28/12/2025

USING YOUR OS

“GETTING WHERE YOU WANT”

howest.be

SCHEDULE

File Systems :

Where is What ?

Users :

Who is Who ?

Ownership :

Who owns What ?

Permissions :

Who can do What Where ?

FILE SYSTEMS

STORING YOUR FILES

howest.be

WHY FILE SYSTEMS ?

File system

- Is a service
- an abstract representation
- of the secondary storage to the OS

MAIN MEMORY VS SECONDARY STORAGE

- Small (MB/GB)
- Expensive
- Fast (10^{-6}/10^{-7} sec)
- Volatile
- Directly accessible by CPU
- Interface: (virtual) memory address
  → “malloc”

- Large (GB/TB)
- Cheap
- Slow (10^{-2}/10^{-3} sec)
- Persistent
- Cannot be directly accessed by CPU
  → Data should be first brought into the main memory
  → “open”

PHYSICAL STORAGES

- File system always acts the same, regardless of underlying physical medium
- A medium is “prepared for storing data”
- Then, a (part of) that medium is mapped unto a location
- Linux : e.g. /mnt/somemount
- Windows : e.g. E:/

PREPARING YOUR STORAGE

PART 1 : PARTITIONING

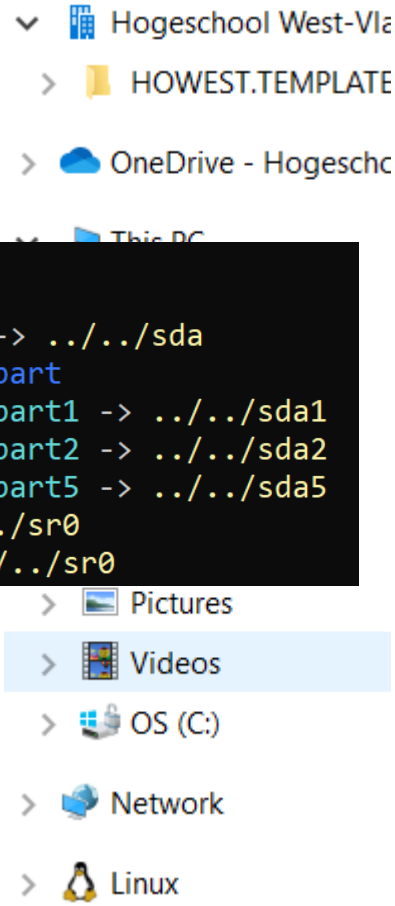
PARTITIONING FOR DUMMIES

- How can I find out what I have ?
  - **Consult your original hardware documentation**
  - **Open up your machine ! :D**
  - /dev/disk/by-path

```
guy@debian-guy:~$ ls -l /dev/disk/by-path/
total 0
lrwxrwxrwx 1 root root   9 Sep  3 13:59 pci-0000:00:10.0-scsi-0:0:0:0 -> ../../sda
drwxr-xr-x 5 root root 100 Sep  3 13:59 pci-0000:00:10.0-scsi-0:0:0:0-part
lrwxrwxrwx 1 root root  10 Sep  3 13:59 pci-0000:00:10.0-scsi-0:0:0:0-part1 -> ../../sda1
lrwxrwxrwx 1 root root  10 Sep  3 13:59 pci-0000:00:10.0-scsi-0:0:0:0-part2 -> ../../sda2
lrwxrwxrwx 1 root root  10 Sep  3 13:59 pci-0000:00:10.0-scsi-0:0:0:0-part5 -> ../../sda5
lrwxrwxrwx 1 root root   9 Sep  3 13:59 pci-0000:02:04.0-ata-2 -> ../../sr0
lrwxrwxrwx 1 root root   9 Sep  3 13:59 pci-0000:02:04.0-ata-2.0 -> ../../sr0
```

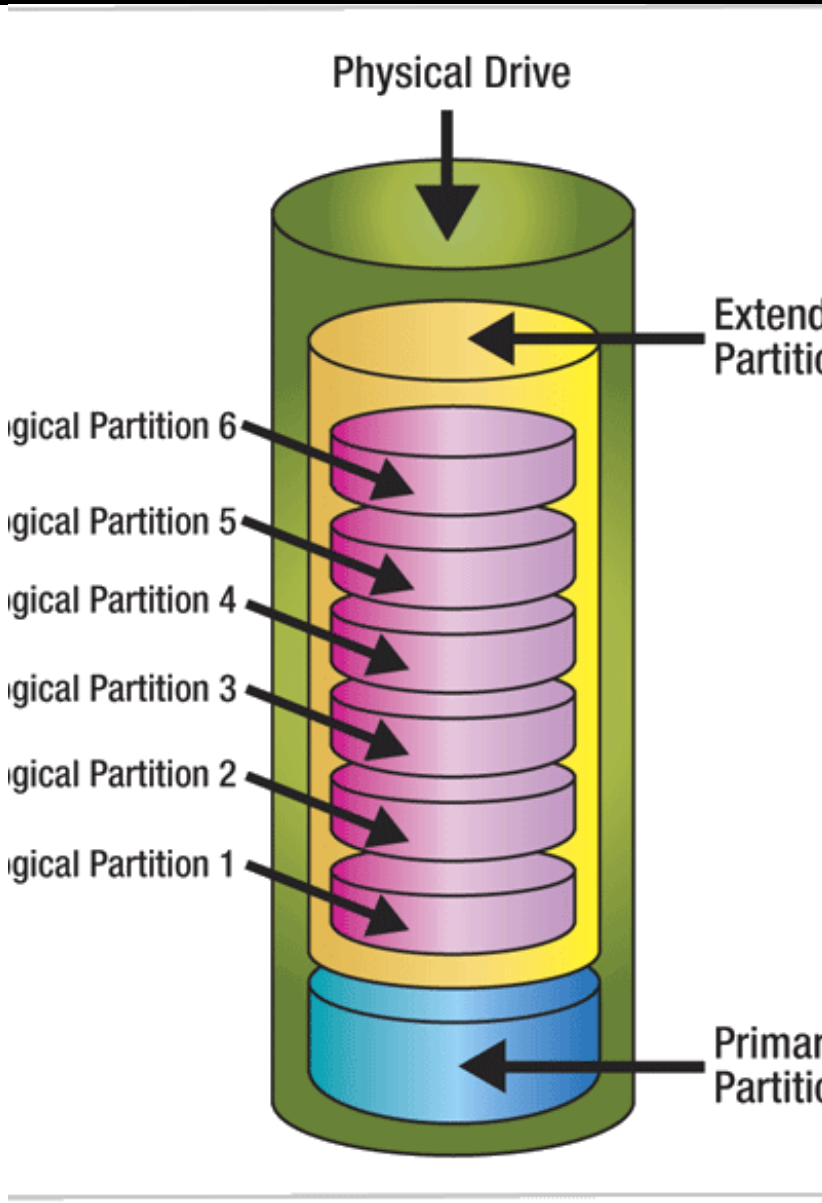
  - /sys/block

```
guy@debian-guy:~$ ls /sys/block
sda  sr0
```



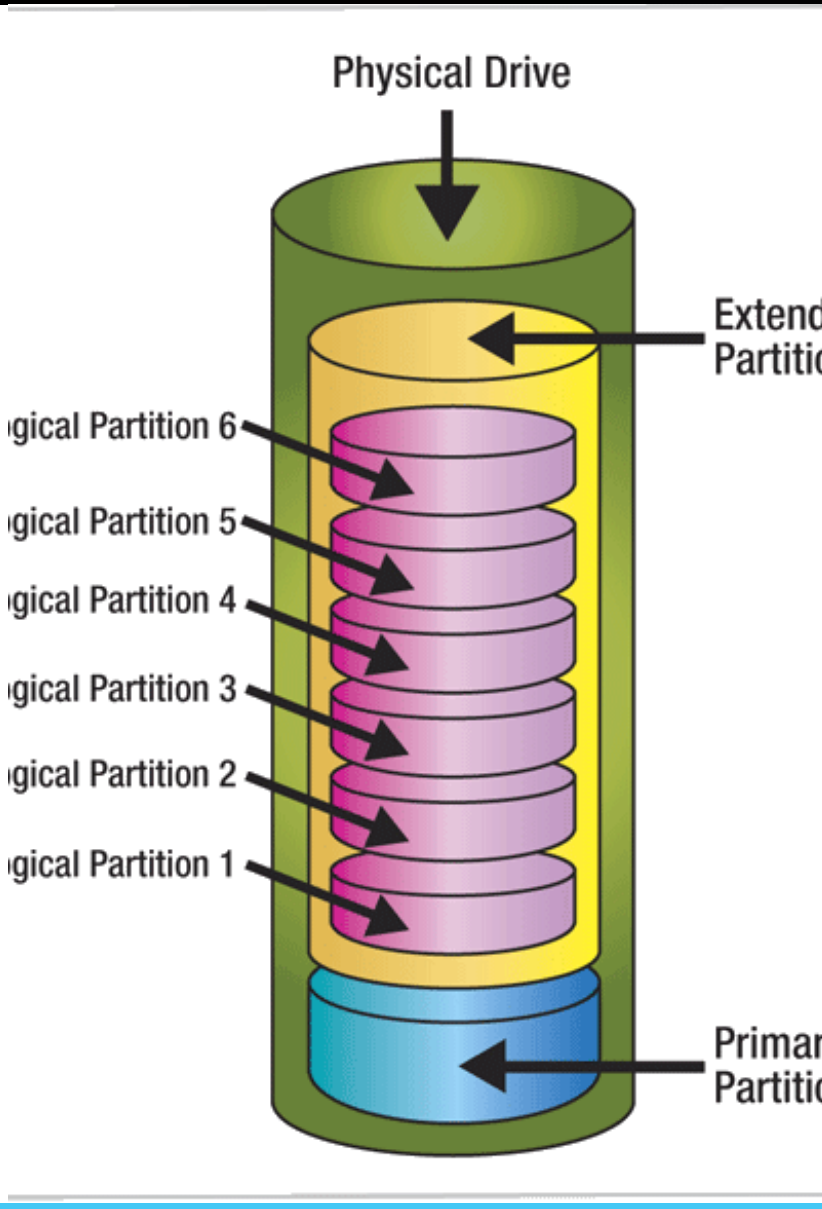
PARTITIONING A STORAGE

- “bucket o’ bits” when you buy it
- Partitioning = making it accessible to an OS by giving it a partition (address) table.
- 2 options :
  - Master Boot Record (MBR)
  - GUID Partition Table (GPT)



PARTITIONING A HARD DISC

- MBR
- First developed, wide backwards compatibility (Legacy BIOS)
- Index located at specific point on disc
- 4 partitions max
- But 1 (and only 1) of them could be the EXTENDED partition
  - “the rest of the disc”
  - That may be split up into 23 logical partitions max
- 32 bit
  - 2 TB discs max



PARTITIONING A HARD DISC

- GPT
- The standard in most modern systems (needs modern UEFI Bios)
- Index redundantly spread across the disc
- 128 partitions max
- 64 bit
  - 9.7 zetabytes discs max (1 zetabyte is about 1 billion terabytes)

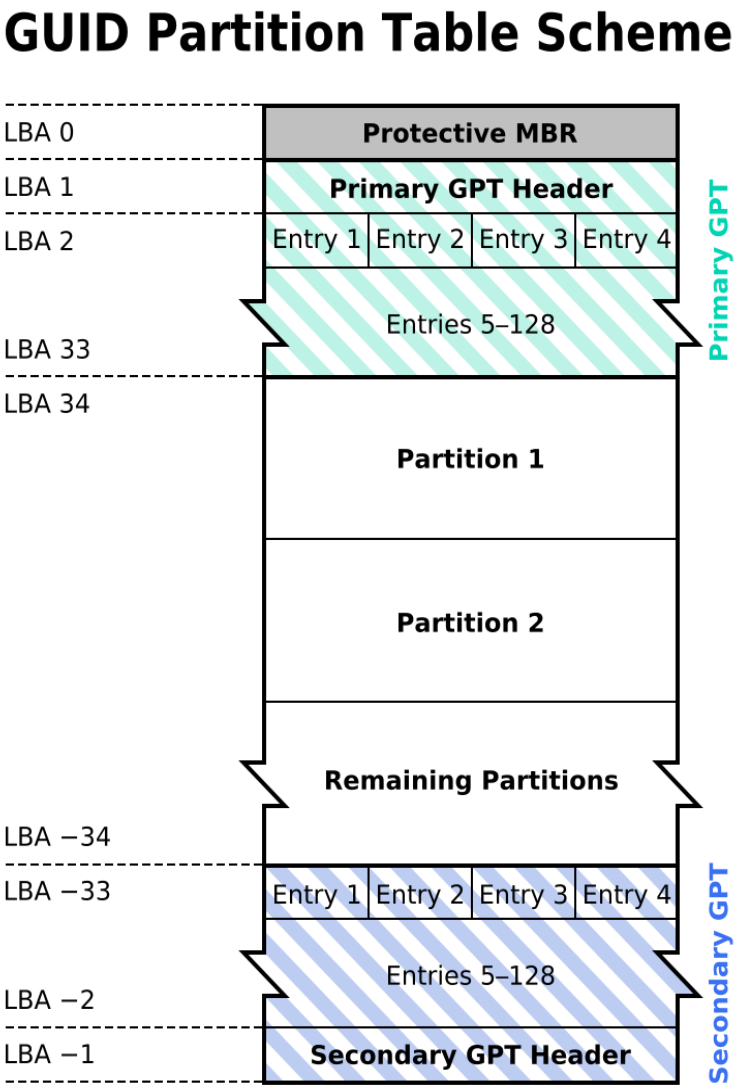
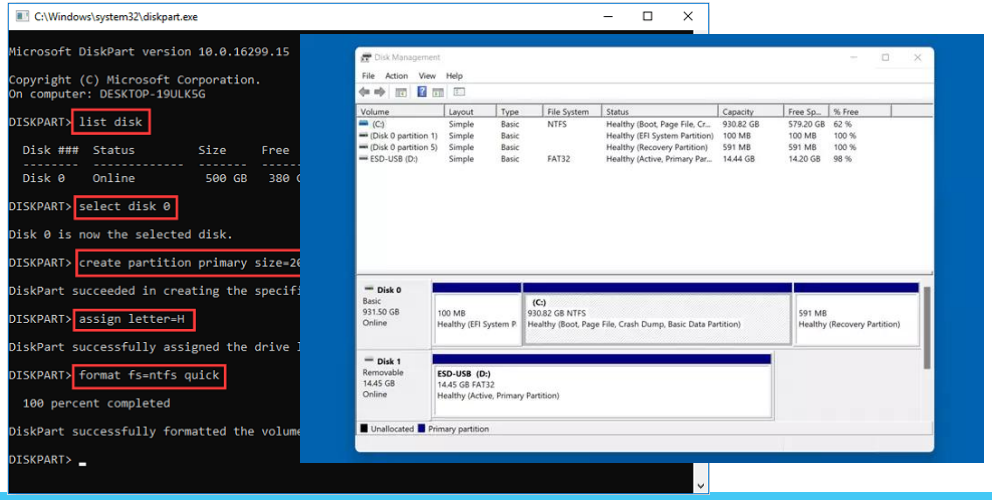


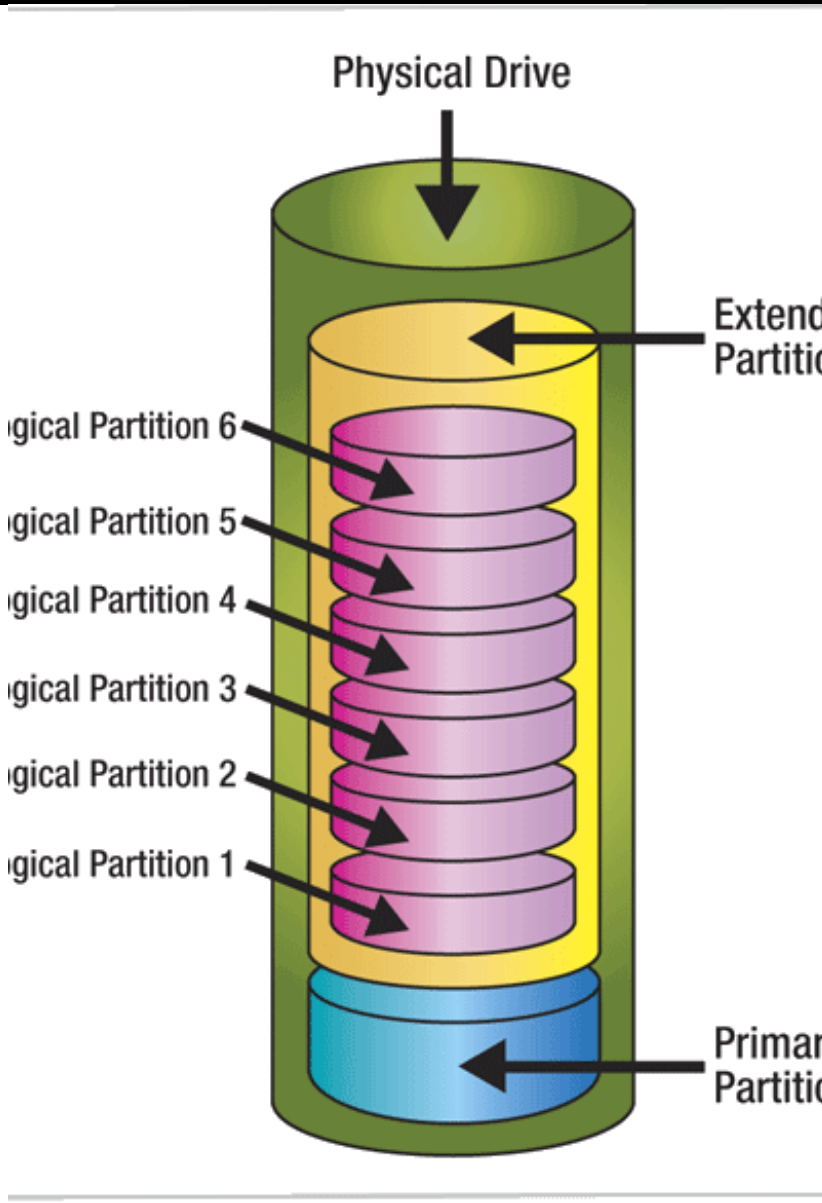
Image source : wikipedia

PARTITIONING IN LINUX

- Commands available
  - **fdisk**, **cfdisk**, **sfdisk**, **parted**
- E.g. /dev/sda
  - SCSI / SATA physical disc
    - One disc per letter
    - So /dev/sdc is the third disc

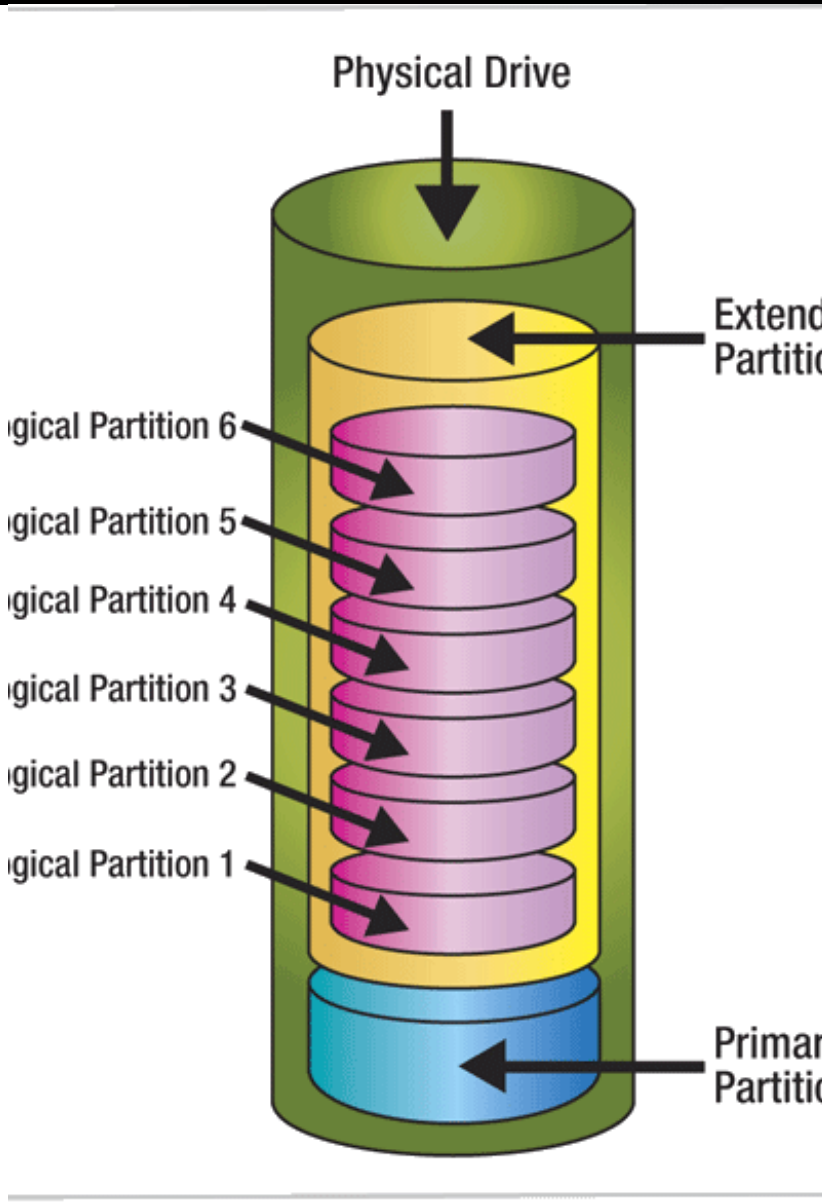
PARTITIONING IN WINDOWS





WHY PARTITIONING AT ALL ?

- Why not read/write directly to /dev/sdX ?
  - YOU CAN !
  - Some applications do that
  - Need root power though
  - Have fun with that one (basically, program your own file system !)
- Everything in 1 partition ?
- Flexibility
- Home directory explosion risk
- THINK BEFORE YOU PART



EXAMPLE PARTITION PLAN

Partition	Size	Description
/dev/sda1	100MB	Will be used to house the /boot partition in which the bootloader configuration and kernel images are stored. Separate partition as none of the files on this partition are needed during regular operations (this partition is not automatically mounted)
/dev/sda2	12GB	Main (root) partition which will host the Linux operating system
/dev/sda3	27GB	Partition which will hold the /home files (files and folders of the users of the system).
/dev/sda4	900MB	Swap partition. The desktop system will have enough physical memory and the system will not use software suspend (write memory content to disk and hibernate the system) so the swap does not need to be as large as the physical memory

PARTITIONING IN ACTION

```
# fdisk /dev/sda
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-12621, default 1): (Press return to use the default "1")
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-621, default 621):
+100M
```

CREATING THE OTHERS

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (97-12621, default 97): (Return again)
Using default value 97
Last cylinder or +size or +sizeM or +sizeK (97-12621, default 12621):
+12288M

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (4041-12621, default 4041): (Return again)
Using default value 4041
Last cylinder or +size or +sizeM or +sizeK (4041-12621, default
12621): +27648M

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 4
First cylinder (12021-12621, default 12021): (Return again)
Using default value 12021
Last cylinder or +size or +sizeM or +sizeK (12021-12621, default
12621): (Return)
```

SETTING THE PARTITION TYPE

```
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 83
Changed system type of partition 1 to 83 (Linux)

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 83
Changed system type of partition 2 to 83 (Linux)

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 83
Changed system type of partition 3 to 83 (Linux)

Command (m for help): t
Partition number (1-4): 4
Hex code (type L to list codes): 82
Changed system type of partition 4 to 82 (Linux swap)
```

```
Command (m for help): w
```

PREPARING YOUR STORAGE

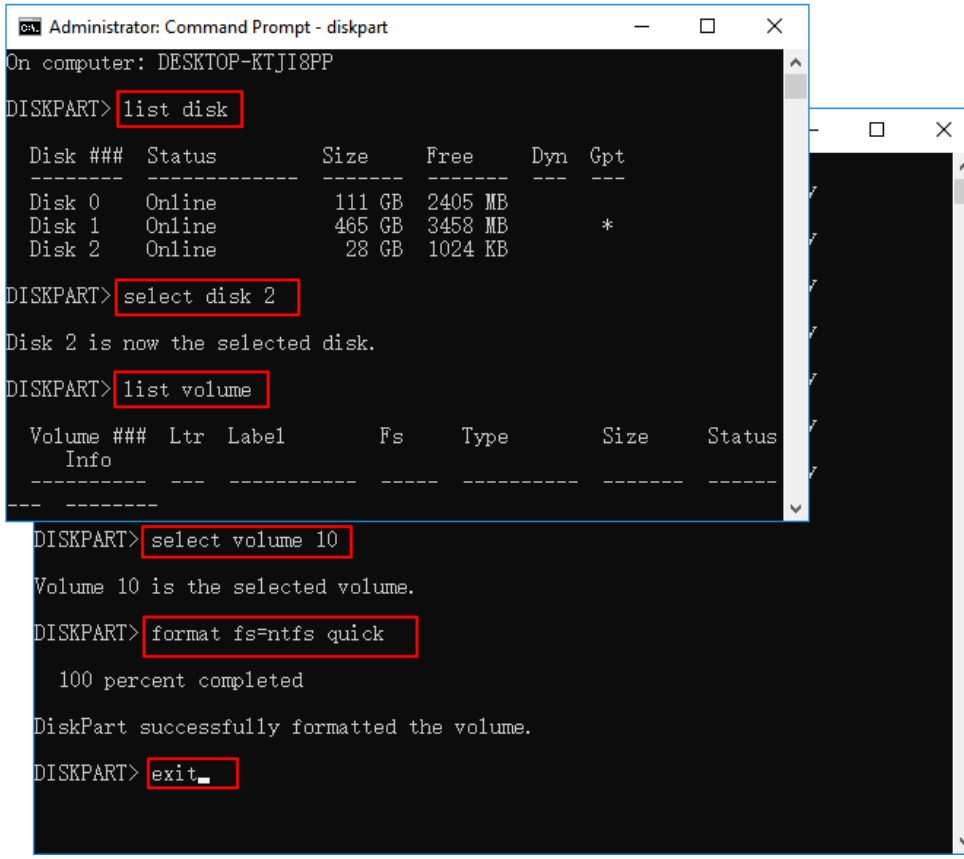
PART2 : FORMATTING

howest.be

FORMAT YOUR PARTITION

- “Format” a storage = putting a filesystem on that storage
- Many filesystem implementations exist!

```
# mkfs.ext2 -L boot /dev/sda1
# mkfs.ext3 -L root /dev/sda2
# mkfs.ext3 -L home /home/sda3
```



FILE SYSTEM FLAVORS (LINUX)

- Ext2
- Ext3
  - Ext2 + journals
- Ext4
  - Ext3 + large file support
- XFS
  - High parallel throughput
  - FAST !
- ZFS
  - From Solaris => Linux
  - File checksum, remote replication , …
  - PETABYTES
- ReiserFS
  - Fast, small files

FILE SYSTEM FLAVORS (OTHERS)

- Windows
  - NTFS
    - Default since Windows NT
  - FAT
    - FAT32 sticks around, but variations exist
  - exFAT
    - For Removable/media, large files support
  - reFS
    - Resilient File System (Windows Server PRO only)
- Apple
  - Apple File System
  - FAT/exFAT

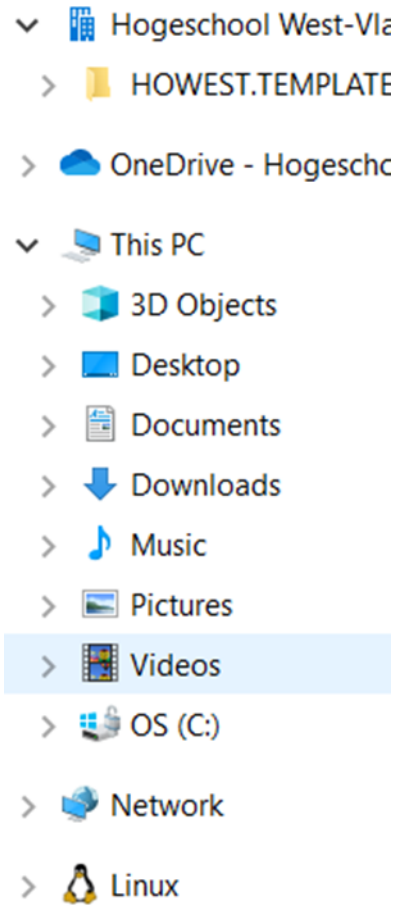
ACCESSING YOUR STORAGE

MOUNTING

FILE SYSTEM CONCEPTS

- Hierarchically structured tree
- Every location has its meaning
- Standardized, but variants exist

```
/
+- bin/
+- ...
+- home/
|  +- thomas/
|  |  +- Documents/
|  |  +- Movies/
|  |  +- Music/
|  |  +- Pictures/          <-- You are here
|  |  |  `- Backgrounds/
|  |  `- opentasks.txt
|  +- jane/
|  `- jack/
+- lib/
+- ...
`- var/
```



THE ART OF MOUNTING

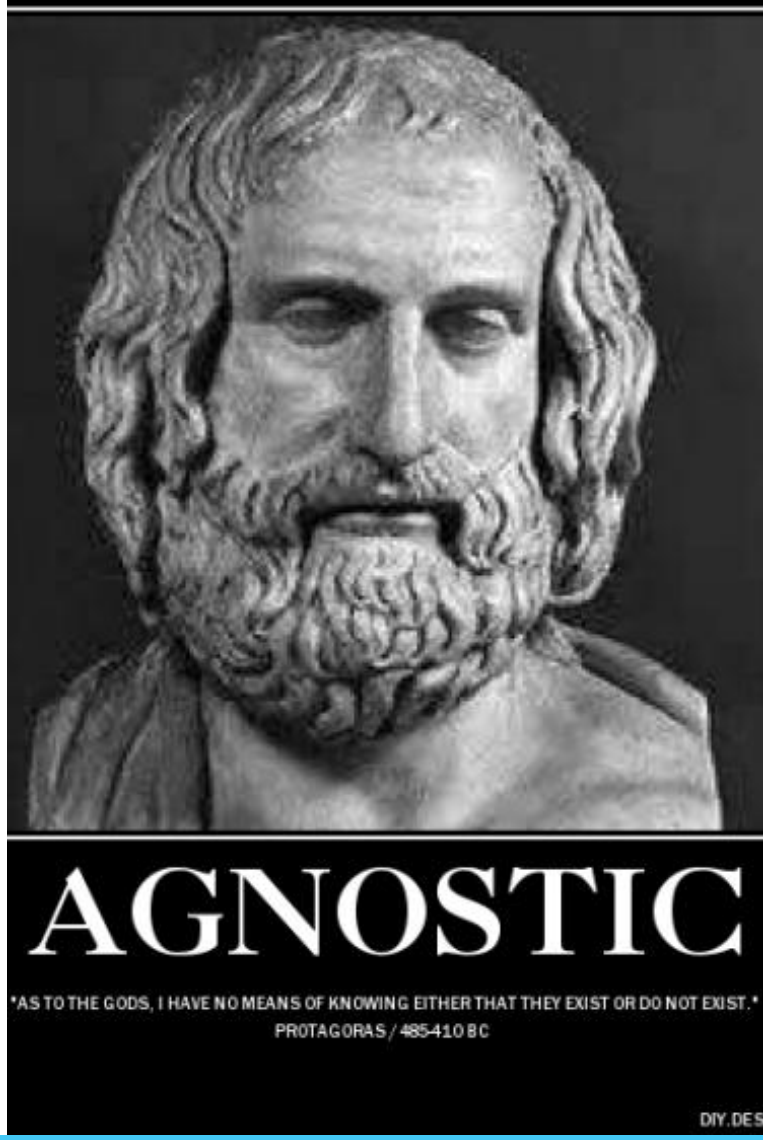
- The root of a file system is stored somewhere
- (when on HDD) On a partition !
- You can combine several partitions to form the complete file system
- Combining one partition with the file system is called *mounting a file system*
- Your file system is always seen as a tree structure, but parts of a tree (a *branch*) can be located on a different partition, disk or even other medium (network storage, DVD, USB stick, ...).

THE ART OF MOUNTING

- identify a location of the file system as being a mount point
  - E.g. /home is the mount point
  - under which every file is actually stored on a different location (everything below /home is stored on the second partition).
- The partition you "mount" to the file system doesn't (need to) know where it is mounted on.
- You can mount the users' home directories at /home (which is logical and not-crazy)
- but you could very well mount it at /srv/export/systems/remote/disk/users.
  - And expose THAT to be mounted across a network

WHAT DOES IT ALL MEAN !!???

- Linux allows your programs to be agnostic
  - Not the religious kind…
  - They just see the directory tree
  - The don't care what is underneath
    - 1 partition
    - X partitions
    - Network drive, cd rom, usb device, /dev/null ☺



MOUNTING AND MAKE IT STICK

```
# mount -t ext3 /dev/sda7 /home
```

Gone after every reboot…

/etc/fstab makes it permanent

```
/dev/sda8  /          ext4    defaults,noatime      0 0
/dev/sda5  none       swap    sw                    0 0
/dev/sda6  /boot      ext4    noauto,noatime        0 0
/dev/sda7  /home      ext4    defaults,noatime      0 0
/dev/sdb1  /media/usb auto    user,noauto,gid=users 0 0
```

Easy mount & unmount afterwards

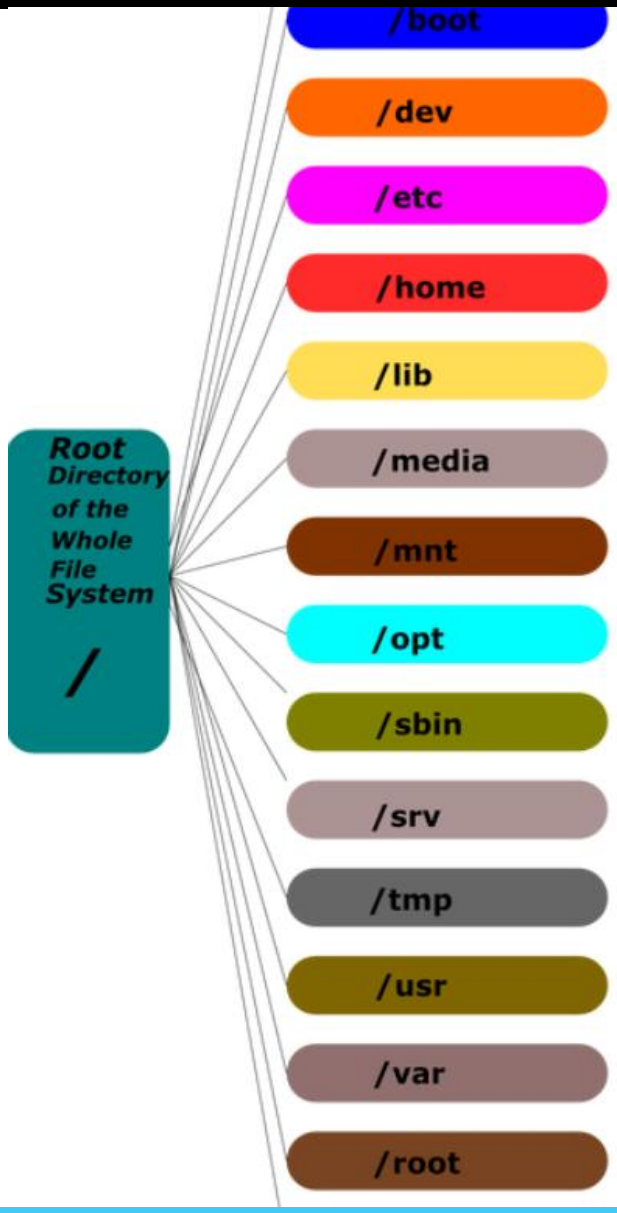
```
# mount /home
# umount /home
```

/ETC/FSTAB IN DEPTH

- The device to mount
- The location to mount the device to (mount point)
- The file system type (auto if you want Linux to automatically detect the file system)
- Additional options
  - "defaults" if you don't want any specific option)
  - noatime (don't register access times to the file system to improve performance)
  - users (allow regular users to mount/umount the device)
- Dump-number
- File check order

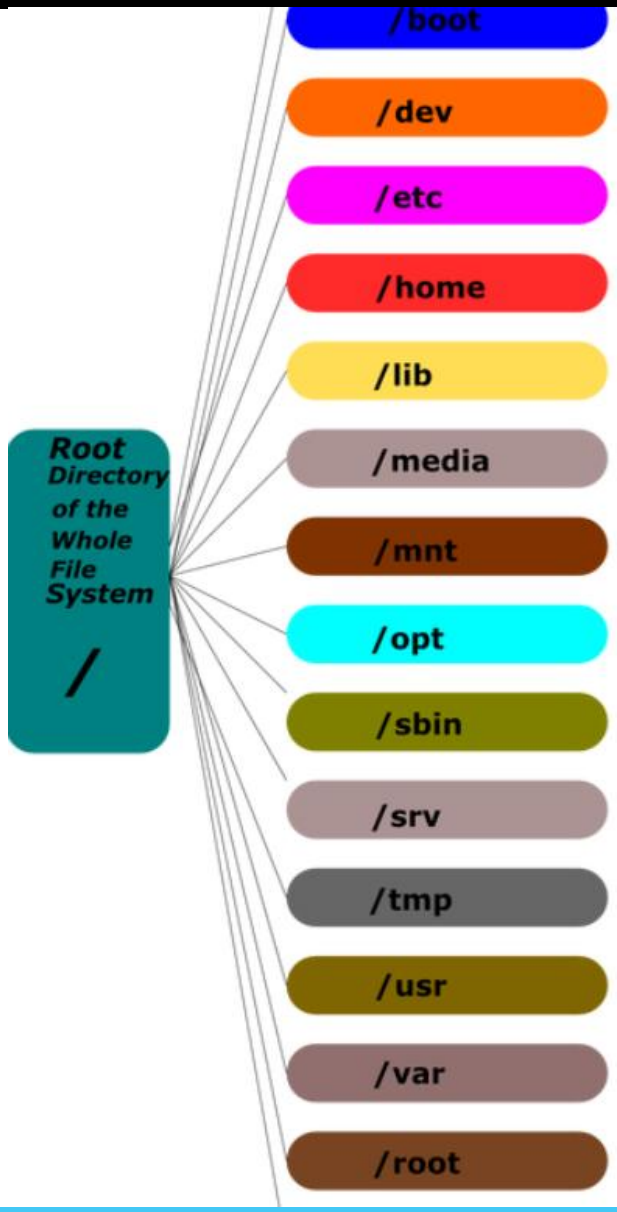
FILE SYSTEM SYSTEM LOCATIONS

- /bin
  - executable programs needed to run the system
  - Migration towards /usr/bin and symbolic links for compatibility
- /etc
  - contains all the configuration files for the system
  - not the user-specific configurations
- /lib
  - system libraries necessary to run the system
  - to run the commands which are located inside /bin.
  - also being migrated towards /usr/lib.
- /sbin
  - executable programs to run the system.
  - /sbin contains programs solely for system administrative purposes



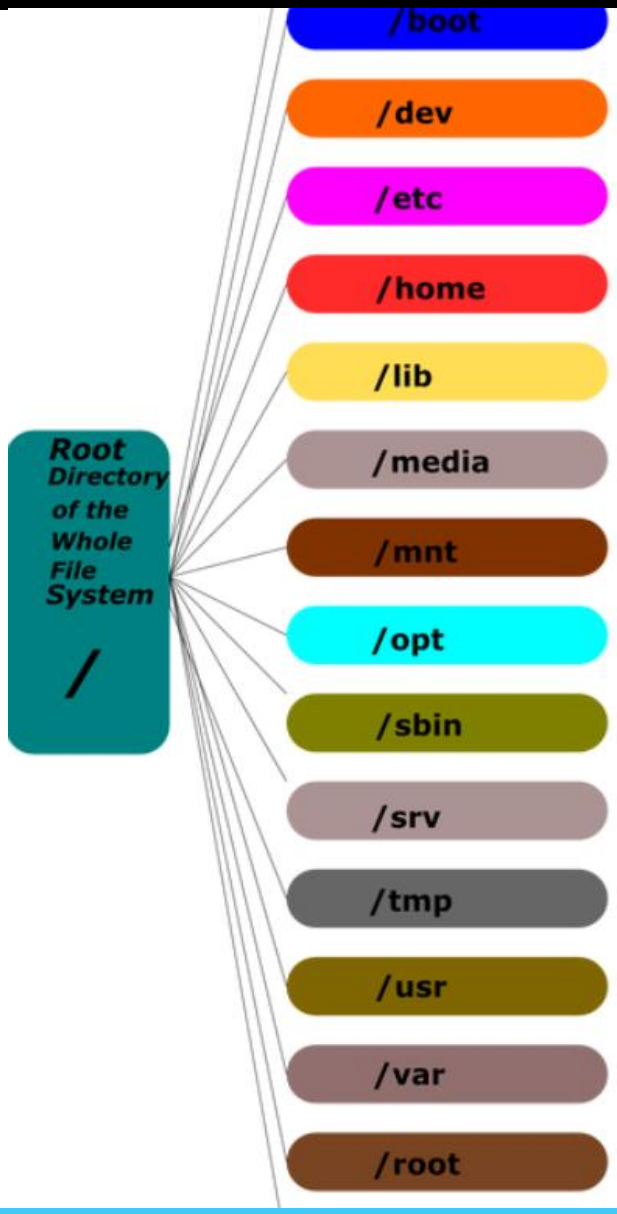
FILE SYSTEM USERLAND LOCATIONS

- /usr
  - root of the userland locations
  - usually the mount point of any separate medium
- /usr/bin
- /usr/lib
- /usr/sbin



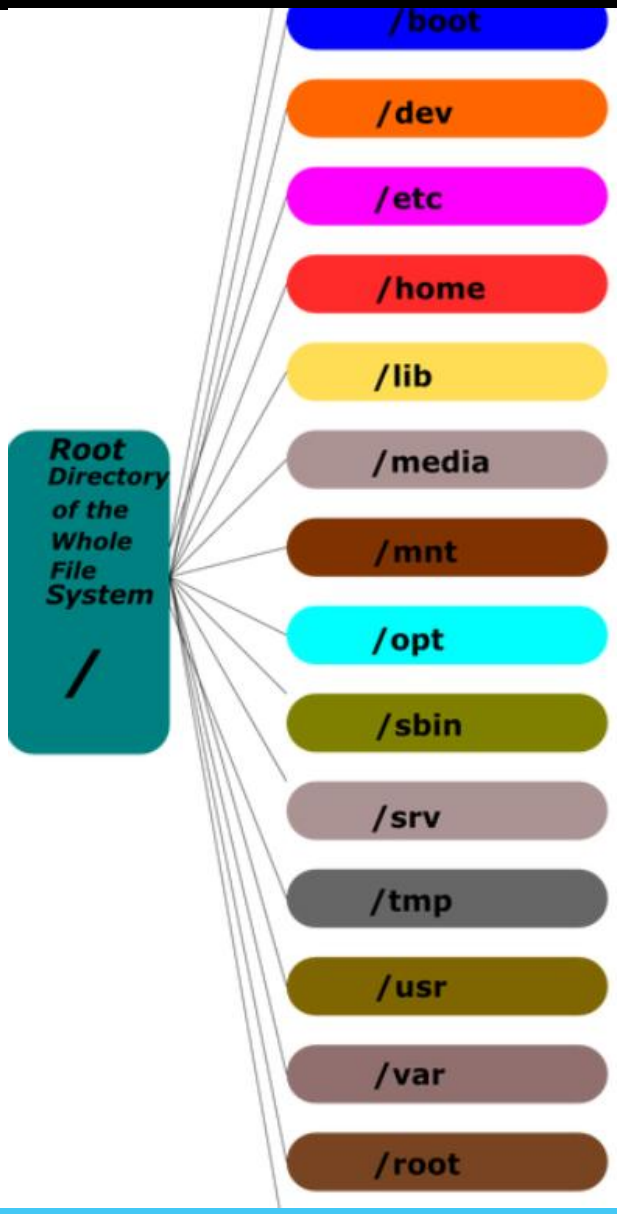
FILE SYSTEM GENERAL LOCATIONS

- /home contains the home directories of all the local users
- /boot
  - contains the static boot-related files
  - not actually necessary once the system is booted
  - E.g. bootloader configuration and kernel image
- /media
  - the mount points for the various detachable storage (like USB disks, DVDs, ...)
- /mnt
  - temporarily mounted media
  - read: not worth the trouble of defining them in fstab



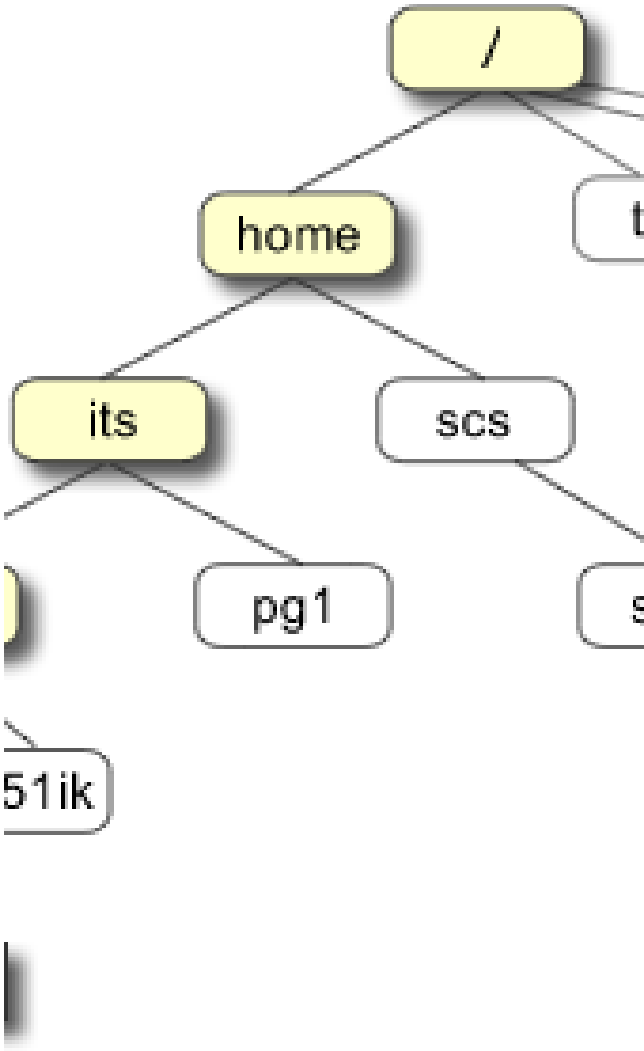
FILE SYSTEM GENERAL LOCATIONS

- /opt
  - add-on packages and is usually used to install applications into
    - which are not
    - provided by your package manager natively
      - =>/usr
    - nor built specific to the local system
      - =>/usr/local.
- /tmp contains temporary files for the system tools.
- /var contains data that changes in size, such as log files, caches, etc.



THE DIRECTORIES

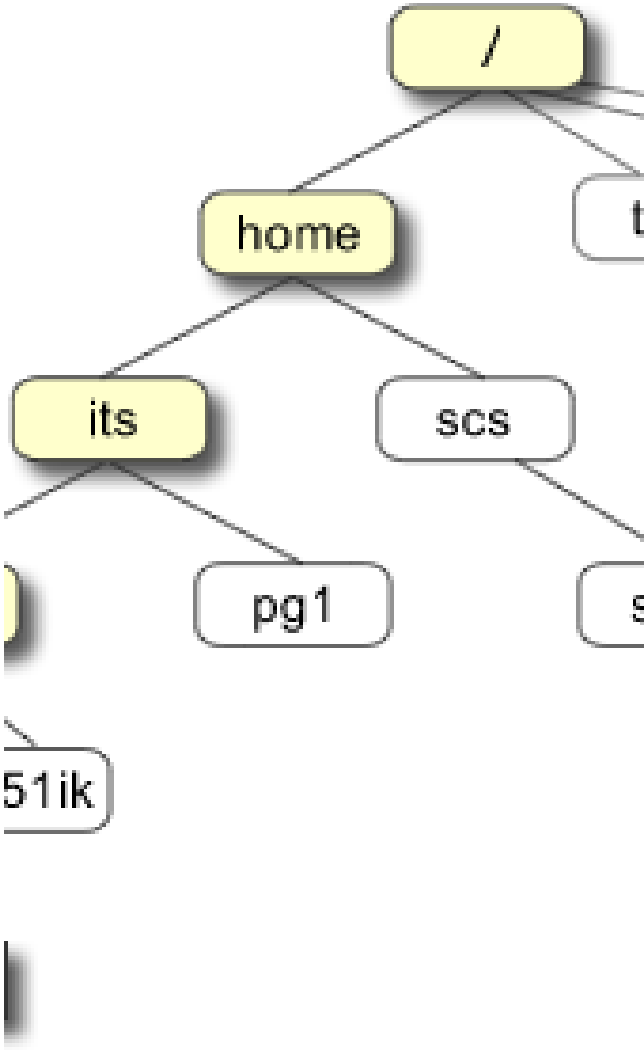
- Files that contain information about other files
- A UNIX directory
  - a file
    - has an owner, group owner, size, access permissions, etc but their meaning differs slightly (see later)
    - has an I-node type structure
    - many file operations can be used on directories
    - Kernel imposes special structure on it when the command to create it used :
      - mkdir
  - whose data is an array or list of (filename, i-node#) pairs.



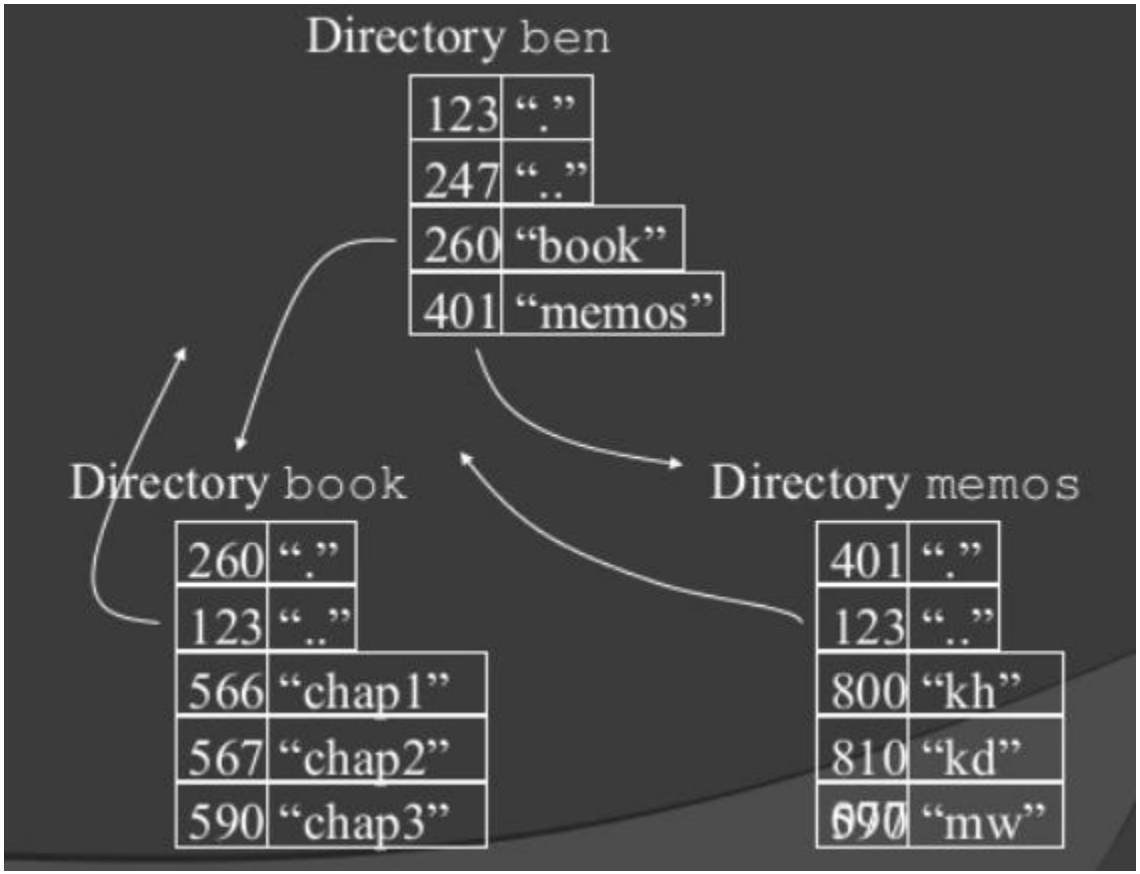
MKDIR AT WORK

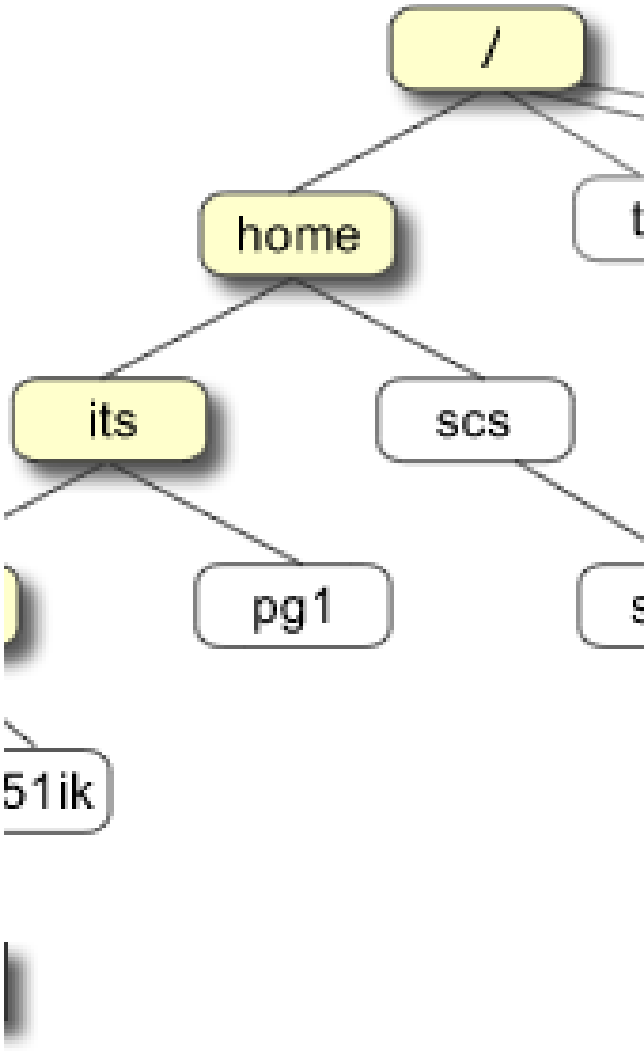
- Mkdir subdir
  - Creates a subdir file and an inode for it
  - The inode and file name are added to the parent file

120	“fred.html”
207	“abc”
135	“bookmark.c”
201	“subdir”



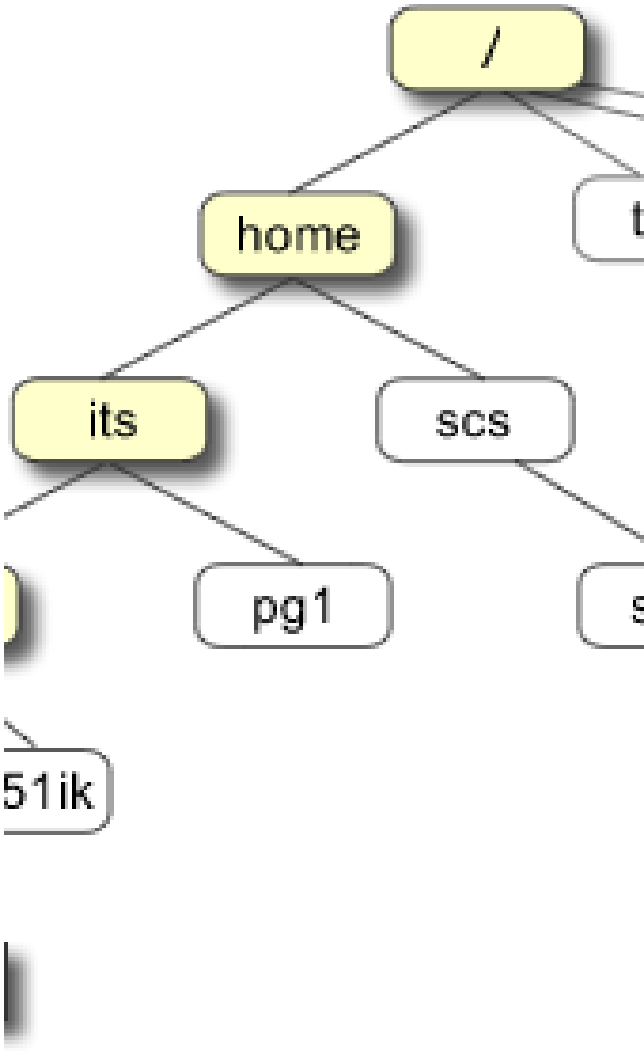
SUBDIRECTORIES





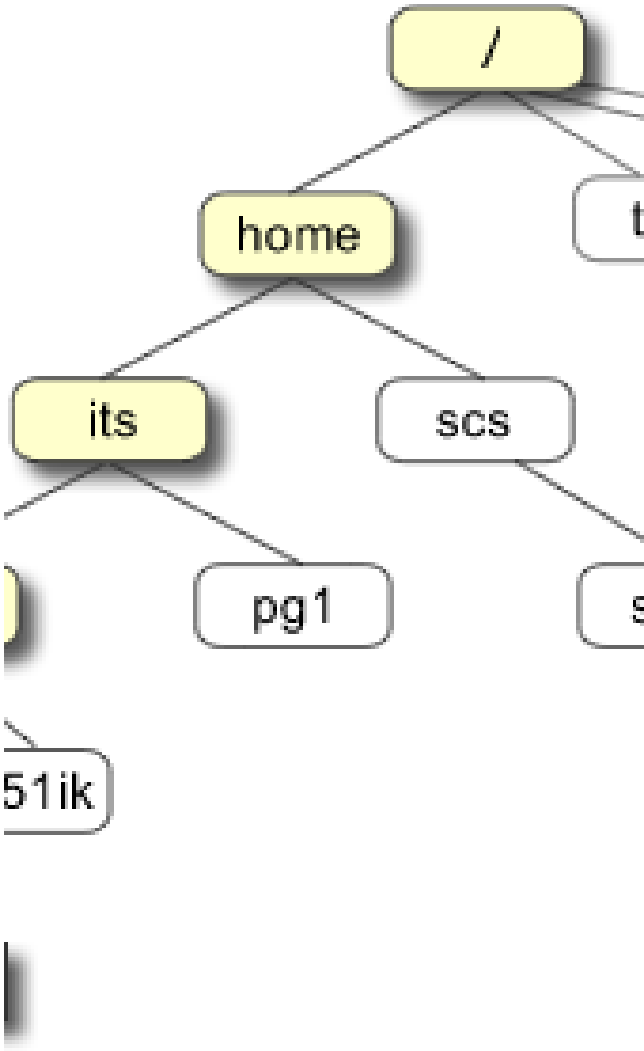
FILES AND INODES (ONLY ON LINUX!)

- Information about each regular local file in a structure called an INODE.
- There is 1-to-1 mapping between the INODE and a file.
- Multiple files may have the same INODE
- Each INODE is identified through its number, a non-negative integer
- The INODE hash array is a list of allocated INODE‘s located at the beginning of the file system



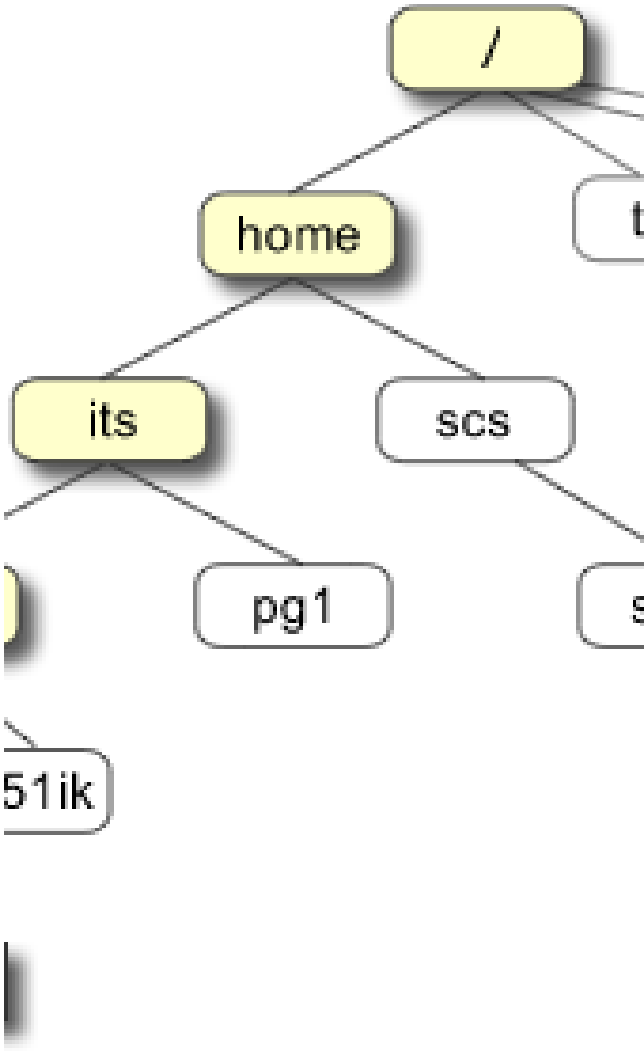
FILES AND INODES (ONLY ON LINUX!)

- INODE structures (UNIX i-node list)
  - are stored on the file system block device (e.g., disk)
  - in a predefined location on the disk. UNIX: the i-node list.
  - Where it is exactly is file system implementation specific.
- INODE numbers have only local meaning (to each file system)
- One file system per partition, one INODE table per file system.



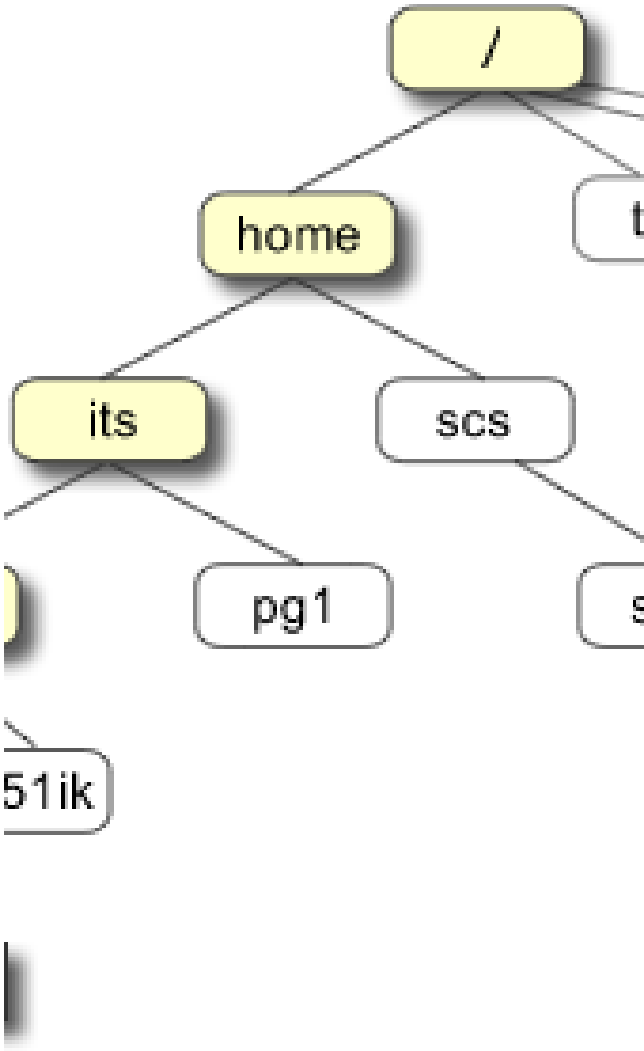
DIRECTORY STRUCTURES

- This process also explains the difference between hard links and soft links.
- A hard link directory entry is a direct pointer to a file INODE.
- A soft link is a pointer to another directory entry.
- In a link, rm clears the directory record.
  - the i-node number is set to 0
  - the file may not be affected
  - The file i-node is only deleted
    - when the last link to it is removed;
    - the data block for the file is also deleted (reclaimed).



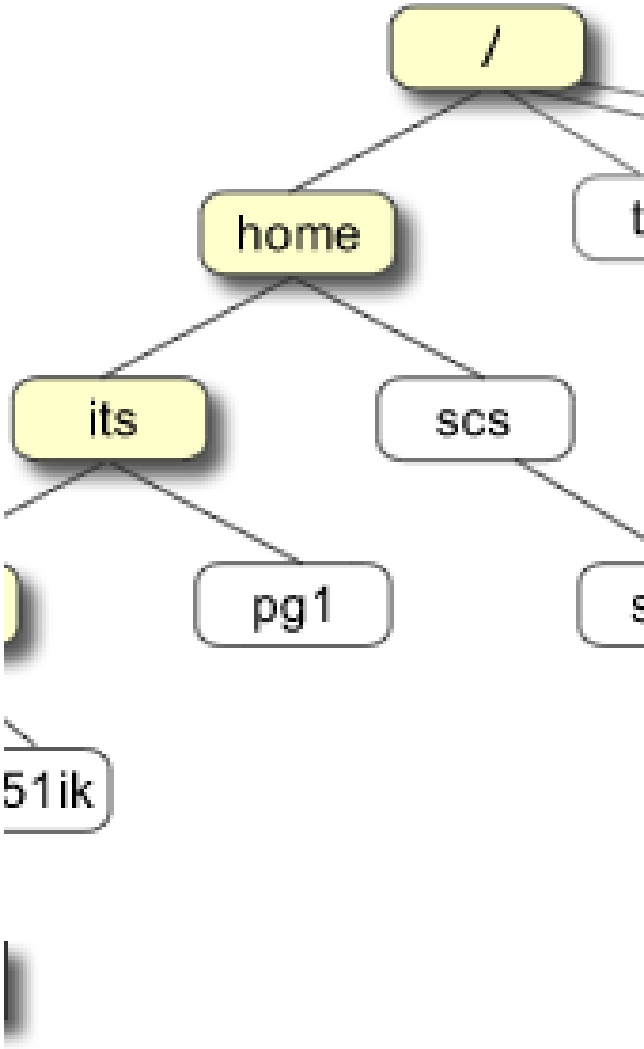
HARD LINKS

inode #2		inode #555		inode #333
. (dot)	2	. (dot)	555	*Disk blocks for the cp / ln / mv file (link count: 3)*
.. (dot dot)	2	.. (dot dot)	2	
home	123	rm	546	
bin	555	ls	984	
usr	654	cp	333	
		ln	333	
		mv	333	



SOFT LINKS

Symbolic link
Create a new inode
Can link directories
Can link files across file system
Does not change hard link count of the linked inode



CREATING LINKS

Linux :

Hard link : ln source destination

Soft link : ln – s source destination

Windows :

ONLY WITH ADMIN POWERS

Hard link : mklink /H Link Target

Soft link : mklink /D Link Target

“hard” Link to DIR : mklink /J Link Target (a.k.a. Directory Junction) does not need admin power, only works on the same filesystem.)

MANAGING YOUR FILES

ARCHIVING

howest.be

ARCHIVING VS. COMPRESSION

- Archiving collapses multiple files into one
  - → A few files or multiple directories
- Compression makes a file smaller
  - → Remove redundant information, replace with a smaller code
  - → Can be applied to individual files, groups of files or entire directory trees

LOSSLESS VS LOSSY COMPRESSION

Lossless:

- Decompressed file is the same as the original
- Doesn't compress as well as lossy
- For data you want to preserve
- Logs, documents, binaries, configuration

Lossy:

- Decompressed file might have lost information from the original
- Drops "unimportant" information from the file to make it compress better
- Images, sound, movies

USING GZIP/GUNZIP/BZIP2/BUNZIP2

gzip foo # removes foo; creates foo.gz

gunzip foo.gz # removes foo.gz; creates foo

gunzip –l foo.gz # shows statistics

bzip2 foo # removes foo; creates foo.bz2

bunzip2 foo.bz2

bunzip2 –l foo.bz2 # DOESN'T EXIST!

TAPE ARCHIVE - TAR

tar –cf foo.tar * # create

tar –tf foo.tar # show info

tar –xf foo.tar # extract

tar –xf foo.tar home/joe

only extract home/joe

tar -czf /dev/st0 /home

tar –czf foo.tgz * # gzip

tar –xjf foo.tbz # bunzip2

TARBALL vs TARBOMB

ZIP

```
zip output.zip file1 file2 file3

zip foo.zip file.doc     # One file

zip –r foo.zip Documents # recurse


unzip –l foo.zip         # show contents

unzip foo.zip            # extract all

unzip foo.zip file1      # just file 1

unzip foo.zip Documents/projectA/*

     # everything under Documents/projectA
```

USERS

howest.be

USER ACCOUNTS AND PASSWORDS

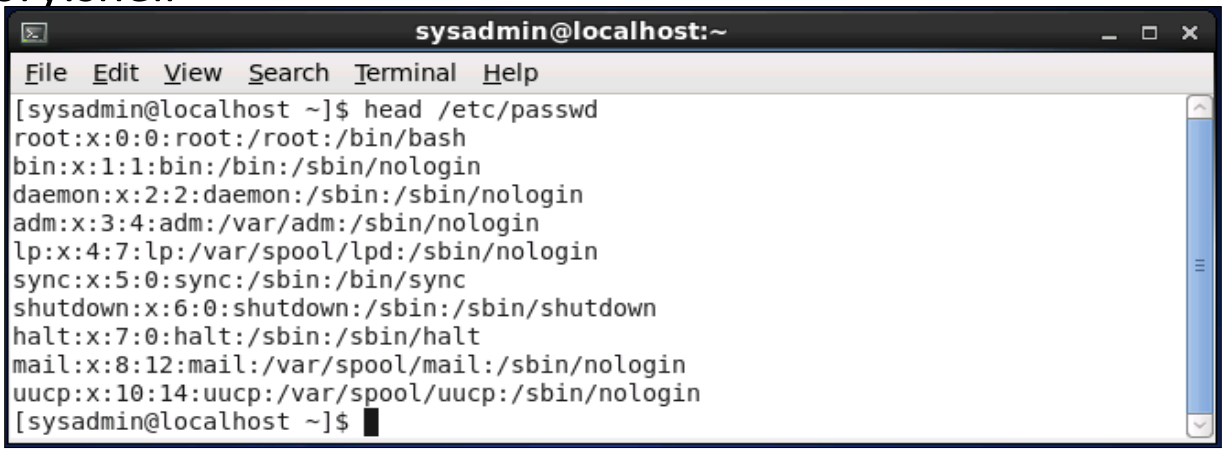
USER ACCOUNTS

- Files in the `/etc` directory contain system configuration data.
- The `/etc/passwd` file defines some of the account information for user accounts.

THE /ETC/PASSWD FILE

- Each line of the `/etc/passwd` file relates to a user account.
- Each line is separated into fields by colon characters. The fields from left to right are as follows:

  name:password placeholder:user id:primary group id:comment:home directory:shell



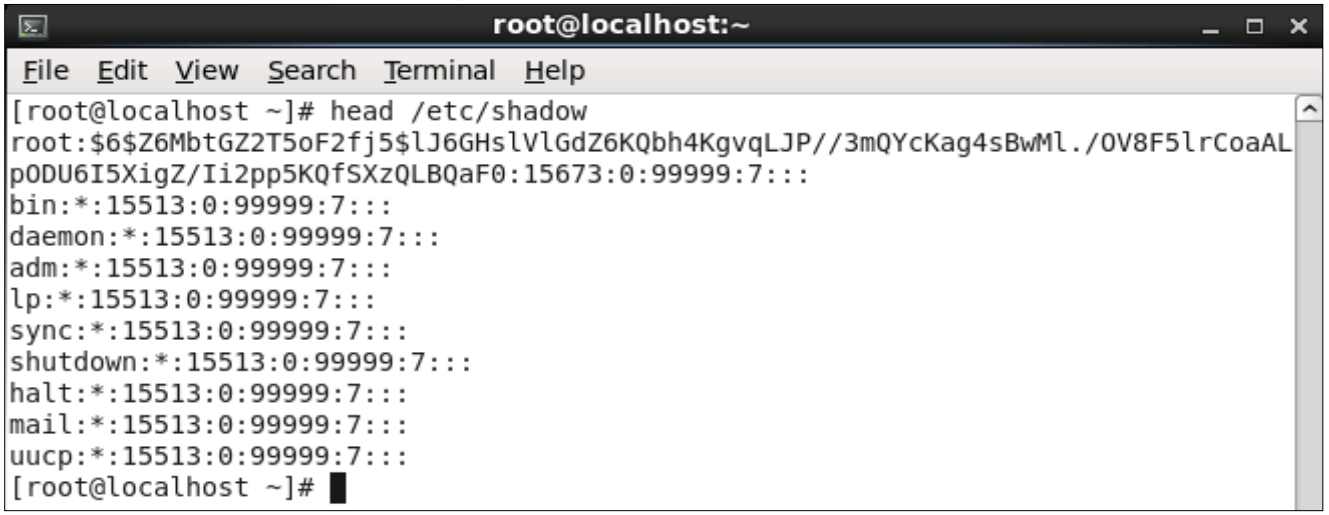
THE /ETC/PASSWD FILE

Field	Example	Description
name	root	This is the name of the account.
password placeholder	x	The **x** in the password placeholder field indicates to the system that the password is not stored here, but rather in the `/etc/shadow` file.
user id	0	Each account is assigned a user ID (UID).
primary group id	0	When a user creates a file, the file is owned by a group id (GID), the user's primary GID.
comment	root	This field can contain any information about the user, including their real (full) name and other useful information..
home directory	/root	This field defines the location of the user's home directory.
shell	/bin/bash	This is the location of the user's login shell.

THE /ETC/SHADOW FILE

- Contains account information related to the user's password.
- The fields of the `/etc/shadow` file are:

name:password:lastchange:min:max:warn:inactive:expire:reserved



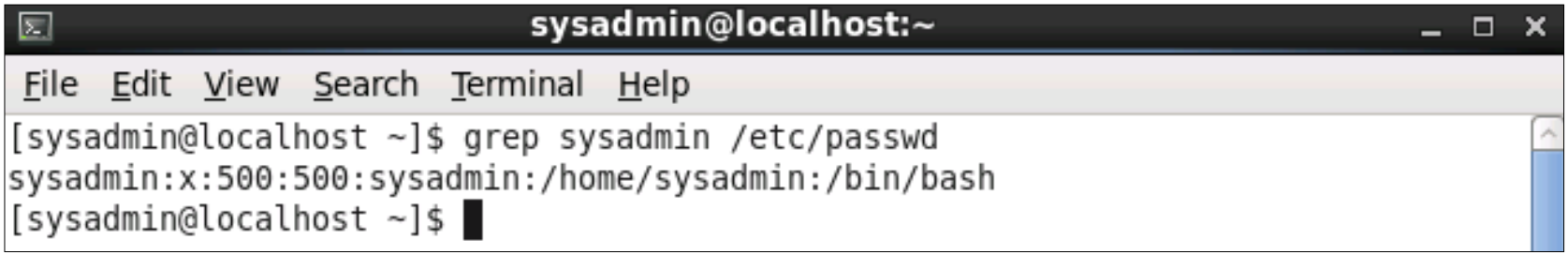
THE /ETC/SHADOW FILE

Field	Example	Description
name	sysadmin	This is the name of the account, which matches the account name in the `/etc/passwd` file.
password	6.....rl1	The password field contains the hashed password for the account.
last change	15020	This field contains a number that represents the last time the password was changed.
min	5	The password can't be changed again for the specified number of days.
max	30	This field is used to force users to change their passwords on a regular basis
warn	7	If the **max** field is set, the **warn** field indicates that the user would be "warned" when the **max** timeframe is approaching.
inactive	60	The **inactive** field provides the user with a "grace" period in which their password can be changed.
expire	15050	This field represents the number of days from January 1, 1970 and the day the account will "expire".

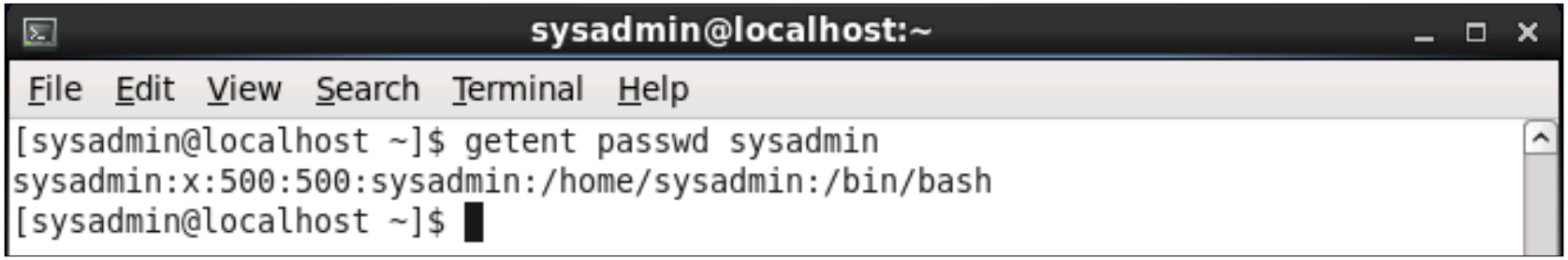
VIEWING ACCOUNT INFORMATION

VIEWING ACCOUNT INFORMATION

- To see the account information for the user name named "sysadmin", use the `grep sysadmin /etc/passwd` command:

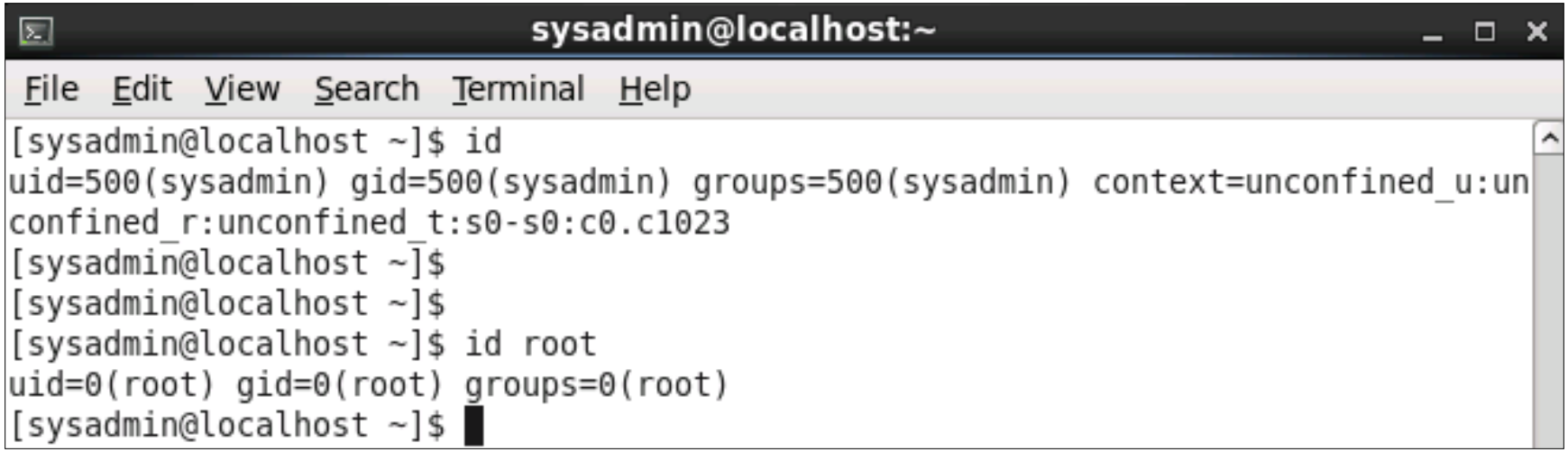


- Another technique is the `getent` command:



VIEWING LOGIN INFORMATION

- To verify your identity you can execute the `id` command:



SYSTEM ACCOUNTS

SYSTEM ACCOUNTS

- System accounts are designed to provide accounts for services that are running on the system.
- Have UIDs between 1-499
- Have non-login shells in `/etc/passwd`
- Have * in password field of `/etc/shadow`
- Most are critical for system operation.
- Only delete a system account when 100% certain it is not needed.

SYSTEM GROUPS

GROUP ACCOUNTS

- Each user can be a member of one or more groups.
- The `/etc/passwd` file defines the primary group membership for a user.
- Supplemental group membership is defined in the `/etc/group` file.
- Either the `grep` or `getent` commands can be used to display group information.

THE /ETC/GROUP FILE

- Each group is defined by this file.
- A colon delimited file with the following fields:

group_name:password_placeholder:GID:user_list

Field	Example	Description
group_name	mail	This field contains the group name.
password_placeholder	x	The "x" in this field is used to indicate that the password is stored in the `/etc/gshadow` file.
GID	12	Each group is associated with a unique Group ID (GID) which is placed in this field.
user_list	mail,postfix	This last field is used to indicate who is a member of the group.

CHANGING GROUPS

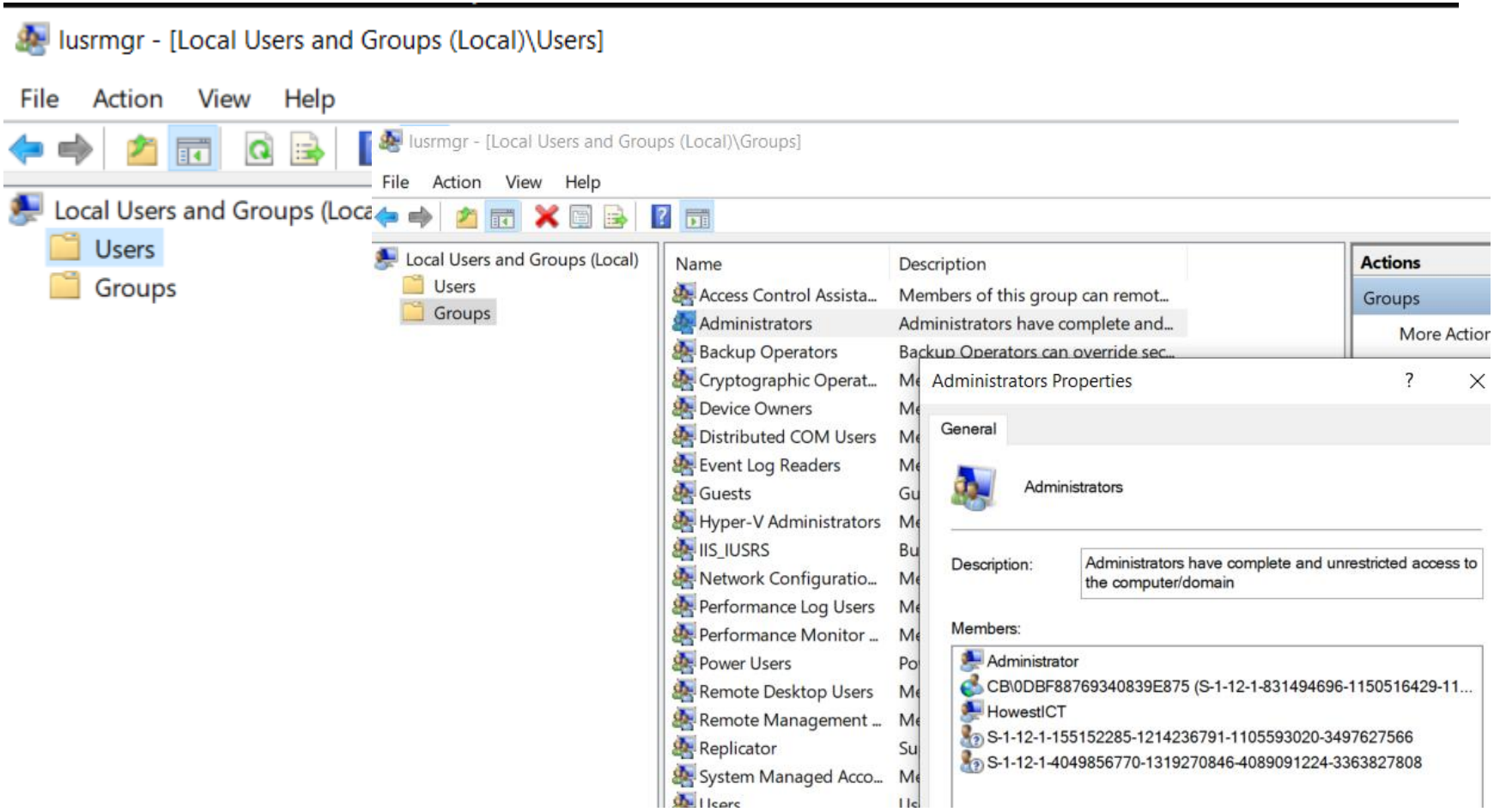
- Create a file that owned by one of your secondary groups by using:

  newgrp group_name

- If you're not a member of the group yet, you'll need to know the group's password !
- Opens a new shell with new primary group.
- Use `id` command to verify new primary group.
- Use `exit` command to return to previous shell.
- May be disabled due to group passwords.

WINDOWS

lusrmg.msc



WORKING WITH ROOT / ADMINISTRATOR

LOGGING IN AS ROOT

- Logging in directly to root account poses a security risk.
- Instead, use the `su --login` or `sudo` command.

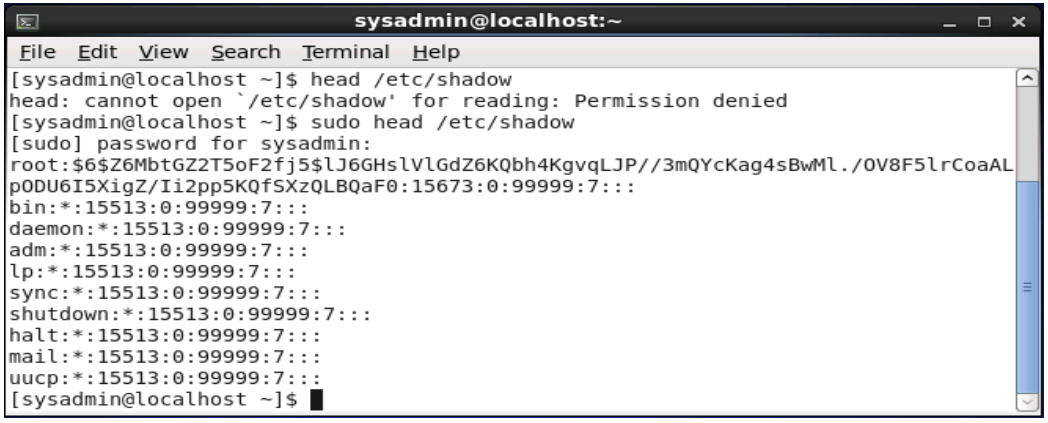
USING THE SU COMMAND

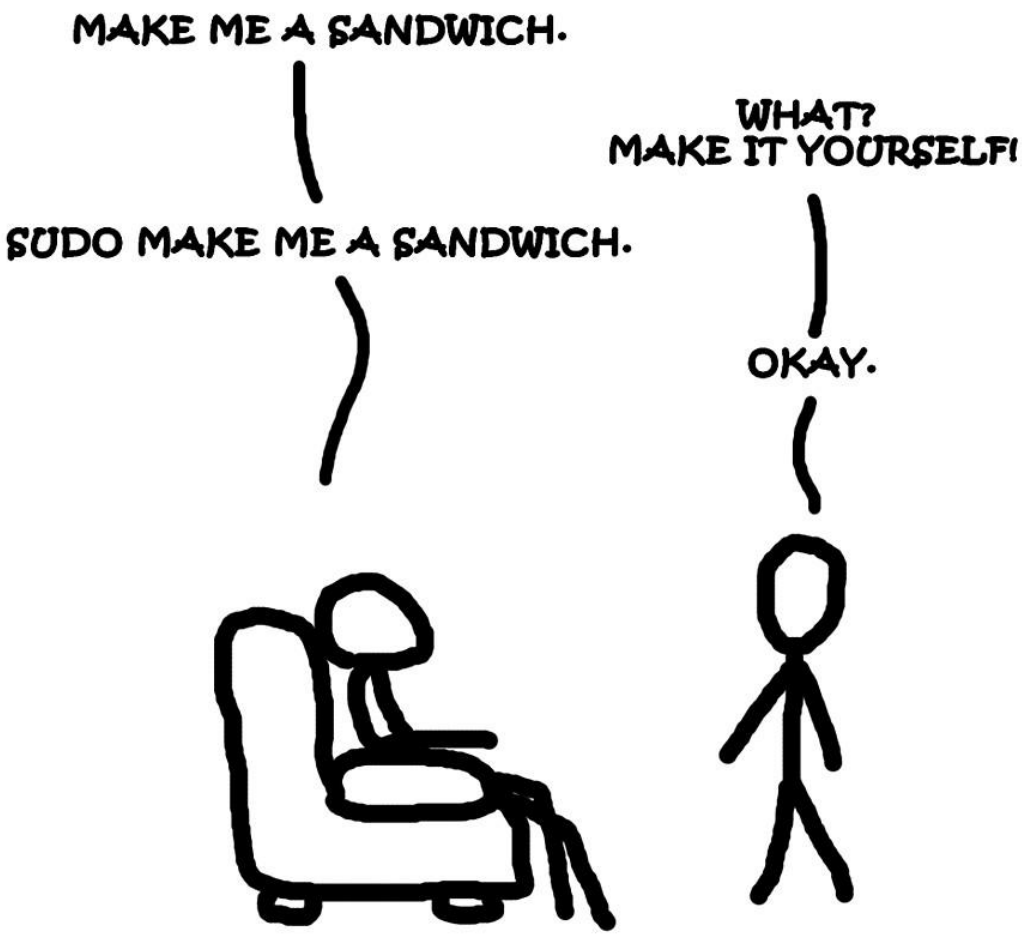
- The `su` command opens a new shell as a different user. (UID changes, but doesn't assume all env.)
  →Example: su user1
- To sign in as if the user had executed a login session
  →Example: su - user1
- Often used to run commands as the root user.
- Use the `--login` option for a full login shell.
- The root user is the default user.
- Use `exit` command to return to original shell.

FUN WITH SUDO

USING THE SUDO COMMAND

- The `sudo` command allows you to execute a single command as a different user.
- Must be set up by installation program or manually after install.
- Prompts user for their own password.




MAKE ME A SANDWICH.
WHAT?
MAKE IT YOURSELF!
SUDO MAKE ME A SANDWICH.
OKAY.

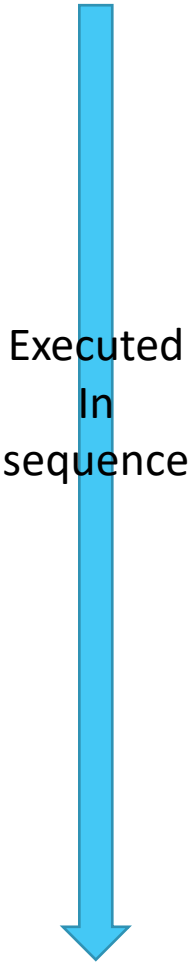
SETTING UP THE SUDO COMMAND

- Configuration is in the `/etc/sudoers` file.
- Modify this file with the `visudo` command.
- Uses `vi`/`vim` editors by default.
- Use the following to modify default editor:

```
export EDITOR=nano
```

A LOOK INSIDE THE /ETC/SUDOERS FILE

```
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

Defaults    env_reset

# Uncomment to allow members of group sudo to not need a password
# %sudo ALL=NOPASSWD: ALL

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```



LOOK AT THAT LINE !

```
root    ALL=(ALL) ALL
```

```
<user list> <host list> = <operator list> <tag list> <command list>
```

(NO)PASSWD:
(NO)EXEC:
(NO)SETENV:
(NO)LOG_INPUT
(NO)LOG_OUTPUT
(NO)MAIL


robm@homebox ~$ sudo su
Password:
robm is not in the sudoers file.
This incident will be reported.
robm@homebox ~$
HEY — WHO DOES SUDO REPORT THESE "INCIDENTS" TO?
YOU KNOW, I'VE NEVER CHECKED.
NICE
NAUGHTY

IN WINDOWS

Sudo for Windows

05/19/2025

Sudo for Windows is a new way for users to run elevated commands (as an administrator) directly from an unelevated console session on Windows.

Read the announcement, which includes a demo video and deep-dive into how Sudo for Windows works.

Prerequisites

The Sudo for Windows command is available in Windows 11, version 24H2 or higher. (Check for Windows updates).

WHO AND W COMMAND

USING THE WHO COMMAND

- Displays a list of users who are currently logged in:

```
[sysadmin@localhost ~]$ who
root      tty2         2013-10-11 10:00
sysadmin  tty1         2013-10-11 09:58 (:0)
sysadmin  pts/0        2013-10-11 09:59 (:0.0)
sysadmin  pts/1        2013-10-11 10:00 (example.com)
```

Column	Example	Description
username	root	Name of the user who is logged in.
terminal	tty2	This column indicates which terminal window the user is working in.
date	2013-10-11 10:00 (example.com)	This indicates when the user logged in.

USING THE W COMMAND

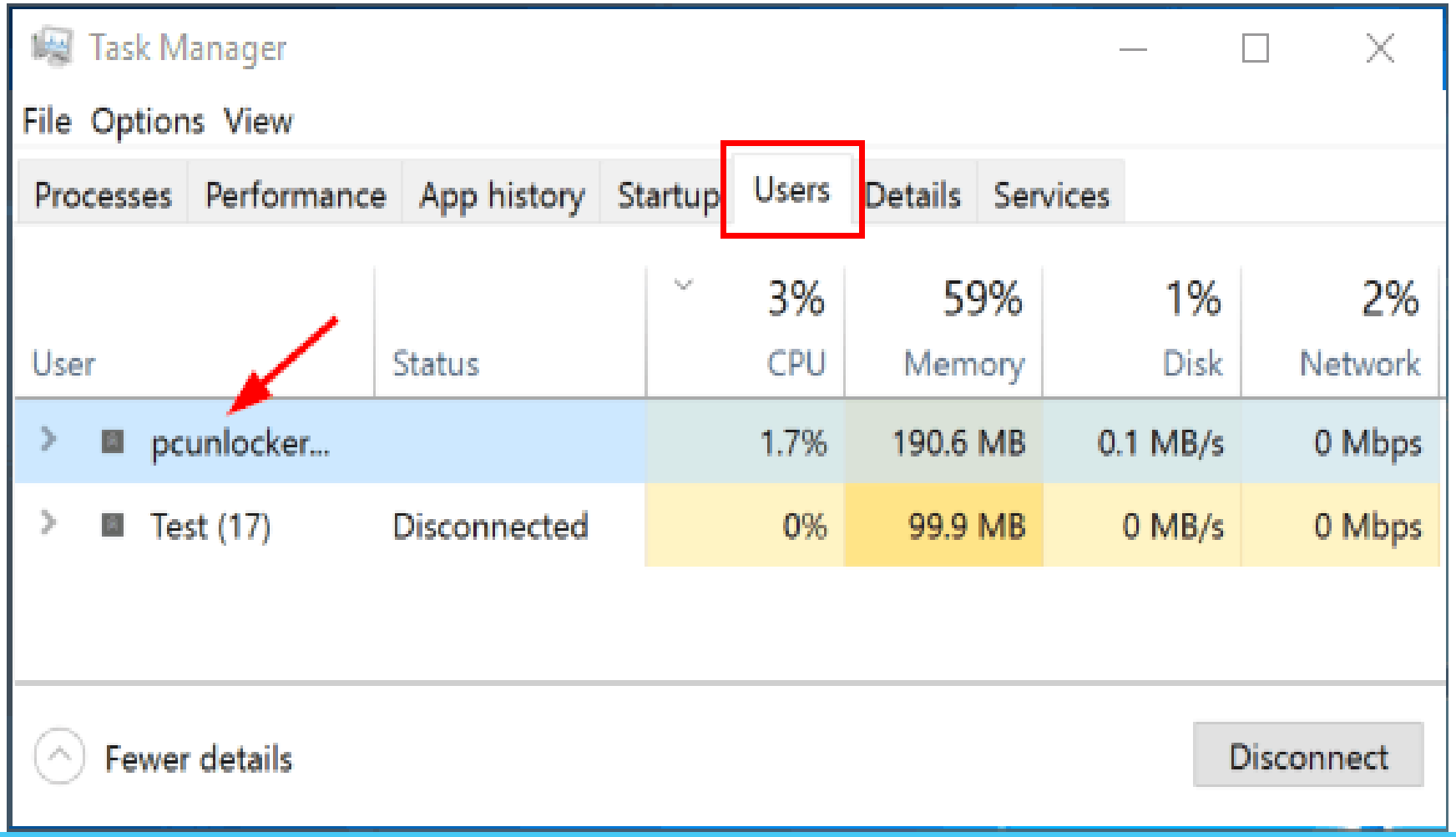
- Displays detailed user and system information:

```
[sysadmin@localhost ~]$ w
 10:44:03 up 50 min,  4 users,  load average: 0.78, 0.44, 0.19
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
root     tty2     -                10:00    43:44  0.01s  0.01s -bash
sysadmin tty1     :0               09:58    50:02  5.68s  0.16s id
sysadmin pts/0    :0.0             09:59    0.00s  0.14s  0.13s who
sysadmin pts/1    example.com      10:00    0.00s  0.03s  0.01s w
```

USING THE W COMMAND

Column	Example	Description
USER	root	This column indicates the name of the user who is logged in.
TTY	tty2	This column indicates which terminal window the user is working in.
FROM	example.com	Where the user logged in from.
LOGIN@	10:00	When the user logged in.
IDLE	43:44	How long the user has been idle since the last command they ran.
JCPU	0.01s	The total cpu time (s=seconds) used by all processes (programs) run since login.
PCPU	0.01s	The total cpu time for the current process.
WHAT	-bash	The current process that the user is running.

USING WINDOWS



OWNERSHIP
FILE MEETS USER
(AND/OR VICE VERSA)

howest.be

VIEWING OWNERSHIP (LS -L)

- To view the ownerships of a regular file, you can use the `ls -l` command:

```
[sysadmin@localhost ~]$ ls -l /etc/named.conf

-rw-r-----. 1 root named 1163 May 13 10:27 /etc/named.conf
```

user owner group owner

- To view the ownerships of a directory file, you can use the `ls -ld` command:

```
[sysadmin@localhost ~]$ ls -ld /etc/named

drwxr-x---. 2 root named 4096 Mar 28  2013 /etc/named
```

user owner group owner

VIEWING OWNERSHIP (STAT)

- Another command that allows you to view ownership information in a more detailed way is the `stat` command:

[sysadmin@localhost ~]$ **stat /etc/named**

File: `/etc/named'

Size: 4096 Blocks: 8 IO Block: 4096 directory

Device: fd00h/64768d Inode: 153995 Links: 2

Access: (0750/drwxr-x---) Uid: (0/ root) Gid: (25/ named)

user owner ↗ group owner ↗

Access: 2013-10-28 16:21:34.949997291 -0700

Modify: 2013-03-28 15:18:54.000000000 -0700

Change: 2013-05-13 09:56:53.831158705 -0700

FILE OWNERSHIP

- Every file is owned by a user and a group.
- If a user creates a file, they will be the user owner of that file.
- The `chown` command can change user ownership of a file, but it can only be used by the root user.
- Although most commands will show the user's account name as the owner, the operating system is actually associating that user's UID as the file owner.

GROUP OWNERSHIP

- When a file is created, the user's primary group is the group owner of the file.
- The user can use the `chgrp` command to change the group owner of a file the user owns, to a group that the user is a member.
- The root user can use the `chgrp` command to change the group owner of any file to any group.
- While most commands will show a group name as the group owner, the system actually tracks group ownership by the GID of the group.

ORPHANED FILES

- If a user is deleted, or has their UID changed, their former UID will show as the owner of their files.
- If a group is deleted, or has its GID changed, the former GID will shown as the group owner of that group's files.

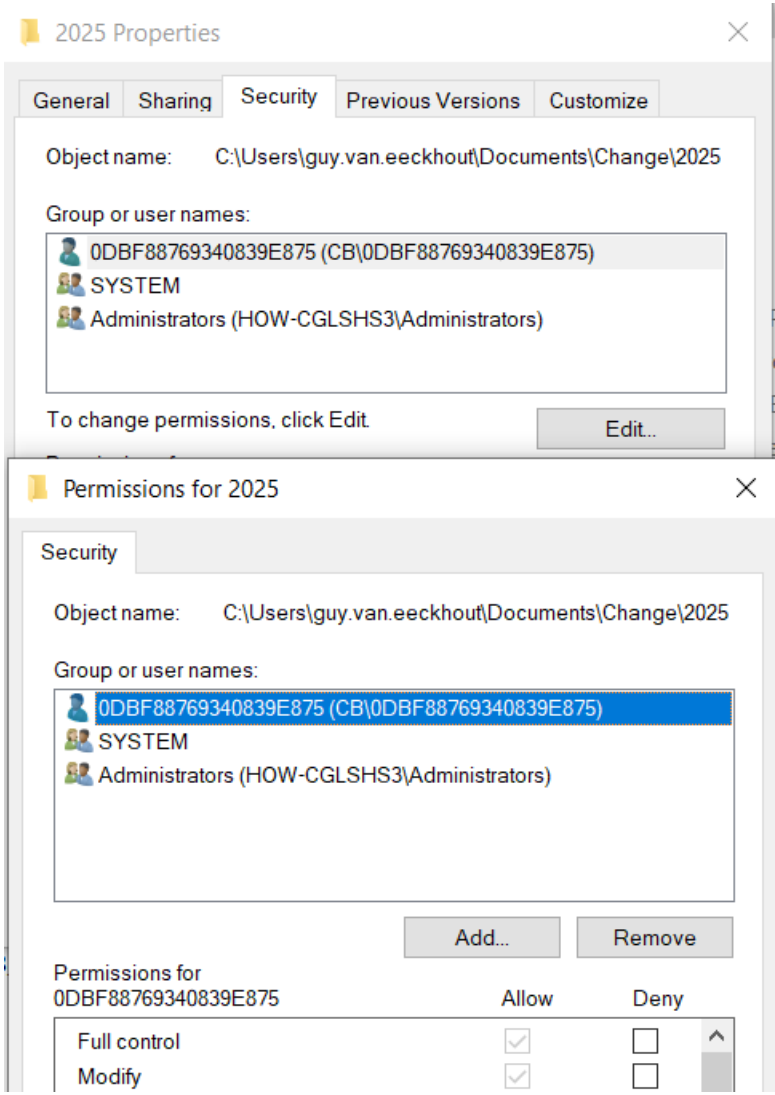
IN WINDOWS

Right click Properties => Security

Edit

A user CAN give his ownership away

You can make your own files unaccessable



IDENTITY INFORMATION

FINDING YOUR IDENTITY

- To see the identity of your current account, and the your group memberships, execute the `id` command:

[sysadmin@localhost ~]$ **id**

uid=500(sysadmin) gid=500(sysadmin) groups=500(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 ***<= SELINUX KERNEL ENABLED***

- Also try the `whoami` command.

VIEWING GROUP MEMBERSHIP

- To list the names of the groups that you have memberships, run the `groups` command:

[sysadmin@localhost ~]$ **groups**

sysadmin research development

- If you are added to a group while logged in, you will have to logout and back in again in order to see your new group membership

CHANGING FILE AND GROUP OWNERSHIP

THE NEWGRP COMMAND

- The `newgrp` command changes your effective primary group by opening a new shell with a different primary group.
- You can join a new group IF YOU KNOW THE GROUP PASSWORD
- Users can use the `newgrp` command to set the primary group to a group they belong *before* they create a file
- The user can return to their original primary group by using the `exit` command
- To permanently change the primary group of the user requires root execute the following command: `usermod -g groupname username`

CHGRP

- A user can change the group that owns the user's files to a group that they belong by using the `chgrp` command.
- The root user can use the `chgrp` command to change the group owner of any file to any group or GID.
- If the `-R` option is used with the `chgrp` command, it will be recursive, acting upon subdirectories and their contents, as well.

CHOWN

- The `chown` command can be used by the root user to change the user owner, the group owner, or both.
- Ordinary users can use `chown` to change the group owner of their files, but since there is `chgrp`, there is no actual need for it.
- Examples:

chown user:group <file|directory>
chown user <file|directory>
chown :group <file|directory>

PERMISSIONS

howest.be

PERMISSIONS

- When you execute the `ls -l` command, the first ten characters of each line are related to file type and permissions:
  - →The first character indicates the file type.
  - →Characters 2-4 are permissions for the user owner.
  - →Characters 5-7 are permissions for the group owner.
  - →Characters 8-10 are permissions for "others" or what is sometimes referred to as the world's permissions. This would be all users who are not the file owner or a member of the file's group.

VIEWING PERMISSIONS

[root@localhost ~]# ls -l /etc/passwd

-rw-r--r--. 1 root root 4135 May 27 21:08 /etc/passwd

- Based on the above command output, the first ten characters could be described by the following table:

File	User Owner			Group Owner			Others		
Type	Read	Write	Execute	Read	Write	Execute	Read	Write	Execute
-	r	w	-	r	w	-	r	-	-

TYPES OF FILES (THE FIRST CHARACTER)

Character	Type of the File
-	A regular file which may be empty, contain text or binary data.
d	A directory file which contains the names of other files and links to them.
l	A symbolic link is a file name that refers (points) to another file.
b	A block file is one that relates to a block hardware device where data is read in blocks of data.
c	A character file is one that relates to a character hardware device where data is read one byte at a time.
p	A pipe file works similar to the pipe symbol, allowing for the output of one process to communicate to another process through the pipe file, where the output of the one process is used as input for the other process.
s	A socket file allows two processes to communicate, where both processes are allowed to either send or receive data.

MEANING OF PERMISSIONS

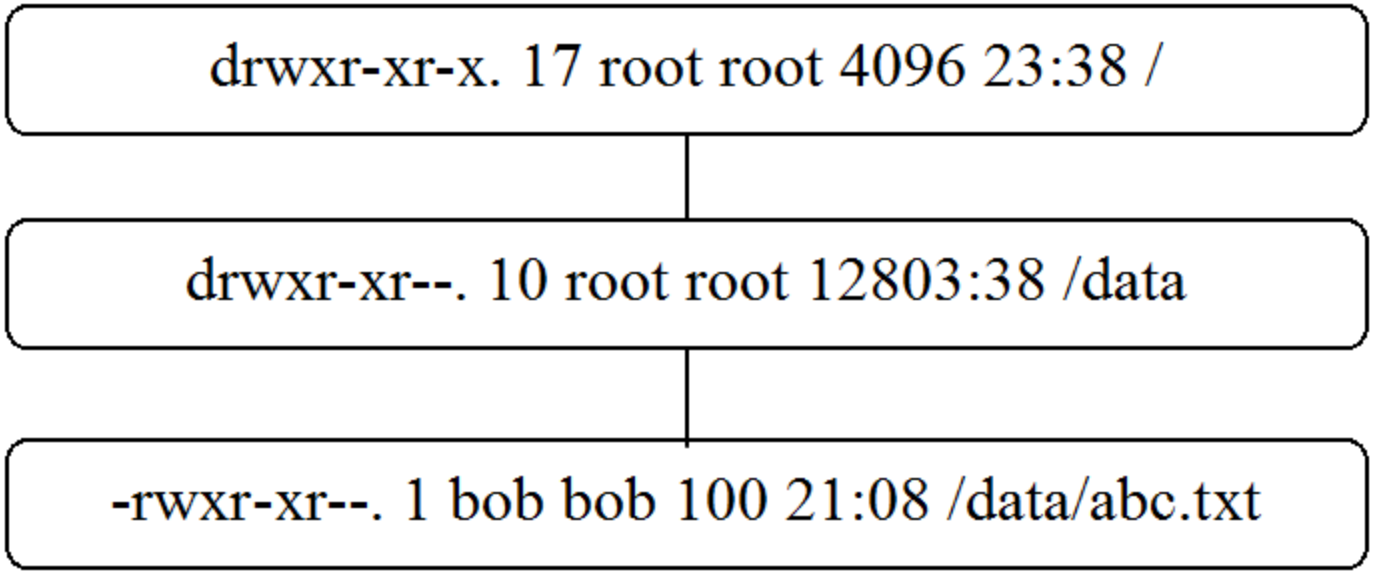
Permission	Meaning on a file	Meaning on a directory
r	The process can read the contents of the file, meaning the contents can be viewed and copied.	File names in directory can be listed, but other details are not be available.
w	The file can be written to by the process, so changes to a file can be saved. Note that **w** permission really requires **r** permission on the file to work correctly.	Files can be added to or removed from the directory. Note that **w** permission requires **x** permission on the directory to work correctly.
x	The file can be executed or run as a process.	The user can use the `cd` command to "get into" the directory and use the directory in a pathname to access files and, potentially, subdirectories under this directory.

UNDERSTANDING PERMISSIONS

- Only one of the three sets of permissions will apply when a user attempts some kind of access on a file:
  - →If you are the user that owns the file, then only the user owner (first 3) permissions apply.
  - →If you are not the user owner, but are a member of the group that owns the file, the group owner (second 3) permissions apply.
  - →If you are not the user owner and you are a not a member of the group that owns the file, then the permissions for the “others” (last 3) will apply.

IMPORTANCE OF DIRECTORY ACCESS

Question: What level of access does bob have to `/data/abc.txt`?



None, because without execute permission on /data there is no way for bob to access the `/data/abc.txt` file.

CHMOD COMMAND

CHMOD

- The `chmod` (change mode) command is used to set or modify permissions.
- To change permissions on a file, you must either be the user owner or root.
- There are two distinct techniques for changing permissions with `chmod`:
  - → symbolic
  - → numeric

USING CHMOD SYMBOLICALLY

- With this technique, you specify *who*, an *operator*, and *what*:

who: specifies whose permissions to alter:

- **u** for user
- **g** for group
- **o** for others
- **a** for everyone

operator: specifies whether to add, remove or assign:

- **+** to add
- **-** to remove
- **=** to set exactly

what: specifies the permission to set on the file:

- **r** for read
- **w** for write
- **x** for execute
- **-** for nothing

CHMOD SYMBOLIC (ALTER) EXAMPLES

- `chmod u+x abc.txt`
  → will alter the execute permission for the user owner.

- `chmod go-rx abc.txt`
  → will alter/remove read and execute for the group owner and others owner.

- `chmod u+wx,g=rx,o-r abc.txt`
  → will alter the write and execute permissions for the user owner (no change to read), will **set** r-x for group owner and alters/removes read permission for “others”.

USING CHMOD (SET) NUMERICALLY

- When using the numeric technique with `chmod`, a three digit number is used to represent the permissions of the user, group and others.
- It is also called the octal method after the octal values that are used to calculate the permissions:
  - →4 = read
  - →2 = write
  - →1 = execute

USING CHMOD NUMERICALLY

- By combining the permissions the values range from 0 to 7:
  - → 7 = rwx
  - → 6 = rw-
  - → 5 = r-x
  - → 4 = r--
  - → 3 = -wx
  - → 2 = -w-
  - → 1 = --x
  - → 0 = ---

- All nine permissions must be specified when using the octal method:
- 777 = rwxrwxrwx
- 775 = rwxrwxr-x
- 755 = rwxr-xr-x
- 700 = rwx------
- 664 = rw-rw-r--
- 640 = rw-r-----

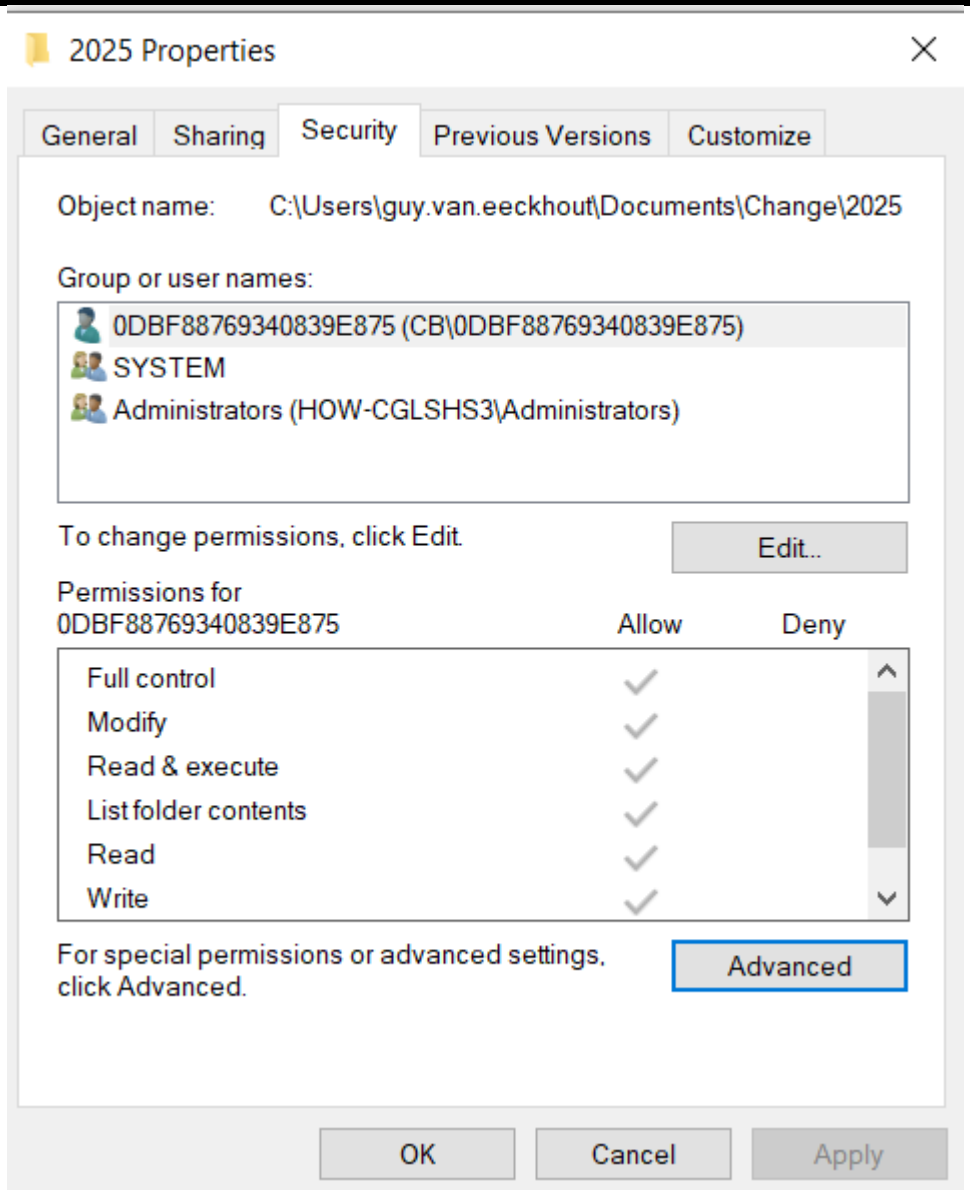
CHMOD NUMERIC EXAMPLES

- `chmod 755 abc.sh`
  → rwxr-xr-x
- `chmod 660 abc.txt`
  → rw-rw----
- `chmod 771 somedir`
  → for rwxrwx--x
- `chmod 400 my.txt`
  → r--------
- `chmod 700 userdir`
  → rwx------

IN WINDOWS

Default permissions using EDIT

Read,
Write,
Modify,
List Folder Contents,
Full Control,
Read & Execute



EXTENDING PERMISSIONS : ACCESS CONTROL LIST (ACL)

WHY ACL ?

- Because o(thers) = “the rest of the world” can be a big place
- E.g. a process like a web server can be granted access to files that reside in a user's home directory, without compromising security by giving the whole world access

ENABLING ACL

- To enable ACL, the filesystem must be mounted with the acl option.

```
# tune2fs -l /dev/sdXY | grep "Default mount options:"
Default mount options:    user_xattr acl
```

SETTING ACL

- The ACL can be modified using setfacl

```
# setfacl -m "u:user:permissions" <file/dir>
```

```
# setfacl -m "g:group:permissions" <file/dir>
```

You will notice that there is an ACL if there is a plus sign after its Unix permissions in the output of ls -l.

```
$ ls -l /dev/audio
```

```
crw-rw----+ 1 root audio 14, 4 nov.   9 12:49 /dev/audio
```

GRANTING PERMISSIONS FOR PRIVATE FILES TO A WEB SERVER

- In the following we assume that the web server runs as the user webserver and grant it access to geoffrey's home directory /home/geoffrey.
- The first step is granting execution permission to webserver so it can access geoffrey's home:

```
# setfacl -m "u:webserver:--x" /home/geoffrey
```

GRANTING PERMISSIONS FOR PRIVATE FILES TO A WEB SERVER

- Since webserver is now able to access files in /home/geoffrey, other no longer needs access, so it can be safely removed:

```
# chmod o-rx /home/geoffrey
```

- getfacl can be used to verify the changes:

```
$ getfacl /home/geoffrey
getfacl: Removing leading '/' from absolute path names
# file: home/geoffrey
# owner: geoffrey
# group: geoffrey
user::rwx
user:webserver:--x
group::r-x
mask::r-x
other::---
```

IN WINDOWS

Default permissions using EDIT

Read,
Write,
Modify,
List Folder Contents,
Full Control,
Read & Execute

Advanced Permissions : ACL !

More granular approach possible

Allow / Deny
Stacks on top of each other
Can get complicated