COMMANDS FOR WORKING WITH SERVICES

1- Managing services with systemd

1.1- Starting, stopping, restarting, enabling, disabling and checking status of services with systemctl

a) On Linux, **systemctl** is a utility that may be used to manage services running on your system. This command should be run as a **root** user.The following options are available with **systemctl**:

- **start** → Starting a service.
- **stop** → Stopping a service.
- **restart** → Restarting a service.
- **status** → Displaying a service status (is stopped or started).
- **enable** → Enabling a service
- **disable** → Disabling a service

b) Synopsis: **systemctl option service_name**

c) Examples:

- **sudo systemctl start ssh** → This command starts the ssh service
- **sudo systemctl stop apache2** → This command stops the apache service
- **sudo systemctl restart cups** → This command restarts the cups service
- **sudo systemctl enable mysql** → This command enables the mysql service
- **sudo systemctl disable ssh** → This command disables the ssh service
- **systemctl status cups** → This command displays basic information about the cups service status
- **sudo systemctl status cups** → This command displays extended information about the cups service

1.2- Listing services and their status

a) **systemctl list-units --type service** → This command shows the list of each service on your system and its current status. Command **sudo** is **not** required.

b) **systemctl list-units --type service --state=running** → This command shows the list of each service loaded, active and running on your system and its current status. Command **sudo** is **not** required.

c) **systemctl list-unit-files --state=enabled** → This command shows the list of enabled services. Command **sudo** is **not** required.

d) **systemctl list-unit-files --state=disabled** → This command shows the list of disabled services. Command **sudo** is **not** required.

e) **systemctl is-enable service_name** → This command shows if a particular service is enabled or not. Command **sudo** is **not** required. Example: **systemctl is-enabled apache2** shows if **apache2** is disabled or not.

f) **sudo netstat -atupn** → This command shows network services running in your system just now. This command should be run as a **root** user. If **netstat** is not installed on your system, you should install a software package called **net-tools**.

1.3- Checking dependencies

a) Sometimes services can include dependencies to other services. A **service dependency** means that **one service needs another service to be running (or started first)** in order to work correctly.

b) In Debian Linux, if a services depends on another services, systemd tries to start the required service. If the required service fails to start, the dependent service will not start.

c) **systemctl list-dependencies service_name** → This command shows dependencies of service_name.

d) **systemctl list-dependencies --reverse service_name** → This command shows the list of services that depend of service_name.

e) Example 1: **systemctl list-dependencies cups** → It shows dependencies of **cups.**

f) Example 2: **systemctl list-dependencies --reverse cups** → It shows services that depend of **cups.**

2- Example of how to add a new service from the scratch

a) Download the source code of a server called **echod.c** from the following URL:

https://www.collados.org/asix1/sm1/tasks/sm1act10/echod.c

b) Compile **echod.c**. Run the command: **gcc -O echod.c -o echod**. A new program **echod** will be created. This program is a server.

c) With root privileges, copy **echod** into **/usr/sbin**.

d) A **systemd service file** is a configuration file that provides information to the system about how to manage a specific service. These files are typically stored in the **/etc/systemd/system** directory and have a **.service** extension. Download a **systemd service file** called **myservice.service** from the following URL:

https://www.collados.org/asix1/sm1/tasks/sm1act10/myservice.service

e) Change the filename of **myservice.service** to **echod.service**. Afterwards, edit the contents of **echod.service** and make the following changes:

- Line 1 → **# Contents of /etc/systemd/system/echod.service**
- Line 3 → **Description=Echod Service**
- Line 9 → **ExecStart=/usr/sbin/echod**

f) Copy, with **roo**t privileges, **echod.service** into **/etc/systemd/system**.

g) Run the following commands with **root** privileges to enable the service:

```
systemctl daemon-reload
systemctl enable echod.service
```

h) Start the new service running with root privileges:

```
systemctl start echod.service
```

i) Check the status of the new service runnig:

```
systemctl status echod.service
```

j) Restart the new service running with root privileges:

```
systemctl restart echod.service
```

and check its status and its PID.

k) Stop the new service running with root privileges:

```
systemctl stop echod.service
```

and check its status

3- Example of how to run a programs, command or bash script during the boot process

a) Create with the help of nano the following bash script:

```
#!/bin/bash
date > /etc/lastStUpdSw
apt-get update
exit 0
```

Save the script with the following name: **StUpdSw.sh**. This is the script that you want to run during the boot process.

About this script:

- It is not a service. When the last line is executed, the script ends.
- It updates the list of packages available for installation on your system.

b) Copy this file to **/usr/sbin** with permissions **755**.

c) Create a file called **StUpdSw.service** in **/etc/systemd/system** with the following content:

```
[Unit]
Description=Update List of Sofware Packages during Boot process

[Service]
ExecStart=/bin/bash /usr/sbin/StUpdSw.sh

[Install]
WantedBy=multi-user.target
```

d) Run the following commands with root privileges:

```
systemctl  daemon-reload
systemctl  enable  StUpdSw.service
```

e) Check the status of the new service runnig:

```
systemctl  status  StUpdSw.service
```

f) Reboot the system. Check that:

- The contents of **/etc/lastStUpdSw** shows last time an **update** of the **software package list** was made during the boot process.
- Run **sudo journalctl -u StUpdSw.service | grep "Started StUpdSw.service" | tail -n 1** and check that during the last boot process the script was started and an update of the software package list was made.
- Check that date and time shown by **/etc/lastStUpdSw** and **journalctl** match.