CS3.304: Advanced Operating Systems (L-T-P: 3-1-1. Credits: 4)

P. Krishna Reddy

E-mail: pkreddy@iiit.ac.in

http://www.iiit.ac.in/~pkreddy

Office: Data Sciences and Analytics Center (DSAC)

Courses You Have Completed

- Digital logic design
  - Build an integrated circuit for a given function.
- Computer Architecture
  - Develop an assembly language program given a problem
- Programming language
  - Develop a high level program to solve a given problem.
    - printf()
    - scanf()
    - fopen()
    - …..

Outline

- **Introduction**
  - **What is an Operating System ?**
- **Coping with the complexity**
- Course topics and grading
- History, development and concepts of Oss
- Different kinds of Computer Systems
- Concept of virtual computer

Questions

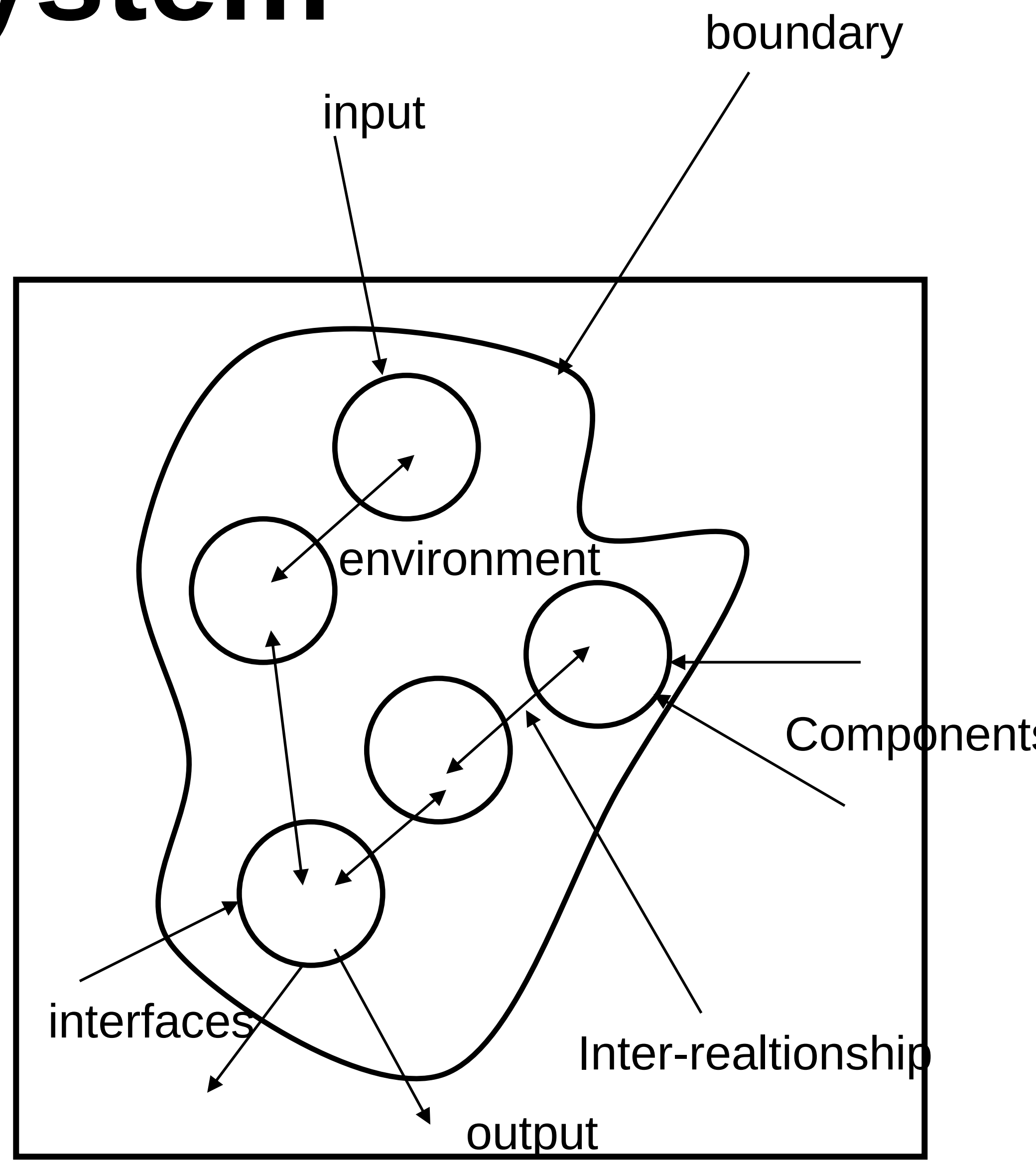
- What is a system ?
- What is an operating system ?
- What is a computer operating system ?

What is a system ?

- A system is an inter-related set of components with an identifiable boundary working together for some purpose.
- System can be natural or fabricated
  - Natural systems: human body or solar system
  - Fabricated systems: cycle, bus, computer, government, boat

System

- A system has nine characteristics
  - Components
  - Inter-related components
  - A boundary
  - A purpose
  - An environment
  - Interfaces
  - Input
  - Output
  - Constraints.



A general depiction of a system

Characteristics…

- Components:
  - A system is made up of components
  - A component is either an irreducible part or aggregation of parts that make-up a system. A component is also called a sub-system.
- Interrelated:
  - The components of interrelated
  - Dependence of one subsystem on one or more subsystems.

Characteristics…

- Boundary (Scope):
  - A system has a boundary, within which all of its components are contained and which establishes the limits of a separating the system from other systems.
- Purpose
  - The overall goal of function of a system. The system’s reason for existing.

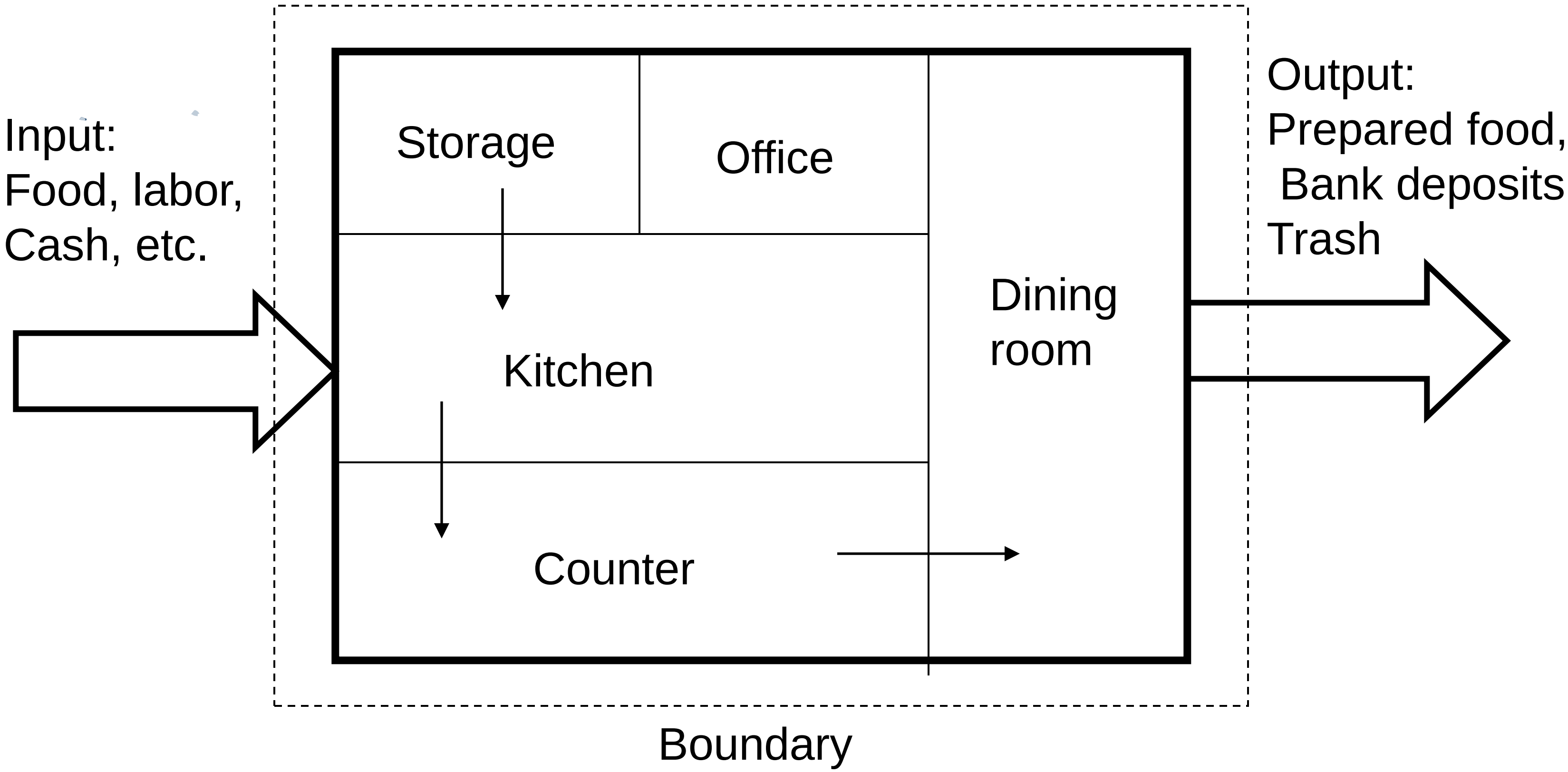
Characteristics…

- Environment
  - Everything external to the system that interacts with the system.
- Interface
  - Point of contact where a system meets its environment or subsystems meet each other.
- Constraint:
  - A limit what a system can accomplish: Capacity, speed or capabilities.

Characteristics…

- Input
  - Whatever a system takes from its environment in order to fulfill its purpose.
- Output:
  - Whatever a system returns to its environment in order to fulfill its purpose.

Example: A fast food restaurant as a system



Environment: Customers, food distributors, banks etc

→ Represents an inter-relationship

Constraints: Popular foods, Health dept., constraints of storage

What is an operating system ?

- Operating system is a system.
- Operating system is a subsystem of any tool.
- Each tool constitutes machine part and operating part.
- **The operating part of a tool is called as operating system of that tool.**
- The purpose of operating system is to facilitate the operation of the underlying machine or tool.
- For some tools, operating system may not exist.
  - Example: Pen.
- For a user, the operating system abstracts the machine part in terms of simple services by hiding the details of the machine. The OS can provide services to users or other subsystems.
- Examples of typical operating systems:
  - Car operating system, Telephone operating system, TV operating system and so on.

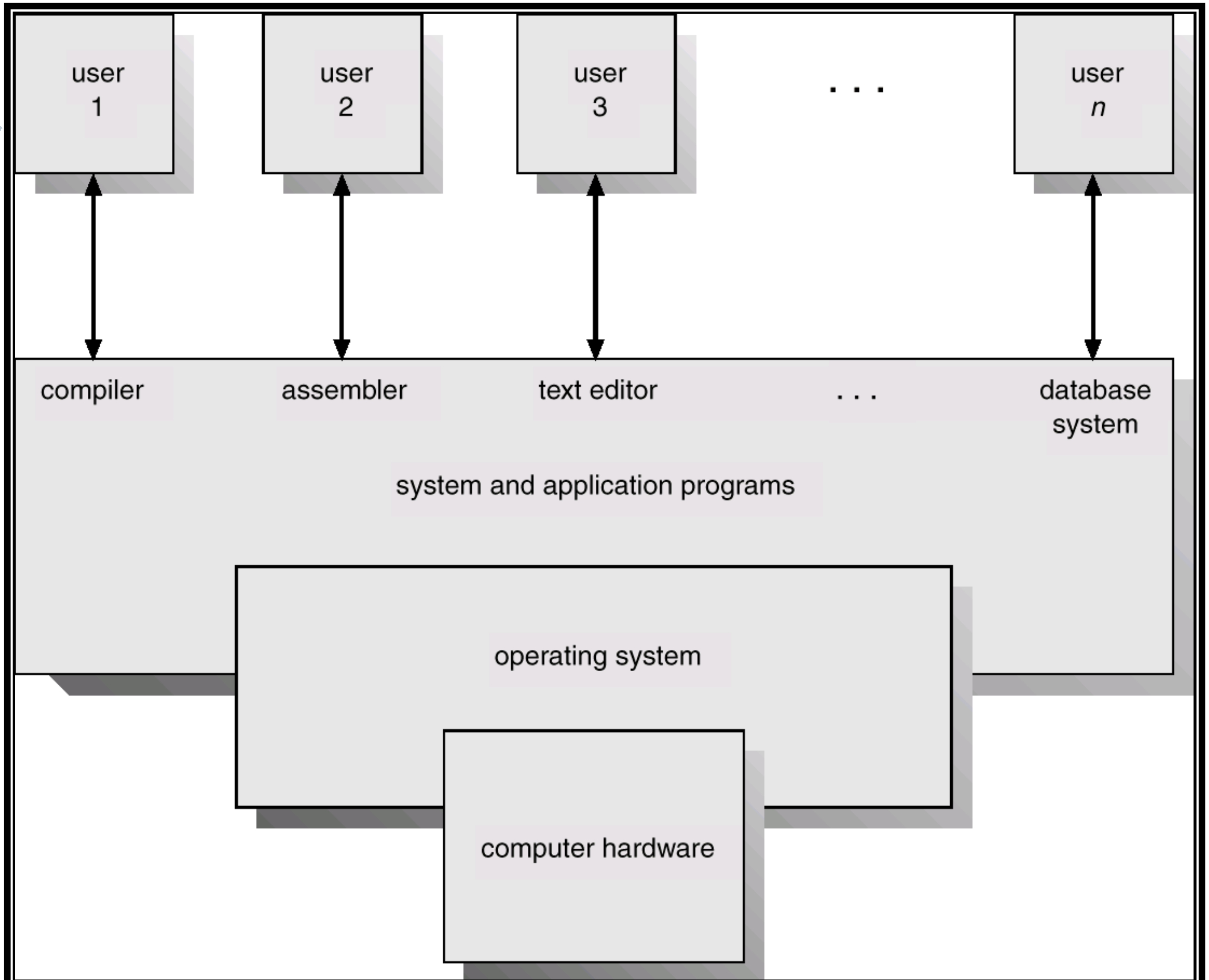
What is a computer operating system ?

- A computer is also a tool that contains the machine part and the operating part.
- The operating part of a computer is called Computer Operating System.
  - The operating system abstracts the underlying hardware in terms of simple services by hiding the details of the hardware. The OS can provide services to users or other subsystems.
- Examples of Computer operating systems:
  - WINDOWS 10, Macintosh, UNIX, SOLARIS, LINUX, Android, MAC IOS and so on.
- In the rest of this course, operating system means computer operating system.

Computer System Components

1. **Hardware** – provides basic computing resources (CPU, memory, I/O devices).
2. **Operating system** – controls and coordinates the use of the hardware among the various application programs for the various users.
3. **Applications programs** – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs), Internet.
4. **Users** (people, machines, other computers).

Abstract View of System Components



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:
  - Make the computer system convenient to use.
  - Use the computer hardware in an efficient manner.

Operating System Definitions…

- **Resource allocator** – manages and allocates resources.
  - Resources: CPU time, Memory Space, file storage space, I/O devices and son on.
- **Control program** – controls the execution of user programs and operations of I/O devices .
- **Kernel** – the one program running at all times (all else being application programs).
- The two goals, efficiency and convenience are sometimes contradictory
- Much of OS theory is concentrates on optimal use of resources.

Sources of Complexity

- “Complex” means “difficult to understand”
- Five signs of complexity
  - Large number of components
    - More component means more complex
  - Large number of inter-connections
    - More interconnections means more complex
  - Many irregularities
    - Lack of regularity, non—repetitive interconnections, exceptions means more complex.
  - A long description
    - The complexity of the computational objects is the length of description.
  - A team of designers, implementers or maintainers
    - If more people are required to maintain the system means more complex

Sources of Complexity: Cascading and interacting Requirements

- More requirements, more complexity
- As they increase, the interactions increases
- Example: telephone
  - Call waiting, call return, call forwarding, call blocking, reverse billing, caller ID, Caller ID blocking, anonymous call rejection, do not disturb and so on

- Principle of escalating complexity
  - Adding a requirement increases complexity out of proportion

Sources of Complexity: Generality

- Generality:
  - Meeting many requirements with single design
  - Applying to variety of circumstances
- Use good judgement how much generality is required
  - Automobile with four independent steering wheels
    - Increases complexity
  - Designing a vehicle one can drive on the high way and use as a boat
    - May not be efficient
- Avoid excessive generality
  - If it is good for every thing, it is good for nothing.

Sources of Complexity: Scaling

- Scaling increases complexity
  - Example 1
    - Small insects absorb oxygen through their skins
    - Large organisms require skins proportion to the cube of their size
      - Addition of lungs and blood vessels
  - Example 2
    - Programmer of four bit microprocessor can write a code to control a toaster in binary language
    - Programmer of video game on 64-bit microprocessor require computers, operating system compiler, video editors, special effect generators and so on

Sources of Complexity: Scaling…

- When environment changes, requirements change
  - Knowledge of how to maintain old equipment may be disappearing
- Original design may lose its relevance.
- Unanticipated requirements
  - Changes to existing system increases propagation of effects
  - A bug introduces another bug
- Bottom-line
  - As systems age, they tend to accumulates more changes and make them more complex.

Sources of Complexity:
Maintaining high utilization

- Desire is to get high performance or high efficiency
- Utilization should be high for scarce resource
- If we try to increase the utilization of the limited resource, complexity increases.

Coping with complexity I

- Modularity
- Abstraction
- Layering
- Hierarchy

Coping with complexity I
Modularity

- Divide-and-conquer technique
  - Analyse or design the system as a collection of interacting subsystems, called modules
  - One can think about each module in an isolated manner.
- N statements; N bugs
- Time to fix a bug is proportional to program.
  - Debugging time =N*N
- Dividing into K modules
  - (N/K)*(N/K)*K
  - (Approx.) (N/K)*(N/K)
- Reduces the debugging time by a factor of K.
- The property of modularity is universal.

Modularity..

- Allows incremental improvement
  - Easy to replace inferior module with an improved one.
- It helps to address the complexity caused due to change.

Coping with complexity I
Abstraction

- The discovery of bug should lead to examining only one module
- There should be little or no propagation effects from one module to another.
  - Several ways to modularize the system
  - Some ways prove better than others.
- Abstraction
  - The ability of any module to treat all others entirely on the basis of external specifications, without need for knowledge about what goes inside.
  - Examples:
    - Finding a compatible DVD player for television set
    - Overnight pocket delivery service
- Well designed and properly enforced modular abstractions are essentially important in limiting the impact of faults and they control the propagation of effects

Coping with complexity I: Robustness principle

- Be tolerant of inputs and strict on outputs
  - Key idea of modern mass production
- The robustness principle plays major role in computer systems.
  - Human interfaces, network protocols, and fault tolerance
- The Safety margin principle
  - Inputs should be within the range
  - Keep track of input ranges and report out of tolerance ranges.

Coping with complexity I: Layering

- Good abstraction minimizes number of interconnections
- Way to reduce interconnections is the method of module organization called "layering"
- Mechanism
  - Set of mechanism to use already complete (lower layer) to create a different complete set of mechanisms. (upper layer)
  - A layer may itself be implemented as several modules
  - A layer only interacts with only peer modules and modules of lower layer and next higher layer.

Coping with complexity I
Hierarchy

- (1) Start with a small group of modules and assemble them into a stable, self-contained subsystem that has a well-defined interface.
- (2) Assemble a small group of subsystems to produce a larger subsystem. Go to (1).
- The result is tree structure known as a hierarchy.
- Hierarchy reduces number of interconnections.

Summary

- Modularity, abstraction, layering and hierarchy provide ways to divide the things up and placing the relevant module in suitable relation one to another.

About computer systems

- Common problems with all complex systems
  - Emerging properties, propagation of effects, incommensurate scaling and trade-offs
  - Examples: space station design, management of economy, communication satellites, design of computer systems
- However, computer systems are different from other systems in two different ways
  - The complexity is not limited by physical laws
  - Rate of change of computer system technology is unprecedented

Computer Systems have no nearby bounds on composition

- They are digital and controlled by software
- Not limited by physical laws
- Analog system
    - Have engineering limitation. Each component contributes to noise.
    - Vibration of electro magnetic radiation
    - Component physical behaviour changes
    - Noise from individual components accumulates
- As number of components increases, at some point of time noise will dominate the behaviour of the system.
    - Natural biological, thermodynamic, macro-economic systems and so on
    - Photocopy of the photocopy is not the same.

- Digital systems are noise free
- Digital logic
  - Range of analog values represent 0/1
  - Gate, flip-flop, a memory chip, a processor or computer system
- When 0 becomes 1, it leads to big mistake and big mistakes are easy to detect
- If a signal does not accumulate noise as it goes through a string of devices, noise does not limit the number of devices to be connected.
- Limit for growth: the ability of human to understand them.
  - A processor chip contains two billion transistors

- Second reason for no nearby bounds is that computer systems are controlled by software.
- The contribution to complexity is worse.
- Hardware has physical limits--- speed of light
- Software appears to have no physical limits.
  - Typical operating system, database system has 10 million program statements.
- It is difficult to get the proper abstraction
- Leaking abstraction
  - Do not perfectly conceal the underlying implementation.
- Leakiness is like noise in analog systems

- It is easy for the designer to misuse the tools of modularity, abstraction, layering, and hierarchy to include some more complexity.
- New features keep getting added.

d(technology)/dt is unprecedented

- Cost for computation and communication dropped 30 percent each year.
- Some components have experienced a greater rate of improvement.
- Different between computer systems and other systems
  - d(technology)/dt
- Complex systems take several years to build, they become obsolete by delivery time.
  - It is not the case with bridge, airplane, civil works

Coping with Complexity

- The right modularity form a sea of plausible alternative modularities
- The right abstraction form a sea of plausible alternative modularities
- The right layering form a sea of plausible alternative modularities
- The right hierarchy form a sea of plausible alternative modularities

Coping with Complexity : Iteration

- Start by building simple working system that meets only a modest number of requirements and evolve the system
  - Small steps will reduce the risk
- Release versions
- Successful iteration
  - Take small steps
  - Do not rush
  - Plan for feedback
  - Study about failures

Coping with Complexity II
Keep it simple

- It is one of the effective method of handling complexity
- Designer must impose self-imposed limits
- Some of reasons for adding new features
  - If new features are added it will be more effective
  - Cost and performance are not factors
  - Each of new feature has demonstrated some where
  - All exceptions are easy to deal
  - Competitor will market a system with new features
  - There is a overconfidence among the designers
- The system designer should keep in mind about the cumulative impact of adding a new feature
- Another principle
  - Avoid sweeping simplifications

Outline

- Introduction
  - What is an Operating System ?
- **Course topics and grading**
- History, development and concepts of Oss
- Different kinds of Computer Systems
- Concept of virtual computer

Objectives

- Traditionally, operating system concepts were developed for single processor (machine).
- Currently, shared memory, shared disk and shared-nothing architectures (computer network) are being employed for building computing systems.
  - Exploiting multiple processor cores and commodity machines to manage diverse computing loads, such as search engine, weather forecasting and mobile-based services.
- The objectives of this course to study the concepts (core as well as recent), which were evolved for building modern operating systems.

Objectives

- Understand the operational part of any computer.
  - To study the concepts (core as well as recent), which were evolved for building modern operating systems.
- Understanding the general principles of OS design.
  - Focus on general-purpose, multi-user systems.
  - Emphasis on widely applicable concepts rather than any specific features of any specific OS.
- Understanding problems, solutions and design choices.
- Understanding the structure of specific OSs: UNIX, LINUX, Solaris, WINDOWS 11, Android

Course topics

- Unit 1: Introduction, Computer System Hardware Review, Networking (9 hours)
- Unit 2: Operating System Structures, Virtual machines, Process and thread management (9 hours);
- Unit 3: CPU scheduling, Process Synchronization, Deadlocks (10 hours);
- Unit 4: Memory management, Virtual memory (10 hours);
- Unit 5: File systems, Hadoop, Map Reduce, Overview of Protection, Security, Multi-media systems, OS design principles (6 hours);

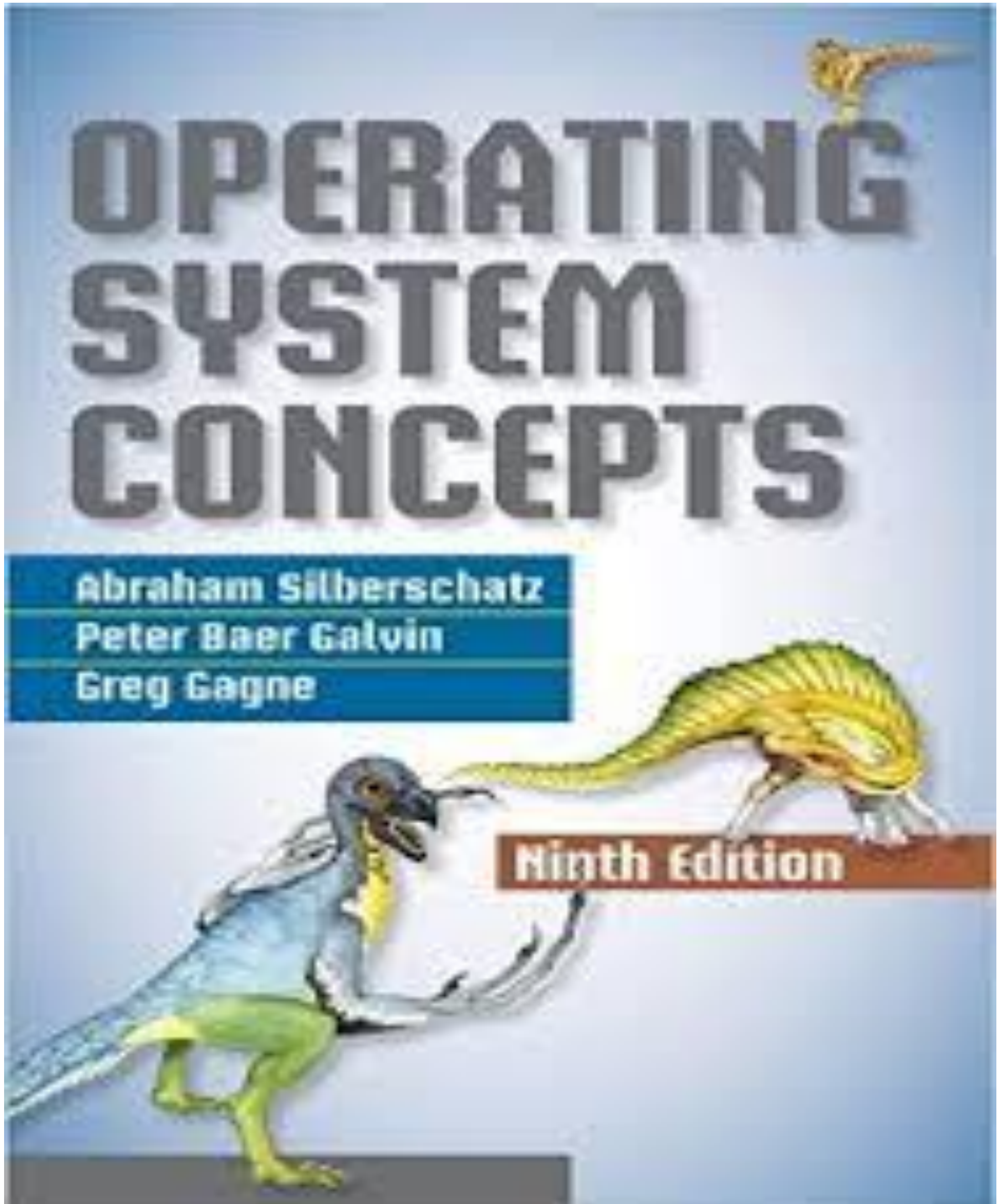
CS3.304: Advanced Operating Systems – Class Schedule (2023)

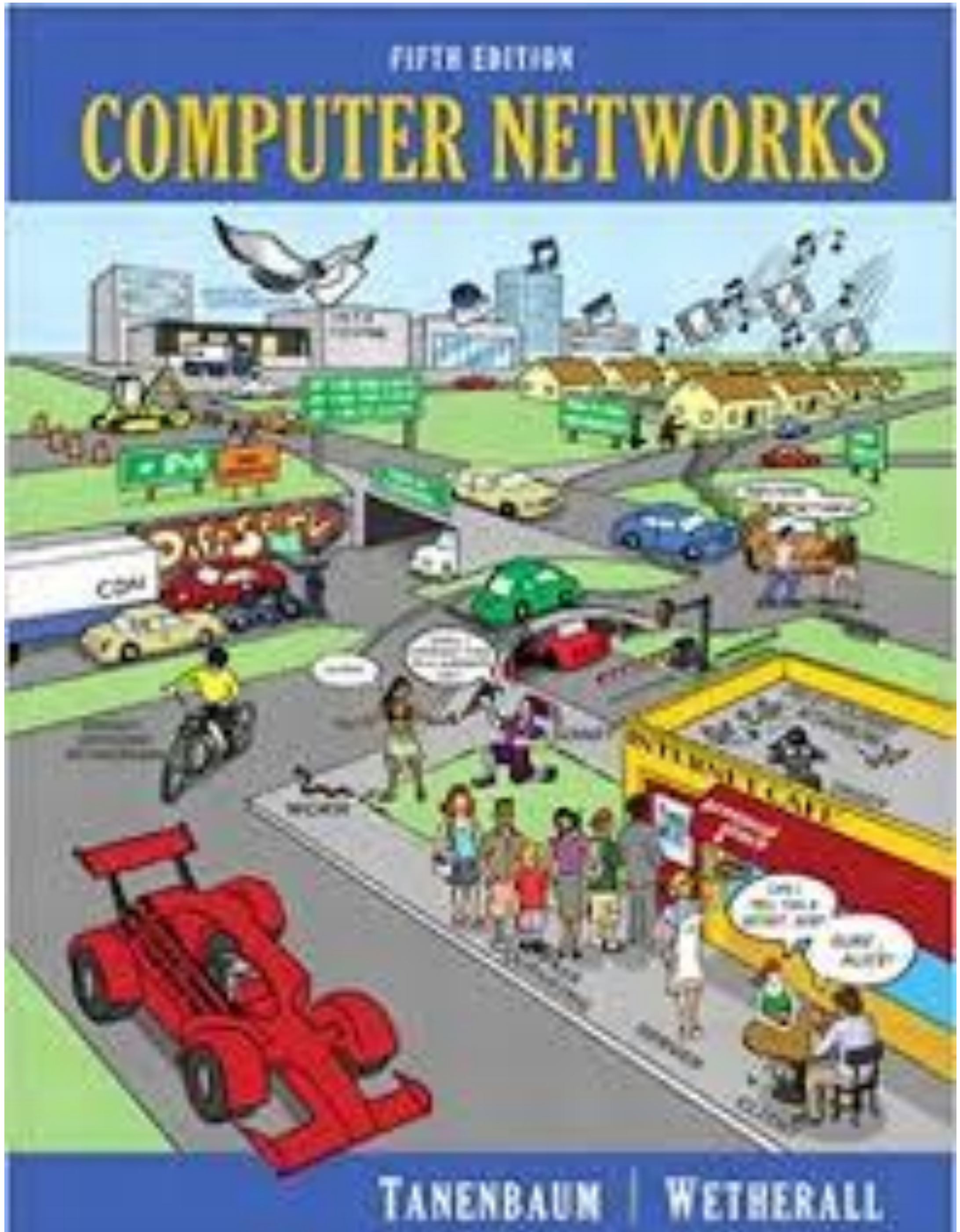
class	Date	Topic	Lab	Assessment
1	1/8/2023 (TUES)	Course Overview		
2	4/8/2023 (FRI)	Concepts of OS: history	Lab 1	
3	8/8/2023 (TUE)	Computer architecture		
4	11/8/2023 (FRI)	System structures		
5	14/8/2023 (MON)	Process concept, inter-process communication		
6	18/8/2023 (FRI)	Concept of Networking, Communication in client-server systems	Lab 2	
7	22/8/2023 (TUE)	Multi-threaded programming		
8	25/8/2023 (FRI)	Virtualization and cloud		
9	29/08/2022	Quiz I		
10	01/09/2023 (FRI)	Process scheduling-1		
11	05/09/2023 (TUE)	Processor scheduling-II	Lab 3	
12	08/9/2023 (FRI)	Processor scheduling-III		
13	12/9/2023 (TUES)	Process Synchronization-1		
14	15/9/2023 (FRI)	Process Synchronization-II		
		Midterm Exam		
15	26/9/2023 (TUE)	Process Synchronization-III	Lab 4	
16	29/9/2023 (FRI)	Deadlocks-1		
17	3/10/2023 (TUE)	Deadlocks-II		
18	6/10/2023 (FRI)	Memory Management-I		
19	10/10/2023 (TUE)	Memory Management-II		
20	17/10/2023 (TUE)	Virtual Memory-I	Lab 5	
21	20/10/2003 (FRI)	Quiz 2		
22	27/10/2023 (FRI)	Virtual memory-II		
23	31/10/2023 (TUE)	Disk Scheduling, RAID		
24	03/11/2023 (FRI)	Basics of File systems		
25	07/11/2023 (TUE)	Hadoop Distributed File System		
26	14/11/2023 (TUE)	Map-Reduce		
27	17/11/2023 (FRI)	Protection, security, Multi-media systems		
28	21/11/2023(TUE)	Trends in modern OS design		

References

- Text books:
  - **Silberschatz, A, Galvin, P, Gagne, G. Operating System Concepts, Addison-Wesley (8th or latest edition).**
  - **Computer Networks (5th Edition) Andrew S. Tanenbaum, David J. Wetherall Prentice Hall.**

- Other References
  - William Stallings, Operating systems, Prentice-Hall, 1998.
  - Operating Systems, Gary Nutt, Pearson Education
  - Charles Crowley, Operating Systems: A design-oriented approach, Tata McGraw-Hill, 1997.
  - Operating Systems: Concepts and Design, Milan Milenkovic, TATA McGRAW-HILL
  - Tanenbaum, A., Modern Operating Systems, Prentice-Hall, second edition, 2000.
  - Research Papers


OPERATING SYSTEM CONCEPTS
Abraham Silberschatz
Peter Baer Galvin
Greg Gagne
Ninth Edition


FIFTH EDITION
COMPUTER NETWORKS
CDN
TANENBAUM | WETHERALL

LAB WORK

- Five mini projects related to the above syllabus will be done by students in the laboratory
- Experiments will be on the exposing the working of several system calls of LINUX OS involving single and multiple real/virtual machines.
  - Installation; reversing a file; Shell writing
  - Process communication
  - Bounded buffer,
  - semaphores,
  - shared memory,
  - threads;
  - Replace "ls" with lookup;
  - Command line for /proc;
  - Memory management
- Note: The lab is very intensive. Please do not ask for the extension of deadline. Each experiment will be evaluated.

Related Research Papers

1. H. M. Levy and P. H. Lipman. Virtual Memory Management in the VAX/VMS Operating Systems. IEEE Computer, 15(3), March 1982, pp. 35-41.

2. Thompson, K., "UNIX Implementation," The Bell System Technical Journal, Vol. 57, No. 6 (July-August 1978), Part 2, pp. 1931-1946.

3. F. J. Corbato and V. A. Vyssotsky, "Introduction and overview of the Multics system," In Proceedings AFIPS 1965 Fall Joint Computer Conference (FJCC), Vol. 27, No. 1, 1965, Spartan Books: New York, pp. 185-196

4. Windows NT and VMS: The Rest of the Story, by Mark Russinovic

5. E. W. Dijkstra, "The Structure of the THE ##Multiprogramming System," Communications of the ACM, Vol. 11, No. 5, May 1968, pp. 341-346

6. C. Daley and J. B. Dennis. Virtual Memory, Processes, and Sharing in MULTICS. Communications of the ACM, 11(5), May 1968, pp. 306-312.

7. Ritchie, D.M., and Thompson, K., "The UNIX Time-Sharing System,"The Bell System Technical Journal, Vol. 57, No. 6 (July-August 1978), Part 2, pp. 1905-1929.

Reading/Practicing Assignments

- Problems will be given
- You have to solve on your own

COURSE OUTCOMEs

- After completion of this course successfully, the students will be able to,
  - CO-1. Explain the concepts of several modern computer operating systems (SOLARIS, LINUX, WINDOWS, MAC, Adroid,…)
  - CO-2: Implement the task on the top of given operating system, in an efficient manner, based on process and thread framework.
  - CO-3. Prescribe the appropriate scheduling/synchronization/memory management/virtual memory/protection for a given application.
  - CO-4. Architect the new application by selecting the appropriate system calls of the given operating system services.
  - CO-5. Develop a distributed application on multi-processor machine or multiple commodity processors connected through a network.

GRADING

Type of Evaluation	Weightage (in %)
Class room test 1	5%
Mid Sem Exam	15%
Class room test 2	5%
End Sem Exam	40%
Attendance	5%
Lab Assignments (mini projects)	30%

Lab Assignments

- You will be given the lab assignments in advance
  - The instructor will provide you with adequate background information to do the assignment
  - Some installations will be necessary. Install well in advance and if you encounter any problems with any installation, you can contact your TA
- Policies for lab
  - You should try to solve any programming issues by yourself first, remember that troubleshooting is how you would actually be learning a lot
  - Every time you encounter any minor programming problem, if you ask the TA or instructor, you will lose out on an excellent learning experience
  - Only if you are unable to solve the problem after putting in a reasonable amount of effort, you should contact your TA or your instructor
  - You should also look online for solutions to your problems, but do not copy code from anywhere
- Grading criteria
  - The overall quality of your lab assignment submission in terms of correctness, quality of code, system design etc.

Plagiarism

- This course has a zero-tolerance policy w.r.t. plagiarism
- Any instance of plagiarism will result in serious penalties (e.g., an F grade for the entire course, among other penalties)
- Forget about doing any kind of plagiarism

Deadlines

- Strict deadlines: You will not be able to submit after the deadline has already passed.

Accessing the Course Materials

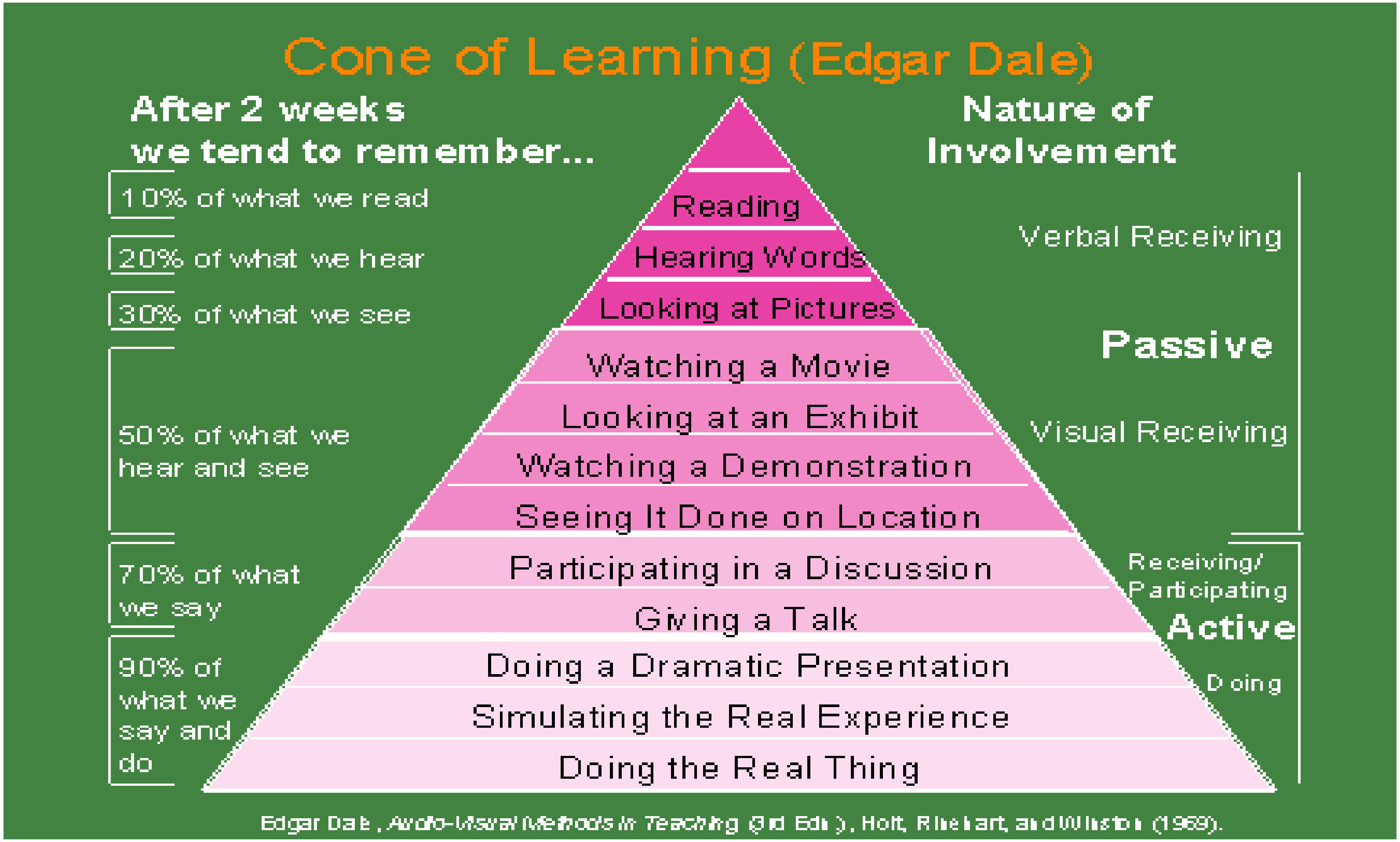
- The presentations, the materials, lab assignments are available on the course portal
  - The slides will be posted prior to each lecture (well in advance)
  - Please ensure that you download the slides & go through them before attending the given lecture
  - Pls. access the course portal regularly
- All students should procure two books, as soon as possible
  - Silberschatz, A, Galvin, P, Gagne, G. Operating System Concepts, Addison-Wesley (8th or latest edition).
  - Computer Networks (5th Edition) Andrew S. Tanenbaum, David J. Wetherall Prentice Hall.

Asking Questions

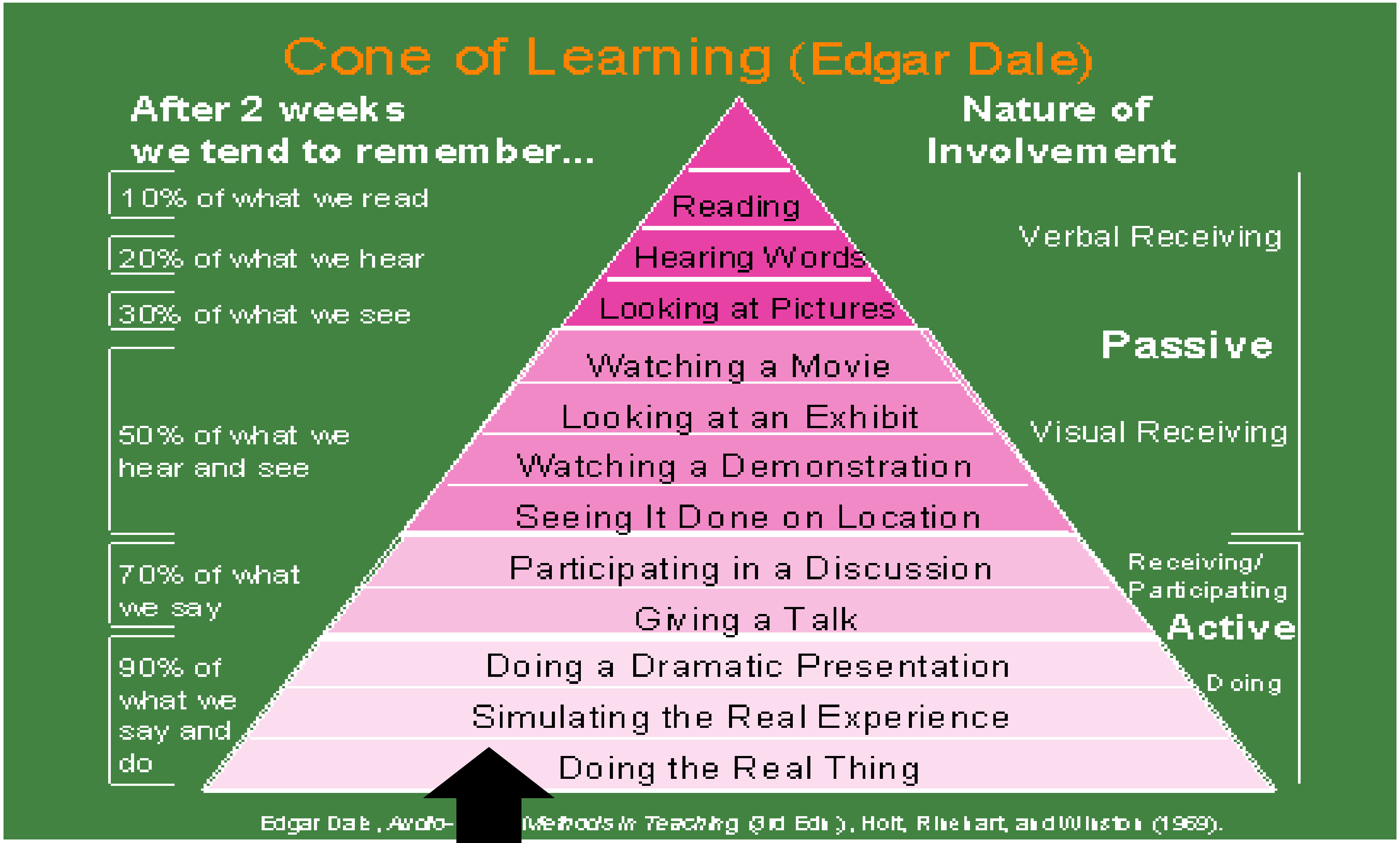
- Theory and lab related questions can be asked through the course portal
  - It is better, because all other students can gain the related knowledge
  - Try to ask as many subject/lab questions as possible.
  - Do not hesitate to ask silly/simple questions regarding lab or theory
    - And try to get the problem resolved as soon as possible

Maximize the benefit from the course

- In the course, we are going study about how a wonderful artifact is being built through the efforts of thousands of researchers
  - Operating system has become part and parcel of every tool
    - Microsoft windows has **50 million** lines of code.
- Focus on a thorough understanding of the concepts, not on memorizing
  - Understand every sentence of the book



Source: http://www.cals.ncsu.edu/agexed/sae/ppt1/sld012.htm



Bottomline: Do the assignments sincerely because it will facilitate you in **INTERNALIZING** the ideas/techniques you learnt in this course.

Source: http://www.cals.ncsu.edu/agexed/sae/ppt1/sld012.htm