Manage Startup Processes and services

When Linux boots up, certain programs are startup automatically. Some of them start up in a specific order. If important applications crash, they will be restarted automatically.

init=initialization system

How does this init system know how to startup applications, what to do when a program crashes and so on…  
It needs specific instructions on how to do its job. And it has all the instructions it needs to do its job in what’s are called systemd units.  
These are simply text files that describe the necessary logic.

And systemd is the name of the collection of tools, components and applications that help start, operate, and manage most of the linux-based OS’s.

Systemd is also the name of the program that starts up as the init system, which can be a bit confusing.

Units can be of various types such as services, socket, device, timer and others.

Timer units let us tell the init system that it should launch a specific application at a specific time.

The focus here is on service units.

These have clear instructions about such things as what command to issue to startup a program, what to do if a program crashes, what command to issue when a program is restarted, and many other things. In a nutshell, a service unit tells the inti system all it needs to know about how it should manage the entire lifecycle of an application.

We can investigate a service by using this command: (ssh is the example service used)

$: systemctl cat ssh.service

If I wanted to edit the service file and modify the instructions, I can use the following command:

$: sudo systemctl edit - -full ssh.service

If I wanted to cancel the edits and return the service file to its factory default settings, I can run the following command:

$: sudo systemctl revert ssh.service

When running the status command, observe if you can see if its enabled (usually highlighted in green)  
If it is enabled, that means that it is configured to startup automatically when the system boots up.

PID = Process Identifier. Every time a program is launched on Linux, a process will startup. The process encapsulates the computer code and resources that are loaded into memory and using the CPU when it needs to execute instructions, and every process has this unique number identifying it. The PID can be used to interact with this process.

Sometimes, we will change the settings of a program. For example, to modify how the SSH daemon works, we could change the settings in the file at /etc/ssh/sshd\_config, after changing the configuration like this, the process currently running will still use the old settings until it is restarted.

One way to force the process to restart and pick up the new settings is to use the systemctl restart command. But this method is very disruptive, its better to use the reload command:  
$: sudo systemctl reload ssh.service

This will reload the service gracefully. \*\* Not all programs support being reloaded this way \*\*

There is a way to try a reload first, and then a restart if the reload is not supported by that app:

$: sudo systemctl reload-or-restart ssh.service

How to check if a service is enabled or disabled  
$: systecmtl is-enabled ssh.service

Some services are very stubborn, if I were to disable them from auto starting at boot and we stop them, I’ll notice later that the process is somehow still running. Somehow, it mysteriously started up on its own, so I’m left wondering about how this happened.

Sometimes services could auto start up other services if they want to do so, so there can be a domino effect. Service one can startup service two, even if you stopped and disabled service two.

There is a brute force way to prevent this from happening and it’s called masking.

$: sudo systemctl mask atd.service

This will point this service to /dev/null (The black hole in Linux)  
You cannot enable or start this service once it is masked.

How to unmask the service  
$: sudo systemctl unmask atd.service

How to get a list of services in the system:  
$: sudo systemctl list-units - - type service - - all

List systemd service files  
$: ls /lib/systemd/system

The services that I will create should be dropped into the /etc/systemd/system/<CustomService.service>

How to reload the entire systemd daemon:  
$: sudo systemctl daemon-reload

To see man pages for specific systemd units, type the following:  
$: man systemd. (type enter after this to locate a specific unit)