UNIT 2: MORE ON OPERATING SYSTEM

Unit Structure

2.0 Learning Objectives

After learning this unit, you will be able to understand:

- Basic of RPC arrangement
- Shared virtual memory
- Concept of Unix Processing
- About Unix Shell

2.1 Introduction

In that context, by allowing the programmer to access and to share "memory objects" without being in charge of their management, virtually shared memory systems want to propose a trade-off between the easy-programming of shared memory machines and the efficiency and scalability of distributed memory systems. We can say that a procedure is an arrangement of closed sequence of instructions which will be controlled through external source. In this case, the data approximations are travelled in all directions which indicate the flow of control. At last, the procedure call is an invention of a procedure.

2.2 Remote procedure Calls

RPC is an influential method for constructing distributed, client-server based request. It stands on extending the concept of conservative or local process calling, subsequently called as procedure that should not be present in the similar address space as the calling procedure. The two procedures possibly will be on the identical system, or they possibly will be on different systems with a network linking them. By means of RPC, programmers of distributed applications stay away from the details of interface with the network. The transport autonomy of RPC cut off the application from the physical as well as logical fundamentals of the data communications method in addition to allows the application to exercise a mixture of transports.

RPC is equivalent to a function call. Similar to a function call, as soon as an RPC is made, the calling opinion is passed to the remote procedure plus the caller waits for a reply to be come back from the remote process. Figure 2.1 shows the stream of activity that takes place all through an RPC call among two networked systems. The client creates a procedure call with the aim of sending a request to the server more over and will wait. The thread is blocked-up from processing in anticipation of either a reply is received or it timed out. When the appeal arrives, the server calls a dispatch routine with the intention of performing the requested service, furthermore sends the reply to the client. Following the RPC call is ended, the client program goes on. RPC purposely supports network applications.

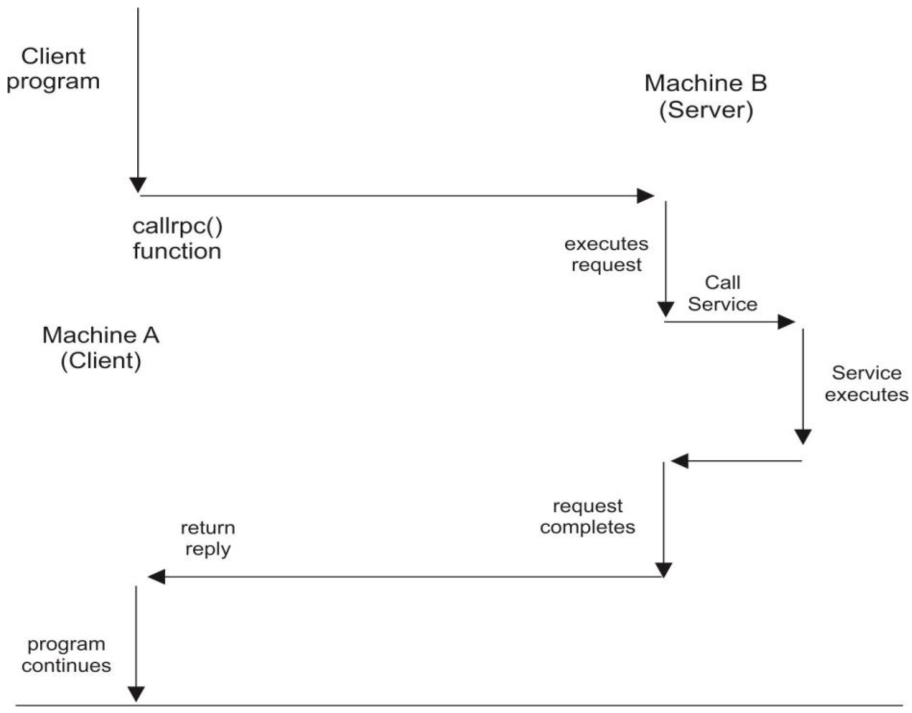

Fig 2.1 Remote procedure Calls

A remote process is exclusively acknowledged by the:-

- program number
- version number
- procedure number

The program number makes out a group of connected remote events, each of which has an exclusive procedure number. A program possibly will consist of one or more versions. Every version comprises of compilation of procedures which are accessible to be called remotely. Version numbers facilitate manifold versions of an RPC protocol to be obtainable at the same time. Every version includes a number of procedures to facilitate remotely. Each procedure has a procedure number.

It is studied that all RPC will a rises in the background of a thread which shows sequential amount of control flow through single execution all the time. It is found that the thread is created and handled by certain application code that is present inside an application thread.

The RPC usage makes use of application threads in order to give equal RPCs as well as RPC run time calls. It is found that an RPC client will be gathered by one or more client application threads which applies RPCs.

While calculating remote procedures, RPC server will employ single or many call threads to present RPC run-time system. In the beginning, the server application thread will brings about several simultaneous calls. The single threaded applications will carry out at least single call thread. It is found that an run-time system will produce call threads in server execution background as shown in Fig 2.2.

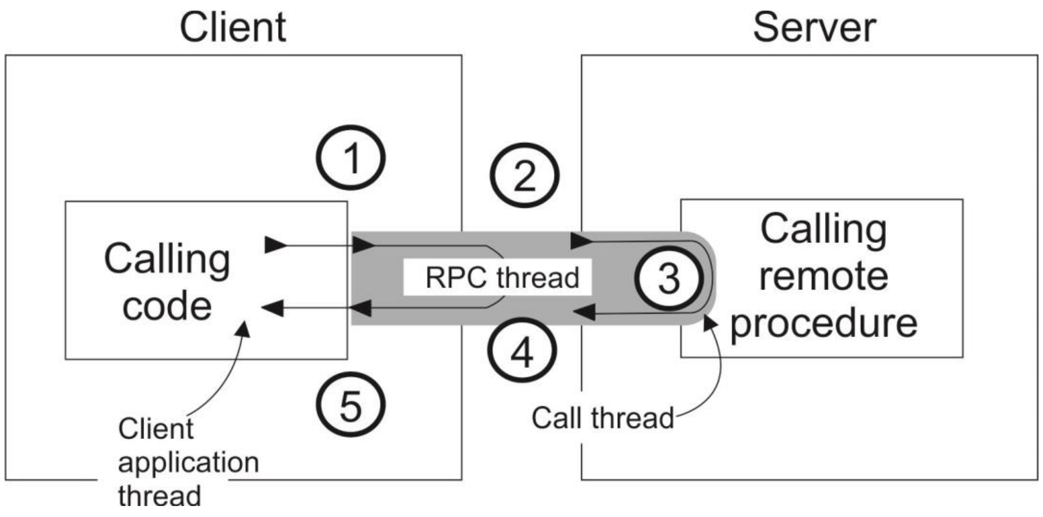

Fig 2.2 RPC arrangement

In the figure, the RPC will get expanded from one corner to corner client along with server execution. When, a client application thread calls a remote process, at that time, it will become the part of rational thread of execution, which will be located as RPC thread. Such types of thread behaves as rational assembly which carry certain portions of RPC that gets increased from one corner point to another corner point with fixed threads of execution in a network. The working part of RPC thread when in execution stage will cover:

- Starting of RPC thread in a client process since client application thread produces RPC.
- Expansion of RPC thread across the diagonals in network to server.
- Movement of RPC thread into call thread in case of remote procedure.
- Execution of remote procedure, where a call thread is the part of RPC thread.
- Giving the network to client by RPC thread.
- Precedence of call result and client application thread by RPC on arrival of RPC thread.

During the lack of RPC, the cancellation and location of thread with local working belong to similar framework. In the presence of RPC, the system will have a remote procedure where both local as well as fraction of cancelled thread's will work.

Check your progress 1

1. Remote procedure calls is a method for creating_____.

a. distributed request	c. Both a and b
b. client-server request	d. None

2. Remote procedure calls is equivalent to_______ call.

a. data	c. message
b. information	d. function

3. In RPC, the remote process is done by_________.

a. program number	c. procedure number
b. version number	d. all

4. RPC run-time system generates ______threads.

a. call	c. message
b. information	d. function

2.3 Distributed shared memory

A distributed shared memory is a method permitting the end-users procedure to right to use shared data with no inter-process communications. We can say that, the objective of DSM arrangement is to create inter-process communications see-through to end-users. It is implemented with the help of both hardware and software. With the idea of programming, two approaches have been studied:-

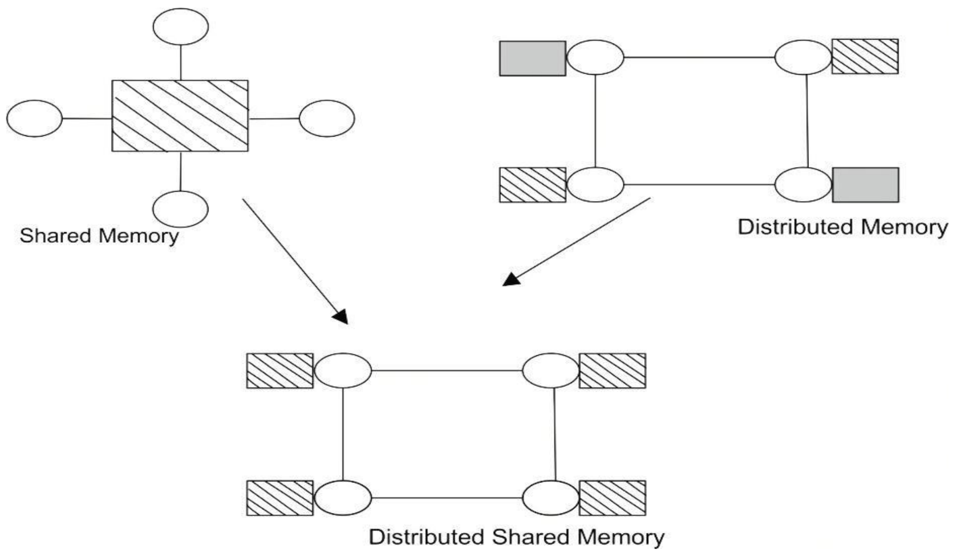

Fig 2.3 distributed shared memory

Shared virtual memory

This idea is very analogous to the well-known thought of paged virtual memory put into practice in mono-processor systems. The essential idea is to set all distributed memories jointly into a single broad address space. Such type of virtual memory has certain drawbacks as they do not allow to take into account the semantics of joint data as well as in this the data granularity is randomly fixed to several page size whatever the type as well as the actual size of the shared data might be. The programmer has no way to give information about such data.

Object DSM

In such class, the joint data such as objects that are variables having an access functions. In his purpose, the user has merely to describe which data (objects) are shared. The complete management of the collective objects (creation, access, modification) is handled by the DSM system. In contradictory of SVM systems which work at operating system layer, objects DSM systems in fact propose a programming model option to the classical message-passing.

In any case, executing a DSM system involves address problems of data position, data access, sharing and locking of data, data coherence. These problems are not definite to parallelism except they have connections through distributed or replicated databases management systems, networks, uniprocessor operating systems and distributed systems.

Methods of Achieving DSM

Hardware - It uses special network interfaces as well as cache coherence circuits.

Software - It modifies OS kernel additionally add a software layer between the operating system as well as in application.

Software DSM Implementation

- Page based –It uses system's virtual memory.
- Shared variable approach- It uses routines to access shared variables.
- Object based- It shares data within collection of objects and gives access to share data by object oriented discipline.

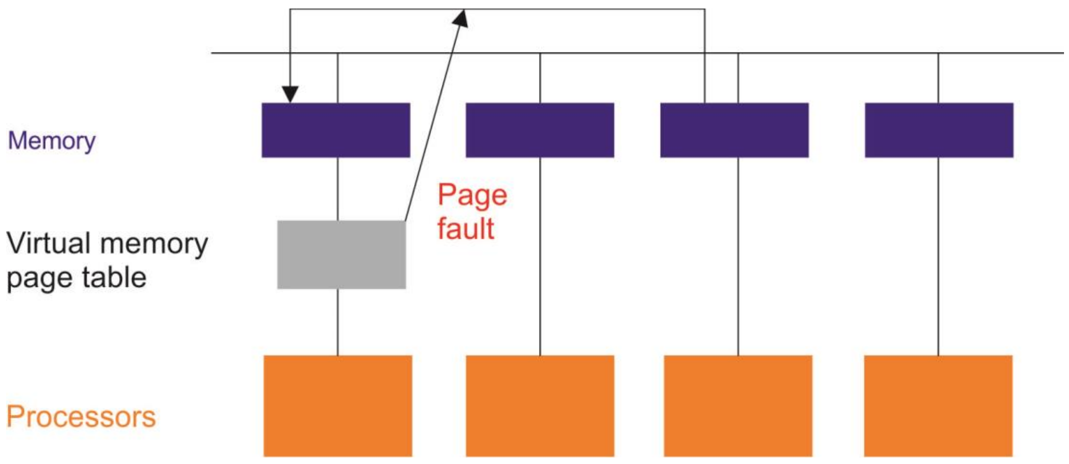

Fig 2.4 objects DSM systems

Advantages of DSM

- It is system scalable.
- It hides message passing and do not open definite sending messages among processes.
- It can use easy extensions to sequential programming.
- It handles difficult and big data bases without replication or sending data to processes.

Disadvantages of DSM

- It may acquire a performance penalty.
- It should be provided for protection against immediate admission to shared data.
- It has small programmer control over real messages individually generated.
- It has a problem in performance of particular that can be difficult.

Consistency Models used on DSM Systems

Release Consistency

An extension of weak consistency shown in fig 2.5 in which the synchronization operations have been specified-

- acquire operation - used before a shared variable or variables are to be read.
- release operation - used after the shared variable or variables have been altered (written) and allows another process to access to the variable(s).

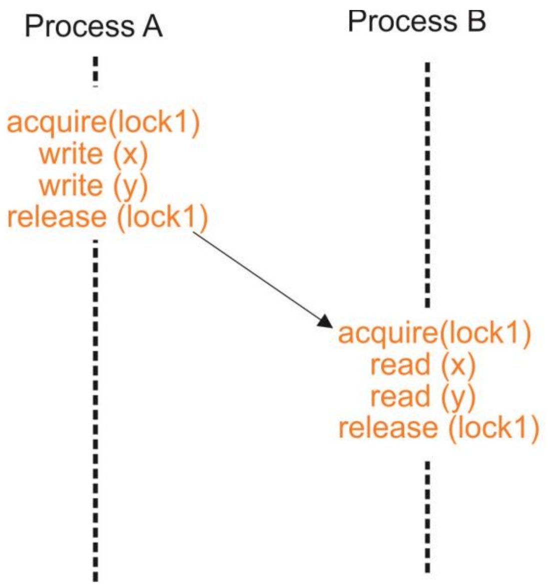

Fig 2.5 release consistency

Typically acquire is done with a lock operation and release by an unlock operation (although not necessarily).

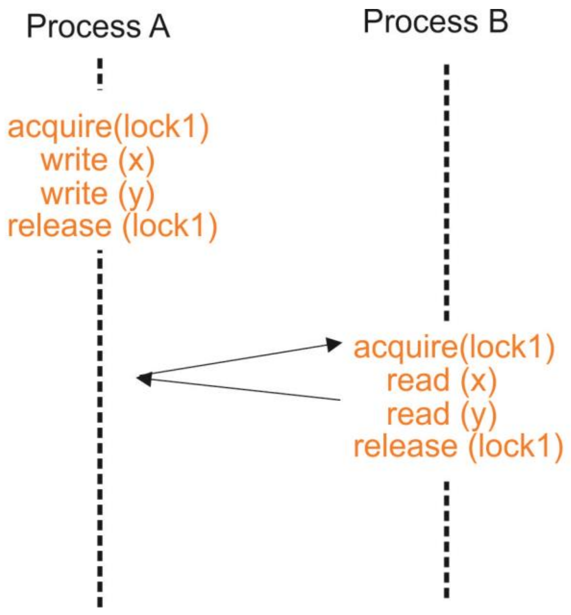

Fig 2.6 Lazy release consistency

Check your progress 2

1. Execution of DSM system involves_______.

    a. address problems of data position

    b. data access

    c. sharing and locking of data_______.

    d. all

2. Which is an advantage of DSM?

    a. Acquires performance penalty.

    b. Should be provided for protection against immediate admission.

    c. Small programmer control over real messages.

    d. It is system scalable.

2.4 UNIX Operating System: Case Studies

Shell

- Shell Serves as an interface among command language user and OS.
- The Shell is user interface and comes in many forms.
- User allowed to enter input when prompted ($ of %)

Unix supports all shells which is running at the same time. The required shell gets loaded at the login position where every alteration can be done by the user. The UNIX command syntax will be executable_file [-options] arguments.

The shell runs a command interpretation loop:-

- Accept command
- Read command
- Process command
- Execute command

Performing a command means developing a child process which is seen working in another shell with the help of forking. In this, the parent process will halt till the child process terminates initially early re-entering command interpretation loop.

It is found that the programs will work in the background by using suffix command line entry which is applied by ampersand (&). The result of this is that the parent will not wait for child process to get completed.

The Processing Environment

Input and Output

During the working of UNIX, there exist three files for a particular process:

- STDIN - Standard input (attached to keyboard)
- STDOUT - Standard output (attached to terminal)
- STDERR - Standard Error (attached to terminal)

As in UNIX, I/O devices results in a unique types of file systems, the STDIO will easily be redirected to different devices along with files who > list _of _users

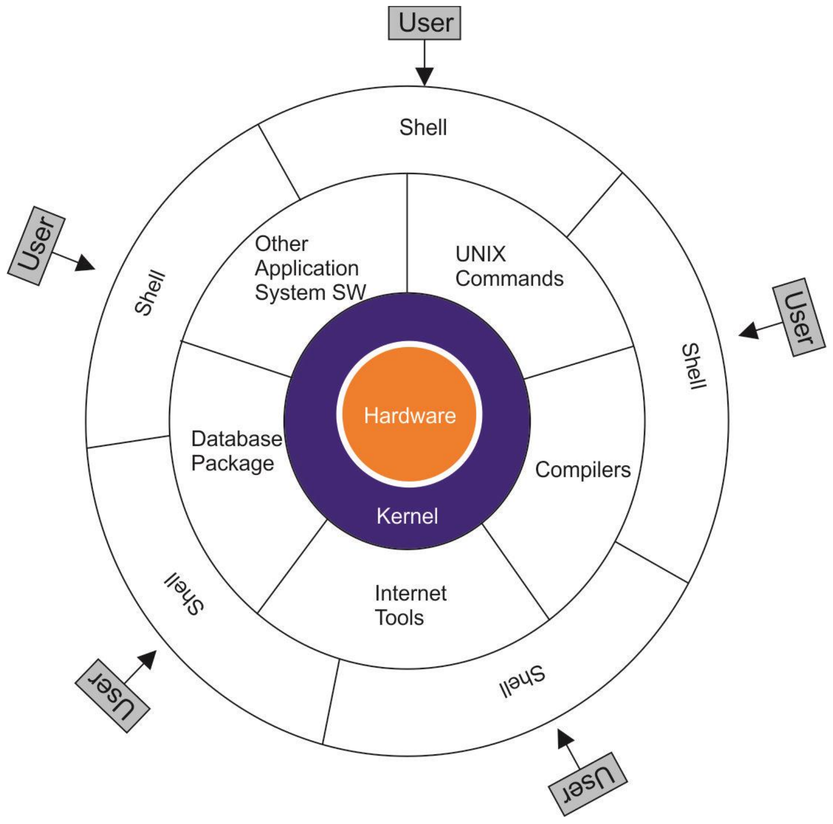

Fig 2.7 Structure of Unix

The Kernel

This is the middle portion of an OS which will give system services to application programs and shell.

Kernel manages certain memory, I/O and Timer processes.

Different process contains different address space for protection.

In Unix sharing of text region is done and changes in process occurs as per environment by calls.

The File System

In UNIX, HDS file is along with root is applied at origin.

The directory in UNIX file has file names and i-nodes.

In this, the subdirectories shows entry of file.

In Unix, directories cannot be directly changed but will change with the help of operating system.

File System is a type of data structure which is present of the disk.

File system contains super block, an arrangement of i-nodes, actual file data blocks as well as free blocks.

In Unix, space allocation is done in case of fixed size blocks.

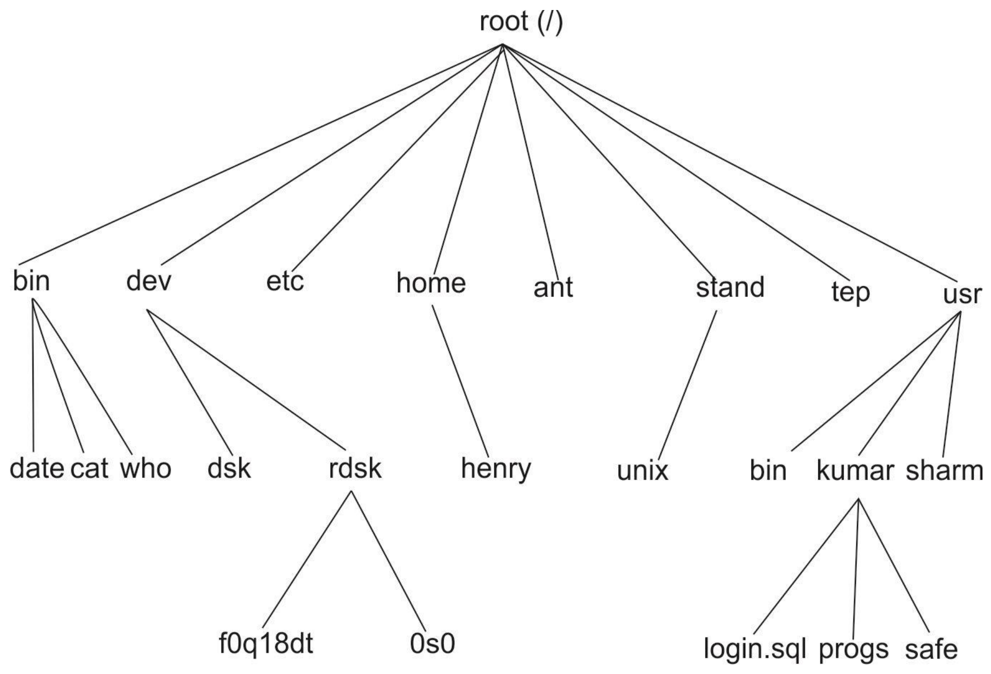

Fig 2.8 File Structure

The i-node contains

- The file owner's user-id and group-id.
- Protection bit for owner, group, and world.
- The block locator.
- File size.
- Accounting information.
- Number of links to the file.
- File type.

The Block Locator

Consists of 13 fields

- First 10 fields point directly to first 10 file blocks.
- 11th field is and indirect block address.
- 12th field is a double-indirect block address.
- 13th field is a triple-indirect block address.

Permissions

- In UNIX, the file and directory contains 3 sets of permission bits.
- These directories are allowed for owner, group and world.
- The system files are copyright by root, wizard, or super user.
- The access of root is restricted to the owners of files.

Setuid

- To modify or change your password, you have to alter/etc/passwd file.
- The root has copyright for passwd command that can be executed by permission to access.
- Setuid is bit which on application with executable file will give similar privileges to user.
- It is similar with many operating system commands.

Process Management

Description of Process Management in SunOS.

Scheduling

- Priority-based pre-emptive Scheduling. Priorities in range -20 to 20.
- Priorities for runnable processes are recomputed every second.
- Allow for ageing, but also increases or decreases process priority based or past behavior.
- I/O – bound processes receive better service.
- CPU – bound processes do not suffer indefinite postponement because the algorithm forgets 90% CPU use in 5*n sec. (where n is the average number of runnable process in the past 60 seconds).

Signals

- Signals are software equivalents to hardware interrupts used to inform processes asynchronously of the occurrence of an event.

Inter-process Communication

- UNIX System V uses semaphores to cantor access to shared resources.
- For processes to exchange data or communicate, pipes are used.
- A pipe is a unidirectional channel between 2 processes.
- UNIX automatically provides buffering scheduling services and,
- Synchronization to processes in pipe line.

Timers

- There are three interval timers associated to each process.
- Initially it will counts to 0 and then produces a signal.
- The initial signal will work regularly, while second work during the process that works with process code.
- The last will work at the time when process works out process code.

Memory Management

- Address Mapping (Virtual Storage) – Paged MMS.
- Virtual address V is dynamically translated to real address (P,D).
- Direct Mapping is used, with the page map held in a high-speed RAM cache.

- Each page map entry contains a modified bit and accessed bit, a valid bit (if the page is resident in PM) and protection bits.
- The system maintains 8 page maps – 1 for the kernel and 7 for processes.
- 2 context register are used – one points to the running process page map and the other to the kernel's page map.
- The replacement strategy replaces the page that has not been active for longest (LRU).

Paging

- Sum OS maintains 2 data structures to control paging.
- The fees list contains empty page frames.
- The loops contains an ordered list of all allocated page frames (except for the kernel).
- The pager ensures that there is always free space in memory.
- When a page is swapped out (not necessarily replaced the system judges whether the page is likely to be used again).
- If the page contains a text region, the page is added to bottom of the free list, otherwise it is added to the top.
- When a page fault occurs, if the page is still in the free list it is reclaimed.

I/O Data

- Data is addressed as byte stream.
- UNIX not put any sort of structure on data, but applications do.
- Manipulation of data appears in any direction.

Devices

- In UNIX, device is unique file type.
- These unique files contain a protection bits to stop read/write operations.
- Certain sensitive devices are not allowed to root while other users uses system calls having setuid bit set.

Generic Unix Command

Command	**Function**
Date	Used to display the current date and time.
Date + % D	Display date only
Date + % T	Display time only
Date + %Y	Display Year part of the date.
Date + % H	Display the hour part of the time.
Cal	Calendar of the current month
Cal year	Displays calendar for all months for the specified year
Call month year	Displays calendar for the specified month of the year.
Who	Login detail of all user as their IP, Terminal NO, User Name,
Who am I	Used to display the login detail of the under.
tty	Used to display the terminal name
uname	Display the operating system
uname-r	Show version number of the OS (kernel).
uname-n	Display domain name of the server
echo"txt"	Display the given text on the screen
echo$HOME	Display the user's home directory
Bc	Basic calculator. Press Ctrl+d to quit.
lp file queue	Allows the user to spool a job along with other in a print
mancmdname	Manual for the given command Press q to exit

History	To display the command used by the user since log on.
Exit / out	Exit form a process. If shell is the only process then logs

Check your progress 3

1. Which is not a shell run command?

   a. delete command c. read command

   b. accept command d. process command

2. The processing files of UNIX are____.

   a. STDIN c. STDERR

   b. STDOUT d. all

3. The central part of Unix OS is______.

   a. kernel c. compilers

   b. shell d. database

4. In Unix, i-node contains_____.

   a. file owner's user-id c. protection bits for owner

   b. group-id d. all

2.5 Let Us Sum Up

In this unit we have learned:

- That a procedure is an enclosed series of instructions which is introduced from and returns the control to an external source.
- It is studied that data approximations can be travelled in all directions along with the flow of control.
- RPC is equivalent to a function call
- A thread is a single chronological flow of control by way of one point of execution at whichever moment.

- A thread formed as well as managed by application code is an application thread.
- Distributed shared memory is a method permitting the end-users procedure to right to use shared data with no inter-process communications.

2.6 Answers for Check Your Progress

Check your progress 1

Answers: (1-c), (2-d), (3-d), (4-a)

Check your progress 2

Answers: (1-d), (2-d)

Check your progress 3

Answers: (1-a), (2-d), (3-a), (4-d)

2.7 Glossary

1. **Kernel -** It is the central part of OS which provides system services to application programs and the shell.
2. **File System -** It is a data structure that is resident on disk.

2.8 Assignment

Explain the Consistency Models used on DSM Systems.

2.9 Activities

Write a DSM system in C++ using MPI for the underlying message-passing and process communication.

2.10 Case Study

Compile and run the remote directory example rls.cand run both client and server on the network.

2.11 Further Readings

1. Distributed Systems, Principles and Paradigms by Tanenbaum.
2. Distributed Systems, Concepts and Design by Coulouris, Dollimore, Kindberg.