**Operation of Running Systems**

1. [Boot, reboot, and shut down a system safely](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#boot-reboot-and-shut-down-a-system-safely)
2. [Boot or change system into different operating modes](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#boot-or-change-system-into-different-operating-modes)
3. [Install, configure and troubleshoot bootloaders](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#install-configure-and-troubleshoot-bootloaders)
4. [Diagnose and manage processes](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#diagnose-and-manage-processes)
5. [Locate and analyze system log files](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#locate-and-analyze-system-log-files)
6. [Schedule tasks to run at a set date and time](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#schedule-tasks-to-run-at-a-set-date-and-time)
7. [Verify completion of scheduled jobs](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#verify-completion-of-scheduled-jobs)
8. [Update software to provide required functionality and security](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#update-software-to-provide-required-functionality-and-security)
9. [Verify the integrity and availability of resources](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#verify-the-integrity-and-availability-of-resources)
10. [Change kernel runtime parameters, persistent and non-persistent](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#change-kernel-runtime-parameters-persistent-and-non-persistent)
11. [Use scripting to automate system maintenance tasks](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#use-scripting-to-automate-system-maintenance-tasks)
12. [Manage the startup process and services (In Services Configuration)](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#manage-the-startup-process-and-services-in-services-configuration)
13. [List and identify SELinux/AppArmor file and process contexts](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#list-and-identify-selinuxapparmor-file-and-process-contexts)
14. [Manage Software](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#manage-software)
15. [Identify the component of a Linux distribution that a file belongs to](https://github.com/StenlyTU/LFCS-official/blob/main/stuff/OperationofRunningSystems.md#identify-the-component-of-a-linux-distribution-that-a-file-belongs-to)

**Boot, reboot, and shut down a system safely**

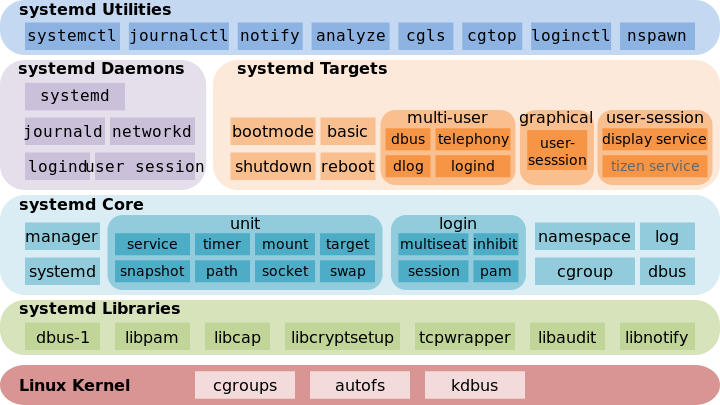
* shutdown -h now - shutdown
* shutdown -r now - reboot

**Boot or change system into different operating modes**

***Boot sequence:***

* POST (PowerOn Self Test) -> Find disk -> Inside disk there's bootloader -> bootloader load kernel -> kernel load init process
* Systemd is the default init process in CentOS
* Systemd starts services. Last service started will be a shell

***Systemd:***

[](https://camo.githubusercontent.com/f5b1cb59adc0cb5d09d2575392cab9d3467ab6d0d9a71b562d96f1df4acc89d8/68747470733a2f2f7777772e6c696e75782e636f6d2f696d616765732f73746f726965732f34313337332f53797374656d642d636f6d706f6e656e74732e706e67)

* Previous versions of Red Hat Enterprise Linux, which were distributed with SysV init or Upstart, implemented a predefined set of runlevels that represented specific modes of operation. These runlevels were numbered from 0 to 6 and were defined by a selection of system services to be run when a particular runlevel was enabled by the system administrator. In CentOS and Red Hat Enterprise Linux 7, the concept of runlevels has been replaced with systemd targets.
* Systemd targets are represented by target units. Target units end with the .target file extension and their only purpose is to group together other systemd units through a chain of dependencies.
* Systemd units are the objects that systemd knows how to manage. These are basically a standardized representation of system resources that can be managed by the suite of daemons and manipulated by the provided utilities.
* Systemd units in some ways can be said to similar to services or jobs in other init systems. However, a unit has a much broader definition, as these can be used to abstract services, network resources, devices, filesystem mounts, and isolated resource pools.
* Systemd was designed to allow for better handling of dependencies and have the ability to handle more work in parallel at system startup.

***Systemd commands:***

* systemctl get-default - It shows default target.
* systemctl list-units --type target --all - It shows all available targets.
* systemctl set-default multi-user.target/runlevel3 - Set multi-user target as default.

***Change target at boot time:***

* If during boot ESC is pressed the grub2 prompt will be showed.
* Highlight a kernel and press 'e'.
* Now is it possible to modify the boot parameter used to load the kernel.

**NOTE**: the changes are not persistent!

E.g systemd.unit=emergency.target can be added to boot system in emergency mode. NOTE: in this modality disk is mounted read only, to mount it read/write, after boot execute mount -o remount,rw /

* When the parameter change is end, press 'Ctrl + x' to boot system

References:

* <https://www.digitalocean.com/community/tutorials/systemd-essentials-working-with-services-units-and-the-journal>
* <https://en.wikipedia.org/wiki/Power-on_self-test>
* <https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files>)

**Install, configure and troubleshoot bootloaders**

The default bootloader in Centos7 is GRUB2.

***Installation:***

* grub2-install /dev/sdb -> Install GRUB on that disk

***Configuration:***

* To change bootloader configuration edit **/etc/default/grub** or **/etc/grub.d/**

vi /etc/default/grub

* The configuration information can be found with:

info -f grub -n 'Simple configuration'

man 7 bootparam -> It shows the kernel boot parameter.

* + Grub useraccount are different than the Linux accounts!
* When changes were made to /etc/default/grub they must be inserted in configuration file used directly by Grub2 which is **/boot/grub2/grub.cfg**. To do this execute:

grub2-mkconfig -o /boot/grub2/grub.cfg

grubby --default-kernel -> Show the default kernel.

grubby --set-default /boot/vmlinuz.. -> Change the default kernel.

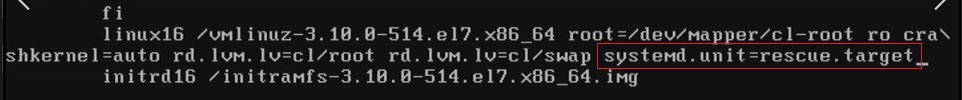
grubby --info=ALL -> Give general information for all kernels.

grubby --info /boot/vmliuz.. -> Give information for specific kernel.

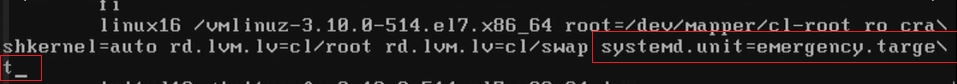
grubby --remove-args='rhgd quiet' --update-kernel /boot/vmlinux... -> Update kernel parameters.

***Troubleshooting:***

* rescue mode - minimal mode, with minimum amount of services loaded. If there is a problem during boot procedure, you can easily fix it.

[](https://github.com/Bes0n/LFCS/blob/master/images/img26.JPG)

* + once you are done press **Ctrl + D** to continue booting.
* emergency mode - In contrast to the rescue mode, nothing is started in the emergency mode. No services are started, no mount points are mounted, no sockets are established, nothing. All you will have is just a raw shell. Emergency mode is suitable for debugging purposes.

[](https://github.com/Bes0n/LFCS/blob/master/images/img27.JPG)

* + mount -o remount,rw / - puts your filesystem in read-write mode. Because in emergency state your file system by default in read-only mode

**Diagnose and manage processes**

***mpstat:***

* yum -y install sysstat
* mpstat -P ALL -u 2 3

CPU usage statistics.

-P Indicate the processor number for which statistics are to be reported, ALL for all cpu

-u Report CPU utilization

2 3 Display three reports at two second intervals.

***ps:***

* ps Processes of which I'm owner
* ps aux All processes.

It will print:

* + user - user owning the process
  + pid - process ID of the process
    - It is set when process start, this means that provide info on starting order of processes
  + %cpu - It is the CPU time used divided by the time the process has been running.
  + %mem - ratio of the process’s resident set size to the physical memory on the machine
  + VSZ (virtual memory) - virtual memory usage of entire process (in KiB)
  + RSS (resident memory) - resident set size, the non-swapped physical memory that a task has used (in KiB)
  + tty - On which process is running.
    - **NOTE**: *?* means that isn't connect to a tty
  + stat - process state
  + start- starting time or date of the process
  + time - cumulative CPU time
  + command - command with all its arguments
    - Those within *[ ]* are system processes or kernel thread
* List processes sorting them by CPU usage: ps -eo pid,ppid,cmd,%cpu,%mem --sort=-%cpu

-e show same result of aux

-o chose columns to show

--sort sort by provided parameter

ppid parent process id

* ps -eo pid,args --forest

--forest show a graphical view of processes tree

* In /proc/[pid]

There is a numerical subdirectory for each running process; the subdirectory is named by the process ID.

* /proc/[pid]/fd

This is a subdirectory containing one entry for each file which the process has open, named by its file descriptor, and which is a symbolic link to the actual file. Thus, 0 is standard input, 1 standard output, 2 standard error, and so on.

* lsof -p pid - Lists open files associated with process id of pid

***Background processes:***

* End a command with & execute a process in background

sleep 600 &

* jobs - List processes in background
  + [1]+ Running -> means that this process is in focus, so you can simply run bg without 1.
* fg pid - To return a process in foreground

***Process priority:***

* ps -eo pid,nice,command
  + nice (NI) is the process priority
* More priorities and more CPU time will be assigned to process
* nice value can be between -20 and 19
* -20 is highest and 19 is lowest
* **NOTE**: only root can assign negative values
* nice -n value command & - It will execute command in background with nice equal to value
* renice re-assign priority to a process
  + renice -n value pid

***Signals:***

* kill pid - Send a SIGTERM to process with pid equal to pid.
* kill -9 pid - Send a SIGKILL to process with pid equal to pid. Overkill!
* kill -number pid - Send a signal that correspond to number to process with pid equal to pid.
* kill -l - List all available signal and corresponding number.
* man 7 signal - Explains every signal. No need to know them by heart!

References:

* <https://superuser.com/questions/117913/ps-aux-output-meaning>
* <http://man7.org/linux/man-pages/man5/proc.5.html>

**Locate and analyze system log files**

***Audit user login:***

* lastlog -> Reports the most recent login of all users(output taken from /etc/passwrd).
* last -> Give information for the last login users.
* last -f /var/log/\*tmp -> It maintains the logs of all logged in and logged out users.
  + *utmp* will give you complete picture of users logins at which terminals, logouts, system events and current status of the system, system boot time (used by uptime) etc.
  + *wtmp* gives historical data of utmp.
  + *btmp* records only failed login attempts or use the command **lastb**.

***Audit Root Access:***

* /var/log/secure -> Hold audit information such as: failed login attempts. On Ubuntu - /var/log/auth.log
* grep sudo /var/log/secure -> Give info when sudo is used.

***General Info:***

* Usually log files are stored in /var/log/
* In Centos many tools use rsyslog to manage logs.
* rsyslog is a daemon that permit the logging of data from different types of systems in a central repository
  + /etc/rsyslog.conf -> configuration file of rsyslog.
  + systemctl status rsyslog -> to check execution status of rsyslog.
  + logger -p cron.crit "someting bad happened" -> Send a log message to cron facility with priority critical. It will be logged in **/var/log/cron**
* logrotate -> Is responsible for rotating the log files.
  + /etc/logrotate.conf -> Configuration for the logrotate.
  + /etc/logrotate.d/ -> You can put your own configuration in that directory.
  + We have logrotate script in */etc/cron.daily* directory.

***Journalctl logs:***

* journalctl -> Opens file system's journal. By default it kept in memory. It's going to be truncated when becomes too big.
* journalctl \_PID=1 -> Search for logs related to the PID number 1.
* journalctl \_UID=33 --since today -> Show the logs for specific user.
* journalctl --dmesg -> Show the kernel logs.
* journalctl --since "10 minutes ago" -> Show journal information for the last 10 minutes.
* mkdir -p /var/log/journal - *journalctl* is not persistent. To make it persistent we need to create directory *journal* in */var/log* dir. After creation of this directory, we will have directory created in journal dir.
* There is also configuration file behind of this, which located in **/etc/systemd/journald.conf**. Here you can specify configuration for your journal. Size, what to store, storage behaviour and so on.

References:

* <https://www.ittsystems.com/what-is-syslog/>

**Schedule tasks to run at a set date and time**

* Daemon that schedule tasks, called jobs, to run at a set date and time is cron
* The schedule of various tasks depend by configuration contained in below files/directories:
  + /etc/crontab
    - Normally isn't edited
      * **NOTE**: It's content can be used as remainder of cron files syntax
    - Each row is a task that must be executed in a scheduled way
    - A special syntax indicates the schedule of each commands
  + /etc/cron.d
    - It contains files with same syntax of /etc/crontab
    - Normally it used by software packages installed in system
  + /var/spool/cron
    - It contains tasks for users
    - Contents can be edited using crontab command
  + /etc/cron.hourly
    - Each script in this directory will be executed every hour
    - Exact time isn't specified but execution is granted, with a combination of deamon cron and anacron
  + /etc/cron.daily
    - Each script in this directory will be executed every day
    - Exact time isn't specified but execution is granted, with a combination of deamon cron and anacron
  + /ect/cron.weekly
    - Each script in this directory will be executed every week
    - Exact time isn't specified but execution is granted, with a combination of deamon cron and anacron
  + /etc/cron.monthly
    - Each script in this directory will be executed every month
    - Exact time isn't specified but execution is granted, with a combination of deamon cron and anacron
  + /etc/cron.allow
    - Only those peope into the allow file are allowed to run cronjobs.
  + /etc/cron.deny
    - Be default all users are allowed to run cronjobs except those users specified into */etc/cron.deny* file
    - NOTE that if neither of these files exists then, depending on site-dependent configuration parameters, either only the super user can use cron jobs, or all users can use cron jobs.
  + /etc/anacrontab
    - Configuration file for anacron.
    - Anacron is not so time dependent. We can specify the time job will run in delay in minutes after the system boots.
* The major difference between cron and anacron is that cron works effectively on machines that will run continuously while anacron is intended for machines that will be powered off in a day or week.

To modify cron jobs:

* crontab -e -> It is used by user to modify his jobs. This command actully creates files in */var/spool/cron* with the name of the user.
* crontab -e -u user -> It is used by root to modify user's jobs
* crontab -u user -l -> Print user's jobs or better show content of file in */var/spool/cron*
* crontab -r -> Delete the crontabs.

***Cron syntax:***

# ┌───────────── minute (0 - 59)

# │ ┌───────────── hour (0 - 23)

# │ │ ┌───────────── day of the month (1 - 31)

# │ │ │ ┌───────────── month (1 - 12)

# │ │ │ │ ┌───────────── day of the week (0 - 6) (Sunday to Saturday;

# │ │ │ │ │ 7 is also Sunday on some systems)

# │ │ │ │ │

# │ │ │ │ │

# \* \* \* \* \* command to execute

* # this line is a comment
* \* always
* 1 0 \* \* \* /command command will be executed one minute past midnight (00:01) every day
* 1-30 \* \* \* \* /command command will be executed every day, every hour at minutes 1 to 30
* \*/10 \* \* \* \* /command command will be executed every 10 minutes, or rather when minutes are 00, 10, 20, 30, 40 and 50.
* 00 \*/2 15 \* \* /command command will be executed the fifteenth day of every month, every two hours
* 00 1-9/2 1 5 \* /command command will be executed on 1st May at 1,00 - 3,00 - 5,00 - 7,00 - 9,00, or rather every two hours from 1,00 to 9,00
* 00 13 2,8,14 \* \* /command command will be executed second, eighth and fourteenth day of each month at 13.00

***at***

* yum -y install at
* **NOTE**: it require that atd demon will be in execution
  + systemctl start atd
  + systemctl enable atd
* at 11:00 open a shell in which inserted commands that will be executed at 11:00
  + ctrl+d close shell
* atq -> Shows scheduled activities identified by an activity ID
* atrm ID -> Will remove from schedule activity with activity ID equal to ID

References:

* <https://en.wikipedia.org/wiki/Cron>

**Verify completion of scheduled jobs**

* Cron will send an email to internal mail spool.
* Enable the logging of crond events.
* Edit the **/etc/rsyslog.conf** and remove comment from this line:

# Log cron stuff

cron.\* /var/log/cron

* systemctl restart rsyslog -> It will restart rsyslog server.

**Update software to provide required functionality and security**

* yum update
* Yum also offers the upgrade command that is equal to update with enabled obsoletes configuration option. By default, obsoletes is turned on in /etc/yum.conf, which makes these two commands equivalent.
* The obsoletes option enables the obsoletes process logic during updates. When one package declares in its spec file that it *obsoletes* another package, the latter package is replaced by the former package when the former package is installed. Obsoletes are declared, for example, when a package is renamed

References:

* <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/ch-yum>
* <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sec-Configuring_Yum_and_Yum_Repositories#sec-Setting_main_Options>

**Verify the integrity and availability of resources**

* /usr/lib/rpm/rpmdb\_verify /var/lib/rpm/Packages -> It will verify the integrity of rpm database.
* rpm -V <package-name> -> Verify integrity of package.
  + **NOTE**: When verifying a package, RPM produces output only if there is a verification failure.

**Verify the integrity and availability of key processes**

* systemctl status processname It will show the status of process with name processname
  + The last rows are the recent logs generated by daemon
* Other command to check processes status:
  + ps
  + pgrep
  + mpstat
* pkill -G 1000 - Kill all processes own by user with ID 1000.
* pgrep -lG 1002 sleep - Show all processes owned by user 1000 related to sleep command.

**Change kernel runtime parameters, persistent and non-persistent**

* lsmod - list kernel modules. Modules loaded automatically when they're needed
* modprobe cdrom - load cdrom module.
* modprobe -r cdrom - unload module cdrom.
* The modules are located in /lib/modules/$(uname -r)/kernel/drivers folder.
* In /proc/sys are contained kernel tunables, parameters that are used to customize the behavior of system.
* Example
  + cd /proc/sys/net/ipv6/conf/all
  + echo 1 > /proc/sys/net/ipv6/conf/alldisable\_ipv6 - Will disable IPv6
  + **NOTE**: This is a runtime change, not permanent
  + **NOTE**: With these files vi cannot be used
* Alternative method: sysctl -w net.ipv6.conf.all.disable\_ipv6=1
* sysctl -a - shows all parameters that can be configured

**To make configuration permanent:**

* cd /etc/sysctl.d
* echo net.ipv6.conf.all.disable\_ipv61=1 > ipv6.conf
  + **NOTE**: the only request is that file will end with .conf
  + **NOTE**: it is better to use: /etc/sysctl.d/99-sysctl.conf link to /etc/sysctl.conf
* sysctl -p reload permanent configuration. Alternative: reboot system

**Some parameters changed often:**

* net.ipv4.ip\_forward=0 disable packet forwarding
* fs.file-max -> maximum number of open files
* kernel.sysrq -> It is a ‘magical’ key combo you can hit which the kernel will respond to regardless of whatever else it is doing, unless it is completely locked up.
* net.ipv4.icmp\_echo\_ignore\_all -> ignore ping

**Use scripting to automate system maintenance tasks**

Bash shell script:

* #!/bin/bash must be first row
* A Bash script is a plain text file which contains a series of commands or/and typical constructs of imperative programming
* It is convention to give files that are Bash scripts an extension of **.sh**
* chmod +x nomefile.sh must be executable
* ./nomefile.sh execute nomefile.sh

References:

* <https://ryanstutorials.net/bash-scripting-tutorial/bash-script.php>

**Manage the startup process and services (In Services Configuration)**

* systemctl command used to manage servers. In Linux servers often are called *daemons*
* systemctl status processname -> It will show the status of process with name processname
  + Active process status eg. inactive, active
  + Loaded unit file name
    - unit file name; enable - This means that daemon will be executed automatically at the next reboot
    - unit file name; disabled This means that daemon won't be executed automatically at the next reboot
  + The last rows are the recent logs generated by daemon
* systemctl start sshd -> It will start sshd daemon.
* systemctl stop sshd -> It will stop sshd daemon.
* systemctl restart sshd -> It will restart sshd daemon.
  + **NOTE**: A restart must be executed each time a daemon configuration file is changed.
* systemctl disable sshd -> Disable the execution of service at bootstrap.
* systemctl enable sshd -> Enable the execution of service at bootstrap.
* systemctl is-enabled sshd -> Check if daemon is enable or disabled in bootstrap sequence.
* systemctl list-unit-files -> List all systemd units object available.
* systemctl list-dependencies sshd.service -> Get dependencies of the service.
* systemctl cat sshd -> Display sshd unit file that systemd has loaded into its system.
* systemctl edit --full sshd -> Will open editor to edit service file.
* systemctl show sshd.service -> This will display a list of properties that are set for the specified unit.
* systemctl mask/unmask nginx.service -> This will prevent the Nginx service from being started, automatically or manually, for as long as it is masked.

***In Services Configuration:***

* /usr/lib/systemd/system/sshd.service - *.service* file goes for any unit file and consist of three parts
  + Unit - generic information about service, dependencies.
  + Service - service definition itself
  + Install - important for target
* [Unit]
* Description=OpenSSH server daemon
* Documentation=man:sshd(8) man:sshd\_config(5)
* After=network.target sshd-keygen.service
* Wants=sshd-keygen.service
* [Service]
* Type=notify
* EnvironmentFile=/etc/sysconfig/sshd
* ExecStart=/usr/sbin/sshd -D $OPTIONS
* ExecReload=/bin/kill -HUP $MAINPID
* KillMode=process
* Restart=on-failure
* RestartSec=42s
* [Install]
* WantedBy=multi-user.target
* systemctl set-property httpd.service MemoryLimit=500M - set memory limit to the httpd.service. Service must be active to apply some changes on it.
* /etc/systemd/system/multi-user.target.wants - we have this directory which contains of following sym links. If we include service in multi-user.target it will create symbolic links
* lrwxrwxrwx. 1 root root 35 Aug 21 15:08 atd.service -> /usr/lib/systemd/system/atd.service
* lrwxrwxrwx. 1 root root 38 Jul 17 14:56 auditd.service -> /usr/lib/systemd/system/auditd.service
* lrwxrwxrwx. 1 root root 39 Aug 14 11:56 chronyd.service -> /usr/lib/systemd/system/chronyd.service
* lrwxrwxrwx. 1 root root 37 Jul 17 14:55 crond.service -> /usr/lib/systemd/system/crond.service
* lrwxrwxrwx. 1 root root 42 Jul 17 14:56 irqbalance.service -> /usr/lib/systemd/system/irqbalance.service
* lrwxrwxrwx. 1 root root 37 Jul 17 14:56 kdump.service -> /usr/lib/systemd/system/kdump.service
* lrwxrwxrwx. 1 root root 46 Jul 17 14:55 NetworkManager.service -> /usr/lib/systemd/system/NetworkManager.service
* lrwxrwxrwx. 1 root root 39 Jul 17 14:56 postfix.service -> /usr/lib/systemd/system/postfix.service
* lrwxrwxrwx. 1 root root 40 Jul 17 14:55 remote-fs.target -> /usr/lib/systemd/system/remote-fs.target
* lrwxrwxrwx. 1 root root 46 Jul 17 14:55 rhel-configure.service -> /usr/lib/systemd/system/rhel-configure.service
* lrwxrwxrwx. 1 root root 39 Jul 17 14:56 rsyslog.service -> /usr/lib/systemd/system/rsyslog.service
* lrwxrwxrwx. 1 root root 36 Jul 17 14:56 sshd.service -> /usr/lib/systemd/system/sshd.service
* lrwxrwxrwx. 1 root root 37 Jul 17 14:55 tuned.service -> /usr/lib/systemd/system/tuned.service

***References:***

* <https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units>

**List and identify SELinux/AppArmor file and process contexts**

* In computer security, mandatory access control (MAC) refers to a type of access control by which the operating system constrains the ability of a subject or initiator to access or generally perform some sort of operation on an object or target. In practice, a subject is usually a process or thread; objects are constructs such as files, directories, TCP/UDP ports, shared memory segments, IO devices, etc. Subjects and objects each have a set of security attributes. Whenever a subject attempts to access an object, an authorization rule enforced by the operating system kernel examines these security attributes and decides whether the access can take place. Any operation by any subject on any object is tested against the set of authorization rules (aka policy) to determine if the operation is allowed.
* In CentOS as MAC is used SELinux
* SELinux can be in three states:
  + *enforcing*: Actions contrary to the policy are blocked and a corresponding event is logged in the audit log.
  + *permissive*: Actions contrary to the policy are only logged in the audit log: /var/log/audit/audit.log
  + *disabled*: The SELinux is disabled entirely.
  + The status can be configured in file /etc/sysconfig/selinux which is symlink to /etc/selinux/config. Changes to this file will be read only after reboot.
  + The state can be changed from *enforcing* to *permissive* and vice versa without system reboot.
  + When the state is set to disable the only way to re-enable SELinux is to change /etc/sysconfig/selinux and reboot.
  + In case if you want to disable SELinux. Change SELINUX=enforcing to SELINUX=disabled and reboot. But you don't want to do that, system need to be secured.
* getenforce - Show the SELinux state.
* sestatus - Show the status of the SELinux.
* setenforce Permissive - set the state to permissive. Only Enforcing or Permissive modes are available.
* setenforce Enforcing - set the state to enforcing.

***Context:***

* On systems running SELinux, all processes and files are labeled in a way that represents security-relevant information. This information is called the *SELinux context.*
* Normally SELinux context is showed with -Z option.
* ls -lZ - show SELinux context of file
* [osboxes@osboxes ~]$ ls -Z /etc/shadow
* -rw-------. root root system\_u:object\_r:shadow\_t:s0 /etc/shadow
* ps auxZ - show SELinux context of processes.
* A SELinux context has the form *user:role:type*
  + To identify easy the user, role and type look at the last two characters: ....\_u:....\_r:....\_t
  + type indicate the type of object
  + unconfined\_t are object not limited by SELinux.
* chcon -t unlabeled\_t /etc/hosts -> Change SELinux type of a file.
* chcon --reference=/path/to/existingfile /path/to/a/newfile -> How to copy context from file to file.
* restorecon /etc/hosts -> Restore SELinux Context from Master file
  + The default context(Master file) is stored in /etc/selinux/targeted/contexts/files/file\_contexts
  + /etc/selinux/targeted/contexts/files/file\_contexts.local stores contexts to newly created files and directories not found in *file\_contexts*
* semanage fcontext -a -t samba\_share\_t /etc/file1 -> Adds the following entry to /etc/selinux/targeted/contexts/files/file\_contexts.local
  + restorecon -v /etc/file1 -> Changes the type to *samba\_share\_t*
* semanage fcontext -a -t httpd\_sys\_content\_t "/home/webdir(/.\*)?" -> Change the context type of the dir and all files in it.

***Booleans:***

* getsebool -a || semanage boolean -l -> To list all SELinux Booleans; semanage give explanation of the bool.
* setsebool -P httpd\_read\_user\_content on - To make permanent change into the specific SELinux Boolean.
* semanage boolean --modify httpd\_read\_user\_content --on - Change the boolean using **semanage** command.

***Ports:***

* semanage port -l | grep 22 -> Show SELinux Port type.
* semanage port -a -t ssh\_port\_t -p tcp 2222 -> Add port 2222 to the ssh ports.
* semanage port -d 82 -t http\_port\_t -p tcp -> Remove port 82 from http ports.

***Advanced usage:***

* Install setroubleshoot on *Centos7*: sudo yum install setools policycoreutils policycoreutils-python
  + *SELinux Troubleshooter* GUI tool can be used for debugging.
* semodule -i dovecot\_dict.pp - Install a new module.
* semodule -R - Reload the SELinux policy, in order to get the new module into effect.

***References:***

* <https://en.wikipedia.org/wiki/Mandatory_access_control>
* <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/chap-security-enhanced_linux-selinux_contexts>

**Manage Software**

***yum***

* packet manager that use RPM packet manager
* yum search keyword

This is used to find packages when you know something about the package but aren't sure of it's name. By default search will try searching just package names and summaries, but if that "fails" it will then try descriptions and URLs.

* *Repository*: collections of software packages used by yum. They are configured in /etc/yum.repos.d
* *Yum* downloads the RPM packages into its cache folder: /var/cache/yum... Use tree after that.
* yum info package -> Information on package
  + If package is installed Repo will be equal to "installed"
* yum install package -> Install package
* yum provides \*/file -> Search package that contain file
* yum remove package -> Remove package
* yum autoremove package -> Remove package plus unused dependencies
* yumdownloader package -> Download the RPM package
  + **NOTE**: require yum -y install yum-utils
* yum deplits nmap -> Show dependencies for nmap.
* yum repolist -> Give available repositories.

***RPM***

* rpm -i file.rpm -> Install file.rpm
* rpm -U file.rpm -> Upgrade file.rpm
* rpm -qa -> List all installed RPM
* rpm -qf file -> Tells to what RPM package file belong.
* rpm -e <package-name> -> Erase the installed package.
* rpm -V <package-name> -> Verify integrity of package.
* rpm -qpi tmux-1.8-4.el7.x86\_64.rpm -> Will print the information of a package. Can be done before installing it.
* rpm -qpl tmux-1.8-4.el7.x86\_64.rpm -> Show what files are included into the package.
  + use -qpc for showing only **config** files.
  + use -qpd for showing only **doc** files.
  + use -qpR for the list of all **dependencies** of that RPM file.

**Identify the component of a Linux distribution that a file belongs to**

* yum provides \*/file -> Search package that contain file
* ldd /path/command -> Show all libraries used by command
* This info is contained in a library cache
* The library cache can be re-build using ldconfing
* The library cache is in /etc/ld.so.cache
* The info for cache are in /etc/ld.so.cache.d/
* The cache is normally re-build each time a new package is installed