Linux Boot Process

Linux boot and startup

- There are two sequences of events that are required to boot a Linux computer and make it usable:
  - ➢ *boot*
  - ➢*startup*.
- The *boot* sequence starts when the computer is turned on, and is completed when the kernel is initialized and system(initd) is launched.
- The *startup* process then takes over and finishes the task of getting the Linux computer into an operational state.

Steps of Boot Process

- BIOS POST
- Boot loader (GRUB2)
- Kernel initialization
- Start systemd, the parent of all processes.(It is also called as initd in some linux old versions)

The boot process

- The boot process can be initiated in one of a couple ways.
  - ➢ if power is turned on, will begin the boot process.
  - ➢ If the computer is already running a local user, including root or an unprivileged user, the user can programmatically initiate the boot sequence by using the GUI or command line to initiate a reboot.

  A reboot will first do a shutdown and then restart the computer.

BIOS POST (Power On Self Test)

- This is the hardware portion of the boot process and is the same for any operating system. When power is first applied to the computer it runs the POST (Power On Self Test) which is part of the BIOS (Basic I/O System).
- This helps to switchon hardware components

- BIOS POST checks the basic operability of the hardware and then it issues a BIOS interrupt, INT 13H, which locates the boot sectors on any attached bootable devices.
- The first boot sector it finds that contains a valid boot record is loaded into RAM and control is then transferred to the code that was loaded from the boot sector.
- The boot sector is really the first stage of the boot loader. There are three boot loaders used by most Linux distributions, GRUB, GRUB2, and LILO.
- GRUB2 is the newest and is used frequently these days than the other older options.

MBR

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components

1) primary boot loader info in 1st 446 bytes

2) partition table info in next 64 bytes (i.e hard disk information)

3) MBR validation check in last 2 bytes.(useful for error checking)

- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

GRUB(version 2)

- GRUB2 stands for "GRand Unified Bootloader, version 2"
- it is now the primary bootloader for most current Linux distributions.
- The primary function of GRUB is to get the Linux kernel loaded into memory and running.
- If you have multiple kernel images installed on your system(multi boot system), you can choose which one to be executed.
- GRUB can also allow the user to choose to boot from among several different kernels for any given Linux distribution.
- This affords the ability to boot to a previous kernel version if an updated one fails or is incompatible with an important piece of software.
- GRUB can be configured using the /boot/grub/grub.conf file.

GRUB

- GRUB displays a splash screen, waits for few seconds, if you don’t enter anything, it loads the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn’t understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-194.el5PAE)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
        initrd /boot/initrd-2.6.18-194.el5PAE.img
```

- As you notice from the above info, it contains kernel and initd image.
- So, in simple terms GRUB just loads and executes Kernel and initd images.

Kernel

- All of the kernels are in a self-extracting, compressed format to save space. The kernels are located in the /boot directory, along with an initial RAM disk image, and device maps of the hard drives.(initramfs – which is initial file system)
- After the selected kernel is loaded into memory and begins executing, it must first extract itself from the compressed version of the file.
- Once the kernel has extracted itself, it loads system(PID 1), which is the replacement for the old SysV init program, and turns control over to it.
- This is the end of the boot process. At this point, the Linux kernel and systemd are running but unable to perform any productive tasks for the end user because nothing else is running.

Init scripts

- **Init** (short for initialization) is the program on Unix and Unix-like systems that spawns all other processes. It runs as a daemon and typically has PID 1.
- The /etc/inittab file was used to set the default run level for the system.
- This is the runlevel that a system will start up on upon reboot.
- The applications that are started by init are located in the **/etc/rc.d** folder.
- Within this directory there is a separate folder for each run level, eg *rc0.d*, *rc1.d,* and so on.
- But in new versions targets are used instead of runlevels
- Multiuser.target analogus to runlevel 3
- Graphical.target analogus to runlevel 5

Runlevels

Run Level	Mode	Action
0	Halt	Shuts down system
1	Single-User Mode	Does not configure network interfaces, start daemons, or allow non-root logins
2	Multi-User Mode	Does not configure network interfaces or start daemons.
3	Multi-User Mode with Networking	Starts the system normally.
4	Undefined	Not used/User-definable
5	X11	As runlevel 3 + display manager(X)
6	Reboot	Reboots the system

Most Linux servers lack a graphical user interface and therefore start in runlevel 3. Servers with a GUI and desktop Unix systems start runlevel 5. When a server is issued a reboot command, it enters runlevel 6.

- To find current default target the command is

```
#systemctl get-default
```

- To set default target syntax :

```
#systemctl set-default <Target.target>
```

- Example

```
#systemctl set-default Multiuser.target
```

- Multiuser.target analogus to runlevel 3
- Graphical.target analogus to runlevel 5

chkconfig

- The chkconfig tool is used in Red Hat based systems (like CentOS)
    - ➢ to control what services are started at which runlevels.
- To display a list of services

```
#chkconfig –list
```

- whether they are enabled or disabled for each runlevel

- But if the targets are used then use command

```
$systemctl list-dependencies [target]
```

```
• root@host:~ # chkconfig --list
• filelimits 0:off   1:off   2:on    3:on    4:on    5:on    6:off
• syslog 0:off       1:off   2:on    3:on    4:on    5:on    6:off
• gpm 0:off          1:off   2:on    3:on    4:on    5:on    6:off
• cpanel 0:off       1:off   2:off   3:on    4:on    5:on    6:off
• kudzu 0:off        1:off   2:off   3:on    4:on    5:on    6:off
• ntpd 0:off         1:off   2:off   3:off   4:off   5:off   6:off
• netfs 0:off        1:off   2:off   3:on    4:on    5:on    6:off
• network 0:off      1:off   2:on    3:on    4:on    5:on    6:off
• netplugd 0:off     1:off   2:off   3:off   4:off   5:off   6:off
• rawdevices 0:off           1:off   2:off   3:on    4:on    5:on    6:off
• ipchains 0:off     1:off   2:on    3:on    4:on    5:on    6:off
• iptables 0:off     1:off   2:on    3:on    4:on    5:on    6:off
• crond 0:off        1:off   2:on    3:on    4:on    5:on    6:off
• anacron 0:off      1:off   2:on    3:on    4:on    5:on    6:off
• cups 0:off         1:off   2:on    3:on    4:on    5:on    6:off
• xfs 0:off          1:off   2:on    3:on    4:on    5:on    6:off
• xinetd 0:off       1:off   2:off   3:on    4:on    5:on    6:off
• httpd 0:off        1:off   2:off   3:on    4:off   5:on    6:off
```

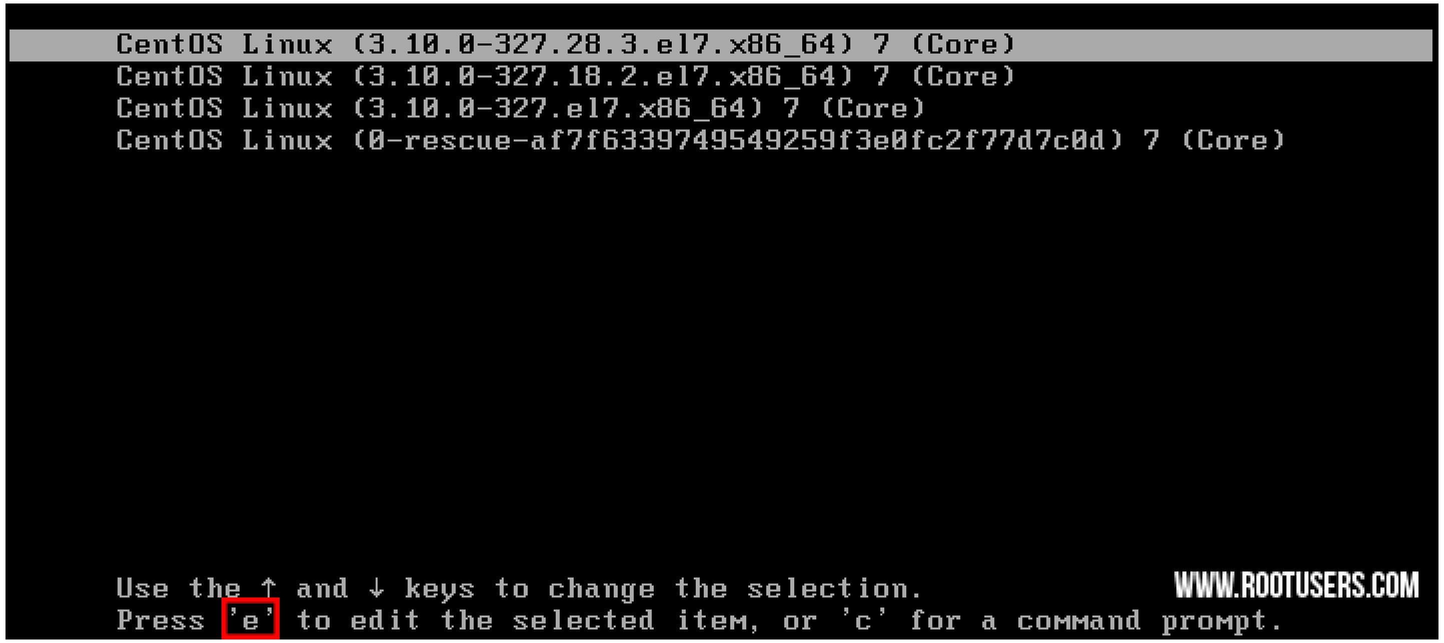
Change root password while booting

How to change root password while booting

- If some times you have forgotten the root password then it can be recovered by breaking boot process.
- But if your grub is password protected then unless you know grub password you cannot recover it
- Steps to recover the root password
- When it shows the OS options to select for booting, press ‘e’ character for editing
- This changes the mode of the system to single user mode
- Then change the root password and reboot the system

Step 1:

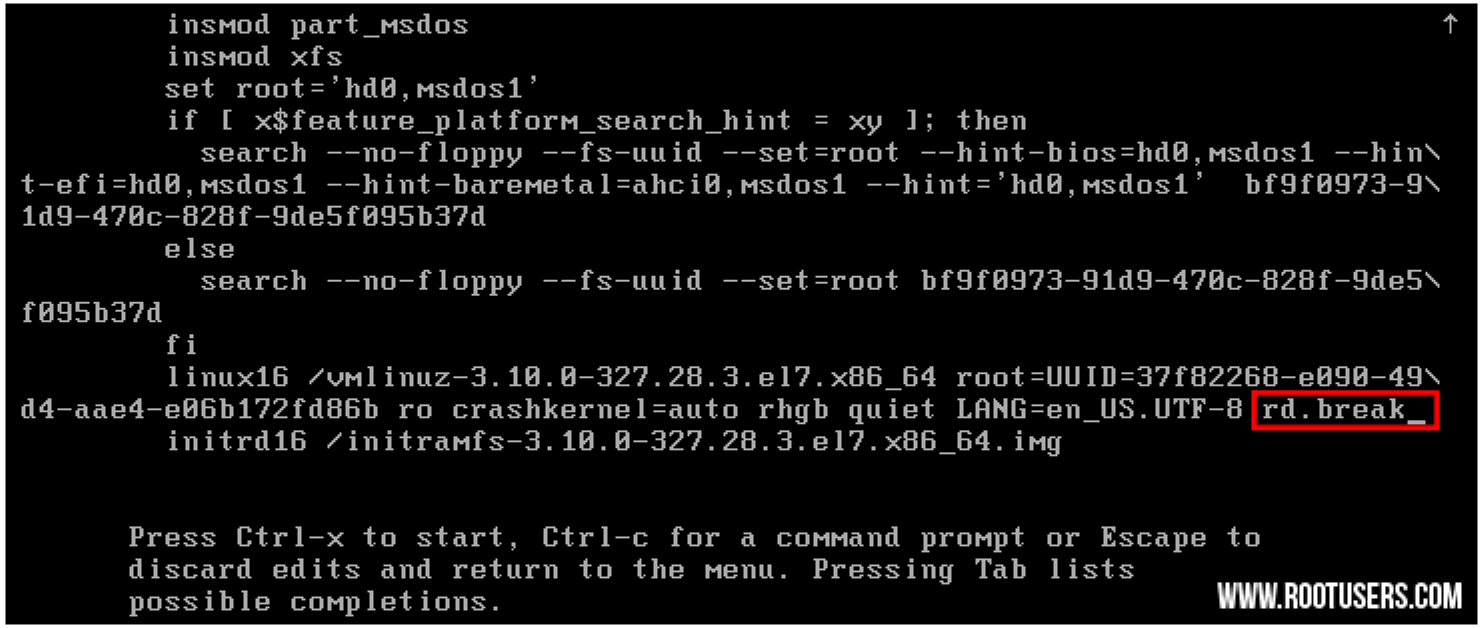
If your Linux system is currently running, reboot it. If it is not yet running, start it up. At the boot menu, press the ‘e’ key to edit the first boot entry.



```
      CentOS Linux (3.10.0-327.28.3.el7.x86_64) 7 (Core)
      CentOS Linux (3.10.0-327.18.2.el7.x86_64) 7 (Core)
      CentOS Linux (3.10.0-327.el7.x86_64) 7 (Core)
      CentOS Linux (0-rescue-af7f63397495492 59f3e0fc2f77d7c0d) 7 (Core)

      Use the ↑ and ↓ keys to change the selection.
      Press 'e' to edit the selected item, or 'c' for a command prompt.
```

Step 2:

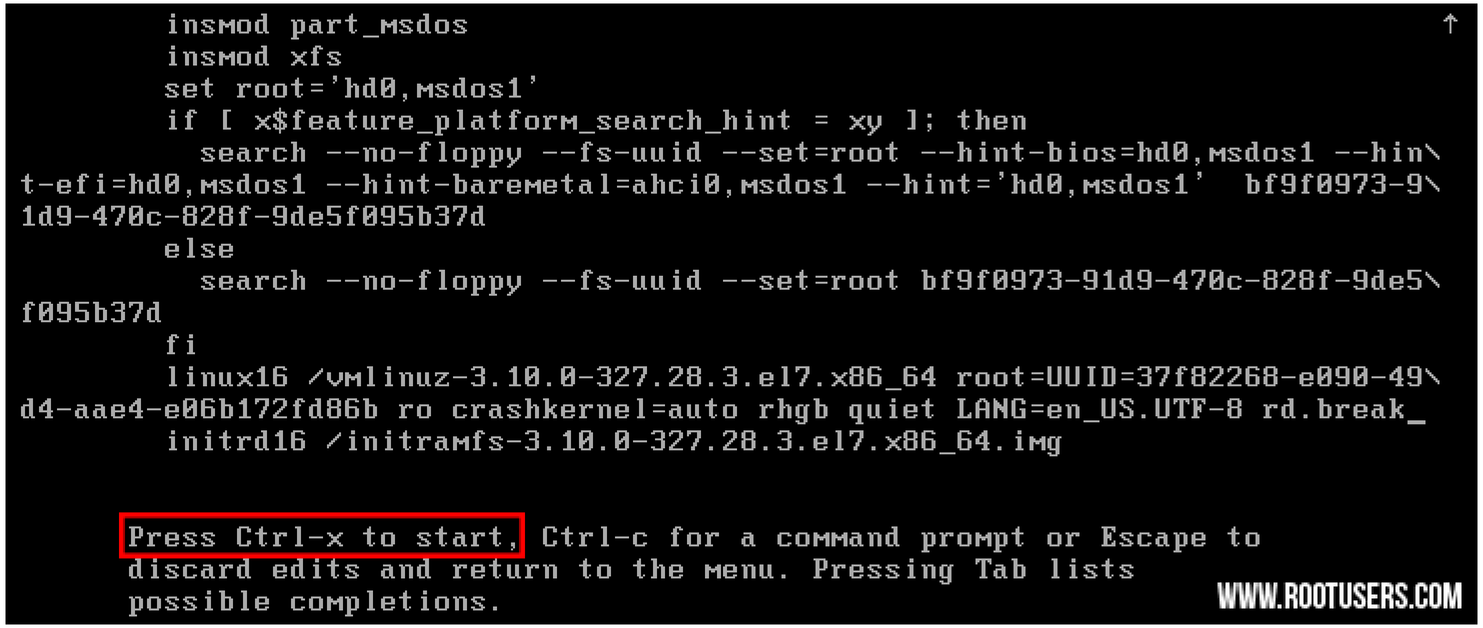
From the grub options, find the line that starts with “linux16” and go to the end of it. Enter ‘rd.break’ without quotes at the end of this line, as shown below.



```
        insmod part_msdos                                                    ↑
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hin\
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1'  bf9f0973-9\
1d9-470c-828f-9de5f095b37d
        else
          search --no-floppy --fs-uuid --set=root bf9f0973-91d9-470c-828f-9de5\
f095b37d
        fi
        linux16 /vmlinuz-3.10.0-327.28.3.el7.x86_64 root=UUID=37f82268-e090-49\
d4-aae4-e06b172fd86b ro crashkernel=auto rhgb quiet LANG=en_US.UTF-8 rd.break_
        initrd16 /initramfs-3.10.0-327.28.3.el7.x86_64.img


      Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
      discard edits and return to the menu. Pressing Tab lists
      possible completions.
```

Step 3:

Press “Ctrl+x” to boot with these options. This will boot to the initramfs prompt with a root shell.

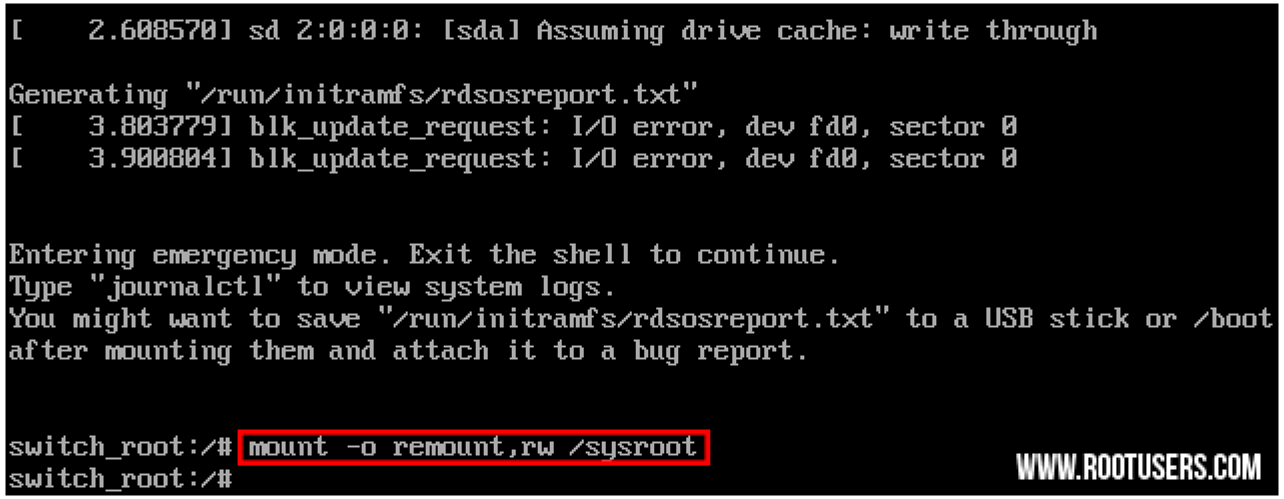


```
        insmod part_msdos                                                    ↑
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hin\
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1'  bf9f0973-9\
1d9-470c-828f-9de5f095b37d
        else
          search --no-floppy --fs-uuid --set=root bf9f0973-91d9-470c-828f-9de5\
f095b37d
        fi
        linux16 /vmlinuz-3.10.0-327.28.3.el7.x86_64 root=UUID=37f82268-e090-49\
d4-aae4-e06b172fd86b ro crashkernel=auto rhgb quiet LANG=en_US.UTF-8 rd.break_
        initrd16 /initramfs-3.10.0-327.28.3.el7.x86_64.img


      Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
      discard edits and return to the menu. Pressing Tab lists
      possible completions.
```

WWW.ROOTUSERS.COM

Step 4:

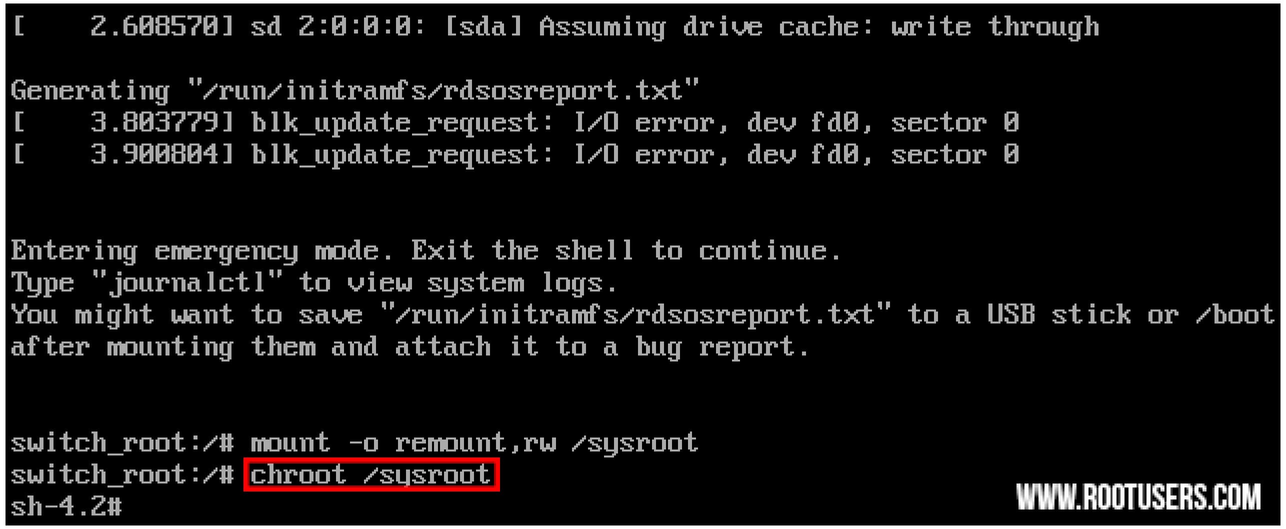
At this stage, the root file system is mounted in read only mode to /sysroot and must be remounted with read/write (rw) permissions in order for us to actually make any changes. This is done with the 'mount -o remount,rw /sysroot' command.



```
[    2.608570] sd 2:0:0:0: [sda] Assuming drive cache: write through

Generating "/run/initramfs/rdsosreport.txt"
[    3.803779] blk_update_request: I/O error, dev fd0, sector 0
[    3.900804] blk_update_request: I/O error, dev fd0, sector 0


Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.


switch_root:/# mount -o remount,rw /sysroot
switch_root:/#
```

Step 5:

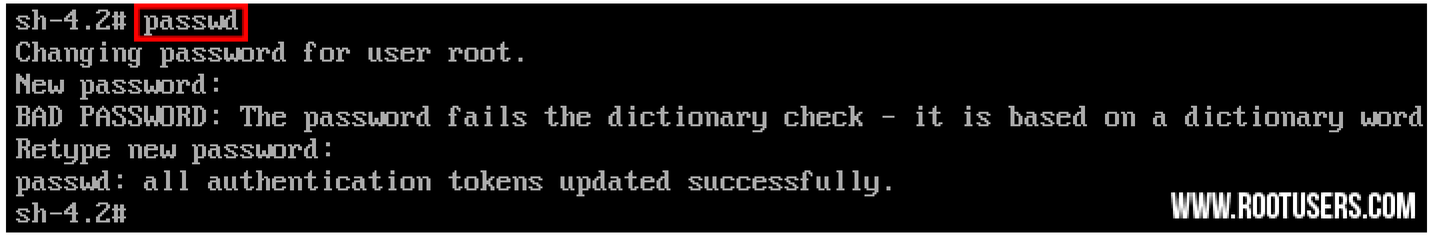
Once the file system has been remounted, change into a chroot jail so that /sysroot is used as the root of the file system. This is required so that any further commands we run will be in regards to /sysroot. This is done by running ‘chroot /sysroot’.



```
[    2.608570] sd 2:0:0:0: [sda] Assuming drive cache: write through

Generating "/run/initramfs/rdsosreport.txt"
[    3.803779] blk_update_request: I/O error, dev fd0, sector 0
[    3.900804] blk_update_request: I/O error, dev fd0, sector 0


Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.


switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
sh-4.2#
```

Step 6:

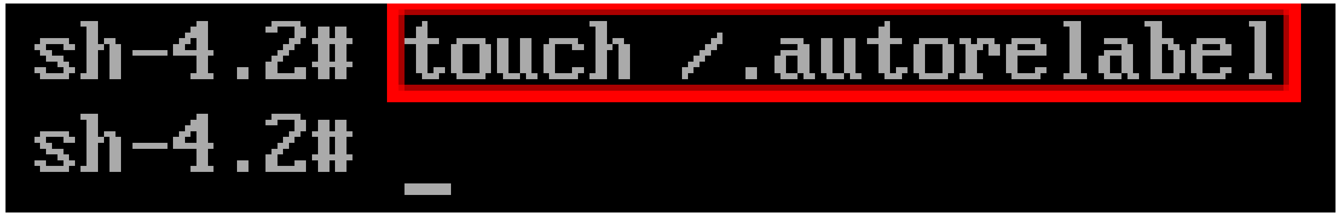
From here the root password can be reset with the ‘passwd’ command.



```
sh-4.2# passwd
Changing password for user root.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
sh-4.2#
```

Step 7:

If you’re not using SELinux, you could reboot at this point and everything would be fine, however by default CentOS/RHEL 7 use SELinux in enforcing mode, so we need to fix the context of the /etc/shadow file. This is because when the ‘passwd’ command is run, it creates a new /etc/shadow file. As SELinux is not running in this mode the file is created with no SELinux contexts, which can cause problems when we reboot. Create the /.autorelabel command using ‘touch’.

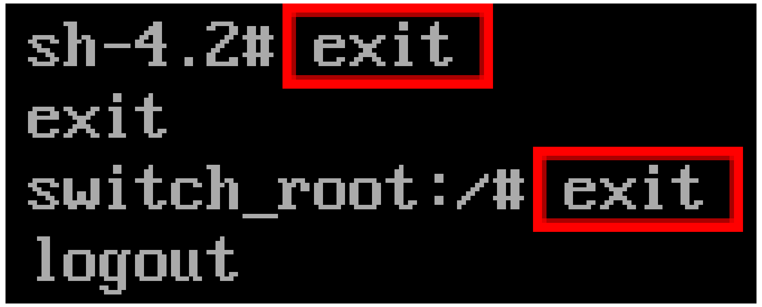


```
sh-4.2# touch /.autorelabel
sh-4.2# _
```

Creating this file will automatically perform a relabel of all files on next boot. Note that this may take some time depending on the amount of files you have on the file system. For a plain vanilla CentOS 7 server, it takes me about 2 minutes to complete.

Step 8:

Enter the ‘exit’ command twice, the first one will exit the chroot jail environment while the second will exit the initramfs root shell and reboot the system.



```
sh-4.2# exit
exit
switch_root:/# exit
logout
```

Once the reboot has completed you will be able to use the root account with your newly set password.

Login sequence

Linux Login sequence

- The login sequence
- It's worth understanding at least those files in the login sequence that will affect things like
- whether you can login in the first place,
- where your initial environment comes from, and so on.

- Following are the files you need to worry about that are invoked/consulted every time you log in (indented in an attempt to show who actually consults who):

- /etc/passwd 1
- /etc/shadow 2
- /etc/group 3

- /etc/profile 4
- /etc/profile.d/*.sh 5

- ~/.bash_profile 6
- ~/.bashrc 7
- /etc/bashrc 8

1. /etc/passwd defines, among other things, the username, numeric UID and GID of the user, the home directory and the login shell. For security reasons, the encrypted password is no longer stored in this file.
2. /etc/shadow represents a more secure place to store things like the encrypted password, password aging information and more. (This is the only file in this list of files that requires root privilege to display.)
3. /etc/group defines the working groups on the host, along with the users who are members of those groups.
4. /etc/profile represents the top-level startup/customization file that is invoked by each user.
5. In addition, this file may, toward the bottom, invoke all of the customization files in the directory /etc/profile.d, which typically represent a small amount of customization on a per-application or per-command basis. If you're confused about where some login environment setting is coming from that seems to be affecting the behaviour of a particular command, this is a good place to look.
6. These two startup files in the user's home directory are for configuring the user's personal environment. In addition, on some Linux systems, the default .bashrc may invoke the global /etc/bashrc file, for one final dose of environment configuration and customization.

- As a final helpful hint, once you get logged in, there are two handy commands that can tell you everything there is to know about your identity and group affiliations:
- $ id
- $ groups

Kernel modules

Add, list, Remove kernel modules

- A kernel module is a program which can be loaded into or unloaded from the kernel upon demand, <u>without necessarily recompiling</u> (the kernel) or rebooting the system, and is intended to enhance the functionality of the kernel.
- If modules are not there , then the kernel would have to be built with all functionalities integrated directly into the kernel image.
- This would mean having bigger kernels, and system administrators would need to recompile the kernel every time a new functionality is needed.
- These are like plugins in windows
- Example – device driver

List All Loaded Kernel Modules in Linux

- In Linux, all modules end with the .ko extension, and they are normally loaded automatically as the hardware is detected at system boot.
- However a system administrator can manage the modules using certain commands.
- Usually, all Linux kernel modules (drivers) are stored in the module directory located that /lib/modules/$(uname -r) directory. To see current modules, type:

```
$ ls /lib/modules/$(uname -r)
```

- To list all currently loaded modules in Linux, we can use the **lsmod** (list modules) command which reads the contents of /proc/modules like this.

```
$lsmod
$lsmod | grep e1000   ----- will check for e1000 module is there or not
```

modinfo command

- modinfo stand for ‘module information‘. This command will show the information about a kernel module. For instance you want to view the information regarding network driver module:

```
#modinfo e100
```

Output of modinfo command clearly show version of this module, description which is showing the manufacture factory, license is GPL and other important information.

- Linux maintains kernel module directory under
- /lib/modules/’uname -r’/kernel/drivers/and configuration files(except for additional configuration file in **/etc/modprobe.d/).** If we want to look at kernel drivers then run the following command.
- ls /lib/modules/`uname –r`/kernel/drivers/

- uname –r ---- this command shows the current kernel version. Your system may contains multiple kernel version
- ls /lib/modules/3.10.0-327.el7.x86_64/kernel/drivers/

Load and unload module

- To load a kernel module, we can use the insmod (insert module) command. Here, we have to specify the full path of the module. The command below will insert the speedstep-lib.ko module.
- # insmod /lib/modules/4.4.0-21-generic/kernel/drivers/cpufreq/speedstep-lib.ko

To Unload the module

- To unload a kernel module, we use the rmmod (remove module) command. The following example will unload or remove the speedstep-lib.ko module.

```
# rmmod /lib/modules/4.4.0-21-
generic/kernel/drivers/cpufreq/speedstep-lib.ko
```

How to Manage Kernel Modules Using modprobe Command

- modprobe is an intelligent command for listing, inserting as well as removing modules from the kernel.
- It searches in the module directory /lib/modules/$(uname -r) for all the modules and related files, but excludes alternative configuration files in the /etc/modprobe.d directory.
- so you don’t need the absolute path of a module; this is the advantage of using modprobe over the previous commands.
- To insert a module, simply provide its name as follows. It will search it

```
# modprobe speedstep-lib
```

To remove module using modprobe

- To remove a module, use the -r flag like this.

```
# modprobe -r speedstep-lib
```

- Note: Under modprobe, automatic underscore conversion is performed, so there is no difference between _ and – while entering module names.

- If we still face problems or errors while loading the modules, then this time we must do debugging.
- By enabling debugging we can find exact error or issue before or after installing the modules.

- ‘-n’ option in the modprobe command can enable this type of debugging. This option will force modprobe command to perform all module loading steps except the final one.

```
#modprobe -vn 'module_name'
```

- To see dependency of the module with using ‘–show-depends‘ in the modprobe command, example is shown below

```
# modprobe --show-depends e1000
#insmod /lib/modules/3.10.0-
327.el7.x86_64/kernel/drivers/net/ethernet/intel/e1000/e1000.ko
```

Forceful loading of module

- If we get any issues while loading the modules or modules not loaded properly.
- To overcome these errors add or load modules forcefully using

‘–force’ option (-f) in the modprobe command.

- #modprobe -f floppy

Creating Yum repository on centos 7

- There are several overheads involved in installing a program in a computer Like the following.

- Is the program compatible with my computer architecture.
- Is that program compatible with my operating system version.
- Does all the programs and libraries required to run a certain program there in the system
- Will the newly installed program conflict with an already installed program

- An installer or program manager, must handle those overhead by itself by not harassing the user.
- Linux, by nature is the best operating system out there(if you configured it the right way).
- The main issue with installing a program's in Linux distribution's is the fact that, different distribution's use different methods to install a program.
- Here, in this post we will be discussing a very famous tool used to install program's in Red Hat Linux system's(even fedora,centos and all red hat like system's). That's none other than the very famous YUM.

Introduction to YUM

- YUM stands for Yellowdog Updater, Modified. Like all other program's in Linux, YUM is also an open source tool.
- It was initially used in Duke University, for managing package installation on their Red Hat based system's.
- These day's its been widely used by almost all Red Hat based system's.
- In fact its the default program installer and package management tool these days.
- visiting the below link of Duke university.

- http://linux.duke.edu/projects/yum/

- What are packages in Linux?
- Red hat Linux,Fedora & all other red hat based distributions uses RPM as their main software installation package tool.
- A Linux software package is nothing but a compressed archive of files, consisting of following, which enables the functioning of that software package.
  - particular product information
  - program files
  - Icons
  - libraries etc.
- RPM is the default package installation tool used in Red Hat Linux. RPM stands for Red Hat Package Manager.

- All required files of an application is compiled in a single file format called with a file extension of .rpm.
- The Red Hat package manager tool, which is installed in all RPM based system, knows how to open and install these .rpm files in the system
- .rpm files are compiled and are ready to install, they also perform a dependency prerequisite check for all required libraries before installing a particular package.

RPM and YUM

- So if RPM is already there, why was YUM made?
- RPM and YUM are completely two different things.
- RPM is the package manager tool which installs the package.
- YUM is a repository management tool which will fetch the appropriate package for your particular version of Linux(along with all other required packages).
- Repositories is an organized collection of packages that YUM uses.
- YUM can use these repositories to fetch the correct and exact version of a particular package compatible for your system.
- Previously before YUM(or before the existence of such repository management tools), the user had to fetch the rpm package for installation, and if a dependency problem arises, the user had to fetch those dependencies from internet or some other sources.

- YUM will contain the URL's(Uniform Resource Locators) of different repositories in its configuration files.
- You can in fact update all the installed applications on your system, with the help of a single YUM command(yum will fetch different packages from appropriate different repositories.)

Create YUM repository step by step

Network configuration

- Linux easily manages multiple network interface adapters.
- Laptops typically include both wired and wireless interfaces, and may also support WiMax interfaces for cellular networks.
- Linux desktop computers also support multiple network interfaces, and you can use your Linux computer as a multi-network client, or as a router for internal networks

Interface configuration files

- Every network interface has its own configuration file in the directory
  /etc/sysconfig/network-scripts.
- Each interface has a configuration file named **ifcfg-<interface-name>X**, where X is the number of the interface, starting with zero or 1 depending upon the naming convention in use; for example /etc/sysconfig/network-scripts/ifcfg-eth0 for the first Ethernet interface.
- Most of the other files in the /etc/sysconfig/network-scripts directory are scripts used to start, stop and perform various network configuration activities.
- Each interface configuration file is bound to a specific physical network interface by the MAC address of the interface.

Network interface naming conventions

- The naming conventions for network interfaces used to be simple, uncomplicated,
- Using ethX made sense to me and was easy to type. It did not require extra steps to figure out what long and obscure name belonged to an interface.
- Unfortunately, adding a new interface often forced the renaming of network interfaces, which broke scripts and configurations. That has all changed, more than once.

- After a short stint with some long and unintelligible network interface names, we now have a third set of naming conventions which seem only marginally better. Names like ethX are still used by CentOS 6.x. The newest naming conventions, with names like eno1 and enp0s3 are used by RHEL 7, CentOS 7, and more recent releases of Fedora. The interface naming convention for RHEL 6 and CentOS 6 is described in Appendix A. Consistent Network Device Naming of the Red Hat Deployment Guide. RHEL 7, CentOS 7, and current versions of Fedora are described in Chapter 8. Consistent Network Device Naming of the Red Hat Networking Guide.

Configuration file examples

- This example network interface configuration file, ifcfg-eth0, defines a static IP address configuration for a CentOS 6 server installation.
- This file starts the interface on boot, assigns it a static IP address, defines a domain and network gateway, specifies two DNS servers, and does not allow non-root users to start and stop the interface.

- # Intel Corporation 82566DC-2 Gigabit Network Connection
- DEVICE=eth0
- HWADDR=00:16:76:02:BA:DB
- ONBOOT=yes
- IPADDR=192.168.0.10
- BROADCAST=192.168.0.255
- NETMASK=255.255.255.0
- NETWORK=192.168.0.0
- SEARCH="example.com"
- BOOTPROTO=static
- GATEWAY=192.168.0.254
- DNS1=192.168.0.254
- DNS2=8.8.8.8
- TYPE=Ethernet
- USERCTL=no
- IPV6INIT=no

- The following interface configuration file, ifcfg-eno1, provides a DHCP configuration for a desktop workstation.
- the DHCP entries, IP address, the search domain, and all other network information are not defined because they are supplied by the DHCP server. Configuration items like the DNS servers can be overridden in the interface configuration file by adding DNS1 and DNS2 lines, as in the previous static configuration example.

- TYPE=Ethernet
- BOOTPROTO=dhcp
- DEFROUTE=yes
- IPV4_FAILURE_FATAL=no
- IPV6INIT=no
- IPV6_AUTOCONF=no
- IPV6_DEFROUTE=no
- IPV6_FAILURE_FATAL=no
- NAME=eno1
- UUID=a67804ff-177a-4efb-959d-c3f98ba0947e
- ONBOOT=yes
- HWADDR=E8:40:F2:3D:0E:A8
- IPV6_PEERDNS=no
- IPV6_PEERROUTES=no

- Configuration options
- There are many configuration options for the interface configuration files. These are some of the more common options:

- DEVICE: The logical name of the device, such as eth0 or enp0s2.
- HWADDR: The MAC address of the NIC that is bound to the file, such as 00:16:76:02:BA:DB
- ONBOOT: Start the network on this device when the host boots. Options are yes/no. This is typically set to "no" and the network does not start until a user logs in to the desktop. If you need the network to start when no one is logged in, set this to "yes".
- IPADDR: The IP Address assigned to this NIC such as 192.168.0.10
- BROADCAST: The broadcast address for this network such as 192.168.0.255
- NETMASK: The netmask for this subnet such as the class C mask 255.255.255.0
- NETWORK: The network ID for this subnet such as the class C ID 192.168.0.0
- SEARCH: The DNS domain name to search when doing lookups on unqualified hostnames such as "example.com"

- Configuration options
- BOOTPROTO: The boot protocol for this interface. Options are static, DHCP, bootp, none. The "none" option defaults to static.
- GATEWAY: The network router or default gateway for this subnet, such as 192.168.0.254
- ETHTOOL_OPTS: This option is used to set specific interface configuration items for the network interface, such as speed, duplex state, and autonegotiation state. Because this option has several independent values, the values should be enclosed in a single set of quotes, such as: "autoneg off speed 100 duplex full".
- DNS1: The primary DNS server, such as 192.168.0.254, which is a server on the local network. The DNS servers specified here are added to the /etc/resolv.conf file when using NetworkManager, or when the peerdns directive is set to yes, otherwise the DNS servers must be added to /etc/resolv.conf manually and are ignored here.
- DNS2: The secondary DNS server, for example 8.8.8.8, which is one of the free Google DNS servers. Note that a tertiary DNS server is not supported in the interface configuration files, although a third may be configured in a non-volatile resolv.conf file.
- TYPE: Type of network, usually Ethernet. The only other value I have ever seen here was Token Ring but that is now mostly irrelevant.
- PEERDNS: The yes option indicates that /etc/resolv.conf is to be modified by inserting the DNS server entries specified by DNS1 and DNS2 options in this file. "No" means do not alter the resolv.conf file. "Yes" is the default when DHCP is specified in the BOOTPROTO line.
- USERCTL: Specifies whether non-privileged users may start and stop this interface. Options are yes/no.
- IPV6INIT: Specifies whether IPV6 protocols are applied to this interface. Options are yes/no.

- If the DHCP option is specified, most of the other options are ignored. The only required options are BOOTPROTO, ONBOOT and HWADDR. Other options that you might find useful, that are not ignored, are the DNS and PEERDNS options if you want to override the DNS entries supplied by the DHCP server.