Control the boot process

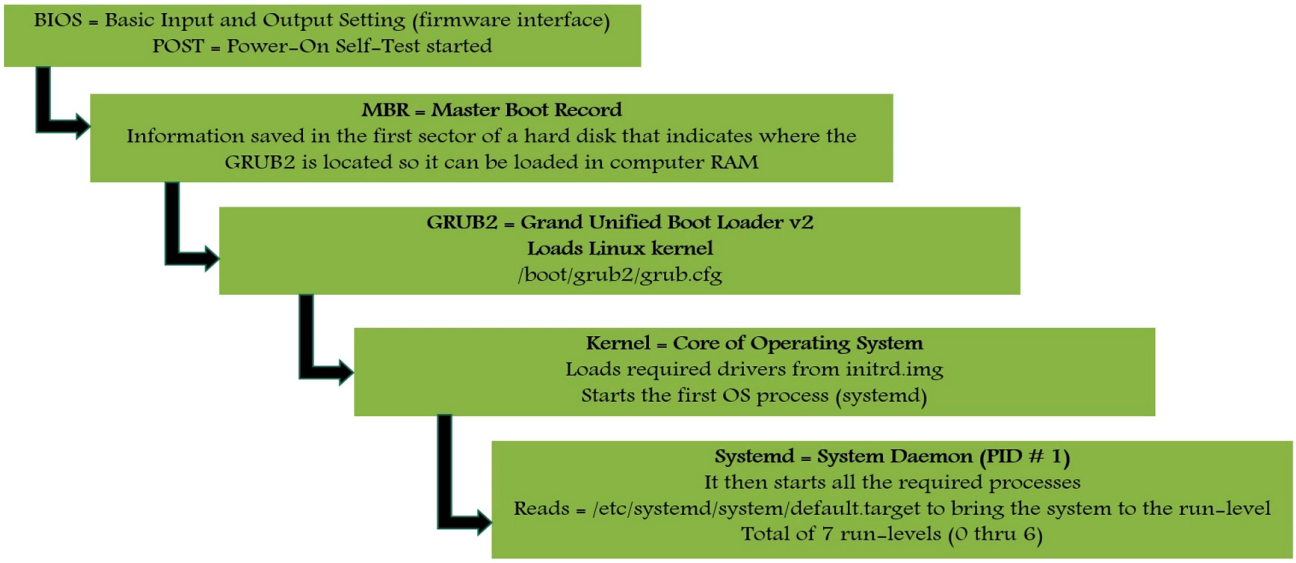
Linux #redhat #boot #run-level

In this note:

- The Linux boot process, set the default target used when booting, and boot a system to a non-default target
- Recover root password
- Repair file system configuration or corruption issues

Linux Boot Process

- The boot sequence changes in CentOS/Redhat 7 and above
- `systemd` is the new service manager in CentOS/RHEL 7 that manages the boot sequence
- It is backward compatible with SysV init scripts used by previous versions of RedHat Linux including RHEL 6
- Every system administrator needs to understand the boot process of an OS in order to troubleshoot effectively
  - What happens if the system doesn't boot, then you'll need to understand the boot sequence to know what is going wrong.



- The firmware comes in with the machine pre-installed by the manufacturer, that firmware is called BIOS which stands for Basic Input and Output Setting.
- When the computer starts up (it takes the power on), and it goes through the post process, The post process selected means that it starts up and then it goes through all the hardware that is attached and checks if the device is OK or not, are there any problems with the hardware?
  - After that, the MBR is loaded, MBR stands for Master Boot Record
  - BIOS looks up, it does the post, and then it goes to the first sector of the disk and finds out that MBR, this MBR has the information of GRUB2
    - GRUB2 stands for Grand Unified Boot Loader v2
    - GRUB2 loads the Linux kernel, that information is is `/boot/grub2/grub.cfg`
    - When you boot up your Linux operating system, you probably don't see the BIOS and posts because it goes through your screen so fast. And MBR is probably not noticed because that's the time that your machine is

looking for that sector. But you'll see at one point on your screen, on your black screen, it's gonna show up as Linux Kernel, and sometimes you probably gonna see 1,2,3,4,5, etc. different kernels that it needs to boot from.

- At that time you could move your cursor up and down and select the kernel you want to load or let the system decide which kernel it's gonna load depending on the GRUB configuration.
    - Now the Kernel comes up, Kernel said "okay GRUB asked me to start"
    - Kernel is the core of the operating system. What it does is that it loads the required drivers from the `init rd.img` file.
    - The kernel job is to load the driver, meaning that every hardware that is attached to your machine. in order for it to work, the Kernel has to communicate to that hardware, so in order for the Kernel to communicate, it has to have that driver that talks to that hardware, so it loads up all those hardware, and then it starts the first, very first OS process that is called `systemd`
    - `systemd` is a new process introduced in the new version of Linux
        - The last step is that `systemd` is started first as process ID # 1 by default.
        - As other processes start they get the incremental numbers of the process.
        - It then start all the required processes that were enabled at the boot time.
        - Then it reads the `/etc/systemd/system/default.target` to bring the system to the run-level
        - Every Linux system has a total of 7 run levels that go from zero to six, which defines where it's gonna start, if it needs a networking single user mode, or if it actually needs to shut down. All those run levels are actually defined in this file.
        - That's how your system becomes online.

How to Reboot/Shutdown the system

- To power off or reboot a system from the command line, you can use the `systemctl` command
- `systemctl poweroff` = stops all running services, unmounts all file systems, and then powers down the system
- `systemctl reboot` = stops all running services, unmounts all file systems, and then reboots the system
- You can also use the shorter version of these commands, `shutdown`, `poweroff` and `reboot`, which are symbolic links to their `systemctl` equivalents

Selecting `systemd` Target

- Systemd is the first Linux process which decides at which run-level the operating system needs to be in
    - run-levels exist from 0 to 6, in total there are 7 run-levels
- These run-levels are now referred to as targets
- The following table lists the most important targets

Target	Purpose
graphical.target	System supports multiple users, graphical and text based logins
multi-user.target	System supports multiple user, text-based logins only
rescue.target	sulogin prompt, bashic system initialization completed
emergancy.target	sulogin prompt, initramfs pivot complete and system root mounted on / read only

- `rescue.target` and `emergaency.target` are the targets that are given to the system if you are going to perform any maintenances, any emergencies, if you wanted to boot off of a CD, or something like that.

To check the current target or run-level

- `systemctl get-default`
- `who -r`

Example using `systemctl` command:

```
[root@localhost ~]# systemctl get-default
```

Output:

```
graphical.target
```

- It means we have both GUI and text enabled activities.

Example using `who -r` command:

```
[root@localhost ~]# who -r
```

Output:

```
         run-level 5  2024-07-08 11:41
```

- The run-level 5 is equivalent to the `graphical.target` run-level
- A target can be a part of another target. e.g., the `graphical.target` includes `multiuser.target` , which in turn depends on `basic.target` and others

View target dependencies

- `systemctl list-dependencies graphical.target | grep target`

Example using `systemctl` command:

```
[root@localhost ~]# systemctl list-dependencies graphical.target | grep target
```

Output:

```
graphical.target
● └─multi-user.target
●   ├─basic.target
●   │ ├─paths.target
●   │ ├─slices.target
●   │ ├─sockets.target
●   │ ├─sysinit.target
●   │ │ ├─cryptsetup.target
●   │ │ ├─integritysetup.target
●   │ │ ├─local-fs.target
●   │ │ ├─swap.target
●   │ │ └─veritysetup.target
●   │ └─timers.target
●   ├─getty.target
```

```
  ├─nfs-client.target
  │ └─remote-fs-pre.target
  └─remote-fs.target
    └─nfs-client.target
      └─remote-fs-pre.target
```

- As you can see the `graphical.target` has a dependency on `multi-user.target` and `multi-user.target` has at the same time a dependency on `basic.target` and so on.

Display the new runlevels/targets

- `ls -al /lib/systemd/system/runlevel*`

Example using `ls` command:

```
[root@localhost ~]# ls - al /lib/systemd/system/runlevel*
```

- Remember the `-a` option will give you ALL the files with the symbolic link as well.

Output:

```
lrwxrwxrwx. 1 root root 15 Mar 19 08:31 /lib/systemd/system/runlevel0.target -> poweroff.target
lrwxrwxrwx. 1 root root 13 Mar 19 08:31 /lib/systemd/system/runlevel1.target -> rescue.target
lrwxrwxrwx. 1 root root 17 Mar 19 08:31 /lib/systemd/system/runlevel2.target -> multi-user.target
lrwxrwxrwx. 1 root root 17 Mar 19 08:31 /lib/systemd/system/runlevel3.target -> multi-user.target
lrwxrwxrwx. 1 root root 17 Mar 19 08:31 /lib/systemd/system/runlevel4.target -> multi-user.target
lrwxrwxrwx. 1 root root 16 Mar 19 08:31 /lib/systemd/system/runlevel5.target -> graphical.target
lrwxrwxrwx. 1 root root 13 Mar 19 08:31 /lib/systemd/system/runlevel6.target -> reboot.target
...
```

- These are the 7 run-levels that exist in Linux
- You could manage the boot process of your system through these targets

Setting default target

- `systemctl set-default graphical.target`

Example using `systemctl` command:

```
[root@localhost ~]# systemctl set-default multi-user.target
```

Output:

```
Removed "/etc/systemd/system/default.target".
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/multi-user.target.
```

- Now if you reboot the system you are not gonna have that graphical view that you are looking at.

- If you run `systemctl get-default` again to check what is the current target you will now see `multi-user.target` but you will have to reboot in order to not see the GUI included in the `graphical.target` run-level

Recover root Password

Method 1

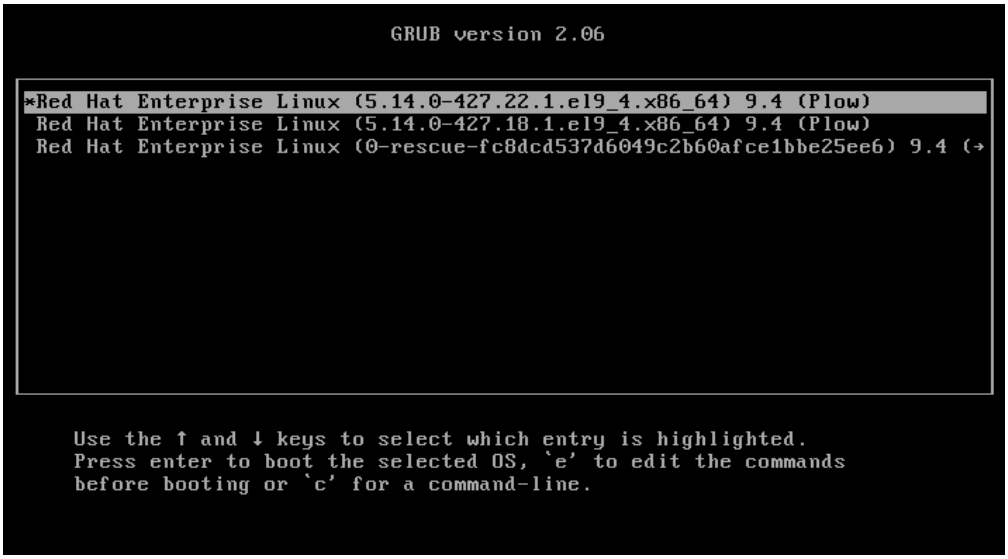
- In order to run the command `passwd root`, you have to be root, and to be root you have to put in the password. Solving this its not that simple.

Steps to recover root password:

- Restart your computer
- Edit the grub file
- Change password
- Reboot

Restarting your computer

- Run `reboot` with admin permissions



- This is how GRUB looks like
  - When you restart your computer and click any key within the 5 sec window that the system gives you you will be in GRUB Kernel selector.
- Once you are in here you could select any of the options:
  - The first one is your actual OS
  - The second one is just the rescue of your OS
- Select the first one (make sure it is highlighted in white)
  - Then press 'e' to edit.

Editing the GRUB file

- Come down to the parameters where you'll see `ro` which means "read only"
- In this case it is located in the 5th line just after `rhel-root`
  - e.g. `.../rhel-root ro ...`
- Replace the `ro` with `rw init=/sysroot/bin/sh`
- Then input the instructions
  - A good administrator always write down their troubleshooting instructions.

- Note how the `ro` was replaced with the `rw init=/sysroot/bin/sh` from the instructions
- Now hit Ctrl + x to boot in single-user mode.

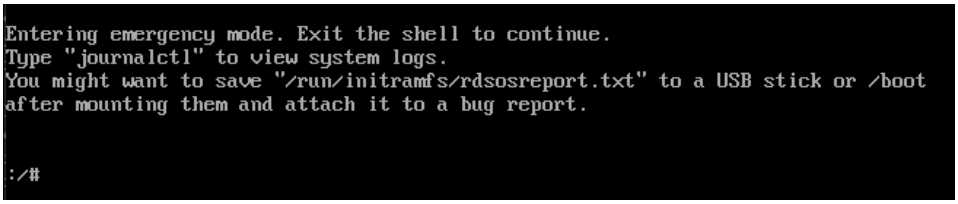
Changing the password

Instructions to change the root password:

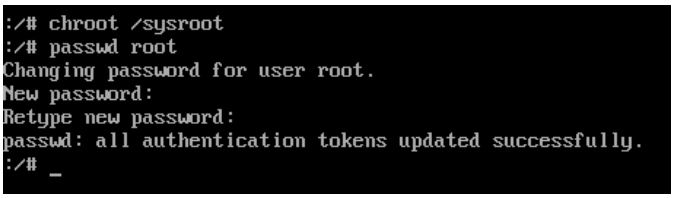
```
rw init=/sysroot/bin/sh
ctrl x
chroot /sysroot
passwd root
```

```
exit
reboot
```

- Note we already covered the first 2 steps



- This is now how the single-user mode would look like
- Now enter the commands shown in the instructions



- Your root password was changed successfully at this point
- Before exiting update SELinux information by running the command: `touch/.autorelabel`
  - In my case this line outputted: `sh: touch/.autorelabel: No such file or directory`

Now exit out of `sysroot`

- You can just type `exit` for this.

Now reboot your system

- Run the `reboot` command.

Once the system comes back online you will have the option to log in as root or as yourself and that's how you change the root password.

Method 2

from How to recover a root password...

1. Find the line that refers to the kernel: There will be a series of 'boot parameters' here: these are instructions passed during the loading of the kernel.
   1. For RHEL/CentOS 7, the line starts with 'linux16'.
   2. For RHEL/Centos 8x, and Fedora the line starts with '`linux`'.
2. Add '`rd.break`' at the end of that line (There are other things you can do here, but for now, this is all you need) **[Note: This change is temporary]**.
3. Now hit `Ctrl-x` to run the edited bootloader script.
4. You'll boot to a 'rescue' prompt that looks like this: `switch_root:/#`.
5. Remount the root partition in read-write mode so that you can run commands. Enter the following: `mount -o remount rw /sysroot` and then hit ENTER.

6. Now type `chroot /sysroot` and hit enter. This will change you into the `sysroot (/)` directory, and make that your path for executing commands.
7. Now you can simply change the password for root using the `passwd` command.
8. Next, before you reboot, you will need to make sure that SELinux allows the file changes. At the prompt ,enter: `touch /.autorelabel`. This will signal SELinux on the next reboot that the filesystem has changed (the changed password) and allow the change to be loaded. This will cause the whole filesystem to be 'relabeled' which might take a while, depending on the size of the filesystem and the speed of the machine, so be aware of this possibility.
9. Type `exit` to leave the chroot environment and enter `reboot`.

Repair Filesystem Corruption

- Filesystem corruption can occur when you either make mistakes in `/etc` configuration files or filesystem become corrupted at the disk level
- In either of the above cases the `systemd` will not be able to boot the system in the defined target and bring the system in emergency mode/target
- The following table lists some common errors and their results

Problem	Result
Corrupt filesystem	systemd attempts to repair the file system. If the problem is too severe for an automatic fix, the system drops the user to an emergency shell
Nonexistent device or UUID referenced in `/etc/fstab`	systemd waits for a set amount of time, waiting for the device to become available. If the device does not become available, the system drops the user to an emergency shell after the timeout
Nonexistent mount point in `/etc/fstab`	The system drops the user to an emergency shell
Incorrect mount option specified in `/etc/fstab`	The system drops the user to an emergency shell

- In any case administrators can use the emergency target to diagnose and fix the issue, because no file systems are mounted before the emergency shell is displayed.
- When using the emergency shell to fix filesystem issues, do not forget to run `systemctl daemonreload` after editing `/etc/fstab` . Without this reload, `systemd` may continue using the old version of the `/etc/fstab` file.
    - If you are having issues with your disk and filesystem corruption is reporting disk errors, then you could run `fsck` ("fs check") command on your filesystem, to repair your filesystem manually.
        - e.g. `fsck /dev/sda`