Unit 4
LINUX Fundamentals

Image source: https://www.itdev.co.uk/blog/get-your-patch-merged-journey-linux-kernel-%E2%80%93-part-3

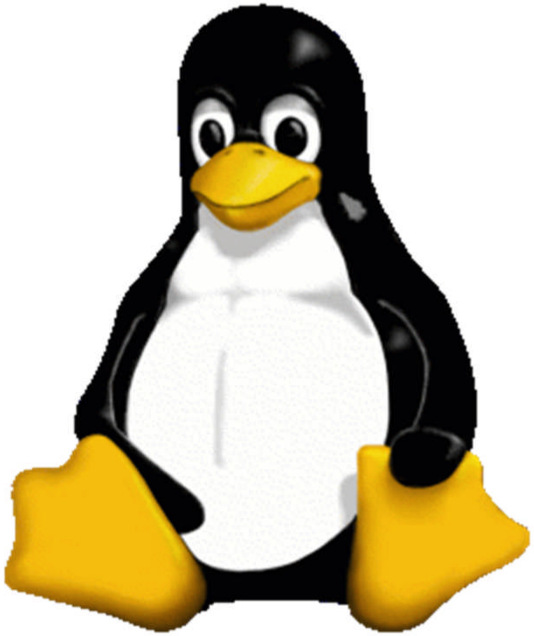
Objective

- Introduction to Linux Kernel
- Linux Architecture
- History of Linux
- Linux Features
- Kernel Design Goals

What is Linux Kernel?

The Linux kernel is a free and open-source, monolithic, modular, multitasking, Unix-like operating system kernel.

It was conceived and created in 1991 by Linus Torvalds for his i386-based PC, and it was soon adopted as the kernel for the GNU operating system, which was created as a free replacement for UNIX.



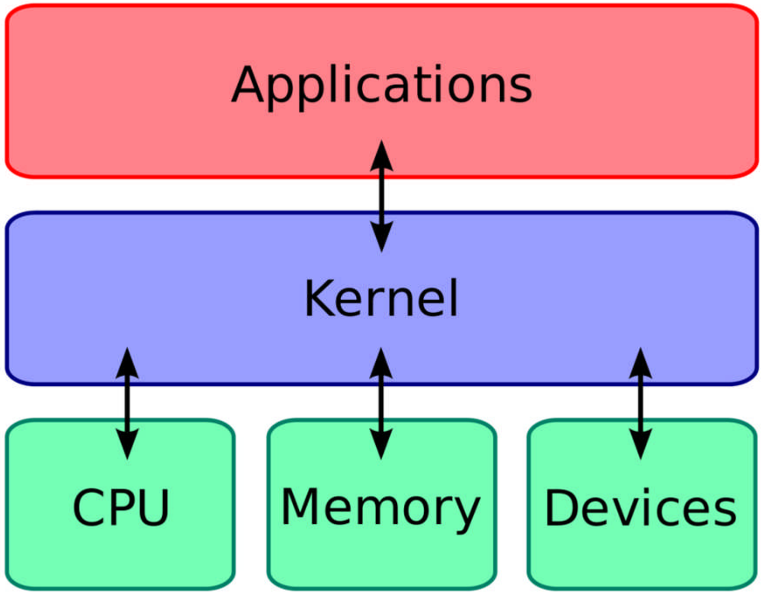
https://en.wikipedia.org/wiki/Linux_kernel

What is a Kernel?

Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

The kernel has following jobs:

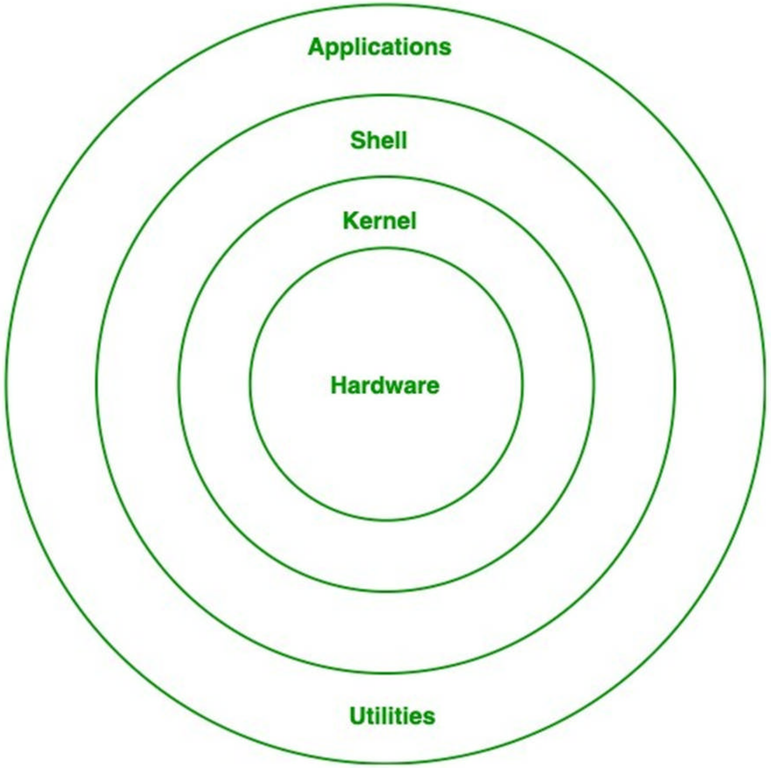
- **Memory management**
- **Process management**
- **Device drivers**
- **System calls and security**



https://blog.digilentinc.com/wp-content/uploads/2015/05/1280px-Kernel_Layout.svg_.png

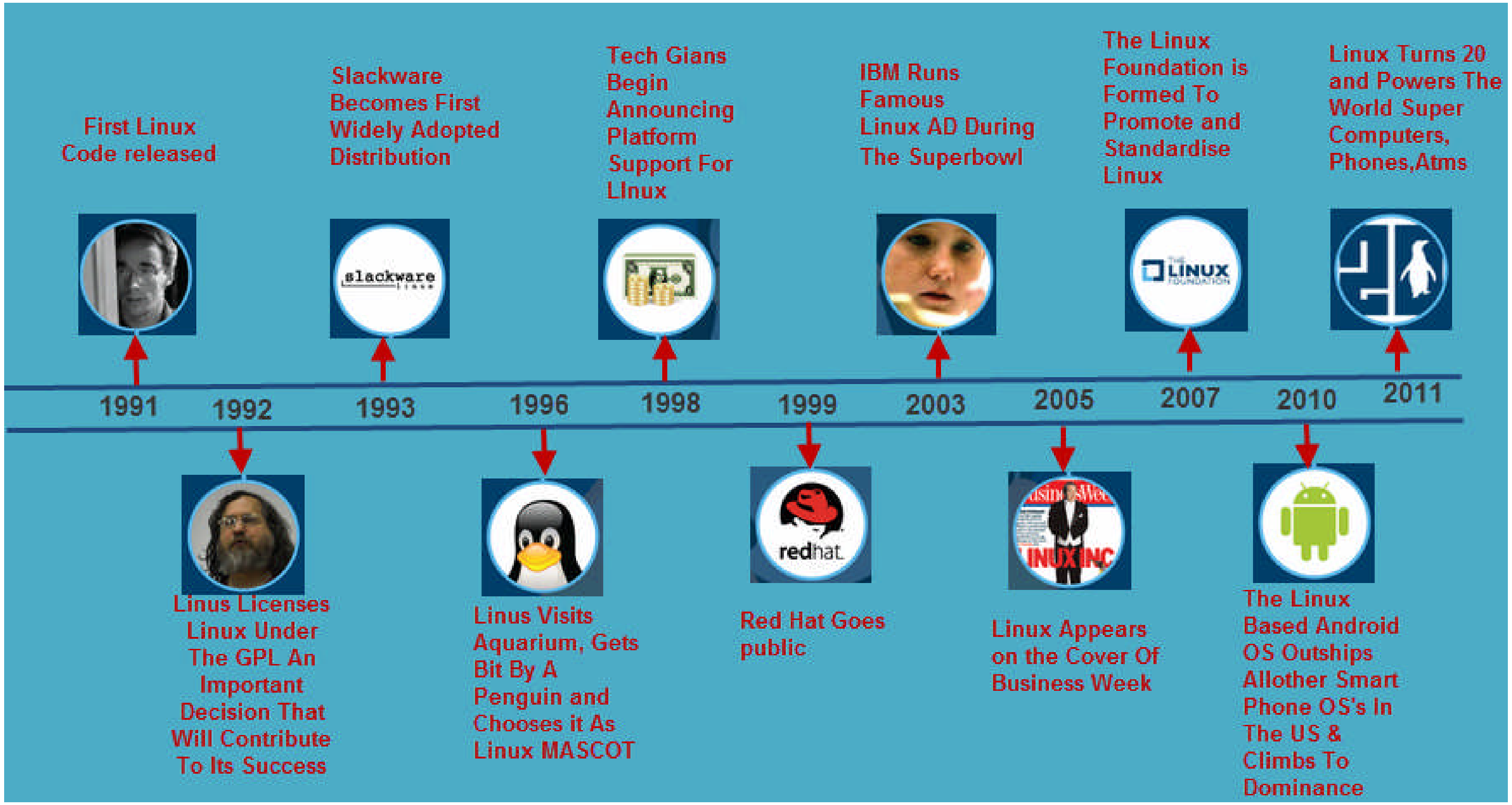
Linux Architecture

- Hardware Layer
- Kernel
- Shell
- Utilities



https://media.geeksforgeeks.org/wp-content/uploads/20200105215737/Untitled-Diagram-215-1.jpg

History of Linux



1991 – First Linux Code released

1992 – Linus Licenses Linux Under The GPL An Important Decision That Will Contribute To Its Success

1993 – Slackware Becomes First Widely Adopted Distribution

1996 – Linus Visits Aquarium, Gets Bit By A Penguin and Chooses it As Linux MASCOT

1998 – Tech Gians Begin Announcing Platform Support For LInux

1999 – Red Hat Goes public

2003 – IBM Runs Famous Linux AD During The Superbowl

2005 – Linux Appears on the Cover Of Business Week

2007 – The Linux Foundation is Formed To Promote and Standardise Linux

2010 – The Linux Based Android OS Outships Allother Smart Phone OS's In The US & Climbs To Dominance

2011 – Linux Turns 20 and Powers The World Super Computers, Phones,Atms

Source: https://www.elprocus.com/linux-operating-system/

Linux Features

- UNIX-like kernel.
- Features:
  - Open source.
  - Preemptive multitasking.
  - Multi-user
  - Hierarchical file system
  - Portable
  - Security

Linux Advantages

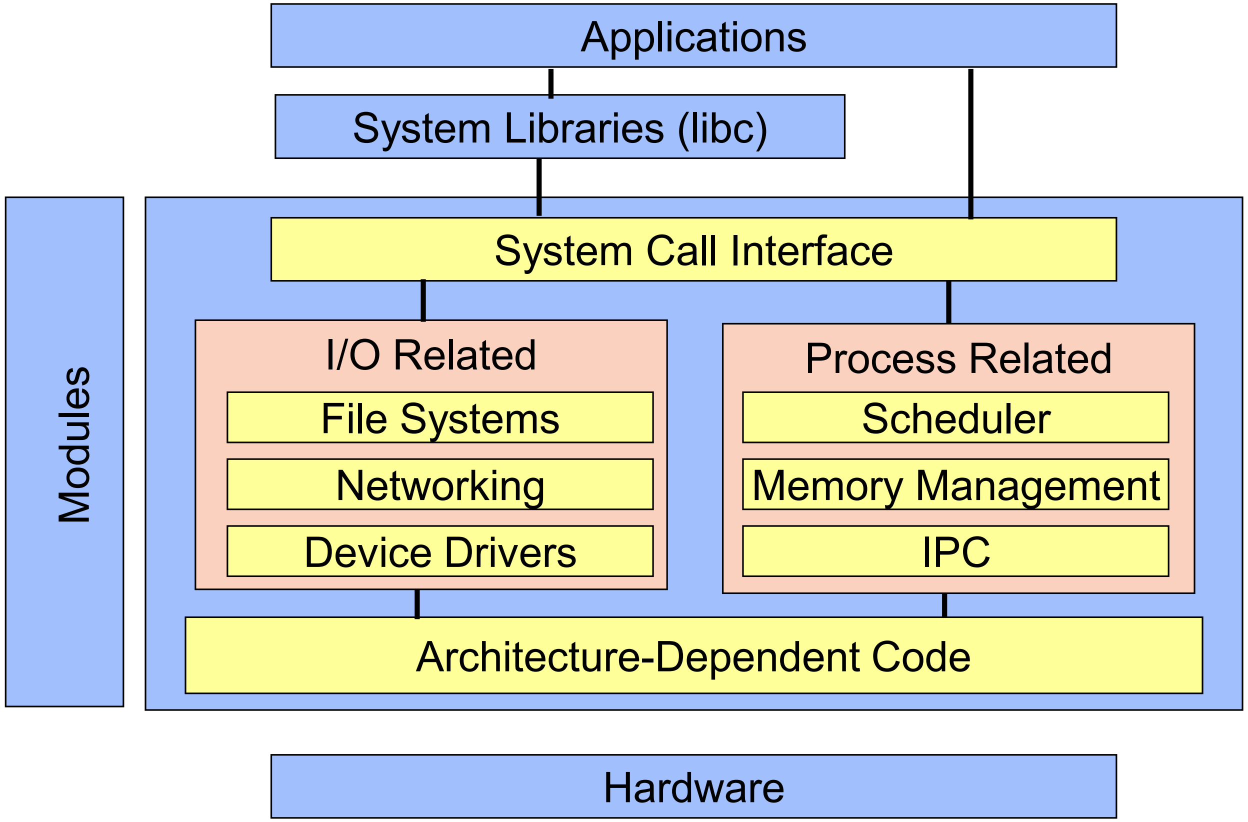
- Low cost
- Stability
- Performance
- Network friendliness
- Flexibility
- Compatibility
- Choice
- Fast and easy installation
- Multi-tasking
- Open Source

Linux Distributions

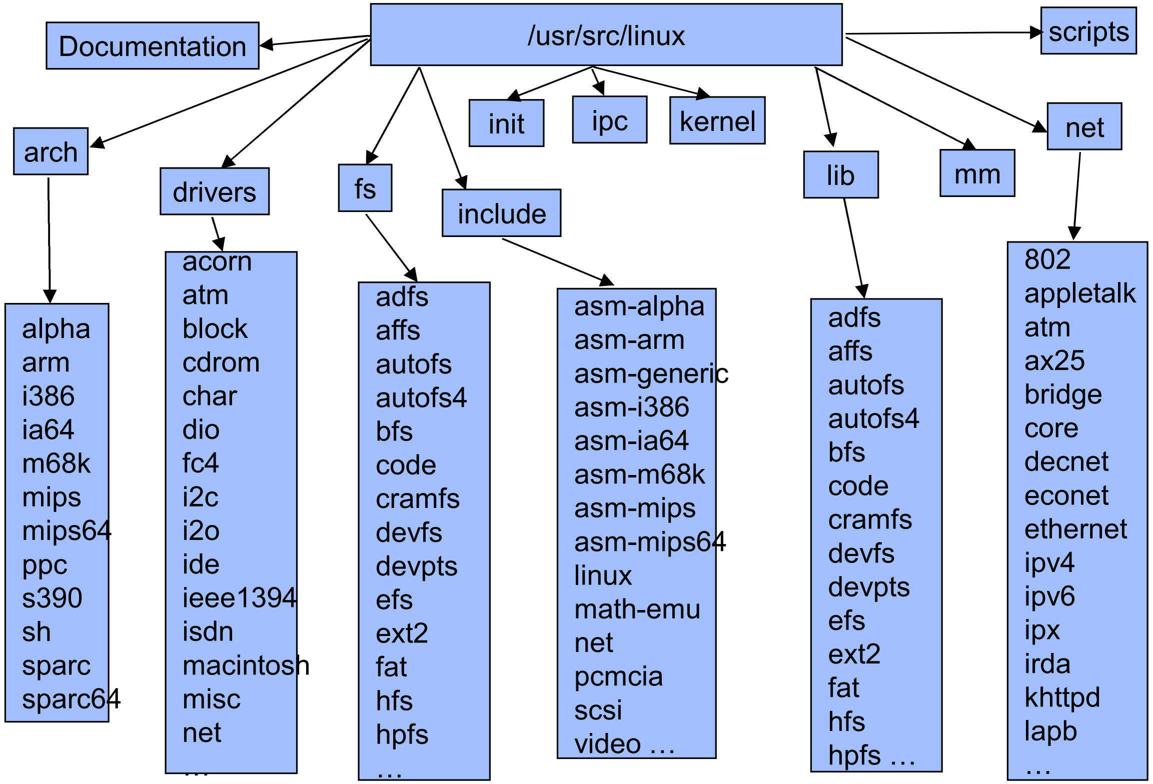
Kernel Design Goals

- Performance: efficiency, speed.
  - Utilize resources to capacity with low overhead.
- Stability: robustness, resilience.
  - Uptime, graceful degradation.
- Capability: features, flexibility, compatibility.
- Security, protection.
  - Protect users from each other & system from bad users.
- Portability.
- Extensibility.

Kernel Design



Linux Source Tree Layout



/usr/src/linux

- Documentation
- arch
  - alpha
  - arm
  - i386
  - ia64
  - m68k
  - mips
  - mips64
  - ppc
  - s390
  - sh
  - sparc
  - sparc64
- drivers
  - acorn
  - atm
  - block
  - cdrom
  - char
  - dio
  - fc4
  - i2c
  - i2o
  - ide
  - ieee1394
  - isdn
  - macintosh
  - misc
  - net
  - …
- fs
  - adfs
  - affs
  - autofs
  - autofs4
  - bfs
  - code
  - cramfs
  - devfs
  - devpts
  - efs
  - ext2
  - fat
  - hfs
  - hpfs
  - …
- include
  - asm-alpha
  - asm-arm
  - asm-generic
  - asm-i386
  - asm-ia64
  - asm-m68k
  - asm-mips
  - asm-mips64
  - linux
  - math-emu
  - net
  - pcmcia
  - scsi
  - video …
- init
- ipc
- kernel
- lib
  - adfs
  - affs
  - autofs
  - autofs4
  - bfs
  - code
  - cramfs
  - devfs
  - devpts
  - efs
  - ext2
  - fat
  - hfs
  - hpfs …
- mm
- net
  - 802
  - appletalk
  - atm
  - ax25
  - bridge
  - core
  - decnet
  - econet
  - ethernet
  - ipv4
  - ipv6
  - ipx
  - irda
  - khttpd
  - lapb
  - …
- scripts

Linux /arch

- Subdirectories for each current port.
- Each contains **kernel**, **lib**, **mm**, **boot** and other directories whose contents override code stubs in architecture independent code.
- **lib** contains highly-optimized common utility routines such as memcpy, checksums, etc.
- **arch** as of 2.4:
  - alpha, arm, i386, ia64, m68k, mips, mips64.
  - ppc, s390, sh, sparc, sparc64.

Linux /drivers

- Largest amount of code in the kernel tree (~1.5M).
- device, bus, platform and general directories.
- drivers/char – n_tty.c is the default line discipline.
- drivers/block – elevator.c, genhd.c, linear.c, ll_rw_blk.c, raidN.c.
- drivers/net –specific drivers and general routines Space.c and net_init.c.
- drivers/scsi – scsi_*.c files are generic; sd.c (disk), sr.c (CD-ROM), st.c (tape), sg.c (generic).
- General:
  - cdrom, ide, isdn, parport, pcmcia, pnp, sound, telephony, video.
- Buses – fc4, i2c, nubus, pci, sbus, tc, usb.
- Platforms – acorn, macintosh, s390, sgi.

Linux /fs

- Contains:
  - virtual filesystem (VFS) framework.
  - subdirectories for actual filesystems.
- vfs-related files:
  - exec.c, binfmt_*.c - files for mapping new process images.
  - devices.c, blk_dev.c – device registration, block device support.
  - super.c, filesystems.c.
  - inode.c, dcache.c, namei.c, buffer.c, file_table.c.
  - open.c, read_write.c, select.c, pipe.c, fifo.c.
  - fcntl.c, ioctl.c, locks.c, dquot.c, stat.c.

Linux /Include

- include/asm-*:
    - Architecture-dependent include subdirectories.
- include/linux:
    - Header info needed both by the kernel and user apps.
    - Usually linked to /usr/include/linux.
    - Kernel-only portions guarded by #ifdefs
        - #ifdef __KERNEL__
        -       /* kernel stuff */
        - #endif
- Other directories:
    - math-emu, net, pcmcia, scsi, video.

Linux /init

- Just two files: version.c, main.c.
- version.c – contains the version banner that prints at boot.
- main.c – architecture-independent boot code.
- start_kernel is the primary entry point.

Linux /ipc

- System V IPC facilities.
- If disabled at compile-time, util.c exports stubs that simply return –ENOSYS.
- One file for each facility:
  - sem.c – semaphores.
  - shm.c – shared memory.
  - msg.c – message queues.

Linux /kernels

- The core kernel code.
- sched.c – "the main kernel file":
  - scheduler, wait queues, timers, alarms, task queues.
- Process control:
  - fork.c, exec.c, signal.c, exit.c etc…
- Kernel module support:
  - kmod.c, ksyms.c, module.c.
- Other operations:
  - time.c, resource.c, dma.c, softirq.c, itimer.c.
  - printk.c, info.c, panic.c, sysctl.c, sys.c.

Linux /mm

- Paging and swapping:
  - swap.c, swapfile.c (paging devices), swap_state.c (cache).
  - vmscan.c – paging policies, kswapd.
  - page_io.c – low-level page transfer.
- Allocation and deallocation:
  - slab.c – slab allocator.
  - page_alloc.c – page-based allocator.
  - vmalloc.c – kernel virtual-memory allocator.
- Memory mapping:
  - memory.c – paging, fault-handling, page table code.
  - filemap.c – file mapping.
  - mmap.c, mremap.c, mlock.c, mprotect.c.

Linux /lib

- kernel code cannot call standard C library routines.
- Files:
  - brlock.c – “Big Reader” spinlocks.
  - cmdline.c – kernel command line parsing routines.
  - errno.c – global definition of errno.
  - inflate.c – “gunzip” part of gzip.c used during boot.
  - string.c – portable string code.
    - Usually replaced by optimized, architecture-dependent routines.
  - vsprintf.c – libc replacement.

Linux /scripts

- Scripts for:
  - Menu-based kernel configuration.
  - Kernel patching.
  - Generating kernel documentation.

Linux File System

Directory	Purpose
/mnt	Standard mount point for external file systems, e.g. a CD-ROM or a digital camera.
/net	Standard mount point for entire remote file systems
/opt	Typically contains extra and third party software.
/proc	A virtual file system containing information about system resources. More information about the meaning of the files in proc is obtained by entering the command man proc in a terminal window.
/root	The administrative user's home directory. Mind the difference between /, the root directory and /root, the home directory of the root user.
/sbin	Programs for use by the system and the system administrator.
/tmp	Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work!
/usr	Programs, libraries, documentation etc. for all user-related programs.

Linux File System

Directory	Purpose
/bin	Common programs, shared by the system, the system administrator and the users.
/boot	The startup files and the kernel, vmlinuz. In some recent distributions also grub data. Grub is the GRand Unified Boot loader.
/dev	Contains references to all the CPU peripheral hardware, which are represented as files with special properties.
/etc	Most important system configuration files are in /etc, this directory contains data similar to those in the Control Panel in Windows
/home	Home directories of the common users.
/initrd	(on some distributions) Information for booting. Do not remove!
/lib	Library files, includes files for all kinds of programs needed by the system and the users.
/lost+found	Every partition has a lost+found in its upper directory. Files that were saved during failures are here.

ls command

ls -a (all) Lists all the files (including .* files)	ls -S (size) Lists the biggest files first
ls -l (long) Long listing (type, date, size, owner, permissions)	ls -r (reverse) Reverses the sort order
ls -t (time) Lists the most recent files first	ls -ltr (options can be combined) Long listing, most recent files at the end

Lists the files in the current directory, in alphanumeric order, except files starting with the "." character

cd and pwd commands

cd <dir>
Changes the current directory to <dir>.

cd -
Gets back to the previous current directory.

pwd
Displays the current directory ("working directory").

cp command

cp <source_file> <target_file>
Copies the source file to the target.

cp file1 file2 file3 ... dir
Copies the files to the target directory (last argument).

cp -i (interactive)
Asks for user confirmation if the target file already exists

cp -r <source_dir> <target_dir> (recursive)
Copies the whole directory.

mv and rm commands

mv <old_name> <new_name> (move)
Renames the given file or directory.

mv -i (interactive)
If the new file already exits, asks for user confirm

rm file1 file2 file3 ... (remove)
Removes the given files.

rm -i (interactive)
Always ask for user confirm.

rm -r dir1 dir2 dir3 (recursive)
Removes the given directories with all their contents.

Creating and removing directories

mkdir dir1 dir2 dir3 ... (make dir)
Creates directories with the given names.

rmdir dir1 dir2 dir3 ... (remove dir)
Removes the given directories
Safe: only works when directories and empty.
Alternative: rm -r (doesn't need empty directories).

File access rights

Use ls -l to check file access rights

3 types of access rights:

- Read access (r)
- Write access (w)
- Execute rights (x)

3 types of access levels

User (u): for the owner of the file
Group (g): each file also has a “group” attribute, corresponding to a given list of users
Others (o): for all other users

Access right examples

-rw-r--r--
Readable and writable for file owner, only readable for others

-rw-r-----
Readable and writable for file owner, only readable for users belonging to the file group.

drwx------
Directory only accessible by its owner

-------r-x
File executable by others but neither by your friends nor by yourself. Nice protections for a trap...

chmod: changing permissions

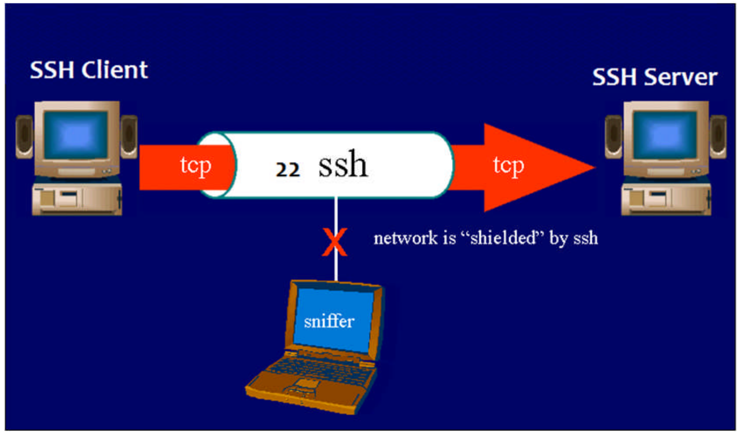
chmod <permissions> <files>
2 formats for permissions:

Octal format (abc):
a,b,c = r*4+w*2+x*1 (r, w, x: booleans)
Example: chmod 644 <file>
(rw for u, r for g and o)

symbolic format. Easy to understand by examples:
chmod go+r: add read permissions to group and others.
chmod u-w: remove write permissions from user.
chmod a-x: (a: all) remove execute permission from all.

SSH

- *ssh* stands for **"Secure Shell"**. It is a protocol used to securely connect to a remote server/system.
- ssh is secure in the sense that it transfers the data in encrypted form between the host and the client.
- It transfers inputs from the client to the host and relays back the output. ssh runs at TCP/IP port 22



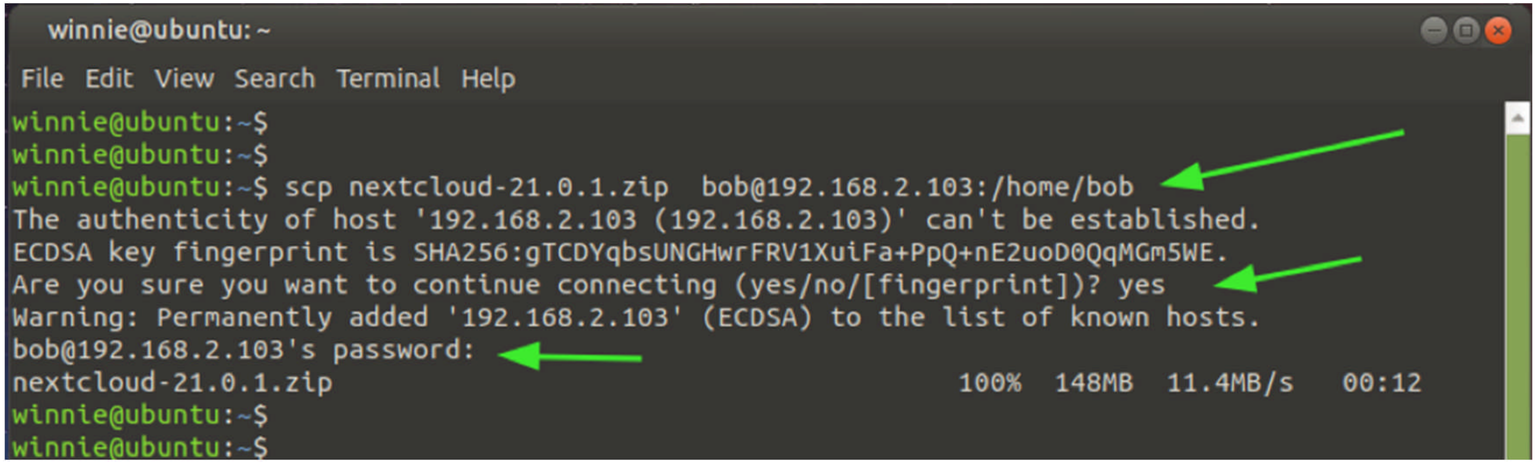
ssh user_name@host(IP/Domain_name) ***Example: ssh root@192.168.1.1***

SCP

- The scp command allows you to copy files over ssh connections.
- This is pretty useful if you want to transport files between computers
- Syntax

  scp examplefile yourusername@yourserver:/home/yourusername/
- *Example*

  scp file1.pdf root@192.168.1.1:/root/Desktop



```
winnie@ubuntu:~$
winnie@ubuntu:~$
winnie@ubuntu:~$ scp nextcloud-21.0.1.zip  bob@192.168.2.103:/home/bob
The authenticity of host '192.168.2.103 (192.168.2.103)' can't be established.
ECDSA key fingerprint is SHA256:gTCDYqbsUNGHwrFRV1XuiFa+PpQ+nE2uoD0QqMGm5WE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.2.103' (ECDSA) to the list of known hosts.
bob@192.168.2.103's password:
nextcloud-21.0.1.zip                                  100%  148MB  11.4MB/s   00:12
winnie@ubuntu:~$
winnie@ubuntu:~$
```

References

1. https://en.wikipedia.org/wiki/Linux_kernel
2. https://www.tutorialspoint.com/operating_system/os_linux.html
3. https://buildmedia.readthedocs.org/media/pdf/lym/latest/lym.pdf
4. https://phoenixnap.com/kb/linux-commands-cheat-sheet
5. https://www.guru99.com/file-permissions.html
6. https://www.hostinger.in/tutorials/linux-commands

THANK YOU