Linux System Administration

PM-TRTNB319

Zsolt Schäffer, PTE-MIK

Legal note

Legal note

Mid-term overview

Linux basics

Lecture 8

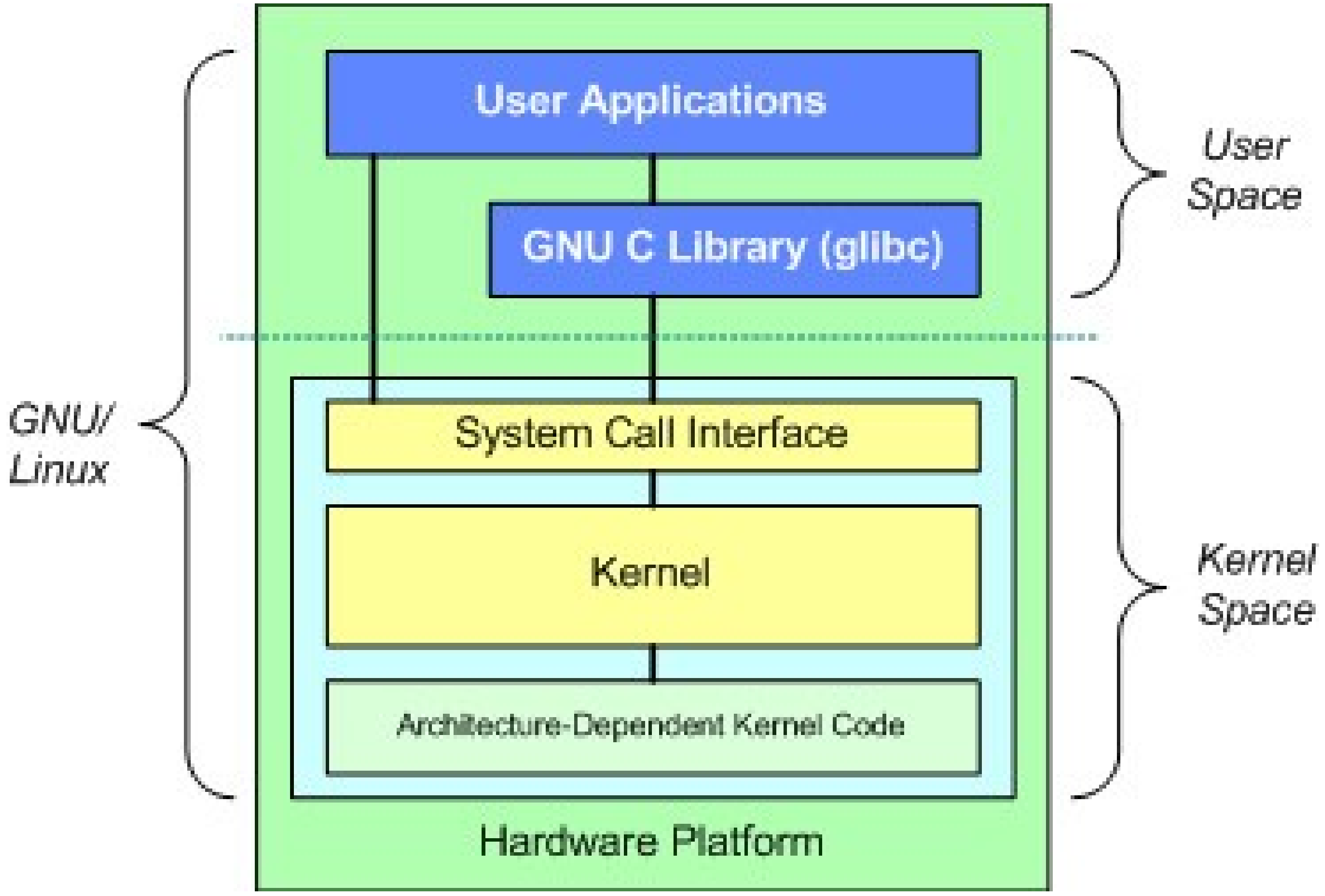
Chapter 1

Linux basic concepts

kernel space

- The kernel itself runs in a **privileged** hardware ring (ringo) on the Intel x86 CPUs. This privilege level will allow access for everything in the computer (all addresses and buses).
- User space processes are separated to their own virtual memory space, they are supervised by the kernel and managed by the kernel's **scheduler**.
- When a process requires a privileged operation (acces-sing an I/O address, reading data from the disk, or create other processes for e.g.), it will ask the kernel to do it. This is called a **system call**. The standardized interface to system calls is provided through the **std. C library** (libc).
- The **device drivers** provide an interface to processes under the /sys virtual filesystem and the device node files under the /dev directory. Device drivers run within the kernel's privileged hardware ring, they are implemented as kernel objects (.ko).

Fundamental architecture of Linux



https://www.ibm.com/developerworks/library/l-linux-kernel/figure2.jpg

Linux basic concepts

terminals

- All interactive user sessions work through a **terminal**: a special character device, which moves in-/output bytes to/from the user. The nature of the terminal is not relevant to process running inside it. For e.g. a terminal can be a direct hardware device file provided by the kernel, or can be a pseudo terminal, which is connected to a software layer. (E.g. a graphical terminal window, or a terminal device that is remotely connected to an other computer via a software TCP/IP tunnel.)
- Terminals usually have a **shell** process (command interpreter and run-time environment) running inside them. A shell runs further processes and handles jobs.
- All processes have 3 **standard streams**, which are by default connected to the terminal that holds the parent process.

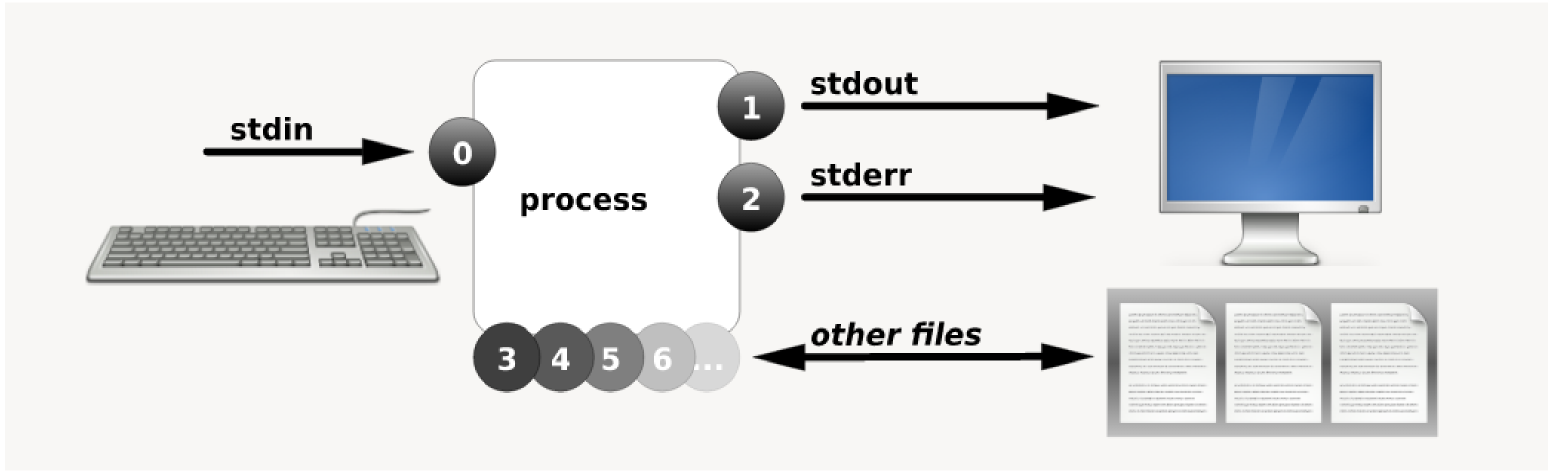
Linux basic concepts

standard streams

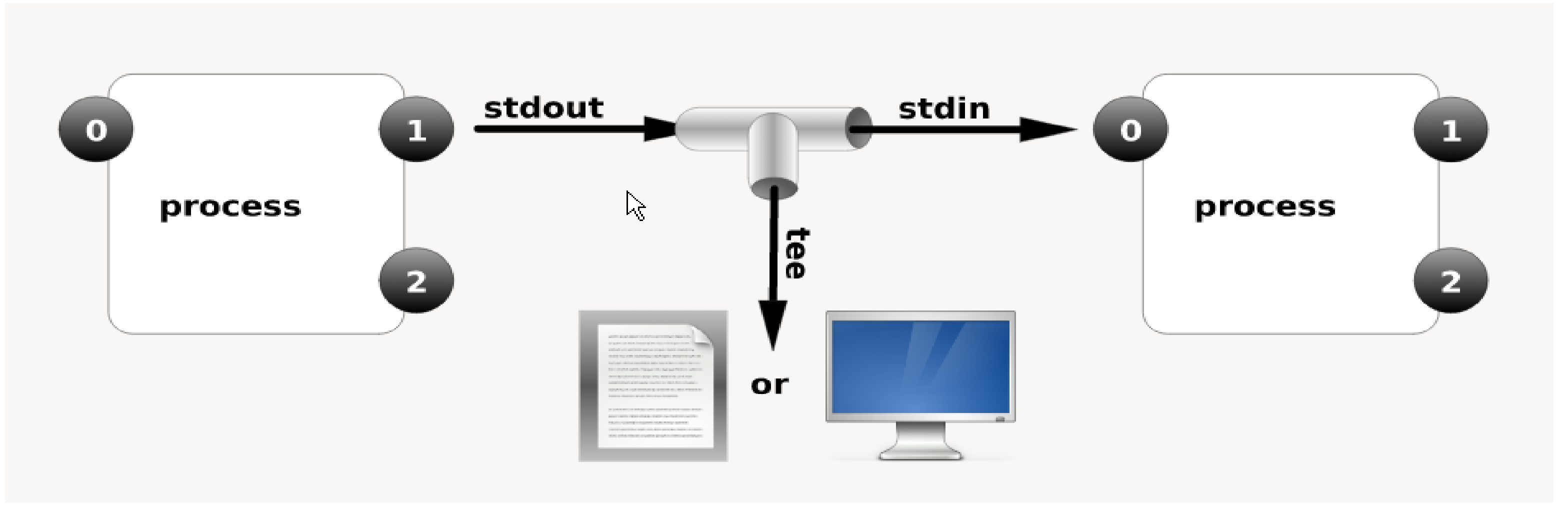
help

- The standard stream specifier (file descriptor) is usually inherited from the parent process, but can be over-ridden with redirects and pipes. Standard streams are represented as files to the processes. (fds 0,1 and 2)
- Daemons are a special type of userspace processes. They have their standard streams redirected in a way, that they don’t communicate with the user directly through a terminal, they read config files and write logs instead. The usual server processes, which provide services to other processes or other computers on the network are implemented as **daemons** (e.g. httpd, dhcpd, mysqld, ftpd, …).
- Use the **man** command to look up the manual pages for a command or a config file. Use **man -k** ***keyword*** to search the man pages for a keyword. Also consider **info** and **apropos**.

Redirecting standard streams



- You can create a named pipes with **mkfifo**.



Redirecting standard streams

- Examples on how to use redirects. Refer to **RH124 CH4.1.** for more.
  - some_program >some_file – redirect the stdout stream
  - some_program 2>some_file – redirect the std error stream
  - some_program 1>some_file – redirect the std out stream
  - some_program >file 2>&1 – redirect stderr to stdout and stdout (both streams) to file. Order is important!
  - >>file means append to file instead of overwriting from start
  - date -s < ~/date.sav – redirect standard input (sets the date to the string specified in date.sav file instead of requesting to type it from keyboard.)
  - cat <<EOF > ~/some_srcipt.sh – read multi-line text from input. Input can be terminated by typing EOF. Then save the output in some_sript.sh. This is called a 'here document'. (shell as text editor)

Pipes

- Processes can be chained using pipes infinitely.
- A pipe can be forked with the **tee** command, which works like a T junction. It forward its stdin to its stdout but also forwards a copy of the stream to the filename given as argument.
- The Linux concept of having a simple, specific, efficient utility (building block) for every task, combined with pipes is very powerful. Some examples:
  - ls -t | head -n 10 | tee ~/ten-last-changed-files | mail student@desktop1.example.com
  - cut -d: -f7 /etc/passwd | uniq | sort
  - dd if=/dev/sda6 | pbzip2 -c -9 | nc hostB -p 2223

    netcat -p 2223 -l | pbzip2 -d | dd of=/dev/sda6

Linux basics

filesystem objects

- In linux everything is a file. A full list of file types with notions (ls command style), and the related command to create the object follows:

-	regular file	touch
d	directory	mkdir
c	character device node	mknod
b	block device node	mknod
p	named pipe	mkfifo
s	socket	socket() syscall
l	symbolic link	ln -s
-	hard link	ln
-	virtual files (/proc, /sys)	(e.g. mount -t proc)

Linux basics

file manipulation

- Use **mkdir** to create an empty directory, use **touch** to create an empty file.
- Use **mv** for moving or renaming a file, use **cp** for copying.
- Use **pwd** to display current working directory, use **cd** to change current working directory.
- Use **ls** to list elements in a specific directory or in the current working directory.
- Use **rm** to remove a filesystem object.
- Use the **file** command to get information about a file’s format and/or header.
- Use **ln** to create a hard link or **ln -s** to create a symbolic link to an existing filesystem object.

Linux basics

file manipulation

- Complete lab practice in **RH124 CH 2.4.**
- Now do the same the smart way!

Shell command line

```
# mkdir Music Videos Pictures
# touch {Music/song{1..6}.mp3,Pictures/snap{1..6}.jpg,Videos/film{1..6}.avi}
# mkdir friends family work
# cp **/*[12].{mp3,jpg,avi} friends
# cp **/*[34].{mp3,jpg,avi} family
# cp **/*[56].{mp3,jpg,avi} work
```

- More on filename globbing in **RH124 CH 2.5.** and lab practice **RH124 CH 2.6.**

Linux basics

file manipulation

- More on find in **RH124 CH4.1.** Also consider the following examples:
- To find all socket type files under /var directory with it's name containing sql and not owned by adam, type the following.

Shell command line

```
# find /var -type s -name "*sql*" ! -user adam
```

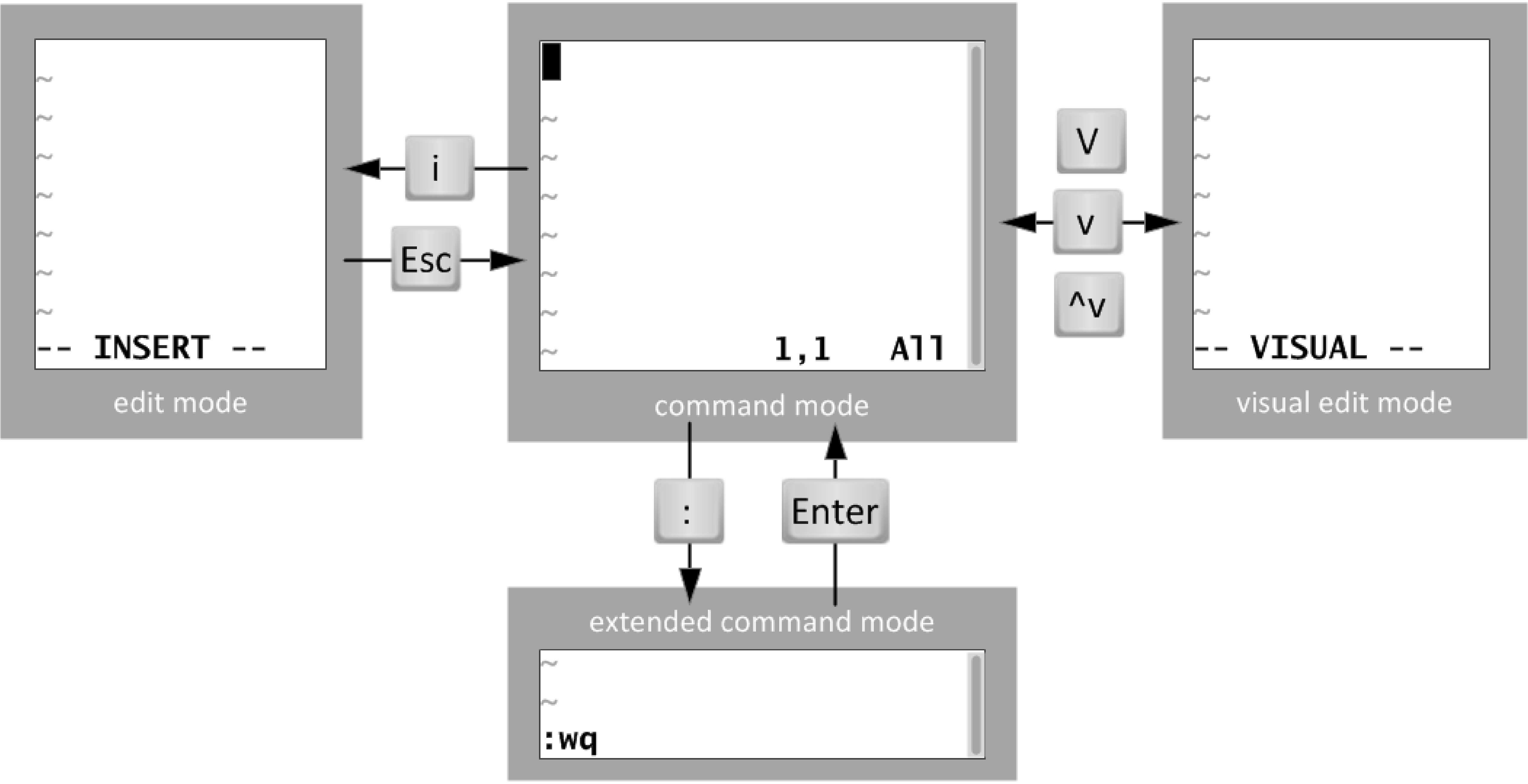
- To find and delete all large and old files owned by bob in external drive (with size larger than 100MB and accessed more then a month ago) type the following.

Shell command line

```
# find /mnt/external -type f -user bob -size
+100M -atime +31 -exec rm {} ;
```

Most important vi commands

- In command mode:
  - press i for insert mode.
  - press a for append mode
  - type dd to delete (cut) whole line
  - type yy to copy whole line
  - type p for paste after cursor, P for pasting before cursor
- Advanced commands:
  - :w to save file
  - :q to quit
  - :q! to quit without saving file
  - :wq to save and quit


-- INSERT --
edit mode
i
Esc
1,1 All
command mode
V
v
^v
-- VISUAL --
visual edit mode
:
Enter
extended command mode
:wq

Linux basics

file manipulation

- Use **find** to locate files in the filesystem directory structure (tree).
- Use **cat**, **less** or **more** to display the contents of files.
- Use **vi** for editing text files.
- Consider Midnight Commander (**mc**) it is a character-graphics dual panel file manager utility.
- Complete lab practice **RH124 CH 4.6.** (visual modes)
- Complete lab practice **RH134 CH 3.3.** (vi basics)
- Complete lab practice **RH134 CH 3.5.** (vi editing)
- Complete lab practice **RH134 CH 3.6.** (vi editing)
- Complete lab practice **RH124 CH 14.5.** (link files)
- Complete lab practice **RH124 CH 14.7.** (find, locate)

Mid-term overview

Users, processes, permissions, security

Lecture 8

Chapter 2

Users and groups

- All system activity in Linux is tied to a user and a group. All processes and all files have a user owner and a group owner. (UID, GID, EUID, RUID, EGID, RGID)
- Locally defined user accounts are stored on disk in the **/etc/passwd** file (public part of account information) and the **/etc/shadow** (secret part, e.g. password hashes, password aging, account expiry) file.
- Local group definitions are set in the **/etc/group** and **/etc/gshadow** files.
- Use **id**, **whoami** to print information about the current terminals (logged-in) user.
- Use **who**, **last** and **lastlog**, to print information about running user sessions and latest login events.

Managing users

- You can manage before-mentioned text files manually. Or you can use the following command line tools. This conveniently also creates a home directory and sets its permissions, besides adding a new user entry.
- The **useradd [switches] username** command can create a user, note the following switches and default values
  - -c to set comment (Gecos) field (empty)
  - -s to specify shell (/bin/sh)
  - -d to specify home directory (/home/[username])
  - -g to specify primary group (UPG)
  - -G to specify list of supplementary groups (none)
  - -u to specify UID (add one to the last used UID)

Shell command line

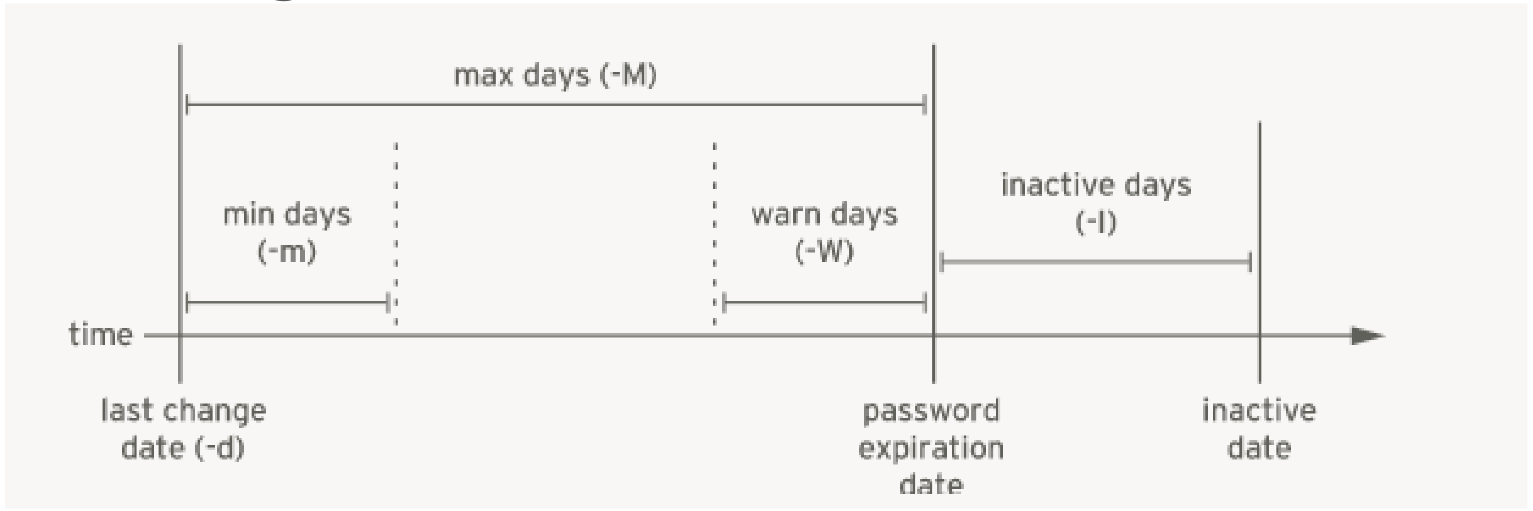
```
# useradd -c "John Smith" -u 1050 -G video jsmith
```

Managing users

- You can modify an existing user with **usermod [switches] username** using similar switches. Also note the -L and -U switches to lock and unlock the password.
- The **userdel username** command removes the user. Using the -r switch also removes the users home directory from the filesystem.
- Note the seemingly conflicted UIDs in **RH124 CH 5.4.** when deleting and adding users.
- You can change your own password with **passwd** command. Only root can change passwords for other users. Root can do it like this: **passwd alice**
- Complete lab practice under **RH124 CH 5.5.**

Managing password aging

- You can adjust the timing fields of the /etc/shadow file with **chage**.



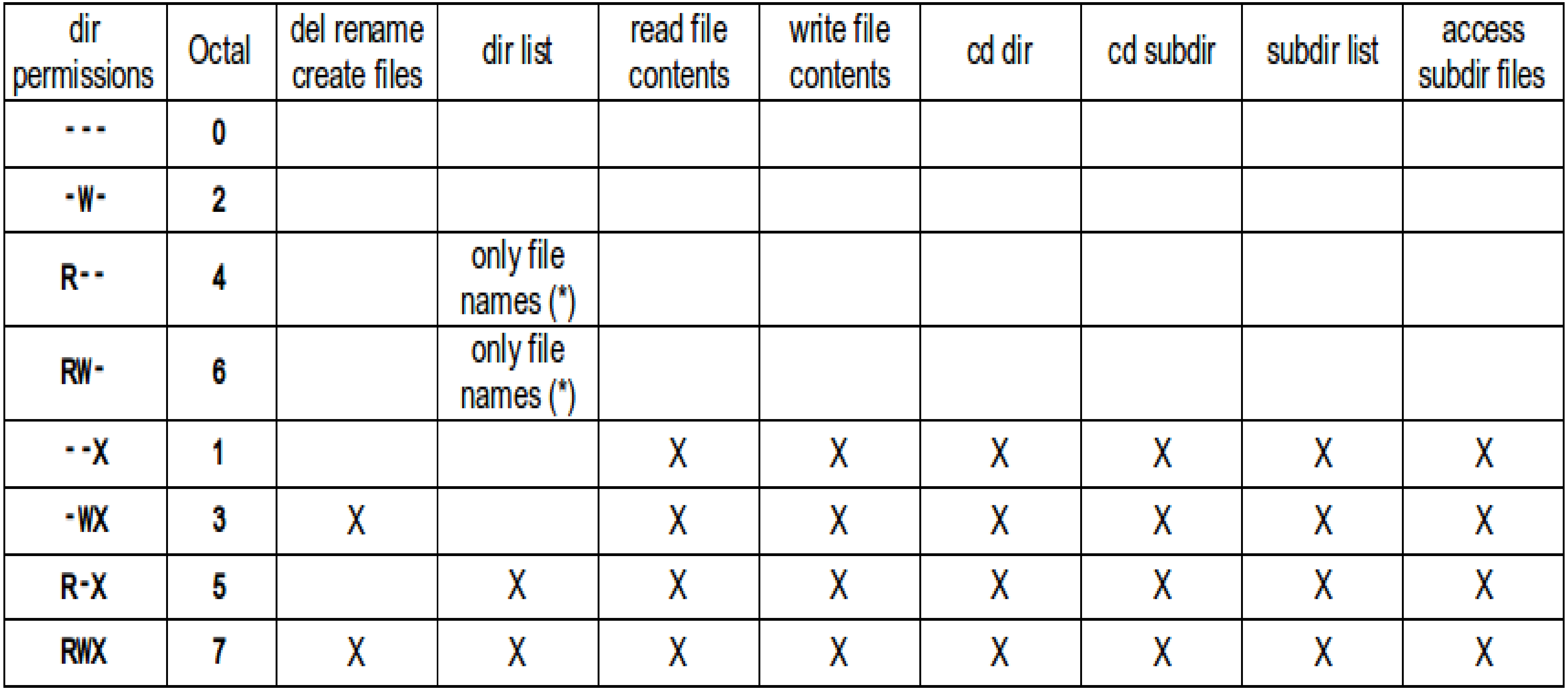
- Use chage -E to set absolute account expiry.
- You can also use the usermod -e command to set expiry.
- Also note the -l, -d options to chage.
- Refer to **RH124 CH5.8.-5.9.**for further information.
- Complete practice **RH124 CH5.10.**

Managing groups

- You can add a new group with **groupadd** command.
- You can modify a group with **groupmod**.
- You can remove a group with **groupdel**.
- You can modify group membership with the usermod command as described earlier. Note the -g -G and -a command switches.
- Refer to **RH124 CH 5.6.** for further
- Complete practice **RH124 CH 5.5.** (create users)
- Complete practice **RH124 CH 5.7.** (group membership)
- Complete practice **RH124 CH 5.9.** (chage -E)
- Complete practice **RH124 CH5.10.** (login.defs, chage)

File permissions

- Each file has a user owner and a group owner. A permission triplet (r,w,x) is stored for every file for the user owner, for the group owner, and a triplet for everyone else (other). There is also a special permission triplet (sticky, setuid, setgid).
- Note the significance of x permission on directories (see table on next slide)
- The owner information of a process (source) that accesses a file (target) is compared by the kernel, it will decide which triplet to apply.
- The ACL scheme allows for additional rules besides the 3 basic triplets. Most importantly ACLs allow for named rules, that target individual users and groups. Each rule in ACL specifies a permission triplet to apply to a user or a set of users/group.

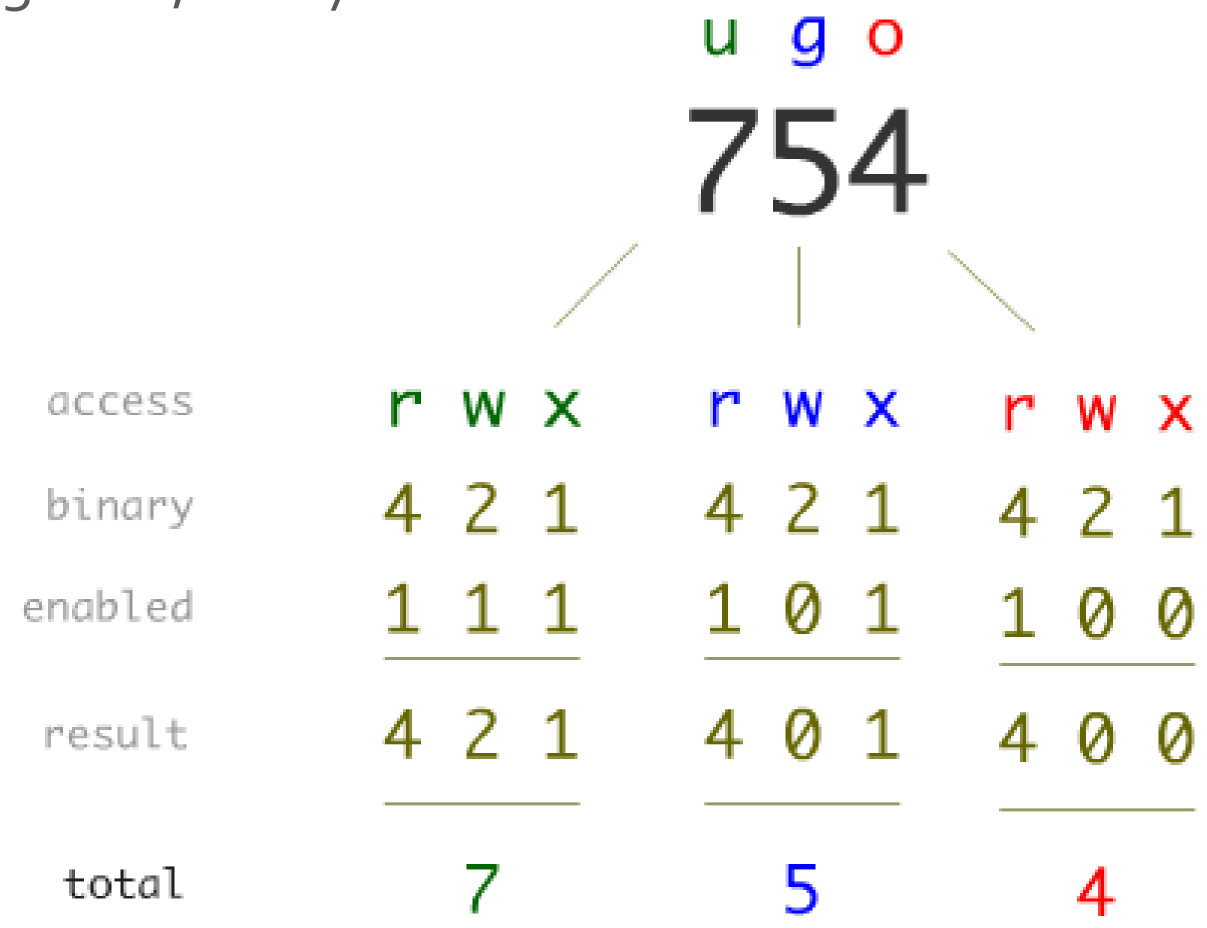


dir permissions	Octal	del rename create files	dir list	read file contents	write file contents	cd dir	cd subdir	subdir list	access subdir files
- - -	0								
-W-	2								
R- -	4		only file names (*)						
RW-	6		only file names (*)						
- -X	1			X	X	X	X	X	X
-WX	3	X		X	X	X	X	X	X
R-X	5		X	X	X	X	X	X	X
RWX	7	X	X	X	X	X	X	X	X

https://i.stack.imgur.com/mZ6qv.png

Octal (numerical) representation

- The octal representation of permissions is also commonly used. Values for special bits are: setuid – 4, setgid – 2, sticky – 1



	u			g			o		
access	r	w	x	r	w	x	r	w	x
binary	4	2	1	4	2	1	4	2	1
enabled	1	1	1	1	0	1	1	0	0
result	4	2	1	4	0	1	4	0	0
total		7			5			4	

https://devopscube.com/wp-content/uploads/2016/05/permissions.png

File permissions

- Use **chmod** to change a files permissions, use **chown** to change a files owner (only root can do that).
- Use the **ls -l** switch to include permissions in the listing.
- The umask is a negative file mode applied to all newly created filesystem objects, use the **umask** command to set the value of the mask.
- Use **getfacl** to print the ACL ruleset for a file. Use **setfacl** to set or modify the ACL list belonging to a file or directory.
- The ACL mask is applied to all named entries and the group owner, before evaluating the triplet.
- Default entries only apply to directories, they will be automatically and recursively inherited to new sub-elements of the directory accordingly.

Reading ACLs

Shell command line

```
# getfacl some_directory
# file: some_directory
# owner: student
# group: controller
# flags: -s-
user::rwx
user:james:---
user:1005:rwx           #effective:rw-
group::rwx              #effective:rw-
group:sodor:r--
group:2210:rwx          #effective:rw-
mask::rw-
other::---
default:user::rwx
default:user:james:---
default:group::rwx
default:group:sodor:r-x
default:mask::rwx
default:other::---
```

Setting ACLs

- An ACL can be set/replaced with the **setfacl -s** command. Use the -m switch to modify an ACL entry, use -x to remove an entry. Use the -d switch together with the former two to modify/remove a default entry.
- The format starts with u: g: o: or m: for user, group, other, and mask entries respectively.
- The next element is an object specifier (user/group name/id). **Mask** and **other** entries have blank specifiers.
- An emtpy user or group specifier corresponds to the user/group who owns the file.
- The last element is the permission set itself. e.g rx

Shell command line

```
# setfacl -m u::rwx,g:sodor:rX,o::- filename
# setfacl -x -d u:james filename
```

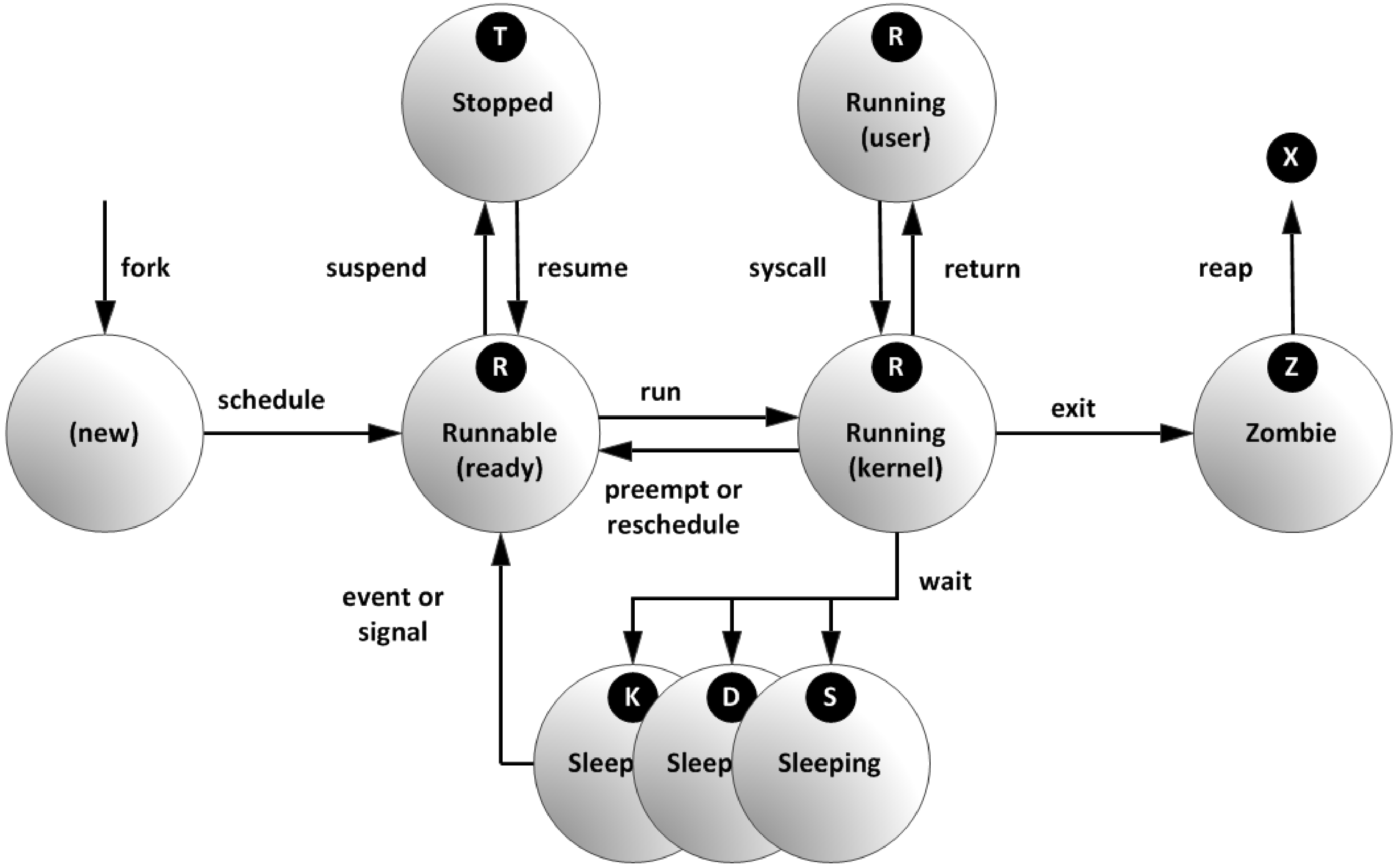
Setting ACLs

- You can use the -R switch to apply ACLs to directories and sub-entries recursively. A permission set containing a X means to apply x permission only to directories, not files.
- You can remove all default ACL entries of a directory with setfacl using the -k switch. The -b switch removes ACLs completely.
- Modifying the ACLs with **setfacl** can cause the mask to be recalculated. To inhibit mask change use the -n switch to setfacl.
- Complete practices **RH124 6.5.** (setgid, umask), **RH124 6.3.** and **6.6.** (chmod, chown)
- Complete practices **RH134 CH6.3** (ACLs Thomas), **RH134 6.4.** (ACLs Holmes)

Processes

- The processes in every operating system are handled by the scheduler. In Linux you can influence how the scheduler prioritizes each processes with the **nice** and **renice** commands. Processes can be listed with **ps, pstree** commands and be managed with the **top** utility.
- The processes can be targeted with signals by the scheduler. Use **kill**, **killall** and **pkill** commands to send signals. Some signals can be handled by the target process. Some signals originate from hardware events, some can be triggered by the user.
- The system load and uptime can be queried with **top** and files in the procfs (**cat /proc/loadavg, cat /proc/uptime**).
- Complete practice **RH124 CH 7.7.** (top), **CH 7.8.** (top, kill) and **RH134 CH 5.3-5.4.** (nice).

Process states



Processes

- Processes are not just created in Linux, a process can only be created by another process by cloning, then forking itself. This enforces the inheritance of security descriptors and privilege levels of child processes. A process owned by root can decrease its privilege level (set EUID, EGID) voluntarily.
- The only way to escalate privileges is through the **setuid** file permission mechanism. Several binaries have this bit set. Two of them (**su** and **sudo**) serve the purpose of running specific commands or whole user sessions as a different effective user (or group).
- Sudo allows for more refined control, the **source** (current) **user's** password may be required. To use su, you need the **target user's** password.
- Complete practice **RH124 CH 5.3.** (su)

Jobs

- Processes and schedulers are operating system concepts. On the other hand jobs are shell related entities. Every pipeline of commands entered in the shell is a job. The processes in a pipe chain are the members of the same job.
- A shell can handle multiple jobs, but only one job can be in foreground. The fg. job is the only one that can get characters and control signals from the terminal input.
- Background jobs can be either be in executing state or can be suspended and resumed later. To start a job right in the background add an **&** (ampersand) to the end of the command. To suspend and send an already executing fg. command to background press **ctrl+z**.

Shell command line

```
# example_command | sort | mail -s "Sort output" &
```

Jobs

- Jobs sent to background with ctrl+z are also immediately paused (suspended).
- To list jobs, use the **jobs** command.
- To bring a job back to foreground use **fg %jobnum**. This will also make it resume.
- To resume a suspended job while still keeping it in the background use **bg %jobnum**.
- Use **ctrl+c** to ask a process to terminate itself cleanly.
- Refer to **RH124 chapter 7.2.** for more.
- Complete practice **RH124 CH 7.3.** (jobs)
- Complete practice **RH124 CH 7.5.** (jobs, pkill)

SELinux

- SELinux is an implementation for the Mandatory Access Control scheme for Linux based on NSA projects.
- In our case the targeted policy is the most important one. It builds mainly on the "type" element of the SELinux context and has rules like the following: “allow httpd_t type processes to open httpd_log type files for append operation.”
  The policy is compiled into a binary (AVC). It is usually not required to modify the policy, it is handled by RH developers and put in the package repository. Though you can refine the effects of the ruleset by managing SELinux booleans with **setsebool** and **getsebool** tools.
- System boot-up defaults for SELinux mode and policy are stored in **/etc/selinux/config**, you can switch between permissive and enforcing mode at runtime with the **setenforce** and **getenforce** commands.

Get SELinux information

- You can use the **sesearch -A | grep [keyword]** command to search the policy for expressions, types, rules. Use the -C -b switch in addition to restrict your search to SELinux booleans and print conditions.
- Try the system-config-selinux graphical tool for managing SELinux.
- Use the -Z switch in general to get SELinux context information. E.g.: **ls -Z**, **ps -Z**, **id -Z**
- You can use **sesearch -T** to look for transitional rules.
- Use the **runcon** command to run a command with specific SELinux context. This command, like others also falls under the rules of transition.
- Use the **man -k selinux_keyword** to search for man pages containing the keyword. E.g. a type, a boolean, ...

Manage SELinux context

- To explicitly set a file’s context use the **chcon** command. Use the -t switch to only alter the type attribute. Use the **restorecon** command to reset a file’s context to the database defined, location dependent, default value. The -R switch stands for recursive, -v stands for verbose.

Shell command line

```
# chcon -t httpd_sys_content_t /var/www/html/app1
# restorecon -Rv /var/www/html
restorecon reset /var/www/html/file1 context
unconfined_u:object_r:user_tmp_t:s0
       -> system_u:object_r:httpd_sys_content_t:s0
# semanage fcontext -a -t httpd_sys_cotent_t '/srv(/.*)?'
# restorecon -RFvv /srv
```

- To change the database of default file-contexts use the **semanage fcontext** command. Use the -l switch to list the current database. Use -d to delete a line, use -a to add a line. Note that the order of evaluation is the same as the order of entry (overlapping rules for subdirs).

Troubleshoot SELinux

- The most common reason for AVC denials is an incorrectly set file context. (E.g. file moved in a new location) → use **restorecon**. This has the most narrow impact on system security, it only affects the given file.
- Use **sealert -l [uuid]** command to get more details on a specific AVC denial. Its ID will be in the system log.
- Complete practice **RH134 CH.7.3.** (setenforce)
- Complete practice **RH134 CH.7.5.** (semanage fcontext)
- Complete practice **RH134 CH.7.7.** (SE booleans)
- Complete practice **RH134 CH.7.9-7.10 .** (sealert, restorecon)

Mid-term overview

Storage

Lecture 8

Chapter 3

Block devices, partitions, filesystems

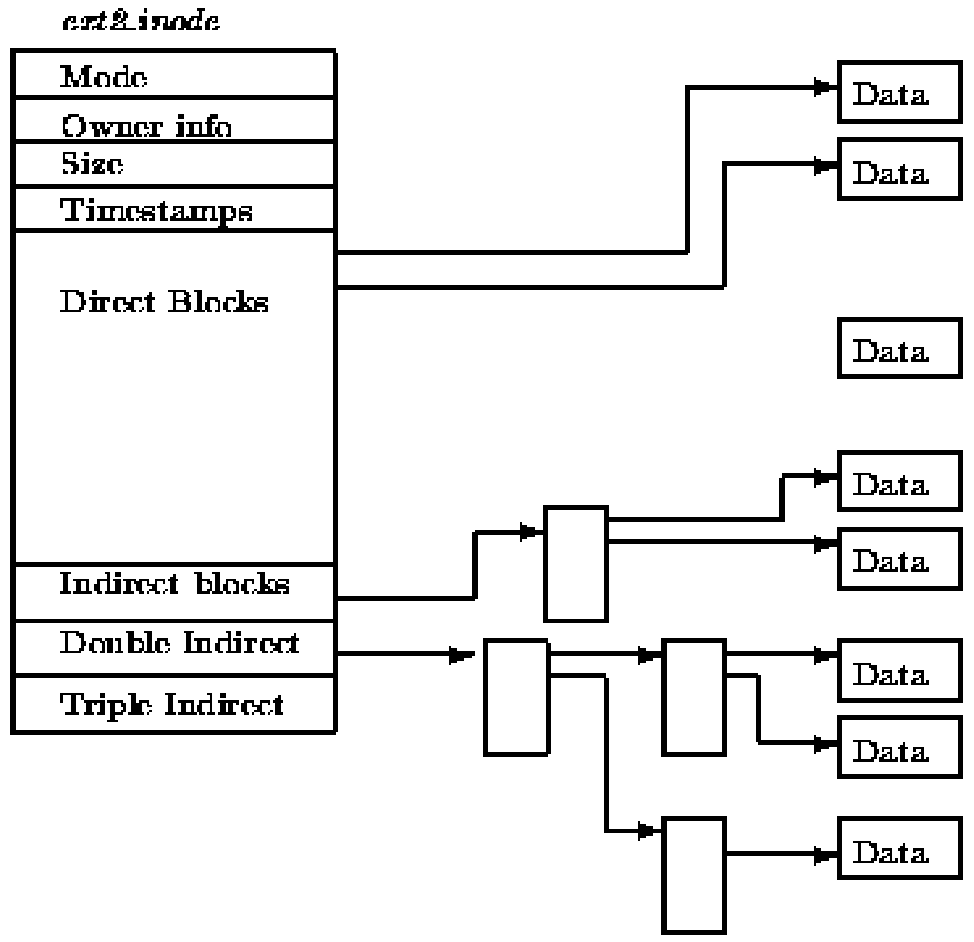
- Block devices are also called raw storage as they only provide a range of hardware addressable storage compartments, no structure or hierarchy is available.
- By partitioning, a block device can be divided into logically separated segments.
- LVM, besides simple logical segmentation, also provides additional features, like space aggregation, live resizing, caching, thin volumes, snapshots, etc… LVM relies on the device-mapper framework, LVs are still raw block devices. Block device can be stacked on each other with dm (for e.g. crypto, cache, other logical block devices.)
- A filesystem standard describes the disk storage format and a set of management algorithms. An FS provides a clear tree structure for organizing data. It may also provide advanced features, like subvolume snapshots.

Block devices, partitions, filesystems

- LVM has 4 major component layers: **physical volumes**, which add up the the extents of **volume groups**, and **logical volumes**, that are carved out of the free extents of a VG. The 4th (optional) layer is for LVM **pools**.
- Filesystems are created on top of block devices (for e.g. partitions, logical volumes or dm-crypt devices).
- **Swap** space can be used to temporarily store infrequently used memory pages on disk. The kernel swaps pages in- and out of the disk to/from memory as needed. Swap is not considered a filesystem.
- The procedure of merging a filesystem into a directory of the existing directory tree is called **mount**ing.
- Unix filesystems have **symbolic link**s (textual path refe-rences, can point outside FS) and **hard link**s (numerical reference to structure/element identifiers within a FS).

The ext2 filesystem storage format

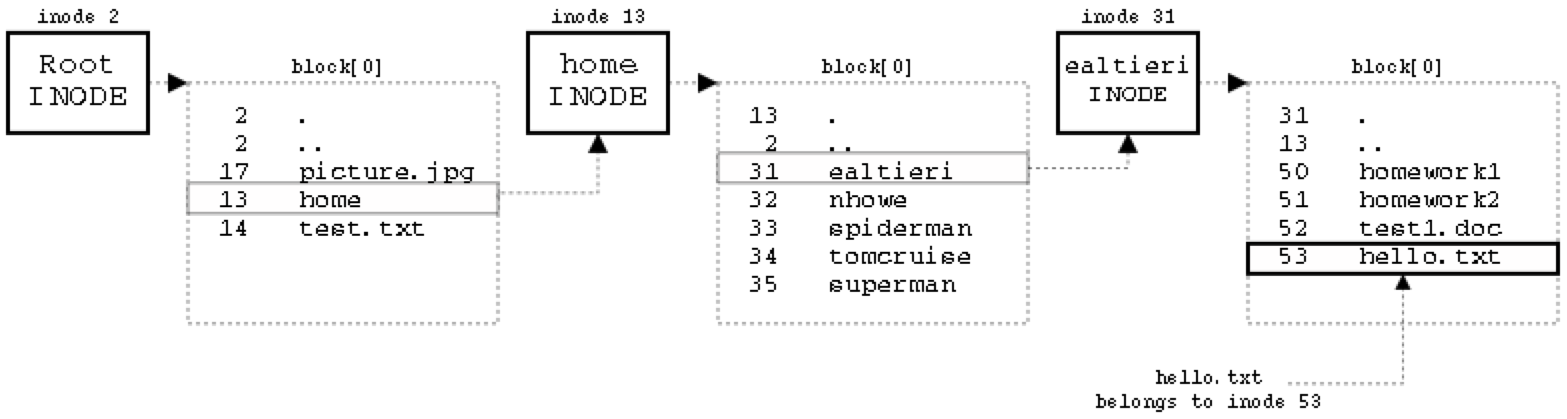
- In case the list of data blocks can’t fit in a single inode, one to three levels of indirection blocks are inserted.



http://www.science.unitn.it/~fiorella/guidelinux/tlk/img84.gif

The ext2 filesystem storage format

- Directory inodes contain a list of lines (records), one for each entity in the directory. A record is a mapping between filename and numerical inode identifier.
- Each record contains 5 fields: inode number, record length, filename length, file type (pipe, symlink, socket, chardev, blockdev, directory), and the filename itself.
- Remember lecture 3! The directory permissions (rwx) now all make sense. Manipulating a directory is the same as modifying or accessing this list of records.



http://cs.smith.edu/~nhowe/262/oldlabs/img/ext2_locate.png

File system management commands

- You can use tho **mount** command to list active mounts or to join an additional FS into the directory tree. Use **umount** for the reverse operation. Use **swapon** and **swapoff** to activate/deactivate swap space.
- Use **lsblk** to list all block devices as they are stacked (tree). Use **blkid** for printing UUID and header type of block devices.
- Use **mkfs.*[fstype]*** for creating filesystems. Use **mkswap** for creating a swap header.
- Use **du** for a file's/dir's space usage, **df** for free space info. Issue **free** for memory and swap usage info.
- Use **lsof** and **fuser** to look up open files and correspon-ding processes.
- Consult the **/etc/fstab** file for auto-activating block devices at system boot.

Partitoning and LVM commands

- Use **fdisk */dev/block_dev_name*** for partitioning disks with the MBR partitioning scheme.
- Use **gdisk** the same way for GPT disks.
- The **cfdisk** utility is a text-menudriven partitioning tool.
- The **pvcreate**, **pvremove**, **pvdisplay**, **pvs** commands can be used to manage the physical volume header on block devices/partitions.
- The **vgcreate, vgremove, vgdisplay, vgs** and **vgchange** commands stand for managing volume groups.
  Use **vgextend** or **vgreduce** to assign/unassign PVs to VG.
- Use **lvcreate**, **lvremove**, **lvchange**, **lvresize** to manage logical volumes. Use **lvdisplay** or **lvs** to print LV information.
- Each filesystem has its own FS resizer tool, e.g. **resize2fs**.

Relevant practices

- Complete practice **RH124 CH.14.3.** (mount)
- Complete practice **RH124 CH.14.8.** (du, blkid, ln, find)
- Complete practice **RH134 CH.9.2.** (fdisk, mkfs)
- Complete practice **RH134 CH.9.4.** (swap)
- Complete practice **RH134 CH.9.5.** (gdisk, swap, mount)
- Complete practice **RH134 CH.10.3.** (lvm)
- Complete practice **RH134 CH.10.5.** (lvextend, xfs_grow)
- Complete practice **RH134 CH.10.6.** (lvextend, xfs_grow)

Mid-term overview

Package manager, boot process, systemd

Lecture 8

Chapter 4

Distributions, repositories, packages

- Linux vendors/families usually have their distinctive package management systems with their own format.
- Package managers serve the purpose of version tracking/managing installed software.
- Packages are organized into package repositories. Repositories are consistent regarding dependencies and compatibility, packages inside a repository are maintained by the same organization.
- The RPM package format also allows for cryptogra-phically signing released software to prevent malware injection.
- In the RH family repositories are stored under the **/etc/yum.repos.d** directory, and they can be enabled/disabled with the **yum-config-manager** command. (--enable=*reponame*, --disable=*reponame)*

yum command

- Yum is an easy to use front-end to the rpm manager. You can use it to install/remove/update/search packages.
- By issuing **yum list** or **yum list installed** you can list all the available/installed packages.
- You can search for patterns in both package name and description text with **yum search all 'keyword'** command. To search for file (remember: a Linux command is a file in /bin or /sbin) use **yum provides *filename** or **yum provides /exact/path/to/file/name**
- Issuing **yum update *packagename*** will update a package to the latest version that has been made available in the repo. Omitting *packagename* will update all packages.
- You can use the **-y** switch to skip the confirmation on doing a yum operation.

yum command

- Use **yum install *packagename*** to install a package from the set of enabled repos. Use **yum localinstall *some-file.rpm*** to install from an rpm from a file in the file-system. Use **yum remove *pattern*** to erase one or more packages. This will also remove packages that depend on the removed package(s).
- Issuing **yum info *package*** will show the detailed description of the package.
- Some software suits are organized into groups, for e.g. "Development tools". You can do group operations by using the group sub-command: **yum group list**, **yum group info *grpname***, **yum group install *grpname***, etc..
- Yum has a history feature: **yum history *[nr]*** will print details of a transaction defined by *nr*. Omitting *nr* will list all recent transactions sorted by tr. number. You can roll back a transaction with **yum history undo *nr*** command.

rpm command

- You may use the **rpm** command for low level interaction with the package manager. The yum front-end actually executes several rpm commands behind the scenes.
- Only the query commands will be discussed in this course. You can use the **-q** switch to rpm to select query mode.
- The **rpm -q -a** command will list all installed packages, **-q -f *filename*** will find what package holds *filename.* Use **-q -i** to display package info, **-q -l** to list files in a package, **-q -c** to list config files only, **-q -d** to list documentation files only. The **-q --changelog** switch will print the version history and comments of a package, **-q --scripts** will show the executable scripts in the package.
- Use **rpm2cpio file.rpm | cpio -idv** to extract all files from an rpm to the current directory.

GRUB

- The self-extracting kernel binary (**vmlinuz**) and the **initrd** image is loaded and executed by the bootloader. The kernel binary itself has **command line arguments**, which can be edited form the **GRUB boot menu** with the e key.
- Boot flow is controlled by the rd.break, rd.debug, systemd.unit=*some*.target kernel cmdline arguments (among many others).
- GRUB configuration is stored in **/boot/grub/grub.cfg** file. This file is auto-generated with the **grub2-mkconfig** command. The configuration for programmatically generating the grub.cfg is stored in **/etc/default/grub** file.
- You can use **grub2-install --root-directory=*/root/of/fs* /dev/*boot_device*** to install a GRUB instance.
- Use **dracut** to generate a new initial RAMdisk image.

systemd

- The **PID=1** process has a special role in the Linux system startup. It is the only userland process executed by the kernel. By executing several other processes PID1 (/sbin/init) is responsible for setting up filesystem mounts, network connections, starting all the server processes, daemons, login terminals.
- Systemd is the PID1 process of many modern Linux systems. The **systemd** suite is a complex system manager with a substantial set of utilities and implicit daemons, like a built-in system logger (systemd-journald) and a network manager (systemd-networkd).
- Systemd has very clear declarative ini style (sections, key=value pairs) **unit files** for services, sockets, mounts. Units are organized into **targets**. Units are executed in **parallel** with respect to their dependency requirements.

systemd

- Switching targets is done with the **systemctl isolate *some*.target**, where target can be for e.g. graphical.tar-get or multi-user.target.
- The default target can be set by **systemctl set-default *some*.target** command.
- (Re)Starting and stopping units is done with **systemctl start/stop/restart *unit.name***, for e.g systemctl restart sshd.service.
- Setting a unit for autostart is done with **systemctl enable/disable *unit.name***.
- Issuing **systemctl mask *unit.name*** will hide a units, it can not be enabled in this state. Issue unmask to revert.
- Use **systemctl status** to query all services or **systemctl status *unit.name*** to query a single unit.

systemd

- Systemd units are stored in /usr/lib/systemd/system. This folder is managed by the package manager. User defined units go to /etc/systemd/system folder. If you want to modify a unit, place a unit file with the same name to the /etc folder of systemd instead, this will override the /usr version of the file with the same name.
- Use **systemctl list-units** or **list-unit-files** to list names or filenames. Use **--type=service** to filter for service units, use **systemctl --failed** to filter to failed units.
- Systemd has integrated kernel cgroups features.
- Use **systemd-cgls** to print the control group tree.
- Issue **systemd-cgtop** for a top-like utility that sorts systemd slices based on resource usage.

Relevant practices

- Complete practice **RH124 CH.13.4.** (yum install)
- Complete practice **RH124 CH.13.6.** (repo enable, yum)
- Complete practice **RH124 CH.13.8.** (rpm local install)
- Complete practice **RH124 CH.13.9.** (repo enable, yum)
- Complete practice **RH134 CH.13.4.** (lost root password)
- Complete practice **RH134 CH.13.6.** (emerg.targ, fstab)
- Complete practice **RH134 CH.13.8.** (grub2-mkconfig)
- Complete practice **RH124 CH.8.2.** (systemd units)
- Complete practice **RH124 CH.8.4.** (s. start/enable)
- Complete practice **RH124 CH.8.5.** (s. start/enable)

Mid-term overview

Networking, firewalld

Lecture 8

Chapter 5

Network Manager

- Network Manager is a resident process that initializes automatic network connections and handles all the network related events (like for e.g. plug of a cable, loss of a WiFi link) while it runs.
- It's global configuration is stored under the **/etc/NetworkManager** directory.
  Connection configurations are to be found under **/etc/sysconfig/network-scripts/ifcfg-*** files.
- It has CLI (nmcli), TUI (nmtui), and GUI (control network, nm-connection-editor) interfaces.
- Device objects in NM represent Linux network interface cards. Connection objects represent settings applicable to interfaces. Each interface can have several connections saved on disk. You can dynamically switch between saved connections.

nmcli command

- Examples for adding connections:
  nmcli con add con-name "default" type ethernet ifname eth1
- **nmcli con add con-name "demo" type ethernet ifname eth1 autoconnect no ip4 172.25.0.10/24 gw 172.25.0.254**
- You can switch between connections by issuing **nmcli con up "demo"** or **nmcli con up "default"**.
- You can administratively disable an interface with **nmcli disconnect dev eth1**. This will also terminate its current active connection.
- You can bring a connection down by **nmcli con down "demo"**. This will not disable the interface, meaning that in the event of a link change, the autoconnect connection for the device will be activated.

nmcli command

- You can modify an existing connection with **nmcli con mod *con-name property value*** like commands. E.g.: **nmcli con mod "static" connection.autoconnect yes**
- Some properties can have multiple values, like for e.g. ipv4.dns, since it is valid to have multiple resolvers configured in a network. You can assign and overwrite previous values the same way, e.g. **nmcli con mod "static" ipv4.dns 8.8.8.8** But in case of multi-value properties you can also prefix the setting with + or – to add/remove an entry, without modifying the other values of the same property. E.g. … **+ipv4.dns 8.8.8.8**
- To delete a connection use **nmcli con del "demo"**.
- To display connection details issue **nmcli con show "*connection-name*"**. To list connections simply omit the connections name. To list devices issue: **nmcli dev status**

Manual IP configuration

- To manually set the ip address of an interface, issue a command like this: **ip addr add 192.168.50.5 dev eth1** To remove: **ip addr del 192.168.50.5 dev eth1**
- To enable/disable an interface, use **ip link set eth1 up/down**.
- To add a routing table entry, issue **ip route add 10.10.20.0/24 via 192.168.50.100 dev eth0**
- To remove it, use: **ip route del 10.10.20.0/24 via 192.168.50.100 dev eth0**
- To set the default gateway: **ip route add default via 192.168.1.1**
- To manually edit the DNS resolver, modify **/etc/resolv.conf** with a text editor.
- To change hostname use: **hostnamectl set-hostname**

Firewalld

- Just like NM firewalld is a resident process with several interfaces (CLI, TUI, GUI). It accepts commands and generates iptables rules at run-time for setting up the kernels Netfilter subsystem accordingly.
- Firewalld divides traffic into zones. Zones are implemented as chains on the iptables level.
- Each zone has a separate list of rules (allowed services). An incoming packet will be classified to a zone, then matched against the zones rules and its fate will be determined.
- An incoming packet is checked for it’s source address, to determine the applicable zone. If no zone rule matches to the source address, then the zone selection decision will be based upon the interface that received the packet. All interfaces belong to exactly one zone.

firewalld services

- A service is defined by a set of ports, for e.g. the ssh service is equivalent to the single port of 22/tcp. The samba-client service consists of allowing two ports, 137/udp and 138/udp.
- A zone can have any number of enabled services. All ports, that do not belong to an enabled service of the zone, will not be available for communication.
- Use the **firewall-config** GUI utility for managing firewalld. Use **firewall-cmd** for command line management.
- Firewalld distinguishes a runtime configuration and saved state stored on disk, called the permanent configuration. Use **firewall-cmd --reload** to drop runtime and load from disk. Use --permanent to add a setting directly to the permanent configuration.

firewall-cmd

- You can print all the services with **firewall-cmd --get-services**. Issue **firewall-cmd --get-zones** for listing the zones, **--get-active-zones** for printing only the zones that have at least one interface in them.
- Use the **--get-default-zone** switch to see the default zone for new interfaces.
- Use **--list-all-zones** to print detailed information about zones (for. e.g. interfaces, enabled services, etc.)
- Use **firewall-cmd --get-services** to list the available services.
- Issue **firewall-cmd --list-all** for printing the whole configuration.
- Use **firewall-cmd --set-default-zone=*zone*** to set the default zone to *zone.*

firewall-cmd

- Take the --**add-source=10.1.0.0/16 --zone=trusted** example to assign a network to a zone. Use **--remove-source=10.1.0.0/16 --zone=trusted** to revert such a change.
- Use the following example to add an interface to a zone: **firewall-cmd --add-interface=etho --zone=trusted** Use --change-interface=etho --zone=dmz, to reassign the interface to a different zone.
- Use **--add-service=mysql --zone=dmz** to enable a service such as mysql to be available in the dmz zone.
- Use **--add-port=17784/tcp --zone=dmz** to allow non-standard service ports to a zone, without creating a service for it.
- Use --remove-port and --remove-service logically.

Relevant practices

- Complete practice **RH124 CH.11.3.** (ip utility)
- Complete practice **RH124 CH.11.5.** (nmcli)
- Complete practice **RH124 CH.11.7.** (ifcfg files manually)
- Complete practice **RH124 CH.11.9.** (hostnamectl)
- Complete practice **RH124 CH.11.10.** (nmcli con)
- Complete practice **RH134 CH.14.2.** (firewall-config GUI)
- Complete practice **RH134 CH.14.3.** (firewalld add service)

Recommended reading

Lecture 8

Chapter 6

Relevant chapters in RH:

- See list in Lectures 1-7

Read by next lecture:

- RH124 CH10 (Logs)
- RH134 CH4 (Scheduling tasks)

Use your rhlearn.gilmore.ca login to access the learning materials.

Thank you for your attention!

Lecture 8, PM-TRTNB319, Linux System Administration

Zsolt Schäffer, PTE-MIK