Unit-V

Distributed Computing : Distributed process management, message passing, remote procedure call, distributed memory management, security in distributed environment, Introduction of Parallel Processing.

Protection an Security goals, Domain of Protections, Security Problems, Authentication, System threats, Encryptions,

Introduction of different Operating systems (Linux, Unix, Windows Server)

Chapter

11

Distributed Computing

Remote file system were used mechanism to take advantage of high speed networking. Use of the file manager interface and storage model constrains the form of distributed programs, Recent O/S uses other ways to support computation, methods that are better for network environment.

The main goal of this chapter is to introduce the most important OS technologies that supports distributed computing. A distributed computing is supported by deriving new, specialized network protocols of higher level.

11.1 Distributed process management

Todays software is evolving rapidly toward distributed and parallel computation. The distribution is supported by shared and distributed memory multi processors and by network of individual machines that effectively provide an abstract form of the distributed memory multiprocessor. The goal of multicomputers is to provide a simple and efficient environment for designing and executing distributed computations, so that the parallelism of application computation can scale up with reasonable cost and effort.

11.1.1 Partitioning of Work

From the application programmer point of view, is to partition the serial computation into units, where there are number of units that all execute simultaneously with a minimum of overhead. If these portioning is perfect and there is no overhead from administration synchronization or communication then the speed is boost up.

For serial application partitioning the application programmer must know the algorithm used in the computation. First the address space is divided into two parts : one part contains software to execute the application function (for example compute the roots of an equation), and second part contains the data.

Level 2 On the left side of diagram above shows the data could be held in one address space, while the functionality of the computation could be partitioned and implemented as independent processes on independent processors. This form of partitioning is called functional partitioning of the computation.

The philosophy behind this functional partitioning is to divide the function into independent stages and then to pass all the data through the different stages record by record, and data management is much smaller. If the computations has some control flow, then some data records may follow a different route through the computational parts than will others.

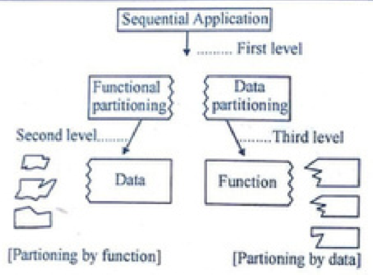


Fig. 11.1.1 : Sequential computation partioning

Level 3 On the right side is an alternative to functional partioning. It divides the data into independent units so that they can be processed by copies of the full sequential functions. This form of partioning is called data partitioning.

The idea behind this partioning is that the functional computation is replicated on number of different machines and then each machine is given a part of the data to be processed. That is each of the processes operating on the different machines is executing in its own address space. Now number of different process of executing simultaneously, each with the responsibility for processing only a fraction of the total data.

The data for each process is loaded into that process's address space. So the data partitioning is sometimes simplex to do than functional partitioning, since a full copy of the original sequential program can simply be given to different processes operating the different machines.

11.1.2 General requirements of Partitioning

What are the general requirements to support distributed computation ? In process management the following major tasks have been identified :

⇒ **Create/delete :** When a computation starts, a single process decides at run time what other process should do ? The O/S must provide facilities to allow the process to create or delete child processes on other machines.

⇒ **Scheduling :** The schedular looks around the network for an appropriate place to execute the program attempting to overlap execution.

⇒ **Synchronization :** In a network the OS generally has to provide an alternative synchronization mechanism based on messages.

⇒ **Deadlock management :** Deadlock detection algorithms rely on the knowledge of the allocation state of the system's resource to determine whether a deadlock exist. In a network, set

of resources includes all resources on every machine, detecting the status of all machines at any given moment.

11.1.3 Architecture of Distributed system

There are various hardware and software architecture are used for distributed computing. It is necessary to interconnect multiple CPUs with some sort of network and also necessary to interconnect processor running on those CPUs with some sort of communication system. Distributed programming can falls into one of various basic architectures or categories : **Client server** architecture connected with smart client code that contacts the server for data. **3-tier architecture** is mostly used in web applications, it moves the client intelligence to a middle tier so that stateless clients can be used, **N-tier** architecture typically is for web applications which forward their request to other enterprise services, **clustered** architecture refers to the cluster of machines that closely coupled and work together, running a shared process in parallel. **Peer to peer** architecture works uniformly and divide there work among all machines, known as peer. Peers can serve both or clients and servers.

11.1.4 Advantages of Distributed Computing

There are some positive factors for using a distributed computing. They are-

1. Resource sharing : Some resources are typically distributed across the system because they cannot be fully replicated at all the sites because it is often neither practical nor cost effective (like I/O files, database, variables, special functions).

2. Inherent : In some application there are some geographically distant to reaching and compute them the computation is inherently distributed.

3. Access remote data : Some data may be too large or too sensitive to be replicated, so to access such super computers users need to log in remotely through some important distributed protocols & middleware.

4. Reliability : A distributed system is more reliable to execute geographical data resources that are not likely to crash at the same time under normal circumstances.

5. Availability : Distributed system resources should be accessible at all times and to improve availability, key components should be replicated.

6. Incremental performance : By resource sharing and remote data acessing the performance/cost ratio is increased than using special parallel machines.

7. Modularity & incremental expendability : Heterogeneous processors may be easily added into the system without affecting the performance as long as those processors are running the some middleware algorithms.

8. Scalability : As the processors are usually connected by a wide area network, adding more processors does not show a direct bottleneck for the communication network.

11.1.5 Disadvantages of Distributed systems

→ Software may not surely available.

→ Network can become saturated. (need additional wiring)

→ Security issuer (only isolated machines is safe)

→ Incremental growth is difficult in practice because of changing Hardware and Software.

11.2 Message Passing

A message is a structured piece of information sent from one agent to another over a communication channel. Message passing mechanisms are the basis of distributed computing in network environments. High performance application domains frequently provide an inter face to the network message passing facility directly to the application programmer, even when the interface is also used by the OS to implement other aspects of distributed environment,

In most applications a message consists of a message identifier and a set of message arguments. The message identifier tells the receiver the purpose or type of the message. The arguments to the message contain additional information that is interpreted based on the type of message. The may contain the object of an action or they may contain the object of an action, or they may contain information used to carry out a request. Message identifiers are usually simple, unique tokens that differentiate one type of message from another. Some simple message protocols get away with using only basic data types, like interger sking and float, for message arguments.

In some very well defined and controlled application environments, message protocol may not need message identifiers at all. The interaction between agents may be so strictly defined that there's no need to specify the type of message being sent because the receiver is already expecting it. HTTP, SSL, TCP/IP are protocols built around some form of message passing. while these other protocols are built around some kind of message passing protocol, that level of the protocol is hidden from the developer by an API of some kind. Likewise, incoming SSL "messages" are processed and mapped into new data objects and method cells on SSL objects. This makes these are complicated but powerful protocols so useful-The application programmer doesn't need to know the details of the protocol at the lower level.

11.3 Remote Procedure call

Remote procedure call (RPC) is an inter process communication technology that allows a computer program to cause a subroutine or procedure to execute in another address space (in LAN or shared network) without the programmer explicitly coding the details for this remote interaction. RPC may be referred to as remote invocation or remote method invocation. By using RPC programmers of distributed applications avoid the details of the interface with the network. The transport independence of RPC isolates the application from the physical and logical elements of the data communications mechanism and allows the application to use a variety of transports.

RPC makes the client/server model of computing more powerful and easier to program. When combined with the ONC RPC GEN protocol compiler clients transparently make remote calls

through a local produce interface.

Remote procedure call (RPC) is a protocol that one program can use to request a service from a program located in another computer in a network without having to understand network details. A procedure call is also sometimes known as a function call or a subroutine call. The requesting program is a client and service providing program is the server. While the server is processing the call, the client is blocked (it waits until the server has finished processing before resuming execution). It can be used for data exchange in distributed file and database systems and harnessing the power of multiple processors. Linux distributions provide an RPC version derived from the RPC facility developed by the open network computing (ONC) group at sun microsystems.

An important difference between remote procedure calls and local calls is that remote calls can fail because of unpredictable network problems. Also, callers generally must deal with such failure without knowing whether the remote procedure war actually invoked.

11.3.1 How does RPC work

An RPC is similar to a functions call when RPC is made the calling arguments are passed to the remote procedure and the sender waits for a response to be returned from the remote procedure. When RPC call between two network system, the client makes a procedure call that sender request to the server and then waits for response. The thread is blocked until reply is received, or times out. When the request arrive, the server cells a dispatch routine that performs the requested service and sender the reply to the client. After the calling process of RPC is completed, the client process continues.

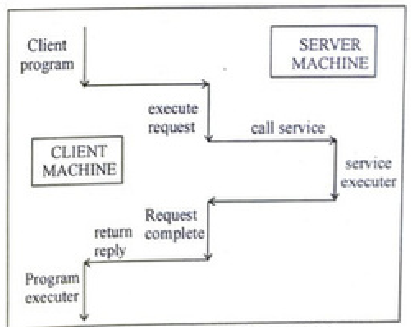


Figure 11.3.1 : Mechanism of RPC

RPC is identified by : program number, which identifies a group of related remote procedures each procedure has a unique program number., **version number** a program may consist of one or more versions. Each version consists of a collection of procedures which are available to be called remotely. Version number enable multiple versions of an RPC protocol to be available simultaneously. **Procedure name** is also a part of RPC which gives a unique identify of procedure.

11.4 Distribute Memory Management

A process has to be memory system, including operating system support for remote disks and remote files. Operating system designers use several different models for handling remote memory. A new interface may be added to allow programs to explicitly reference primary memory allocated at remote hosts. (See figure A)

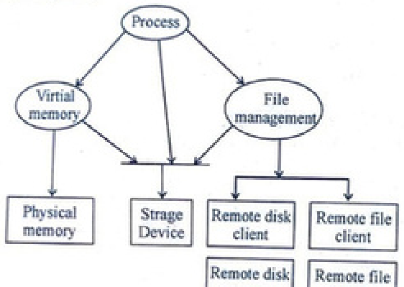


Figure 11.4 (A) : Normal Interface to the memory system

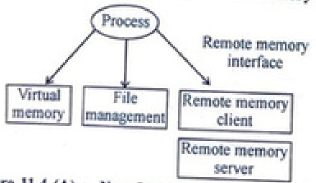


Figure 11.4 (A) : New Interface to Remote memory

Distributed objects one a special case of distributed shared memory, some operating system designers use a single local object interface while others provide both a local & remote object interface so that local object references will not be "slowed down" by having to use a general protocol that can references local and remote objects (see figure B)

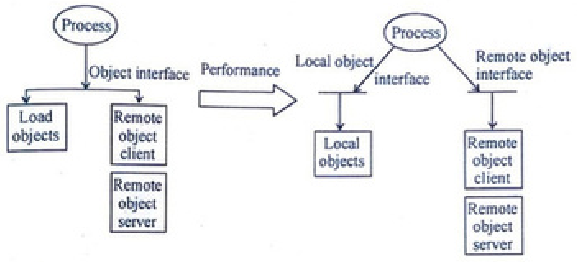


Figure 11.4 (B) : Distribution object

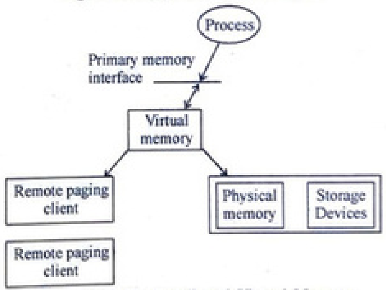
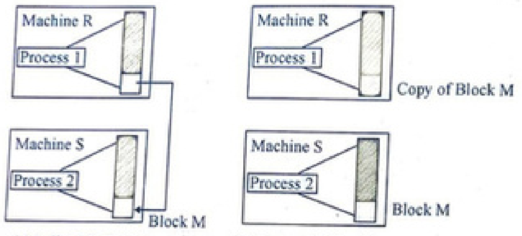


Figure 11.4 (C) Distributed Virtual Memory

Figure C : indicates how distributed virtual memory extends a local system's paging syste across the network. A page server manages secondary memory so that the missing page can be retrieved from the server rather than from the local disk when a page faults occur.

The figure provide several alternatives for representing a remote address space. The distributed memory is shared memory since it is used by two or more processes executing on two or more machines. Adistributed memory appears as a collection of blocks of primary memory. The block specification are derived from the programming language, so they may be objects, derived data structures or dynamically allocated memory.

Lets have an example of two general approaches of distributed memory design. We have two figures a and b in fig. (a) the operating system for machine S allocates a block of memory, M, to be shared by processes 1 and 2 executing on machines R and S. When process 1 references the distributed memory, the reference is translated to a server request in machine S. The communication between the distributed memory client in R and the server in S uses a suitable network protocol.



(a) Indirect Reference (b) Copy of Inforuation

In fig. (b) ilustrates are alternative approach. Here, machine R makes a copy of block M in the address space of process 1. Now, each reference by process 2 to block M is local in machine S and each reference by process 1 to block M is local in machine R. As in all other cases in which copies are created, the problem arises when process 1 writes into blocked M. The copies in machine R and machine S are inconsistent. The distributed- memory system must provide a mechanism to ensure memory coherence if blocks are cached.

There are several fundamental issues must be considered when designing distributed memory systems :

- **Memory Interface :** Should the model employ a distinct interface for referencing network memory, or can it reuse the existing primary memory interface to reference remote memory ?
- **Location transparency :** How much knowledge should the process have about the location of the remote part of the address space ?
- **Unit of Sharing :** What should the unit of sharing be in the address space ? The unit could be data structures, pages, segments, or some other unit.
- **Name management :** Information will have to be imported and exported by naming the unit to be shared. How should this be handled ?
- **Implementation efficiency :** Assume the two processes and their address spaces are on different machines, what are efficient implementations of the remotely stored shared memory.

There are two classes of architecture do implement distributed memory.

11.4.1 Multi computer

Many multi processes are built as a "multi computer" with several different processors having access to the entire memory of the machine.

11.4.2 Network of Machines

Another class of distributed memory designs provides a logical shared memory interface using the packet based network to support access to the memory blocks.

11.4.3 Remote memory

Remote memory refers to any of a broad spectrum of approaches in which the distributed memory is accessed using an interface that offers from the normal primary memory interface & from the file interface.

The remote memory interface extends the conventional programming model suggested by non veumann machines

11.4.3.1 Memory interface : Two issues are involved in designed the memory interface-

How is the memory read from & written to once it has been declared ?

How is memory declared to be remote ? Means, how can remote memory be mapped into a process's address space.

11.4.3.2 Memory unit sizes- Remoe memory is natural for the size of the units to be defined explicitly by the programming rather than by the system. Pure location transparency implies that there is a global address space completely hidden the physical location of any addressing appearing in it. for example : if remote memory is allocated at the transport address <net,host, port>, the process must be able to reference the block & offset at the given server address.

11.4.3 Distributed virtual memory

Shared remote memory inherently has an aspect of additional abstraction to it. This is because it is treated as if it were local memory even though it is physically allocated on a remote machine and accessed using network protocols.

Virtual memory incorporates its own memory mapping mechanism as an inherent part of its operation. The technique uses this mechanism to implement distributed memory.

Virtual memory references differ from physical memory references because each virtual address is mapped to a physical address prior to being referenced within the physical memory module. In distributed virtual memory, distributed virtual addresses are mapped into a shared virtual address space. The shared unit address identified the location of the target memory location if it is loaded in the local machine's primary memory, or its location an a global secondary memory indicated in figure below :

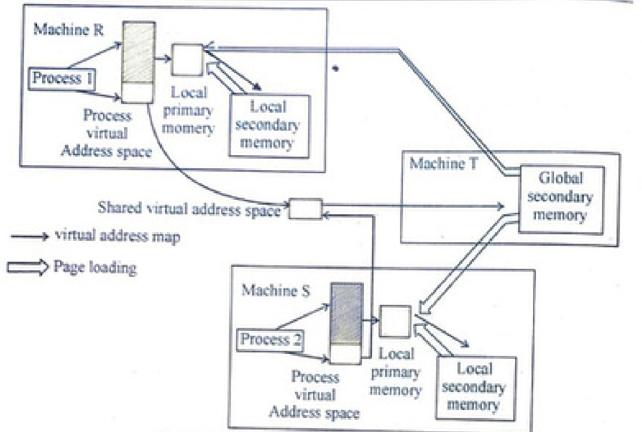


Figure 11.4.3 : Distributed virtual memory

When a page from the private part of a process's address space is loaded, it is obtained from the process's local secondary memory image. When a page that is mapped to the shared part of the virtual address space is loaded, it is obtained from a shared, global secondary memory location that is accessible by more than one process.

The shared, global secondary memory as residing on a seperate machine is shows in figure. However, the critical aspect is that the shared page is referenced via a server process on the local machine or on a remote machine.

Distributed virtual memory is very attractive from the programmer's side. It can be allocated and referenced in the same manner as local virtual memory. From the memory manager's perspective, the machinsm for constructing the virtual address space needs to address the usual network naming and transparency issues.

The global secondary memory server must register its service with an appropriate name server so that clients can bind to the server at runtime. There are several ways to invoke the server name binding. The simplest is for he programmer to call an initialization procedure looks up the page server's transport layer address, <net, host, port>, establishes a connection to it, and then allows the process to begin executing. When the local memory manager detects a missing page fault, it determines, if the page is in the distributed memory or local virtual memory. If it is disributed, fault handler uses the connection to the page server to retrieve the missing page and place it in the local primary memory.

11.4.4 Distributed objects

Object oriented programming has become a popular for defining computations because it inherently relies on message as the exclusive means of interaction among components.

Object are difficult for an OS to manage efficiently because they can be as small as on integer or as large as a bitmap image. With distributed memory, the difficulty is moving objects around on the network, so that when are object is being heavily used by another object, the two are loaded on the same machine. Thus object mobility is a key issue in implementing distributed objects while similar to load balancing, object mobility differs in the sense that only portions of address spaces are moved, with the object names being maintained in a global address space. Emerald is an example of a distributed object system.

11.5 Security in distributed Environments

Security strategies must be as diverse & customized as are individual computing environments. Many large organization have a vast number of computers in use in many different areas.

To computing functions typically report to several different parts of the organization rather than to a centralized system or security group. Yet most security function still attempts to operate as a centralized control & support function.

The security function has been centralized because-

The client or user areas do not have the expertise them selves.

The client or user areas do not fully accept or under stand security requirements.

The client or user areas cannot provides the necessary level of reliability in terms of process & counter balances.

The system support functions are largely centralized.

Although these arguments may be legitimate they can no longer be used to justify maintaining centralized system security in a business environment in which systems management is distributed. If an organization must maintain this, it should be prepared for large & costly bureauracis for lock of support from the people who need security services.

11.5.1 Security controls for Distributed computing

For many organization distributed processing has become the most prevalent mode of computing. In an organization with a midrange distributed environment, the responsibility for all aspects of ownership and maintenance of the distributed system typically follows distributed model.

11.5.2 New Role of Security

The role of the security department depends on the organization's node of systems management.

In a Centralized Environment

The security department's role is to provide centralized security management and contralized security tools as well as to support and exist with centralized compliance management. This means establishing appropriate centralized administration & support procedures that are managed constituently and easily dutiable. Further it means a centalized responsibility for documenting what resources exist & what rules should apply in managing these resources.

In a Decentralized Environment

Security management functions should follow system management function. In the case of a centralized system, for example, a person in the payroll department should be designated to manage local security administration needs. If systems management both provides access to and uses locally managed payroll resources, the security support function should be separated from the system management function.

11.5.3 Automated security tools

The decentralized are as that most need guidance seldom have staff people with sufficient interest and experience to determine and implement the necessary standards & then to manage the decentralized security functions. Such security management tools come in two forms:

Automated tools that enable an another to determine whether the security implementation is in compliance with the organization's policies & standards.

11.5.4 Maintaining security in the future

For security to be maintained in the coming years, data security managers must know how and when to look for technical expertise, and what expertise to look for. The most pragmatic approach is to establish environment specific standards that can be managed on an exception basis. One these standards are in place and procedures for identifying & responding to exception conditions have been established the organization can be supported with a simple monitoring & reporting process that identifies and reporting process that identifies and responds to any non standard activity.

The process of establishing can be under taken either by using external exports on a limited basis or by bringing together a task force of internal exports. In either case, data security managers can acts as facilitators, project managers gathering a flexible body of expertise and resources when they are needed.

11.6 Introduction of Parallel Processing

Parallel operating systems are the interface between parallel computers (or computer systems) and the applications (parallel or not) that are executed on them. They translate the hardware capabilities into concepts usable by programming languages. Parallel operating system are primarily concerned with managing the resources of parallel machines. This takes is very challenging : Application programmers demand all the performance possible many hardware configurations exist and change very rapidly, yet the operating system must increasingly be compatible with the main

stream versions used in personal computers and workstations due to both user pressure and to the limited resources available for developing new versions of these system.

The architecture of a parallel operating system is closely influenced by the hardware architecture of the machines it runs on. Parallel operating system in particular enable user interaction with computers with parallel architecture. The physical architecture of a computer system is therefore an important starting point for understanding the O/S that contrast it. There are two classifications of parallel computer architectures : **Flynn's** and **Johnson's.**

11.6.1 Flynn's classification of Parallel architecture

Flynn divides computer architectures along in two areas according to the number of data sources and the number of instruction sources that a computer can process simultaneously. This leads to four categories of computer architecture :

SISD : Single Instruction single data. This is the most widespread architecture where as single processor executes a sequence of instructions that operate on a single stream of data. These computers typically run either a version of the microsoft windows O/S or one of the many variants of the UNIX O/S (sum microsystem's salaries, IBM's AIX, Limuse.....) although these O/S include support for multiprocessor architectures and are therefore not strictly operating system for SISD machines.

MIMD : Multiple Instruction multiple data. These machines constitute the core of the parallel computers in use today. The MIMD design is a more general one where machine are composed of a set of processors that execute different programs which access their corresponding datasets. These architectures have been a success because they are typically cheaper than special purpose SIMD machines since they can be built with off the shelf components.

MISD : Multiple instruction single Data, No existing computer corresponds to the MISD model which was included in this description mostly for the sake of completeness. However, experts could envisage special purpose machines that could use this model. For example "Cracker computer" with a set of processors where all are fed the some stream of ciphered data which each tries to crack using a different algorithm.

SIMD : Single Instruction multiple data. The SIMD architecture is used mostly for super computer vector processing and calculus. SIMD computers is based on distributing sets of homogeneous data among an array of processors. SIMD machines can be further divided into pipelined and parallel SIMD. An example of a SIMD machine is the connection machine (e.g. CM-2), built by the thinking machines corporation and the MasPar MP-2 by the masPar computer corporation. This computer is based on a 2 dimensional lattice of up to 16 K processing elements driver by a VAX computer that server as a front end to the processing elements.

Exercise

Very Short Questions (Up to 20 words)

1. What should distributed system do ? [Raj. Univ. 2008]
2. Disadvantage of distributed system is ? [Raj. Univ. 2005]
3. Define RPC. [Raj. Univ. 2008]
4. What is centralized and Decentralized Environment ?
5. Explain Distributed virtual memory.

Short Questions (up to 80 words)

1. Write about client server computing. [Raj. Univ. 2004]
2. Explain security in distributing system.
3. Explain Message passing in distributed system.
4. What is meant by RPC ? Explain its working.
5. Define Automated security tools.

Long Questions

1. How you maintain security in the future ?
2. Discuss the architecture of distributed system.
3. Write short notes one :
   (i) RPC
   (ii) Distributed processing system
4. What are the main advantages and disadvantages of distributed computing ?
5. Explain the memory management of distributed system.

■■■

Chapter

12 Protection and Security

12.1 Protection

Processes in (user & system processes) an OS must be protected from one another's activities protection and security, Which is measure of confidence that the integrity of a system and its data will be preserved. Internal protection is of no use if the computer system is accessed by an unauthorized individual who can remove some important file or insert a virus into the system and makes it non functional

12.2 Protection an security• Goals

The protection to be enforced into a system should have the following goals.

The OS should enable the users to safety share common logical address space such as files etc. It should also enable the users to share a common physical address space such as memory in such system, the goal should be to prevent accidental and intentional distractive behavior.

Protection can improve reliability by detecting latent error's at the interface between component subsystem. A protection oriented system provides means to distinguish between authorized and unauthorized usage.

Ensure fair and reliable resource usage, Each program. Component active in a system should use the system resources only in accordance with certain policies.

12.3 Domain of Protection

A computer system is a collection of processes and objects (resources). Object means hardware objects (such as the CPU, disk, memory segments) and software objects (such as files programs and semaphores).

Each object has special name. The operation that are possible may depend on the object. For Example, on CPU can be only executed on, memory can be read or written, disk can be read and written, data files can be created, deleted, opened, read, written and closed A process should allowed to access only those resource it has been authorized to access. At any time a process should be able to access only those resources that it currently requires to complete its task.

This second requirement, "need to know" principle, so that it should be able to access the resource which it currently needs.

12.3.1 Access Rights

Protection should be provided to those resources. Which may be currently used by a process.

Each domain defines a set of objects and type of operation that may be invoked on each object. The ability to execute an operation on an object is an access right. A domain is a collection of access rights. This means that domain is a set of <object, right> pair.

Right refers to the operations that can be invoked by a process on the corresponding object. These right are also know as access rights of a particular process on the corresponding object.

eg.- if a process is executing in domain D < = data files [read, write] >; it can perform only read and write operation on the data file named 'F'.

The association of a process with a domain may be static or dynamic.

The static if the set of resources available to the process is fixed through out the process's lifetime.

In dynamic establishing dynamic protection domain is more complicated than establishing static protection domain's. If the static association protection can be implemented by strictly adhering to the "need to know" principle. This implies that the domain should keep changing at different instances of a process execution, as and when there is a changing in the use of resources by this process.

For example, consider a process P that needs read access (on a disk) in one phase and write access in another phase. Then in the first phase the domain should include only read access right on disk i.e. <disk, read> and in the later phase, the domain should be modified to contain <disk, write>

In dynamic a process can switch from one domain to another. A new domain may be created with the changed content and the process may be switched to this new domain.

A domain can be realized in a variety of ways.

Each user may be a domain. The set of objects that can be accessed depends on the identity of the user. Domain switching occurs, when the user is changed, when one user logout & another is login.

Each user may be a domain. The set of objects that can be accessed depends on the identity of the process. Domain switching occurs, when one process sends a message to another process and then waits for a response.

Each procedure may be a domain. The set of objects that can be accessed corresponds to the local variable defined writing the procedure. Domain switching occurs when a procedure call is made.

12.3.2 Access matrix

The access matrix model is the policy for user authentication and has several implementations such as access control lists (ACL) and capability lists. It is used to describe which users have access to what objects (resources) The access matrix model consists of 4 major parts.

⇒ A list of objects.

⇒ A list of subjects

⇒ A function I that return an object type

⇒ The matrix itself, with the objects making the columns & the subjects making the rows

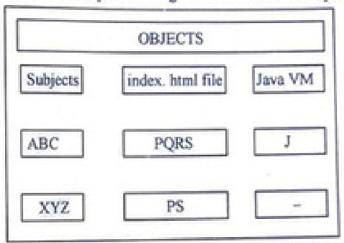


Figure 12.3.2 (a) : Example Access Matrix

In the cells where a subject and object meet lie the rights the subjects has on that object. In figure shows an example access matrix.

An access matrix has several standard operations :

Entry of a right into a specified cell

Removal of a right from a specified cell

Creation of a subject

Removal of a subject

Removal of a object.

In figure users as subjects and states that ABC can read write execute & copy the index html file. XYZ can only read of copy this file. The subject can also be processes and procedures. It means that subjects are in fact the domains, which can be realized either as a user process or a procedure.

Object / Domain	File 1	File 2	Printer
D1	Read write	execute	output
D2	–	Execute	–
D3	–	Read write	output

Figure 12.3.2 (a) Generalized Access matrix

Advantages : It provides an appropriates mechanism for defining and implementing strict control for both the static and dynamic association between process and domains.

Disadvantages :

1. Mechanisms must be enforced to protect the access matrices them selves from change.
2. Access matrices are inefficient for storage of access rights in a computer system because they tend to be large.

12.4 The Security Problem

Computer resources must be guarded against unauthorized access, mailcions, destruction or alteration and accidental introduction of inconsistency. These resources include information stored in the system (both data and code), as well as the CPU, memory, disks, tapes and networking that are the computer.

Security misuse of the system can be either accidental or intention. It easier to protect against the accidental misuse then the protect against the intentional misuse.

The intentional misuse can be of the following disks-

- Theft of information
- Unauthorized modification of data
- Unauthorized destruction of data.

System can be protected from such threats at two levels.

Physical : The sits containing the computer system must be physically secured so that the intruders can be prohibited to enter the site. Both the machine rooms and the terminals or workstations that have access to the machines must be secured.

Human users should be carefully screened so that the chance of authorized a user who may give access to an intruders is reduced.

Operating system : The system must protect itself from accidental or purposeful security breaches. A stack flow could allow the launching of an unauthorized process.

Network : Much computers data in modern systems travels. Over private leased lines shared lines shared lines like internets, wireless connection, or dial up lines. These data could be just as harmful as breaking into a computer, and interrupt of communication could constitute a remote denial of service attack, users use of and trust in the system.

12.4 Authentication

Authentication of the user of the system is one of the major security issues associated with an operating system. Generally authentication of a user is based on-

(i) User identifier and password (passwords)

(ii) An finger prints retina pattern or signature (Biometrics techniques)

(iii) Badge card possessed by a user (Artifacts)

(i) Passwords : The password is the most common authentication mechanism based on sharing of a secret. In passwords based system each user has a passwords, which may initially be assigned by the system administrator. Many systems allow users to change their passwords. The system stores all user passwords and uses them to authenticate the users. Passwords are popular because they request no special hardware and are relatively easy to implement.

(ii) Artifact based authentication : The artifact used for user authentication include machine readable badges (usually with magnetic strips) and electronic smart cards.

Badge or card readers may be installed in or near the terminals and user are required to supply the artifact for authentication.

In many systems, Artifacts identification is coupled with the use of a passwords. For example, In some companies badges are required for employee to gain access to the organization's gate.

Smart cards can augment this scheme by keeping even the user's passwords with in the card itself, which allows authentication with out storage of passwords in the computer.

(iii) Bio metrics Techniques : Authentication mechanism is based on the unique characteristics of each user. Some user characteristics can be established by means of biometrics techniques.

(iv) Password vulnerabilities : There are two common ways to gives password. One way is for the (either human or program) to know the user or to have information about the user. People use obvious information (such as the names of their cats or spouse) as their passwords. The other way is do use all passable combination of valid password characters until the password is found.

(v) Enrypted passwords : One problem with all these approaches is the difficulty of keeping the password secret with in the computer. Each user has a password. The system contains a function that extremely difficult the designers hope impossible to invert but is simple to compute.

When a user presents a password it is encoded and compared against the stored encoded password.

Even of the stored encoded password is seen, it cannot be decoded, so the passwords cannot be determined. Thus, the password file does not need to be kept secret.

(vi) One time password : To avoid the problems of password sniffing and shoulder surfing a system could use a set of paired passwords. When a session begins, The system randomly selects and presents one part of a password pair, the user must supply the other part.

This approach can be generalized to the use of an algorithm as a password. The algorithm might be an integer function for example. The system selects a random integer and presents it to the user. The user applies a function and replies with the correct result. The system also applies the function if the two results match, access is allowed.

These fall into two categories-

(i) Physiological characteristics, such as finger prints capillary patterns in the retina hand geometry, and facial characteristics.

(ii) Behavioral characteristics, such as signature dynamics, voice pattern and timing of key strokes.

Advantage : Largely increased accuracy of user Authentication.

- Reduction of errors in security conscious environments.

Disadvantage : Increased cost

- Potential invasion of privacy.
- Reluctance of some users.

12.5 Encryption

Encryption is one such mechanism which allows such data to be scrambled so that even if some one intercepts it on the network, it is not readable to him/her.

Thus the basic purpose of encryption is to make the data transfer secure over the network.

Encryption works as :

1. It transforms information from ("clear") to coded information ("ciphertent"), which cannot be read by outside parties.

2. This transformation process is controlled by an algorithm & a key.

3. The main challenges in using this approach is the development of encryption schemes that are impossible to break-

These are two kinds of encryption (i) "Symmetrical Encryption" or secret key which users a single key to encrypt and decrypt the transmitted data.

(ii) Asymmetrical encryption which uses "private key" in which one key is used to encrypt and another to decrypt the transmitted data.

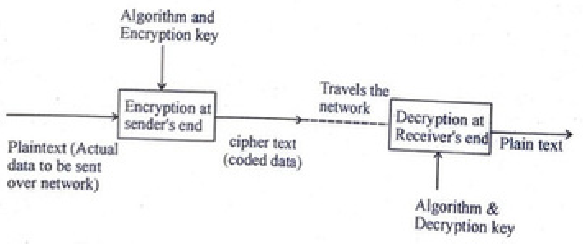

Algorithm and Encryption key
Encryption at sender's end
Travels the network
Decryption at Receiver's end
Plain text
Plaintext (Actual data to be sent over network)
cipher text (coded data)
Algorithm & Decryption key

Figure 12.5 : General Encryption & Decryption mechanism

12.6 Program Treats

Processes are the only means of accomplishing work on a computer. Therefore, Writing a program that creates a breach of security, or causing a normal process to change its behaviors & create a breach is the common goal of crackers.

12.6.1 Viruses, worms and Trojans

A computer virus is a small program written to alter the way a computer operator and its executes without the permission or knowledge of the user. A program is a virus if it meets two criteria-

1. It must execute itself
2. It must replicate itself

Worms are basically the programs that replicate them selves from the system to system without the use of a host file. They contain malicious code that can cause major damage to the files, software and data on a computer.

There are several ways that worms can get into a computer. The most common is through the interned and E-mail. Pretty park Worm is a privalent example. Trojan horse are impostors files that claim to be some thing describe but in fact are malicious.

A very important distinction from true viruses is that they do not replicates them selves as versus do. Trojan contain malicions code, that when triggered cause loss of data.

12.7 System and Network Threats

Program threats use a breakdown in the protection mechanism of a system to attack programs. In contrast system and network threats involves the abuse of services & network connections. System and network threats create a situation in which OS resources & user files are misused. Sometimes a system & network attack is used to launch a program attack.

The more open an operating system is more services it has enabled and the more functions it allows, it is that a bug is available to exploit. Increasingly operating system strive to be secure by default.

Exercises

Part I (Very Short Answer)

1. Define security in details ?
2. Explain the Access Authorization ?
3. Disadvantage of access matrix ?

Part II (Short Answer)

1. Explain the domain the protection ?
2. Describe the security problem ?

Part III (Long Answer)

1. Explain the threats monitoring in details ?
2. What are two advantage of encrypting to data stored in the computer system ?
3. Compare capability lists and access list ?

■■■

Chapter

13 Case Studies-Linux, Unix and Window Server

Introduction of different operating system (Linux, Unix, Window Server)

13.1 Linux Operating System

Linux is a free open source operating system based on unix. linux was originally created by Linux torvalds with the assistance of developers from around the global.

Linux is very powerful OS it is gradually becoming popular though out the world.

13.1.1 Linux 2.0

Released in June 1996, 2.0 added two major new capabilities.

- Support for multiple architectures, including a fully 64 bit native Alpha port
- Support for multiprocessor architectures

Other new features included :

- Improved memory management code
- Improved TCP/IP performance
- Support for internal kernel threads, for handing dependencies between loadable modules, and for automatic loading of modules on demand.
- Standardized configuration interface

Available for Motorola 68000 series processors, Sun Sparc systems, and for PC and PowerMac systems 2.4 and 2.6 increased SMP support, added journaling file system, preemptive kernel, 64 bit memo.

13.2 Components of linux system

Linux operating system has three components -

13.2.1 Kernel

Kernel is the core part of linux. It is responsible for all major activities of this operating system. It is consist of various modules & it interacts directly with the underlying hardware.

13.2.2 System library

Its are special function or programs using which application programs or system utilities accesses kernel's features.

13.2.3 System utility

System utility programs are responsible to do specialized individual level tasks.

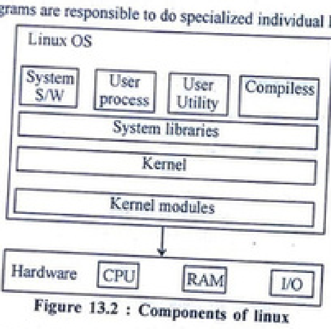


Figure 13.2 : Components of linux

13.3 Kernel mode vs user mode

Kernel component code executes in special privileged mode called kernel mode with full access to all resources of the computer. This is represent a single process, executes in single address space & do not require any content switch & hence is very efficient and fast.

Support code which is not required to run in kernel mode is in system library user programs & other system programs works in user mode which has no access to system hardwares and kernel code.

User programs/utilities use system libraries to access kernel function to get system level tasks.

13.4 Basic Features

Portable : Part ability means softwares can works on different types of hardwares in some way. Linux kernel & application programs support their installation on any kind of hardware platform.

Open source : Its freely available & it is community based development project.

Multi user : It means multiple user can access system resources. Like memory/ram/application program at some time.

Multi programming : It means multiple application can run at same time security linux provide user security using authenfication features like password protection/controlled access to specific files/encryption of data.

Shell : Linux provides a special interpreter program which can be used to execute commands of the OS.

Hierarchical File System : Linux is a multi programming system means multiple application can run at same time.

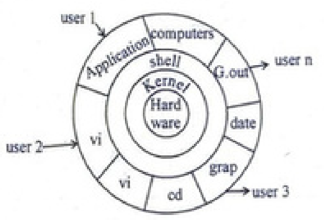


Figure 13.4 : Architecture

Linux system Architecture is consists of following layers-

- **Hardware layer :** Hardware consists of all peripheral devices (RAM/HDD/ CPU etc.)
- **Kernel :** Core component of operating system interacts directly with hardware provides low level services to upper layer components.
- **Shell :** An interface to kernel hiding compliantly of kernel's functions from users. Take commands from user & executes kernel's functions.
- **Utilities :** Utility programs giving user most of the function abilities of an operating system.

13.5 Unix

Unix is an operating system which was first development in 1960's, and has been under constant development ever since. By operating system, we mean the suit of programs which make the computer work. It is stable multi-user. Multi-tasking system for servers desktops & laptop.

Unix system also have a graphical user interface similar to microsoft window which provides an easy to use environment.

It is popular multi-user multi tasking operating system.

13.6 Types of UNIX

There are many different versions of unix although they share common simulates the most popular varieties of UNIX are SON solans GUU/LINUX & Mac OS X.

13.6.1 The UNIX operating system

The unix operating system is made up of three parts the kernel the shell the program.

The kernel : The kernel of the UNIX is the sub of the operating system. It allocates time and memory to programs and handles the file store & communication in responses to system calls.

The Shell : The shell acts as an interface between the user and the kernel when a user logs in the login program checks the user name & password & then starts another program called the shell.

Files and processes : Every thing in UNIX is other a file or a process.

A process is an executing program identified by a unique PI(Process unidentified) A file is a collection of data. They are created by users using text editors running computers etc.

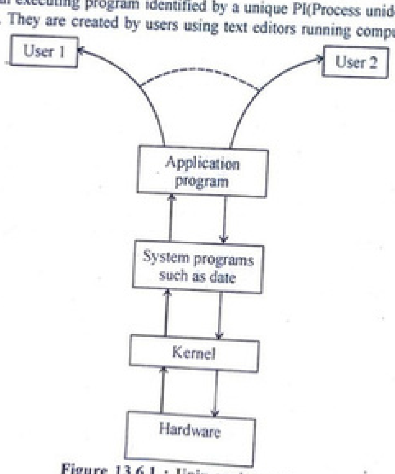


Figure 13.6.1 : Unix environment

Unix operating system can be divided into pieces. These pieces can be thought of as building blocks of the OS. The lowest level of the operating system controls the computer's hareware,

while higher levels of the operating system provide higher level services by manipulating the lower levels of the system.

In figure, at the most basic level of the operating system is a compact collection of software which directly controls.

Control part of the operating system is terminal the kernel. It is the unix kernel, which controls your keyboard & mousse your screen & the various disk drivers other peripheral attached to your computer.

At the highest level of the operating system software hierarchy is the UNIX shell. The UNIX shell acts as a command interpreter. It is piece of software which provide you with a system prompt & which interprets the commands you type of seconds them to the proper levels of the operating system for processing.

13.6.2 Unix File System

In unix operating system, as in many other operating systems, data is stored in files. In UNIX, a file is a collection of data stored together on a mass storage device (such as a disk drive) and given a name. File's in UNIX can contain any sort of data, and can be of virtually any size.

You might for instance, have file, which contain's a digitized picture of your car, another, which contains your grocery list, and another. Which contains the text of a paper you are writing for a class other files contain the executable code for varions commands you will use.

Files are stored in a hierarchical system of directories. When a directory is contained with in another directory it is termed as a subdirectory. The directory it is contained in is called its parent directory. A unix file system is the collection of all directories, sub dirctories and files.

Full path names : In UNIX system there is a single, top level directory. Because the structure of the UNIX file system is some times referred to as a tree, owing to its hierarchical nature, is called root directory. In all cases, the root directory's name is the same : a single slash,/.

Relatives Pathnames and the current working Directory

As an added convenience all directories in a UNIX file system contain two named directories, one called (pronounced dot) and another called .. (pronounced dot-dot). The can be used as short and for your current working directory. The name .. can be used as shartand for your current working directing.

For example, if current directory had the full path names/user/local/lib/emacs.

If you instead used the relative path name :

.. / foo

You shall be referring to the file whose full path name is :

/usr/local/lib/foo

Stadnard directories :

A description of few standard directories like :

/ dev	Where special device files are kept. There is one special file for each device.
/ bin	Where executable system utilities like sh, cp, rm are kept only the utilities available as binary files are kept
/etc	Where system configuration files and databases are kept.
/lib	Where operating system and programming libraries are kept
/ tmp	Contains temporary data files
/ usr/local	The top level directory for storing application software and data written or modified by local system staff.
/usr/local/bin	Where locally written or locally modified application software is stored. Many useful application reside here.
/usr/local/bin	where configuration files for locally adapted software and parts of some locally adapted applications reside.
/usr/man	The manual pages (help pages) are kept here

13.6.3 Process management in UNIX

Processes are actives entities in the UNIX system. Each process runs a single program and has a single thread of control. In other words, it has one program counter, which keeps track of the next instruction to be executed.

UNIX is a multi programming system, so multiple, independent processes may be running at the same time.

Process creation : Processes are created in UNIX in a simple manner. The fork ( ) system call creates an exact copy of the original process. The calling process is called the parent process. The new process is called the child process. The parent and child each have their own private memory images. If the parent sub sequently changes any of its variables, the changes are not visible to the child, and visa versa.

Open files are shared between parent and child. That is, if a certain file was open in the parent before the fork ( ), It will continue to be open in both : the parent and the child after wards. Changes made to the file by either one will be visible to the other.

The fact that the memory images, variable registers, and every thing else are identical in the parent and child leads to a small difficult : how do the processes know which one should run the parent code and which one should run the child code ? The solution is that the fork () system call returns a zero to the child and a non zero value, known as the pid (process identifier of the child) to the parent. As shown in Figure.

```
pid = fork ( ); /* if the succeeds, pid > 0 in the parent */
if (pid < 0)
{
    /* fork failed, usually because memory or some table is full */
}
else if (pid > 0)
{
    /* parent code */
```

```
}
else
{
    /* child code */
}
```

Figure 13.6.3 : Process creation in UNIX

For example : When a child terminates, the parent is given the pid of the child that just finished. This can be important because a parent may have many children. Since children may also have children, an original process can build up an entire tree of children grand children and further discendants.

When the system is booted, the kernal starts a process called init. This process then reads a file/etc/ttys that tells how many terminals the system has and provide certain information describing each one. Init then forks off a child process for each terminal and goes to sleep until some child terminates.

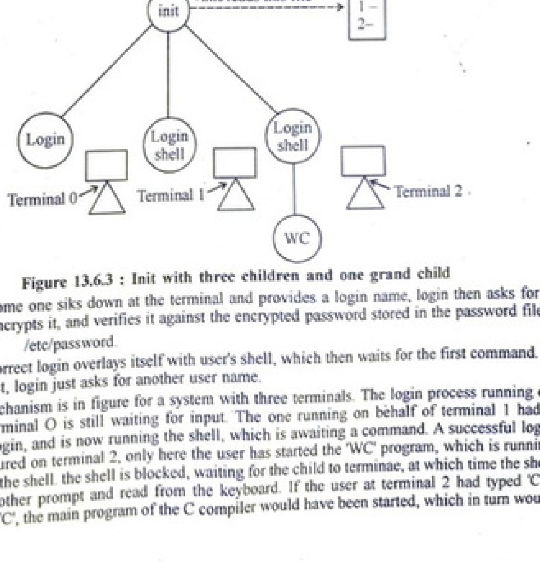


Figure 13.6.3 : Init with three children and one grand child

When some one siks down at the terminal and provides a login name, login then asks for a password, encrypts it, and verifies it against the encrypted password stored in the password file :

/etc/password.

If it is correct login overlays itself with user's shell, which then waits for the first command. If it is incorrect, login just asks for another user name.

This mechanism is in figure for a system with three terminals. The login process running on behalf of terminal O is still waiting for input. The one running on behalf of terminal 1 had a successful login, and is now running the shell, which is awaiting a command. A successful login has also occured on terminal 2, only here the user has started the 'WC' program, which is running as a child of the shell. the shell is blocked, waiting for the child to terminae, at which time the shell will type another prompt and read from the keyboard. If the user at terminal 2 had typed 'CC' instead of 'WC', the main program of the C compiler would have been started, which in turn would have forked off more processes to run the various compiler passes.

13.6.4 Memory management in UNIX

When the UNIX kernel is first loaded in to memory (at boot time), it sets aside a certain amount of RAM for itself as well as for all system and user processes. Main categories into which RAM is divided are :

Text : To hold the text segments of the running processes.

Data : To hold the data segments of the running processes.

Stack : To hold the stack segments of the running processes.

Standard memory : This is area of memory, which is available to all the running programs if they need it.

Buffer cache : All the read and writes to the file system are cached here first. You may have experienced situations where a program that is writing to a file does not seem to work (nothing is written to the file). You wait a while, then a sync occurs, and the buffer cache is dumped to disk and you see that the file size increases.

13.7 Command language of UNIX :

Commands	Purpose
bc	Programmable calculator
cat	Cancate nate file and write them to standard output
CC	Compile a C program
Cmp	Compare two fiels to see if they are identical
Comm	Returns lines common to two sorted files
cp	make copy of a file
date	Return the date and time
echo	Returns the arguments
ls	List files in a directory
MK dir	Make a directory
WC	Count characters, words, and lines in a file
Sort	Sort a file consisting of ASCII times
pwd	Displays present working directory
Paste	Combine multiple files as columns in a single file
ls	List file in a directory
tr	Translate character codes
Sh	Invoke the shell
rm	remove a file
rm dir	Remove a directory
lp	Print a file on the line printer
cut	make a column in a document into a separte file
ln	create link to a file

13.8 Feature

User portability : Even with a relatively poor user interface non programmer users have adopted UNIX. The primary reason for this is because. UNIX can run on so many different computer system ranging form small desktops to the largest computer in the world. Once a user has learned UNIX, the skills can be used on many different systems. This ability for a user to work on many different makes of computer system without retraining is called "user portability".

Open system : An open system is a system, which allows application portability. System inter operability & user portability between many different computer hardware platform. UNIX is a good computer of an open system.

User Interface : There has been extensive work to improve the user interface to UNIX. The most dramatic effort has been the adulation of windows interfaces on top of UNIX such as X-windows, open look etc. These interfaces do not change UNIX itself but are built on the top of UNIX to provide a more intuitive interface to UNIX.

Microsoft windows server : A number of academic departments in Arts. Sciences and Engineering have institute the use of microsoft window server. Due to the complexity and variety of these system, ITS must restrict the hardware & operating system software versions that it can support. All departments operating microsoft window server are required to have operators/or system administrators. Is tier support with a working knowledge of their operating system as well as familiarity with installed hardware and applications.

Separating system : The windows 2008 server product is supported in its current release & service future server products will be supported after sufficient time has been given for iITS to evaluates them upon their release. Window server product be used as a platform for distributed multi user function & not as a substitute for window XP or vista winning standard productivity & effect applications.

Hardware Requirement : Minium requirements for existing servers.

- In Xeon 3.0 GHz CPU
- At least 2 gigabytes of memory.

Back up : For an existing server to supported we a back up schedule must be in place and the tapes readily available screens will be conferred with a tape drive to used for backup purposes.

Power Protection : Servers must be property power protected. At a minimum the server most be attached to uninterrupted power supply.

Firwall : It is required that any server on the network be placed behind an appliance level firewall. UIT & ITS both offer this service. New servers should have a forwall trickled as a part of the purchase owners of existing servers should seriously consider taking advantage of one of the services placing a forwall between the server & the network.

Network Access : All network & LAN access for laboratories & tenders will need to be attained from online from.

The supported of compose connection solution.

All Affiliate account need responsibility person listed under account description.

13.9 Differences between Linux and UNIX

	Linux	Unix
Cost	Linux can be freely distributed, downloaded freely, distributed through magazines, Books etc.	Different flavors of Unix have different cost structures according to vendors
Developed	Linux kernel is developed by the community. Linus torvalds oversees things	Three bigest distributions are Solaris (Oracle), AIX (IBM) & HPUX Hewlett Packard. And Apple Makes OSX, an unix based os..
Price	Free but support is available for a price	Some free for development use (solaris) but support is available for a price.
Security	Linux has had about 60–100 Viruses listed till date. None of them actively spreading nowadays.	A rough estimate of UNIX viruses is between 85–120 viruses reported till date.
File system support	Ext2, Ext3, Ext4, Reiser FS, Xfs, Btrfs, FAT, FAT32, NTFS	jfs, gpfs, hfs, hfs+, ufs, xfs, zfs format
Architectures	Originally developed for Intel's x86 hardware, ports available for over two dozen CPU types including ARM	is available on PA-RISC and itanium machines. Solaris also available for x86/x64 based systems. OSX is Power PC (10.0-10.5)/×86(10.4) / × 64 (10.5-10.8)
Examples	Ubuntu Fedora red hat, debia, Archlinux, Android etc	OS X, Solaris, All linux
Processors	Dozens of different kinds.	x86/x64, sparc, Power, itanium, PA-RISC, powerPC and many others.
Text mode interface	BASH (Bourne Again Shell) is the Linux default shell. It can support multiple command interpreters.	Originally the Bourne Shell. Now it's compatible with many others including BASH, Korn & C.

13.10 Usage of Linux and Unix

Linux OS is great of small to medium sized operations, and today it is also used in alrge enterprises where UNIX was considered previously as the only option. A few years back, Lunix was considered as an interesting acedemic project, but most big enterprises where networking and multiple user computing are the main concerns; people didn't consider Linux as an option. But today, with major software vendors porting their applications to Linux, and as it can be freely distributed, the OS has entered the mainstream s a vaible option for Web serving and office applications.

But there are some circumstances where UNIX is the obvious choice, or used to be. If an enterprise used massive symmetric multiprocessing systems, or systems with more than eight CPUs, they needed to run UNIX in the past. UNIX was far more capable in handling all the processes more effectively han Linux. However since 2004 more of the world's biggest supercomputers now run Linux than unix. Since 2011 Linux powers over 90% of the top 500 servers.

13.11 Linux-Unix Differences in Cost & Distribution

Linux can be freely distributed, as it is an open Source OS, So anyone can get a copy of Linux from books, magazines, or from the internet also. For server versions, organizations typically pay distributors for a support contract, not the software. The major distributors are RED HAT, Mandrake. For server hardware, IBM, HP, Dell are the major ones.

UNIX is costly as compared to Linux; the midrange UNIX servers are priced in between $ 25,000 and $ 249,999 (including hardware). The major distributors are HP, IBM. A high end UNIX server can cost up to $ 500,000. According to IDC, Gartner, IBM is the market leader in UNIX servers, HP is in 2nd position and SUN is in the third position.

Commercial UNIX is usualy custom written for each system, making the original cost quite high, whereas Linux has base packages also. In this respect, Linux is closer in its model to Windows than a commercial UNIX OS is. At the time of purchasing a UNIX server, users get a Vendor assistance plan on setting up and configuring the system. But with Linux, Vendor support must be purchased separately.

13.12 Threats and Security : Unix vs. Linux

Both of the operating systems are vulnerable to bugs but Linux is far more responsive in dealing with the threats. Linux incorporated many of the same characteristics and functions found in UNIX, including the segmentation of the user domain in a multi-user environment, the isolation of tasks in a multi-tasking environment, a password system that can be encrypted and/or located remotely and much more. As Linux is an open system OS, the bugs can be reported by anyone in the user/developers forum, and within days it can be fixed. But for UNIX, this is not the case, and user has to wait for a while, to get the proper bug fixing patch. The open source community delivers faster because it does not have to go through the endless development cycles of commercial-based operating systems.

At the same time, as an open source operating system, it is supported by tens of thousands of developers worldwide. To reiterate, this allows for better innovation and quicker-to-market features than anything UNIX can provide.

13.13 Market and Future of Linux and Unix

According to International Data Corp. (IDC). Linux has grown faster than any other server

OS over the past few years. Linux user base is estimated to be about more than 25 million machines, compared to 5.5 million for combined UNIX installations.

13.14 Windows 2003 server

Server install step by step. This will give you an idea on how to do. This when and if the time arises for you to either reinstall the operating system, install for the first time or upgrade. Your present operating system. This install guide is fully graphical and will take you through each step of the installation.

What you will require to do this walk through is

1. A windows 2003 net server CD
2. A computer with CD-ROM access.

Windows server 2003, standard additions delivers intelligent file and printer sharing more secure internet connectivity centralized desktop policy management and web solution that connect employees partners.

Key features : Update management

Remote access

Server hardware support

Application verification

File services

Assisted support

Directory services.

Windows server, provide a comprehensive server platform that is easy to deploy manage & use help business to lower ITC. Window server, helps organization achieve the greatest return on their sower.

13.15 Difference betweenWindows Server 2012 and Window 8

When talking about microsoft platform, the organization develops two types of operating systems namely client operating systems and network operating systems.

13.15.1 Client Operating System (COS)

The client operating systems that Microsoft develops are the ones that can be installed on a standalone PC or virtual machine, and can also be connected to a network infrastructure in any organization. The client operating system are capable of working as the network clients, and in almost all cases they are eligible to receive the services that the server in the network infrastructure offers.

The client operating systems are not powerful enough to provide their own services, and therefore they cannot be replaced by a full-fledged servers. Also, the client operating systems are comparatively lightweight and do not require too much of processors, memory, or other hardware resources as is the case with the network operating system (servers).

Some examples of the client operating systems are :

1. Disk Operating System or DOS
2. Windows 3.1
3. Windows 3.11
4. Windows 95
5. Windows 98
6. Windows ME
7. Windows 2000 professional
8. Windows XP
9. Windows Vista
10. Windows 7
11. Windows 8

13.15.2 Network Operating System (NOS)

A network operating system is an OS that has been developed by the microsoft to provide the services to all the client computers (client operating systems) that a network infrastructure has. It is the Active Directory Domain Services (AD DS) and some other services that differentiate a network operating System from a client operating system.

When the Active Directory Domain Services (AD DS) are installed on a network operating system, the OS becomes a full-fledged domain controller that then becomes capable of managing the entire domain and the client computers that the domain has from a central location.

With the help of the Active Directory Domain Services, the administrators can implement centralized security policies, which are then applicable on all the client computers that are added to the active directory Domain.

Some examples of network operating system that microsoft has developed are :

1. Windows 2000 Server (Entire Family)
2. Windows server 2003 (Entire Family)
3. Windows Server 2008 (Entire Family)
4. Windows Server 2012 (Entire Family)

Since the client operating systems are not required to offer their services to othr computers in network, they can be installed on a normal computer. On the other hand, because the network operating systems are especially installed/deployed in a network so that they can provide the services to other client computers that the network has, they must be installed on a machine that has decent hardware configuration.

13.16 Difference between window 7 and window 8

Windows 7 and Windows 8 both are operating System developed by Microsoft. Windows 8 is the latest Operating System from Microsoft with has got lot of new features are being discussed on the web. Following are the point wise **differences between microsoft Windows 7 and Windows 8.**

Windows 7	Windows 8
Windows 7 is the latest operating system released by microsoft.	Windows 8 is the upcoming operating System by Microsoft.
Windows 7 was released to manufacturing on July 22, 2009	Window 8 will be released in sometime in 2012. It was announced at the 2011 Consumer Electronics Show in Las Vegas.
Windows 7 codename was Blackcomb	Windows 8 codename was Jupiter
It includes new features like touch and handwriting recognition, supoort for virtual hard disks, improved performance on multicore processors, improved boot performance, Direct Access, kernel improvements, UAC (User Account Control), Windows Action Center, Jumplist, etc.	It Supports System on a chip (SoC) and mobile ARM processors
It started using Windows Aero.	It includes new features like Hybrid Booth which uses advanced hibernation functionality far faster shutdown times, portable Workspace, ability to install Window 8 on a USB Storage, Modern Task Manager, native ISO mounting, Modern Reader to Read PDF files, etc.,
It removed some features (present in Vista) like classic start menu, taskbar features, windows explorer and windows media player features.	It started using windows metro user interface.
Minimum hardware requirements for running Windows 7 32-bit are 1 GHz × 86 processor, 1 GB Ram, 16 GB Hard Disk space and DVD Drive.	
Maximum hardware requirements for running windows 7 64-bit are 1 GHz X 86–64 processor, 2 GB Ram, 20 GB hard Disk space and DVD Drive.	

13.17 Difference between Window and Linux

	Linux	Windows
What is it ?	Linux is an example of Open Source software development and a Free Operating System (OS). In the case of Peach OSI the OSI stands for Open Source Initiative.	Windows is the family of operating systems (OS) from Microsoft, which is the with out a doubt the most famous OS in the world.
Cost	Linux and thus peach OSI can be freely distributed, downloaded freely, distributed through magazines, books etc. There are priced versions for Linux also, but they are normally cheaper than Windows.	For desktop or home use, Windows can be expensive. A single copy can cost around $ 50 to $ 450 depending on the version of Windows you want to use.
Manufacturer	The Linux kernel is developed by the community. Linus Torvalds, the original author of Linux, oversees things.	Microsoft created the Windows operating system, but allows other computer manufactures to distribute their own computers with Windows pre-installed.
Threat detection and solution	In case of Linux, threat detection and solution is very fast, as Linux is mainly community driven and whenever any Linux user posts any kind of threat, several developers start working on it from different parts of the world, immediately.	After detecting a major threat in Window OS, Microsoft generally releases a patch that can fix the problem and it can take more than 2/3 months. Sometimes sooner, Microsoft releases patches and updates weekly.
Examples	Ubuntu, Fedora, Red Hat, Debian, Archlinux, Android, Peach OSI etc.	Windows 8, 8.1, 7, Vista, XP
Processors	Dozens of different kinds.	Limited but will run on most (80%)
Company/ developer	Linux Torvalds	Microsoft
Source Model	Source model	Closed/Shared source

Exercise

Part I (Very Short Answer)

1. What is windows ?
2. Define the linux ?
3. What features of window 7 ?
4. What is commands ?

Part II (Short Answer)

1. Explain window server ?
2. Difference between unix and linux ?
3. Difference between window 7 and window 8 ?

Part III (Long Answer)

1. What is the purpose of master file table in windows ?
2. Explain how unix manages a process address space ?
3. List and complex the key characteristics of window lunix & linux operating system.
4. What features of Unix is details ?

□□□

Unitwise Exercises

Unit-I

1. What is the fundamental difference between network and distributed operating system ?

Ans. : The basic difference between network operating system and distributed operating system is that network operating system allows the user to access remote computer system only if he/she knows the address of those computer system, whereas distributed operating system is used by the user to share data and programs across many computers that are networked together. Here, the user need not know the address of these computer from which the data should be fetched.

2. Explain the role of operating system in details ?

Ans. : When the power to a computer is turned on, the first program that runs is usually a set of instructions kept in the computer's ROM that examines the system hardware to marke sure every thing is functioning property. This post on self test (post) check's the CPU, memory and basic input output systems for errors and stores a the result in a special memory location. Once the post has successfully completely, the software loaded in ROM.

The operating system's tasks in the msot general fall into six categories.

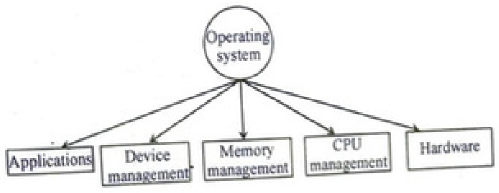


Figure 1 : Role of an operating system

3. Describe the computer system operating in details ?

Ans. : A computer functional diagram is CPU, memory and number of devices make our modern general purpose computer system.

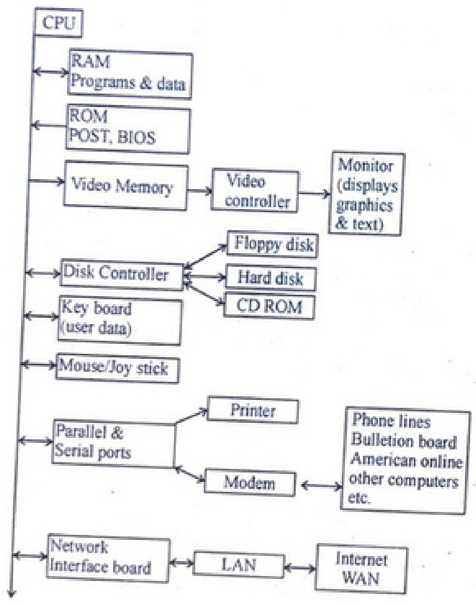


Figure 2 : Computer functional Block diagram

4. Explain the Direct Memory Access with diagrams ? What was the necessity of DNA ?

Ans. : Direct Memory Access is another advanced mechanism where I/O device is given full freedom to transfer block of data to/from memory with out going through the CPU. In figure explains this concept. This method is known to increase the speed of overall computer operation.

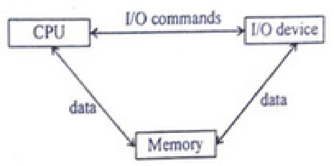


Figure 3 :

Necessity of DMA : If the I/O operation becomes very fast then CPU faces a situation in which it receives more and more interrupts with in a given time interval and thus CPU is left with hardly any time to execute a process. I/O operation using DMA :

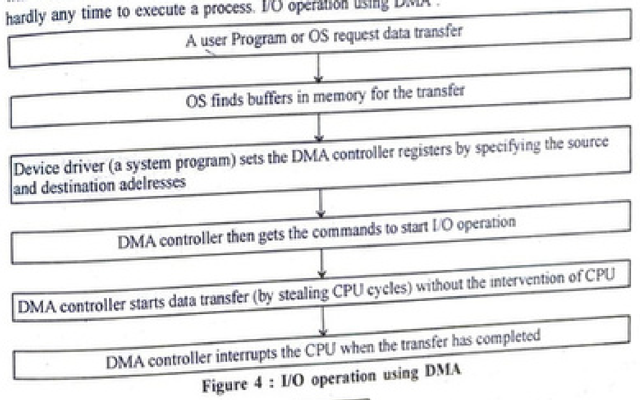

A user Program or OS request data transfer
OS finds buffers in memory for the transfer
Device driver (a system program) sets the DMA controller registers by specifying the source and destination adelresses
DMA controller then gets the commands to start I/O operation
DMA controller starts data transfer (by stealing CPU cycles) without the intervention of CPU
DMA controller interrupts the CPU when the transfer has completed

Figure 4 : I/O operation using DMA

Unit-2

1. List the differences among short-term, medium term and long term Schedulers ?

Ans. : (i) Short-term scheduler selects from jobs in memory those jobs that are ready to executs & allocates the CPU to them.

(ii) Medium term Scheduler used especially with time sharing as an intermediate Scheduling level. A swapping scheme is implemented to remove partially run programs from memory and rain state them later to continue where they left off.

(iii) Long term scheduler determines which jobs should be brought into memory (from secondary storage device) for processing.

2. Differentiate between user level and kernel level threads ?

Ans. : (i) User level threads are unknown by the kernel whereas kernel is aware of the kernel threads.

(ii) Kernel thread used to be associated with a process whereas every user thread belongs to a process.

3. Justify that multi threading can take the best advantage of multiple processors ?

Ans. : Multiple processors can be best utilized when multi threading is implemented because in multi threading, multiple threads should execute simultaneously and this simultaneous execution can be done efficiently in a multi processing environment with each thread being executed on a separate process. This will in crease the speed of a multi threaded program. For example, a web server that sevices each request in a separate thread if uses multiple processors to execute these threads, can become very fast as compared to when it uses only a single processor.

4. Consider a system with a set of processes P_1, P_2 and P_3 and their CPU burst times, priorities and arrival times being mentioned as below :

Process	CPU burst time	Arrival time	Priority
P_1	5	0	2
P_2	15	1	3
P_3	10	2	1

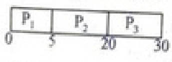
Assceme 1 to be the highest priority and calculate the following :

1. Average waiting time using FCFS, SJF (preempture and non-preemptive) and priority (preemptive and non-preemptive) scheduling mechanisms.

2. Average turn around time using FCFS, SJF (preemptive & non preemptive) and priority (preemptive and non preemptive) scheduling mechanisms.

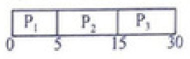
3. Assume time quantum to be 2 units of time. Calculate average waiting time and average turn around time using round robin scheduling.

Ans. : 1. Gantt chart for the different scheduling algorithms will be :



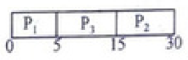
FCFS

Average waiting time $= \frac{0+5+20}{3} = \frac{25}{3}$ units of time



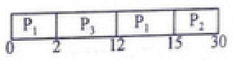
SJF preemptive

Average waiting time $= \frac{0+5+15}{3} = \frac{20}{3}$ units of line



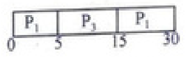
SJF non-preemptive

Average waitng time $= \frac{0+5+15}{3} = \frac{20}{3}$ units of time



Priority (preemptive)

Average waiting time $= \frac{0+2+10+15}{3} = \frac{27}{3}$ Units of time



Priority non preemptive

Average waiting time $= \frac{0+5+15}{3} = \frac{20}{3}$ units of time

2. FCFS

Turn around time = time of completion of job-time of submission of job

turn around time for	P_1	$= 5 - 0 = 5$
turn around time for	P_2	$= 20 - 1 = 19$
turn around time for	P_3	$= 30 - 2 = 28$

average turn around time $= \frac{5+19+28}{3} = \frac{52}{3}$ units of time

SJF (preemptive)

Average turn around time $= \frac{5+13+28}{3} = \frac{47}{3}$ units of time

SJF (non-preemptive)

Average turn around time $= \frac{5+13+28}{3} = \frac{47}{3}$ Units of time

SJF priority (Preemptive)

Turn around time for $P_1 = 15 - 0 = 15$

Turn around time for $P_2 = 30 - 1 = 29$

Turn around time for $P_3 = 12 - 2 = 10$

Average turn around time $= \frac{15+29+10}{3} = \frac{54}{3}$ units of time

SJF (non-preemptive)

Average turn around time $= \frac{5+29+13}{3} = \frac{47}{3}$ units of time

3. RR scheduling

P₁	P₂	P₃	P₁	P₂	P₃	P₁	P₂	P₃	P₂	P₃	P₂	P₃	P₃	P₃	P₃

0 2 4 6 8 10 12 13 15 17 19 21 23 25 27 29 30

waiting time for $P_1 = 0 + 4 + 4 = 8$

waiting time for $P_2 = 2 + 4 + 3 + 2 + 2 = 13$

waiting time for $P_3 = 4 + 4 + 3 + 2 + 2 = 15$

Average waiting time $= \frac{8+13+15}{3} = \frac{36}{3} = 12$ units of time

Average turn around time $= \frac{(13-0)+(23-1)+(30-2)}{3}$

$= \frac{47}{3}$ Units of time

$= \frac{13+22+28}{3} = \frac{63}{3} = 21$ units of time.

5. Explain Deadlock characteistics ?

Ans. : When a deadlock will occur. If these four conditions occur simultaneously, then we have deadlock :

(i) Mutual Exclusion : A resource can be used by only one process at a time.

(ii) Hold and wait : At least one process is holding a resource and needs to acquire other resources that are being hold by other processes.

(iii) No preemption : If a process holds a resources, it can not be preempted.

(iv) Circular wait : There must exist a set (ρ_0, ρ_1,.......ρ_n) of waiting proceses such that ρ_0 is waiting for a resource that is held by ρ_1, ρ_1 is waiting for a resource held by ρ_2.... and so an and finally ρ_n is waiting for a resource held by ρ_0.

6. List three overall strategies in handing deadlocks.

Ans. : (a) Ensure system will never enter a dealock state.

(b) Allow deadlocks, but devise schemes to recover from them.

(c) Pretend deadlock donot happen.

Unit-3

1. Explain the difference between logical and physical addresses.

Ans. :

- Logical addresses are those generated by user programs relative to location 0 in memory.
- Physical addresses are the actual addresses used to fetch and store data in memory.

2. Explain the difference between internal and external fragmentation.

Ans. : Internal fragmentation is the area in a region or a page that is not used by the job occupying that region or page. This space is unavailable for use by the system until that job is finished and the page or region is released.

3. Why are segmentation and paging sometimes combined into one scheme?

Ans. : Segmentation and paging are often combined in order to improve upon each other. Segmented paging is helpful when the page table becomes very large. A large contiguous section of the page table that is unused can be collapsed into a single segment table entry with a page table address of zero. Paged segmentation handles the case of having very long segments that require a lot of time for allocation. By paging the segments, we reduce wasted memory due to external fragmentation as well as simplify the allocation.

4. List ways of reducing the context switch time.

Ans. : a. Minimize the amount of memory of a process to be swapped.

b. Increase the speed of the disk used for swapped-out processes.

c. Overlap swapping and program execution.

5. What advantages does segmentation have over paging?

Ans. : a. Can associate protection with critical segments, without protecting other items.

b. Can share code and data with other users, tightly, so that only the desired portions are shared.

6. When do page faults occur? Describe the actions taken by the operating system when a page fault occurs.

Ans. : A page fault occurs when an access to a page that has not been brought into main memory takes place. The operating system verifies the memory access, aborting the program if it is invalid. If it is valid a free frame is located and I/O requested to read the needed page into the free frame. Upon completion of I/O, the process table and page table are updated and the instruction is restarted.

7. List benefits of having only part of a program in memory.

Ans. : a. Simplifies the programming task.

b. More user-programs can run concurrently in newly freed memory.

c. Less swapping of entire programs, thus less I/O.

8. List six steps to process a page fault.

Ans. :

a. Check page-frame table in PCB. If address invalid, abort program.

b. If address valid, but its page not current, bring it in.

c. Find free frame.

d. Request I/O for the desired page.

e. Update the page-frame table in PCB.

f. Restart the instruction.

9. List ways to implement LRU, to determine which page is victim.

Ans. :

a. Use hardware counter and/or clock. Page table contains time that is updated after each use. Page with oldest time is victim.

b. Use stack of page numbers. On each use, page number is moved to top of stack, and rest are shifted down. Bottom number is the victim.

c. Use a reference bit that is set after use, but cleared after time intervals. Victim is one with bit cleared.

d. Use a reference byte. After given time interval, the byte in each page is shifted right, but page in use has sign-bit set. Victim is page with lowest-valued byte.

10. Describe the page-to-frame translation.

Ans. : Logical address is split into page offset and page number. Scan page table for page number; corresponding entry is the frame number, which is combined with page offset to give physical address.

Unit 4

1. Explain the purpose of the open and close operations.

Ans. :

- The open operation informs the system that the named file is about to become active.
- The close operation informs the system that the named file is no longer in active use by the user who issued the close operation.

2. What is a file?

Ans. : A named collection of related data defined by the creator, recorded on secondary storage.

3. What is a sequential file?

Ans. : A file that is read one record or block or parameter at a time in order, based on a tape model of a file.

4. What advantages are there to this two-level directory?

Ans. : Users are isolated from each other. Users have more freedom in choosing file names.

5. What disadvantages are there to this two-level directory?

Ans. : Without other provisions, two users who want to cooperate with each other are hampered in reaching each other's files, and system files are inaccessible.

6. How do we overcome the disadvantages of the two-level directory?

Ans. : Provide links from one user directory to another, creating path names; system files become available by letting the command interpreter search your directory first, and then the system directory if file needed is not in first directory.

7. How can we protect files on a single-user system?

Ans. : a. Hide the disks.

b. Use file names that can't be read.

c. Backup disks.

d. On floppies, place a write-disable-tab on.

8. List four ways systems might provide for users to protect their files against other users.

Ans. :

a. Allowing user to use unprintable characters in naming files so other users can't determine the complete name.

b. Assigning password(s) to each file that must be given before access is allowed.

c. Assigning an access list, listing everyone who is allowed to use each file.

d. Assigning protection codes to each file, classifying users as system, owner, group, and world (everyone else).

9. Explain first-fit, best-fit, and worst-fit methods of allocating space for contiguous files.

Ans. :

- First-fit: Scan available blocks of disk for successive free sectors; use the first area found that has sufficient space; do not scan beyond that point.
- Best-fit: Search for smallest area large enough to place the file.
- Worst-fit: Search for largest area in which to place the file.

10. What information is needed for disk I/O?

Ans : a. Whether input or output.

b. Disk address.

c. Memory address.

d. Amount of data to be moved.

11. What are the main differences between capability lists and access lists?

Ans. : An access list is a list for each object consisting of the domains with a nonempty set of access rights for that object. A capability list is a list of objects and the operations allowed on those objects for each domain.

12. Capability lists are usually kept within the address space of the user. How does the system ensure that the user cannot modify the contents of the list?

Ans. : A capability list is considered a "protected object" and is accessed only indirectly by the user. The operating system ensures the user cannot access the capability list directly.

Unit 5

1. To build a robust distributed system, you must know what kinds of failures can occur.

Ans. :

a. List possible types of failure in a distributed system.

b. Specify which items in your list also are applicable to a centralized system.

2. What is the need-to-know principle? Why is it important for a protection system to adhere to this principle?

Ans. : A process may access at any time those resources that it has been authorized to access and are required currently to complete its task. It is important in that it limits the amount of damage a faulty process can cause in a system.

3. Make a list of security concerns for a computer system for a bank. For each item on your list, state whether this concern relates to physical security, human security, or operating system security.

Ans. : In a protected location, well guarded: physical, human. Network tamperproof: physical, human, operating system. Modem access eliminated or limited: physical, human. Unauthorized data transfers prevented or logged: human, operating system. Backup media protected and guarded: physical, human. Programmers, data entry personnel, trustworthy: human.

4. What are the advantages of encrypting data stored in the computer system?

Ans. : Encrypted data are guarded by the operating system's protection facilities, as well as a password which is needed to decrypt them. Two keys are better than one when it comes to security.

5. How were the design goals of UNIX different from those of other operating systems dur- ing the early stages of UNIX development?

Ans. : Rather than being a market-oriented operating system, like MULTICS, with definite goals and features, UNIX grew as a tool to allow Thompson and Ritchie to get their research

done at Bell Labs. They found a spare PDP-11 system and wrote UNIX to help them with text-processing requirements. It therefore exactly suited their personal needs, not those of a company.

6. What type of operating system is Windows XP? Describe two of its major features.

Ans. : A 32/64 bit preemptive multitasking operating system supporting multiple users.

i. The ability automatically to repair application and operating system problems.

ii. Better networking and device experience (including digital photography and video).

7. List the design goals of Windows XP. Describe two in detail.

Ans. : Design goals include security, reliability,Windows and POSIX application compatibility, high performance, extensibility, portability and international support.

i. Reliability was perceived as a stringent requirement and included extensive driver verification, facilities for catching programming errors in user-level code, and a rigorous certification process for third-party drivers, applications, and devices.

ii. Achieving high performance required examination of past problem areas such as I/O performance, server CPU bottlenecks, and the scalability of multithreaded and multiprocessor environments.

8. What are the responsibilities of the I/O manager?

Ans. : The I/O manager is responsible for file systems, device drivers, and network drivers. The I/O manager keeps track of which device drivers, filter drivers, and file systems are loaded and manages buffers for I/O requests. It furthermore assists in providing memorymapped file I/O and controls the cache manager for the whole I/O system.

9. How does NTFS handle data structures? How does NTFS recover from a system crash? What is guaranteed after a recovery takes place?

Ans. : In NTFS, all file-system data structure updates are performed inside transactions. Before a data structure is altered, the transaction writes a log record containing redo and undo information. A commit record is written to the log after a transaction has succeeded. After a crash the file system can be restored to a consistent state by processing the log records, first redoing operations for committed transactions and undoing operations for transactions that did not successfully commit. This scheme does not guarantee that user file contents are correct after a recovery, but rather that the file-system data structures (file metadata) are undamaged and reflect some consistent state that existed before the crash.

10. What is RPC ? Why it is important ?

Ans. : Remote Procedure Call (RPC) is a protocol that one program can use to request a service from a program located in another computer in a network without having to understand network details. (A procedure call is also sometimes known as a function call or a subroutine call.) RPC uses the client/server model. The requesting program is a client and the service-providing program is the server. Like a regular or local procedure call, an RPC is a synchronous operation requiring the requesting program to be suspended until the results of the remote procedure are returned. However, the use of lightweight processes or threads that share the same address space allows multiple RPCs to be performed concurrently.

■■■

BIBLIOGRAPHY

[Arnold et al. 1974]

Arnold, J.S., et al. Design of tightly-coupled multiprocessing programming. IBMSJ, Vol. 13, No. 1, 1974, pp. 61-87.

[Abernathy et al. 1973]

Abernathy, D.H., et al. Survey of Design Goals for Operating Systems. In three parts. OSR Vol. 7, No. 2, April 1973, pp. 29-48; OSR Vol. 7, No. 3, July 1973, pp. 19-34; OSR Vol. 8, No. 1, Jan. 1974, pp. 25-35.

[Almasi & Gottleib 1994]

Almasi, G.S., & Gottlieb, A., Highly Parallel Computing. Benjamin/Cummings, Redwood City, CA, 1994.

[Andrews et al. 1987]

Andrews, G.R., et al. The Design of the Saguaro Distributed Operating System. TSE, Vol. SE-13, No. 1, Jan. 1987, pp. 104-118.

[Bach 1986]

Bach, M.J. The Design of the UNIX System. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[Barak & Litman 1985]

Barak, A. & Litman, A. MOS: A Multicomputer Distributed Operating System. SPE, Vol. 15, No. 8, Aug. 1985, pp. 725-737.

[Bierman 1984]

Bierman, K.H. The TurboDOS Operating System. ACM SIGSMALL Newsletter, Vol. 10, No. 3, Aug. 1984, pp. 24-34.

[Bic & Shaw 1988]

Bic, L., and Shaw, A. The Logical Design of Operating Systems (second edition). Prentice-Hall, Englewood Cliffs, NJ, 1988.

[Buzen & Gagliardi 1973]

Buzen, J.P., and Gagliardi, U.O. The Evolution of Virtual Machine Architecture. NCC, Vol. 42, 1973, pp. 291-300.

[Buckley 1986]

Buckley, F.J. An Overview of the IEEE Computer Society Standards Process. Proceedings of the IEEE Computer Standards Conference, 1986, pp. 2-8.

[Brown et al. 1984]

Brown, R.L., et al. Advanced Operating Systems. COMPUTER, Vol. 17, No. 10, Oct. 1984, pp. 173-190.

[Coffman & Denning 1973]

Coffman, E.G., and Denning, P.J. Operating Systems Theory. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[Cutler et al. 1976]

Cutler, D.N., et al., The Nucleus of a Real-Time Operating System. ACM 1976, pp. 241-246.

[Cheriton 1984]

Cheriton, D.R. The V Kernel: A Software Base for a Distributed System. SOFTWARE, Vol. 1, No. 2, April 1984, pp. 19-42.

[Cheriton 1982]

Cheriton, D.R. The Thoth System: Multiprocess structuring and Portability. North Holland, Amsterdam, 1982.

[Deitel 1984]

Deitel, H.M. An Introduction to Operating Systems (revised first edition). Addison-Wesley, Reading, MA, 1984.

[Digital 1981b]

RSX-11M Guide to Writing an I/O Driver. Digital Equipment Corp., Maynard, MA, 1981.

[Feder 1984]

Feder, J. The Evolution of UNIX System Performance. BLTJ, Vol. 63, No. 8, Part 2, Oct. 1984, pp. 1791-1814.

[Frank & Theaker 1979a]

Frank, G.R., and Theaker, C.J. The Design of the MUSS Operating System. SPE, Vol. 9, No. 8, Aug. 1979, pp. 599-620.

[Georg et al. 1984]

Georg, D.D., et al. A General-Purpose Operating System Kernel for a 32-bit Computer System. HPJ, Vol. 35, No. 3, March 1984, pp. 28-34.

[Grampp & Morris 1984]

Grampp, F.T., and Morris, R.H. UNIX Operating System Security. BLTJ, Vol. 63, No. 8, Part 2, Oct. 84, pp. 1649-1672.

[Gaglianello & Katseff 1985]

Gaglianello, R.D., and Katseff, H.P. Meglos: an Operating System for a Multiprocessor Environment. Proceedings of the 5th International Conference on Distributed Computing Systems, Denver, CO, May 1985, pp. 35-42.

[Goldberg 1973]

Goldberg, R.P. Architecture of Virtual Machines. NCC 1973, pp. 309-318.

[Goldberg 1974]

[illegible]

Goldberg, R.P. A Survey of Virtual Machine Research. COMPUTER, Vol. 7, No. 6, June 1974. pp. 34-45.

[Havendar 1968]

Havendar, J.W. Avoiding Deadlock in Multitasking Systems. IBMSJ, Vol. 7, pp. 74-84, 1968.

[Habermann 1969]

Habermann, A.N. Prevention of System Deadlocks. CACM, Vol. 12, No. 7, July 1969, pp. 373-385.

[Habermann 1976]

Habermann, A.N. Introduction to Operating System Design. Science Research Associates, Chicago, 1976.

[Interdata 1974]

Interdata Corp. Interdata Announces 32-bit MultiTasking OS (product announcement). COMPUTER, Vol. 7,N7, July 1974, p. 57.

[IBM 1963]

IBSYS Operating System. C28-6248, IBM Corp., 1963.

[Joseph et al. 1984]

Joseph, M., et al. A Multiprocessor Operating System. Prentice-Hall International, 1984.

[Janson 1985]

Janson, P.A. Operating Systems: Structures & Mechanisms. Academic Press, New York, 1985.

[Kernighan & Pike 1984]

Kernighan, B., and Pike, R. The UNIX Programming Environment. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[Kildall 1981]

Kildall, G. CP/M: A Family of 8 and 16-bit Operating Systems. BYTE Vol. 6, No. 6, June 1981, pp. 216-232.

[Lynch 1972]

Lynch, W.C. An Operating System designed for the Computer Utility Environment. In [Hoare & Perrott 1972], pp. 341-350.

[Lampson et al. 1981]

Lampson, B.W., Paul, M., and Siegert, H.J., eds. Distributed Systems: Architecture and Implementation. Springer-Verlag, New York, 1981.

[Meyer & Seawright 1970]

Meyer, R.A., and Seawright, L.H. A Virtual Machine Time Sharing System. IBMSJ, Vol. 9, No. 3, 1970, pp. 199-218.

[Milenkovic 1987]

Milenkovic, Milan. Operating Systems: Concepts and Design. McGraw-Hill, New York, 1987.

[Nutt 1977]

Nutt, G.J. A Parallel Processor Operating System Comparison. TSE, Vol. SE-3, No. 6, Nov. 1977, pp. 467-475.

[Peterson & Silberschatz 1985]

Peterson, J., and Silberschatz, A. Operating System Concepts (second edition). Addison-Wesley, Reading, MA, 1985.

[Peterson 1981]

Peterson, G.L. Myths about the Mutual Exclusion Problem. Information Processing Letters, Vol. 12, No. 3, June 1981, pp. 115-116.

[Ousterhout et al. 1980]

Ousterhout, J.K., et al. Medusa: An Experiment in Distributed Operating System Structure. CACM, Vol. 23, No. 2, Feb. 1980, pp. 92-105.

[Ritchie & Thompson 1974]

Ritchie, D.M., and Thompson, K. The Unix Timesharing System. CACM, Vol. 17, No. 5, May 1974, p. 365.

[Raimondi 1976]

Raimondi, D.L. A Distributed Real-Time Operating System. IBMSJ, Vol. 15, No. 1, 1976, pp. 81-101.

[Svoboda 1981]

Svoboda, L. Performance Monitoring in Computer Systems: A Structured Approach. OSR, Vol. 15, No. 3, July 1981, pp. 39-50.

[Tanenbaum 1987]

Tanenbaum, A.S. Operating Systems: Design and Implementation. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[Turner 1986]

Turner, R.W. Operating Systems: Design and Implementation, Macmillan Publishing Co., New York, 1986

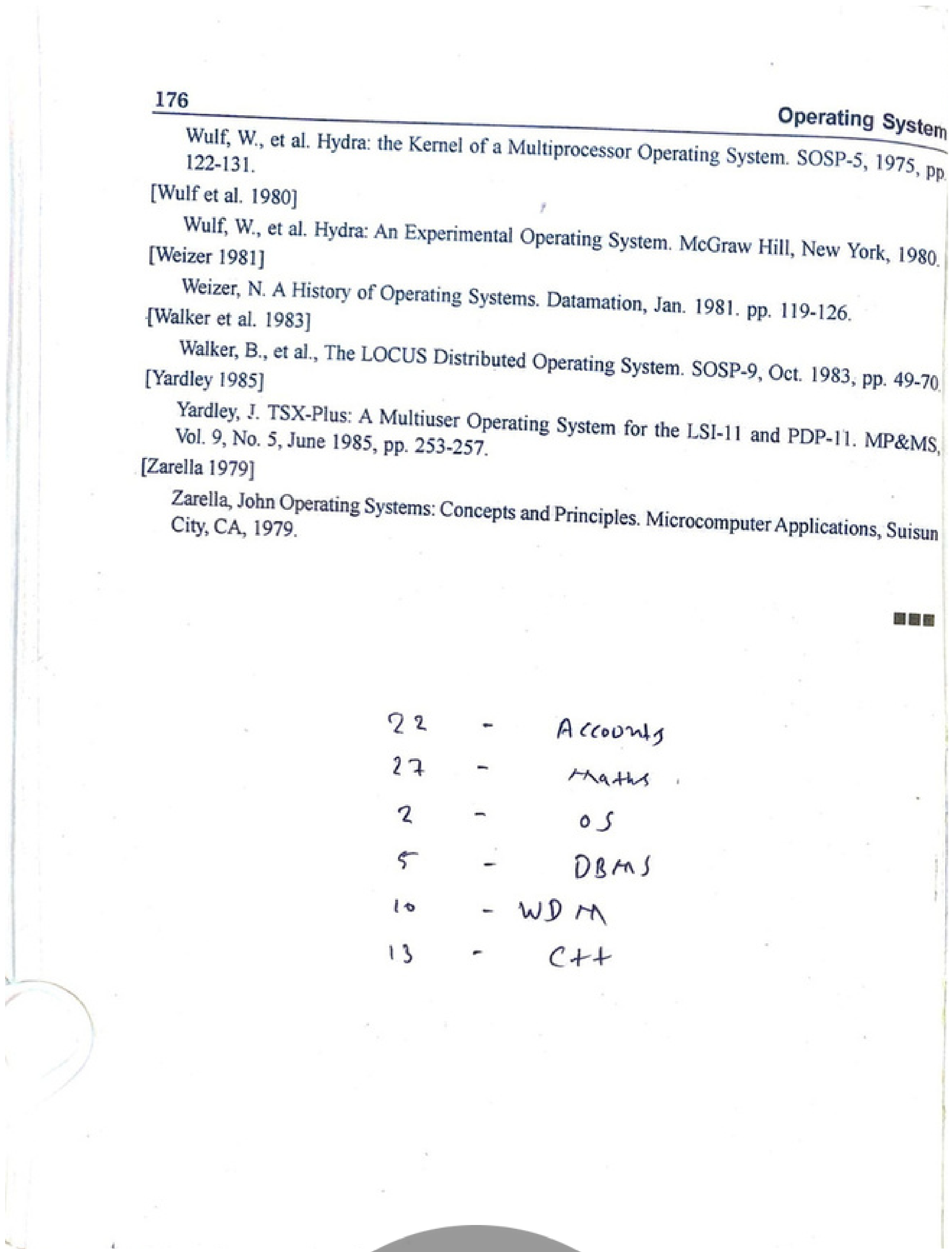
[van der Linden & Wilson 1980]

van der Linden, F., and Wilson, I. Real-time executives for microprocessors. MP&MS, Vol. 4, No. 6, Jul/Aug. 1980, pp. 211-218.

[Van Tassel 1972]

Van Tassel, D. Computer Security Management. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[Wulf et al. 1975]



Wulf, W., et al. Hydra: the Kernel of a Multiprocessor Operating System. SOSP-5, 1975, pp. 122-131.

[Wulf et al. 1980]

Wulf, W., et al. Hydra: An Experimental Operating System. McGraw Hill, New York, 1980.

[Weizer 1981]

Weizer, N. A History of Operating Systems. Datamation, Jan. 1981. pp. 119-126.

[Walker et al. 1983]

Walker, B., et al., The LOCUS Distributed Operating System. SOSP-9, Oct. 1983, pp. 49-70.

[Yardley 1985]

Yardley, J. TSX-Plus: A Multiuser Operating System for the LSI-11 and PDP-11. MP&MS, Vol. 9, No. 5, June 1985, pp. 253-257.

[Zarella 1979]

Zarella, John Operating Systems: Concepts and Principles. Microcomputer Applications, Suisun City, CA, 1979.

■■■

22	-	Accounts
27	-	Maths
2	-	OS
5	-	DBMS
10	-	WDM
13	-	C++