, directory management, mounting;
  - ***Distributed File System***
- I/O Management
  - Device drivers, disk scheduling
  - **Linux Device Driver & Kernel Programming**

53