UNIX OS: It's a powerful OS. Initially developed by Ken Thompson, Dennis Ritchie at AT&T Bell labs in 1970. It is prevalent among scientific, engineering, and academic institutions due to its most appreciative feats. like multitasking, flexibility, and many more. In UNIX, the file system is a hierarchical structure of files & dir. where users can store and retrieve info. using the files.**Features of UNIX: Multitasking:** A UNIX os is a multitasking os that allows you to initiate more than one task from the same terminal so that one task is performed as a foreground and the other task as a background process. **Multi-user:** The UNIX os allows multiple users to access computer resources simultaneously through time-sharing, which is managed by a scheduler that divides CPU time into segments assigned to each user on a scheduled basis. **Portability:** This feature makes the UNIX work on different machines and platforms with the easy transfer of code

to any computer system. Since a significant portion of UNIX is written in C language, and only a tiny portion is coded in assembly language for specific hardware. **File Security and Protection:** The UNIX os provides various levels of security to ensure the protection of files and systems. This includes assigning usernames & passwords to individual users for authentication, granting file access permissions (read, write, execute), and encrypting files to make them unreadable. **Command Structure:** UNIX commands are easy to understand and simple to use. eg: "cp", mv etc. where cp means copy and mv means move. **Communication:** In UNIX, communication is an excellent feature that enables the user to communicate worldwide. It supports various communication facilities provided using the write command, mail command, talk command, etc.

Diff b/w unix & winos: UNIX: It comes with cli . It is a free and open-source os. It is more secure because all system updates require explicit user permission. It supports multiprocessing. It is fully case-sensitive, and files can be considered separate files. It is a command-based os. In a UNIX system, hardware support is limited. Some hardware could not have drivers built-in. It uses the Unix File System (UFS), which includes the STD.ERR and STD.IO file systems. Unix and its distros are well known for their high level of stability. Creating a bkup and recovery sys. in UNIX is time-consuming, but it is becoming easier with the release of new Unix distros. **Win:** It comes with a GUI. It is a licensed OS . It is less secure than the UNIX OS . It doesn't support multiprocessing. It has case sensitivity as an option. It is a menu-based OS . Almost all hardware has drivers available. It makes use of the New Technology File System & the File Allocation System32.

Although Win has become more stable in recent years, it still falls short of the reliability offered by Unix systems. It contains a built-in backup and recovery sys. that makes it more user-friendly. **Internal Cmds:** Cmds which are built into the shell. For all the shell built-in cmds, exec of the same is fast in the sense that the shell doesn't have to search the given path for them in the PATH variable, and also no process needs to be spawned for executing it. eg: source, cd, fg, etc. **External Cmds:** Cmds which aren't built into the shell. When an external cmd has to be executed, the shell looks for its path given in the PATH variable, and also a new process has to be spawned and the cmd gets executed. They are usually located in /bin or /usr/bin. eg, when you execute the "cat" cmd, which usually is at /usr/bin, the exec /usr/bin/cat gets executed. eg: ls, cat etc.

Unix Architecture: Kernel: This is the heart of the os. It interacts with the hardware and most of the tasks like memory mgmt, task scheduling and file mgmt. **Shell:** The shell is the utility that processes your req. When you type in a cmd at your terminal, the shell interprets the cmd & calls the prog. that you want. The shell uses standard syntax for all cmd. C Shell, Bourne Shell and Korn Shell are the shells available with UNIX. **Cmds&Utilities:** There are various cmd & utils which you can make use of in your daily tasks. ex. cp, mv, cat,grep. **Files& Directories:** All the data of Unix is organized into files. All files are then organized into dir. These dir are further organized into a tree-like structure called the fs.**UNIX booting process: 1BIOS:** This is the first process that starts when you turn on the comp & it determines the list of bootable devices available in the sys., such as Hard Disk, CD/DVD-ROM, Floppy Drive, USB etc. The os boots from the Hard Disk where the MBR

contains the primary boot loader. **2Bootloader(bl):** This phase involves loading the bl (MBR&GRUB) into mem. to bring up the kernel. MBR is the first sector of the hard disk and cannot load the kernel directly, so a bl with fs driver is req. GRUB is used with the details of the fs in /boot/grub.conf & file sys. drivers. To be very brief this phase incl. loading of the bl (MBR&GRUB) into mem to bring up the kernel. MBR It is the first sector of the Hard Disk with a size of 512B. The first 434 - 446B are the primary bl, 64B for partition table and 6B for MBR validation timestamp. **GRUB loads the kernel in 3 stages GRUB Stage1:** The primary bl takes up less than 512B of disk space in the MBR - too small a space to contain the instructions necessary to load a complex os. Instead the primary bl performs the function of loading either the stage 1.5 or stage 2 boot loader.

Process in UNIX: In Unix/Linux, when a program/command is executed, a new process is created and provided with a unique instance consisting of resources it may use. Each process is assigned a unique 5-digit process ID (PID) which Unix uses to track them. Used up PIDs can be reused for newer processes since each process has a unique PID, and at any point in time, no two processes with the same PID can exist in the system.**Phases of process creation: Initializing a process: Method 1: Foreground Process :** Every process when started runs in foreground by default, receives input from the keyboard, and sends output to the screen.When a command/process is running in the foreground and is taking a lot of time, no other processes can be run or started because the prompt would not be available until the program finishes processing and comes out. **Method 2: Background Process:** It runs in the background without keyboard input and waits till keyboard

input is required. Thus, other processes can be done in parallel with the process running in the background since they do not have to wait for the previous process to be completed. Adding & along with the command starts it as a background process. **Tracking ongoing process:** ps (Process status) can be used to see/list all the running processes. For more information -f (full) can be used along with ps. For single-process information, ps along with process id is used. **Stopping a process:** When running in foreground, hitting Ctrl + c (interrupt character) will exit the command. For processes running in background kill command can be used if it's pid is known. If a process ignores a regular kill command, you can use kill -9 followed by the process ID.

GRUB Stage 1.5: Stage 1 can load stage 2 directly, but it is normally set up to load stage 1.5. This can happen when the /boot partition is situated beyond the 1024 cylinder head of the hard drive. GRUB Stage 1.5 is located in the first 30KB of Hard Disk immediately after MBR and before the first partition. This space is utilized to store file sys. drivers and modules.This enabled stage 1.5 to load stage 2 to load from any known loc on the fs i.e. /boot/grub **GRUB Stage 2:** This is responsible for loading kernel from /boot/grub/grub.conf and any other modules needed. Loads a GUI interface i.e. splash image located at /grub/splash.xpm.gz with list of available kernels where you can manually select the kernel or else after the default timeout value the selected kernel will boot. The original file is /etc/grub.conf of which you can observe a symlink file at /boot/grub/grub.conf **3Kernel:** is the core of the os , responsible for mgmt of all sys. processes. It configures hardware and mem. , decompresses the initrd img., & loads the necessary drivers. Progs like insmod and rmmod help in loading and unloading kernel modules. The kernel mounts the root partition as read-only, runs the init process, and unmounts the initrd img., freeing up all the occupied mem. **4Init Process:** Executes the sys to boot into the run level as specified in /etc/inittab. Next as per the fstab entry fs's integrity is checked and the root partition is re-mounted as read-write (earlier it was mounted as read-only). **5Runlevel scripts:** These scripts are located in the /etc/rc.d dir. Depending on the selected runlevel, the init process executes startup or stopping scripts located in subdir named rc0.d through rc6.d. The startup scripts begin with "s," while the stopping scripts begin with "k." The /etc/rc.d/rc.local file is executed every time the runlevel changes.

Types of acc: Root acc: This is aka superuser & would have complete control of the sys. A superuser can run any cmd without any restriction. This user should be assumed as a sys. admin. **Sys acc:** Sys acc are those needed for the operation of system-specific comp ex mail accounts & the sshd acc.These acc are usually needed for some specific function on your sys and any modifications to them could adversely affect the sys. **User acc:** User acc provides interactive access to the sys for users & groups of users. General users are typically assigned to these acc & usually have limited access to critical sys files and dir. Unix supports a concept of Group Acc which logically groups a number of acc. Every acc would be a part of another group acc. A Unix group plays important role in handling file permissions & process mgmt.**Creating user in unix:** "useradd -d homedir -g groupname -m -s shell -u userid acc name" **where: -d** homedir Specifies home dir for

the acc, **-g groupname** Specifies a group acc for this acc **-m** Creates the home dir if it doesn't exist, **-s shell** Specifies the default shell for this acc, **-u userid** You can specify a user id for this acc, **accname** Actual acc name 2b created.

File perm: File ownership is an imp comp of Unix that provides a secure method for storing files.**Owner perm:** The owner's perm determine what actions the owner of the file can perform on the file. **Group perm:** The group's perm determine what actions a user, who is a member of the group that a file belongs to, can perform on the file. **Other perm:** The perm for others indicate what action all other users can perform on the file.**File Access Modes:** The perm of a file are the first line of defense in the security of a Unix sys. The basic building blocks of Unix perm are the

read, write, and exec perm **Read:** Grants the capability to read, i.e., view the contents of the file. **Write:** Grants the capability to modify, or remove the content of the file. **Execute:** Users with execute perm can run a file as a prog.

Link is a pointer to a file or dir. It allows more than one file name to refer to the same file. Creating links provides a shortcut to access a file. Links allow more than one file name to refer to the same file, elsewhere.

Hard Links: hard links are pointers that share the same Inode value and reference the same physical file location as the original file. They are more flexible and can still be linked even if the original or linked files are moved within the same file system, but not across different file systems. The number of links for a file is shown in the "link" column when using the ls -l

command. Removing a hard link only reduces the link count and doesn't affect other links, and renaming the original file doesn't affect the hard links. **Soft Links:** This is a file that contains a separate Inode value pointing to the original file, similar to a shortcut in Windows. Soft links can be linked across different file systems and reflect changes made to the original file, but if the original file is deleted or moved, the soft linked file becomes a "hanging" link. Soft links only contain the path for the original file and not its contents. If the original file is removed, the soft link becomes a "dangling" link that points to a nonexistent file. Soft links can link to directories.

Types of files: 1OrdinaryFiles: An ordinary file is a file on a sys that stores data, text, or prog instructions such as txt or img. These files are typically located within or under a dir file and dont contain other files. **2Directories:** Dir in UNIX store both special and ordinary files & are similar to folders in Win/Mac OS. Each dir file contains an entry for every file and subdir it houses, with each entry consisting of a filename & a unique identification no. for the file or dir, known as the inode no. **3Special Files:** Special files in UNIX and Linux rep. physical devices like printers, tape drives, & terminals used for I/O operations. These files appear in a file sys like ordinary files or dir and are of two types: character special files and block special files. Character and block special files are used for device I/O operations on UNIX systems. **4Pipes:** In UNIX, cmd can be linked together using a pipe symbol "|", which acts as a temp file to hold data from one cmd until it is read by another. Pipes provide one-way flow of data, with the output of the first cmd serving as the input to the second. To create a pipe, place the vertical bar between two cmds . For ex, "who | wc -l". In the long-format output of "ls -l", named pipes are marked with the "p" symbol. **5Sockets:** A Unix socket is a special file that facilitates advanced inter-process communication, often used in a client-server app framework. It functions like a stream of data similar to network streams, but all transactions are local to the fs. In the long-format output of "ls -l", Unix sockets are marked by the "s" symbol. **6Symbolic Link:** A sym link, also called a soft link, is a file that references another file in the file system using a txt form of the path. It appears to have its own name, but it actually points to the file it references. If the source file is deleted or moved, the sym link will not function properly, but deleting the soft link itself will not affect the data file.

Unix file system (fs) is a logical method of organizing & storing large amt of info in a way that makes it easy to manage.The Unix fs organizes & stores data through files & dir in a tree-like structure called the fs. All data in Unix is stored as files, which are organized into dir, & these dir are organized into a hierarchy known as the dir tree. The "root" dir is located at the top of the fs & is rep. by a "/", while all other files are descendants of the root. **Dir/Files&their desc: / :** The slash alone denotes the root of the fs tree. **/bin :** Stands for "binaries" & contains certain fundamental utilities, such as ls or cp, which are generally needed by all users. **/boot** : Contains all the files that are required for a successful booting process. **/dev :** Stands for "devices". Contains file rep of peripheral devices & pseudo-devices. **/etc :** Contains system-wide config files & system db. Originally also contained "dangerous maintenance utilities" such as init,but these have typically been moved to /sbin or elsewhere. **/home :** Contains the home dir for the users. **/lib :** Contains sys lib & some critical files such as kernel modules or device drivers. **/mnt :** Stands for "mount". Contains fs mount points.**/root :** The home dir of the "root" superuser, which is the system admin, is usually located on the initial fs & not in the "/home" dir, which could be a mount point for another fs. **/tmp :** The temp dir is a loc for storing temp files, & on many sys, it is cleared upon startup. The dir may have tmpfs mounted atop it, meaning its contents do not survive a reboot, **/usr :** Originally the dir holding user home dir, its use has changed. It now holds executables, libraries, and shared resources that are not sys critical, like the X Win Sys, KDE, Perl, etc. **/usr/bin :** This dir stores all binary progs distributed with the os not residing in /bin, /sbin etc.

Layers of UNIX OS- While working with UNIX OS, several layers of this sys. provide interaction b/w the pc hardware and the user. **Layers in UNIX os: 1Hardware:** This layer of UNIX consists of all hardware-related info. in the UNIX env.. **2Kernel:**The kernel is the core of the os, responsible for managing all sys. processes. It configures hardware and mem. decompresses the initrd image, and loads the necessary drivers. Prog. like insmod and rmmod help in loading and unloading kernel modules. The kernel mounts the root partition as read-only, runs the init process, and unmounts the initrd img, freeing up all the occupied mem. **3TheShell:** Shell is a cli interpreter in UNIX that interprets user commands and calls the desired prog. It maintains a history of cmds typed in by the user and can repeat them using the cursor keys or the 'history' cmd. UNIX has various built-in

cmds like cat, mv, grep. **Types of Shell in UNIX:** Bourne Shell: This is simply called the Shell. It was the first Shell for UNIX OS. It is still the most widely available Shell on a UNIX system. **CShell:** The C shell is another popular shell commonly available on a UNIX system. The C shell was developed by the University of California at Berkeley and removed some of the shortcomings of the Bourne shell. **Korn Shell:** This Shell was created by David Korn to address the Bourne Shell's user-interaction issues and to deal with the shortcomings of the C shell's scripting quirks. **4Application Prog Layer:** It is the outermost layer that executes the given external apps. UNIX distributions typically come with several useful applications programs as standard.