CL-2006 Operating System

LAB - 07

System Configuration, Bootloader, Runlevels, Software Installation & System Services

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
Spring 2023

<u>Objective</u>

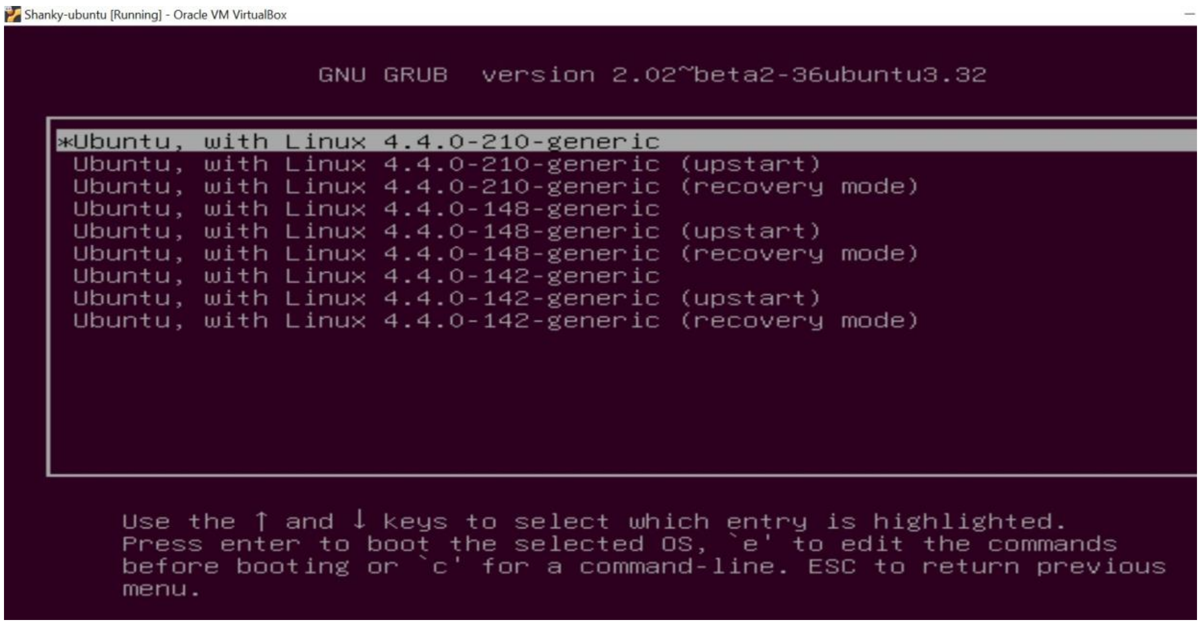
The objective of this lab is to gain a perspective and control over your operating system. This will involve managing system configuration, services, bootloader and run levels along with software installation.

<u>System Configuration/Bootloader</u>

The GRUB Bootloader is the default Ubuntu bootloader and most commonly used bootloader for Linux. Using GRUB, we can modify the startup of the system according to the desired output. This has many applications like selecting a specific kernel, or setting the default operating system and even setting up “dual-boot”.

Here are the steps to select a kernel using a VM in Oracle VirtualBox:

1. Log in directly to the Ubuntu device console using Oracle VirtualBox.
2. Reboot your Ubuntu machine using the shutdown command (shutdown -r now).
3. Once your device restarts, press and hold Shift while loading Ubuntu GRUB, and you will see the GRUB bootloader menu. If you don’t select anything, GRUB will boot the system with the default kernel and proceed with the booting process.
4. You should select Advanced options for Ubuntu either when you need to boot with a different kernel other than the default kernel. Or if you face any issues within the system like file corruption, kernel not loading correctly.
5. After you select the Advanced options for Ubuntu, a Ubuntu GRUB menu will show up. In the menu, you should see various kernels options you choose. If you don’t select an option, the first displayed kernel will load by default.
6. As you can see below in the image, there are two additional options for every kernel: upstart and recovery:
    a. (upstart): an event-based replacement for the /sbin/init daemon, which handles starting tasks and services during boot, stopping them during the shutdown, and supervising them while the system is running.
    b. (recovery mode): to recover the kernel from any failures in the system.



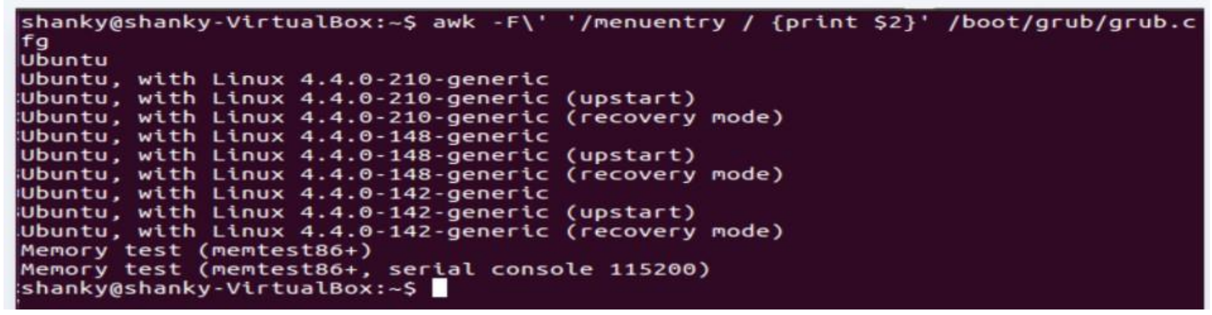
```
                     GNU GRUB  version 2.02~beta2-36ubuntu3.32

 *Ubuntu, with Linux 4.4.0-210-generic
  Ubuntu, with Linux 4.4.0-210-generic (upstart)
  Ubuntu, with Linux 4.4.0-210-generic (recovery mode)
  Ubuntu, with Linux 4.4.0-148-generic
  Ubuntu, with Linux 4.4.0-148-generic (upstart)
  Ubuntu, with Linux 4.4.0-148-generic (recovery mode)
  Ubuntu, with Linux 4.4.0-142-generic
  Ubuntu, with Linux 4.4.0-142-generic (upstart)
  Ubuntu, with Linux 4.4.0-142-generic (recovery mode)

      Use the ↑ and ↓ keys to select which entry is highlighted.
      Press enter to boot the selected OS, `e' to edit the commands
      before booting or `c' for a command-line. ESC to return previous
      menu.
```

7. Once logged into the device, the uname command verifies if the specified kernel is loaded. The -r option prints the kernel name.

The following steps can be followed to setup the default operating system:

1. List the GRUB boot menu entries by running the awk command as shown below. The awk command will display all the entries stored in /boot/grub/grub.cfg. The contents of the file will show up on the terminal. Pay attention to the lines. You will need the number associated with the line ahead, which starts with zero, i.e. the Ubuntu line is 0 and Ubuntu, with Linux 4.4.0-210-generic is 1.

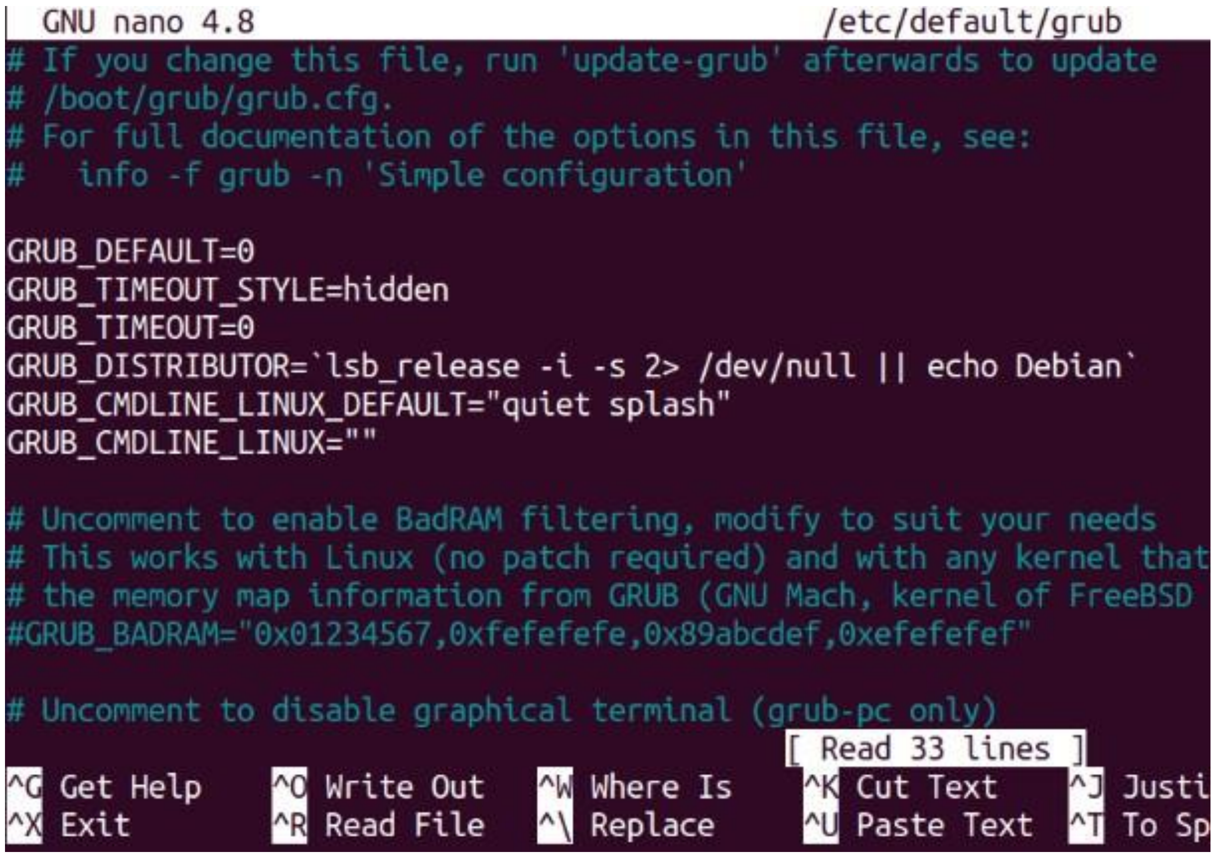
```
awk -F\' '/menuentry / {print $2}' /boot/grub/grub.cfg
```



```
shanky@shanky-VirtualBox:~$ awk -F\' '/menuentry / {print $2}' /boot/grub/grub.c
fg
Ubuntu
Ubuntu, with Linux 4.4.0-210-generic
Ubuntu, with Linux 4.4.0-210-generic (upstart)
Ubuntu, with Linux 4.4.0-210-generic (recovery mode)
Ubuntu, with Linux 4.4.0-148-generic
Ubuntu, with Linux 4.4.0-148-generic (upstart)
Ubuntu, with Linux 4.4.0-148-generic (recovery mode)
Ubuntu, with Linux 4.4.0-142-generic
Ubuntu, with Linux 4.4.0-142-generic (upstart)
Ubuntu, with Linux 4.4.0-142-generic (recovery mode)
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)
shanky@shanky-VirtualBox:~$ 
```

2. Edit the grub config file /etc/default/grub using your preferred text editor (nano, vi, gedit), set the value of GRUB_DEFAULT to the line number of the desired OS entry. The number will be according to the output of the awk command ran above. If you set GRUB_DEFAULT to 0, the first OS entry in the GRUB boot menu entry will boot. If you set it to 1, the device boots

the second OS entry by default, and so on. In the screenshot below, the first OS Entry Ubuntu is set to boot by default.

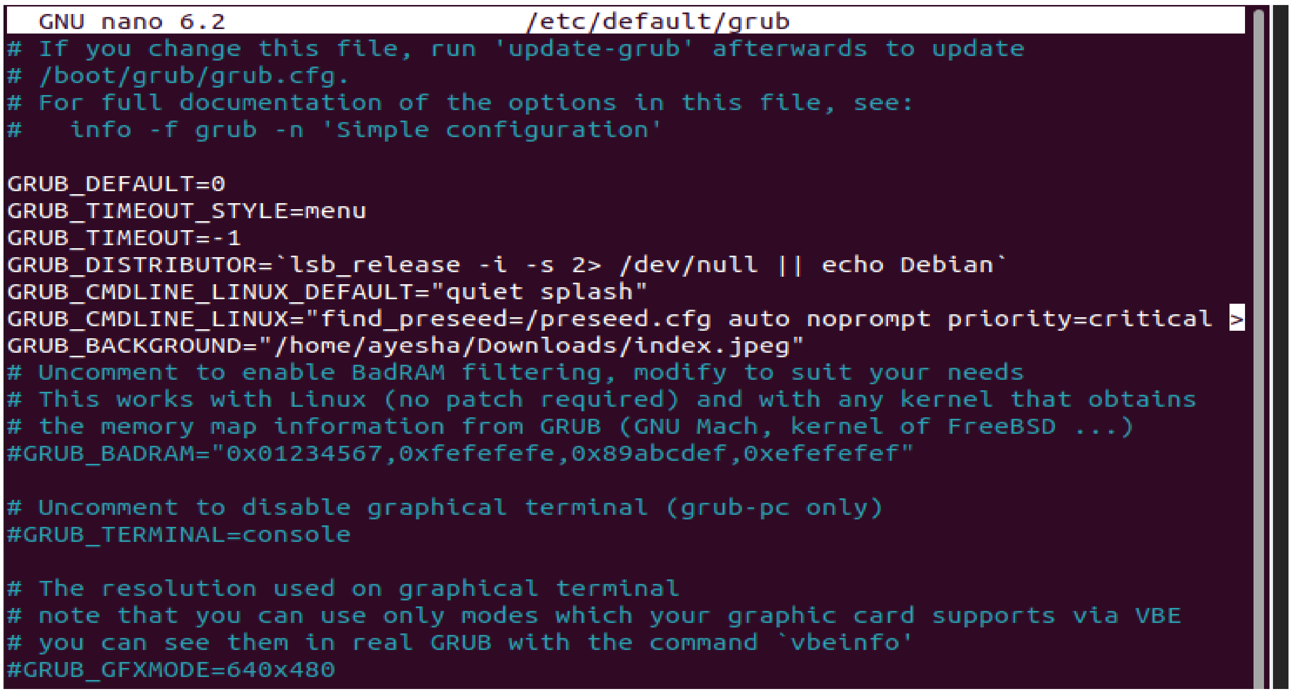


```
  GNU nano 4.8                                    /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
                                              [ Read 33 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justi
^X Exit         ^R Read File    ^\ Replace      ^U Paste Text   ^T To Sp
```

In the above image, you can see the default presets. Some information about these:

- **GRUB_DEFAULT** will set the default operating system
- **GRUB_TIMEOUT_STYLE** if set to hidden means it will not display the boot menu until the boot menu key is pressed, if it is set to “menu”, then it will display the Grub boot menu.
- **GRUB_TIMEOUT** sets the default boot delay. If set to 0, it means the operating system will be booted immediately.

3. Once you save the changes in the GRUB configuration file, make sure to run the **update-grub** command. The **update-grub** command takes the changed configuration file and applies it to the system.



```
  GNU nano 6.2                        /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=-1
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt priority=critical >
GRUB_BACKGROUND="/home/ayesha/Downloads/index.jpeg"
# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480
```

Here:

- **GRUB_TIMEOUT_STYLE=menu**
- **GRUB_TIMEOUT=-1**

Add another line as:

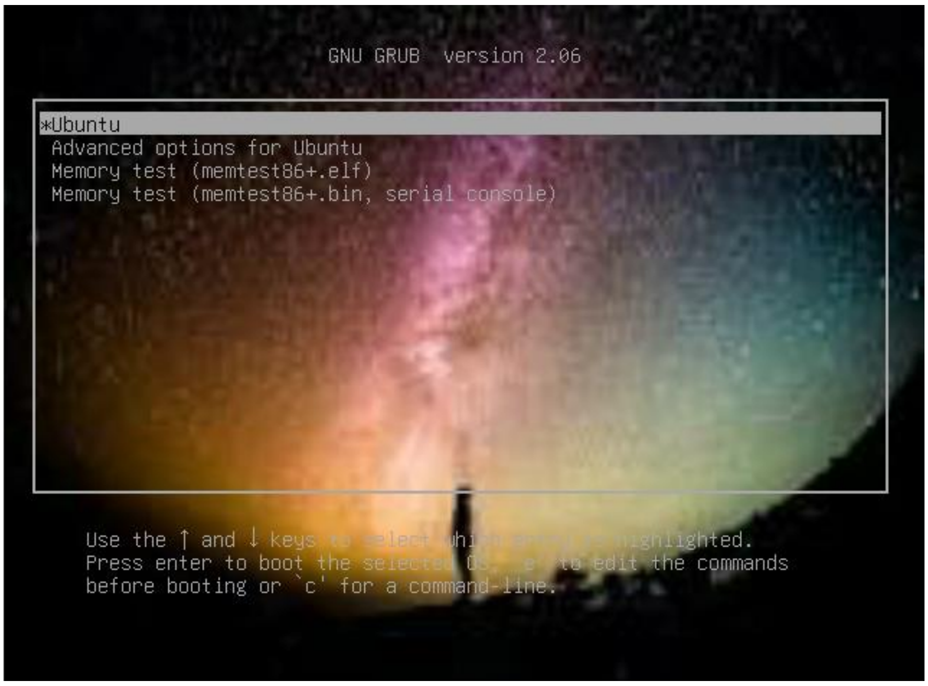
GRUB_BACKGROUND=”your image path”

Then save and exit the file by CTL+S, CTL+X

Write

- **sudo update-grub**
- **sudo reboot**

Your changes in the grub file will reflect on reboot,



```
                    GNU GRUB  version 2.06

 *Ubuntu
  Advanced options for Ubuntu
  Memory test (memtest86+.elf)
  Memory test (memtest86+.bin, serial console)

      Use the ↑ and ↓ keys to select which entry is highlighted.
      Press enter to boot the selected OS, `e' to edit the commands
      before booting or `c' for a command-line.
```

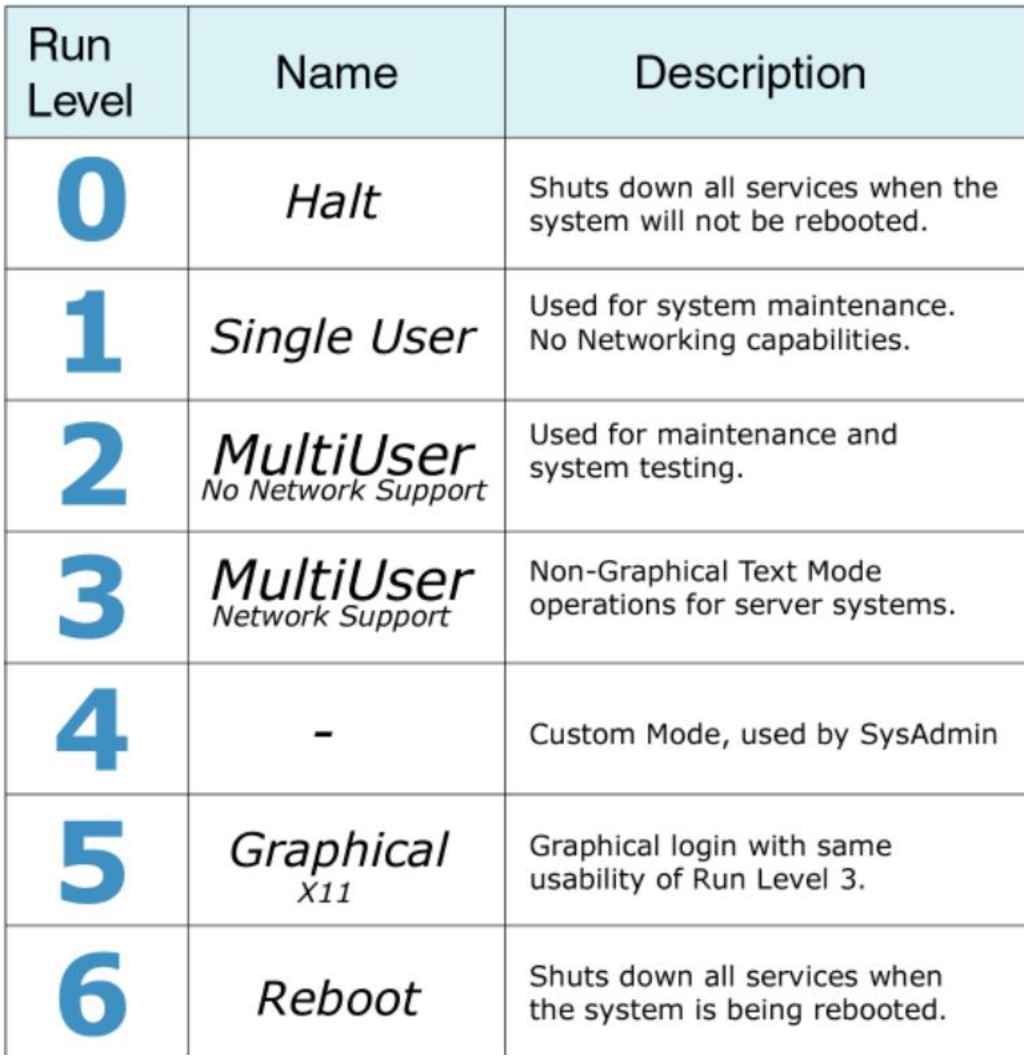
<u>RUNLEVELS IS LINUX:</u>

A runlevel is an operating state on a on the <u>Linux</u>-based system. Runlevels are numbered from zero to six.

Runlevels determine which programs can execute after the <u>OS</u> boots up. The runlevel defines the state of the machine after boot.

Systems administrators set the <u>default</u> runlevel of a system according to their needs, or use the runlevel command to find out the machine's current runlevel to assess a system. For example, the runlevel can indicate whether or not the system's network is operational. Use the runlevel command /sbin/runlevel to find the current and previous runlevel of an operating system.

Each basic level has a different purpose. Runlevels 0, 1, 6 are always the same. Runlevels 2 to 5 are different depending upon the Linux distribution in use. Only one runlevel is executed when the system is booted. They are not implemented sequentially. For example, either runlevel 4 or 5 or 6 is executed, not 4 then 5 then 6.



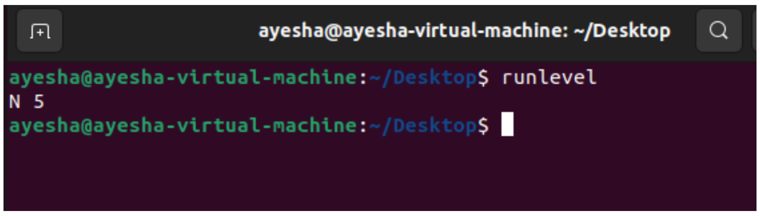
Run Level	Name	Description
0	*Halt*	Shuts down all services when the system will not be rebooted.
1	*Single User*	Used for system maintenance. No Networking capabilities.
2	*MultiUser* *No Network Support*	Used for maintenance and system testing.
3	*MultiUser* *Network Support*	Non-Graphical Text Mode operations for server systems.
4	-	Custom Mode, used by SysAdmin
5	*Graphical* *X11*	Graphical login with same usability of Run Level 3.
6	*Reboot*	Shuts down all services when the system is being rebooted.

Users can modify the present runlevels or even create new ones if needed. Runlevel 4 is typically for user-defined runlevels.

Booting a system into different runlevels solves certain problems. For example, if a machine fails to boot due to a damaged configuration file, refuses to allow the user to log in due to a corrupted /etc/passwd file or if you forget your password, you can solve these problems by booting into single-user mode.

LINUX COMMANDS TO CHECK & CHANGE RUNLEVELS:

- type “runlevel” command in terminal to check your run level



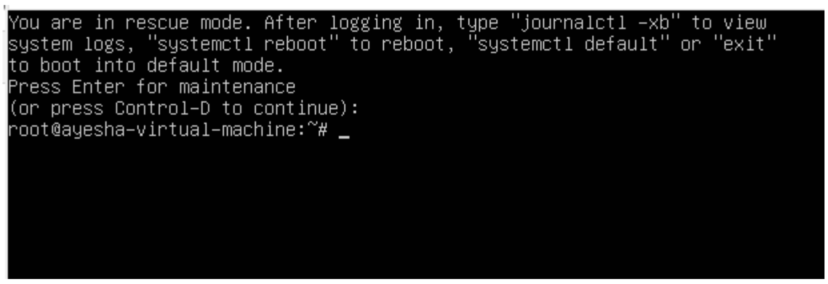
```
ayesha@ayesha-virtual-machine:~/Desktop$ runlevel
N 5
ayesha@ayesha-virtual-machine:~/Desktop$
```

- To temporarily change the runlevel on your Ubuntu Linux system use **telinit** or directly invoke **init** command.

  The following linux command will change runlevel to **1**:

  ```
  # init 1
  OR
  # telinit 1
  ```

  It will take you to single user non graphical mode:



```
You are in rescue mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or "exit"
to boot into default mode.
Press Enter for maintenance
(or press Control-D to continue):
root@ayesha-virtual-machine:~# _
```

- Now change to a default runlevel on Ubuntu Linux is to edit Grub's default startup sequence. Locate **/etc/default/grub** file and edit line, type:

  sudo nano /etc/default/grub

  find this line:

  ```
  GRUB_CMDLINE_LINUX=""
  ```

To include your desired runlevel. For example, to change default runlevel to **5** edit the above line to desired runlevel. For example, to change to runlevel 5 insert:

```
      GRUB_CMDLINE_LINUX="5"
```

and run Grub update command with administrative privileges:

```
# update-grub

# sudo reboot
```

After reboot the system will now boot to runlevel 5 with graphical user interface.

Installing Software on Linux using code

Software on Linux can be installed through a few various means. Sometimes it’s most appropriate to directly use the terminal, however using package managers can also be quite user friendly. In this tutorial, we will demonstrate how to install any software on Linux which is in a tarball format. Tarball is a term which refers basically to a compressed format of a lot of files, the naming scheme of a tarball can vary, for example TGZ, TBZ, TXZ, TZST, or with double file extension, e.g. TAR.GZ, TAR.BR, TAR.BZ2, TAR.XZ, TAR.ZST.

The following steps can be followed to install any software on Linux in tarball format:

1. By default, Ubuntu (or Debian-based Linux distributions) does not come with the tools required. You need to install the package build-essential for making the package and checkinstall for putting it into your package manager. These can be found on the install CD or in the repositories, searching in Synaptic Package Manager or the command-line apt-get Run the following series of commands to prepare your system for the upcoming software installation:
    - **sudo apt-get install build-essential checkinstall**
    - **sudo apt-get install cvs subversion git-core mercurial**
    - **sudo chown $USER /usr/local/src**
    - **sudo chmod u+rwx /usr/local/src**
2. You must uncompress the tar file using tar command. Assuming the file is named as “filename.tgz”, to uncompress it the command would be:

    tar xfvz filename.tgz

3. After this, executing the 3 popular commands i.e. configure, make & make install will complete the software installation process:
    - **./configure**
    - **make**
    - **make install**

Let’s study in detail what these 3 commands do exactly:

1. The **configure** script basically consists of many lines which are used to check some details about the machine on which the software is going to be installed. This script checks for lots of dependencies on your system. For the particular software to work properly, it may be requiring a lot of things to be existing on your machine already.

    When you run the configure script you would see a lot of output on the screen , each being some sort of question and a respective yes/no as the reply. If any of the major requirements are missing on your system, the configure script would exit and you cannot proceed with the installation, until you get those required things. ./configure may fail if it finds that dependencies are missing.

    The main job of the configure script is to create a Makefile. This is a very important file for the installation process. Depending on the results of the tests (checks) that the configure script performed it would write down the various steps that need to be taken (while compiling the software) in the file named Makefile.

2. **make** is actually a utility which exists on almost all Unix systems. For make utility to work it requires a file named Makefile in the same directory in which you run make. As we have seen the configure script's main job was to create a file named Makefile to be used with make utility. (Sometimes the Makefile is named as makefile also)

   make would use the directions present in the Makefile and proceed with the installation. The Makefile indicates the sequence, that Linux must follow to build various components / sub-programs of your software. The sequence depends on the way the software is designed as well as many other factors.

   The Makefile actually has a lot of labels (sort of names for different sections). Hence depending on what needs to be done the control would be passed to the different sections within the Makefile or it is possible that at the end of one of the section there is a command to go to some next section.

   Basically the make utility compiles all your program code and creates the executables. For particular section of the program to complete might require some other part of the code already ready, this is what the Makefile does. It sets the sequence for the events so that your program does not complain about missing dependencies.

3. As indicated before make uses the file named Makefile in the same directory. When you run make without any parameters, the instruction in the Makefile begin executing from the start and as per the rules defined within the Makefile (particular sections of the code may execute after one another..thats why labels are used..to jump from one section to another). But when you run make with install as the parameter, the make utility searches for a label named install within the Makefile, and executes only that section of the Makefile.

   The **install** section happens to be only a part where the executables and other required files created during the last step (i.e. make) are copied into the required final directories on your machine. E.g. the executable that the user runs may be copied to the /usr/local/bin so that all users are able to run the software.

   Similarly all the other files are also copied to the standard directories in Linux. Remember that when you ran make, all the executables were created in the temporary directory where you had unzipped your original tarball. So when you run make install, these executables are copied to the final directories.

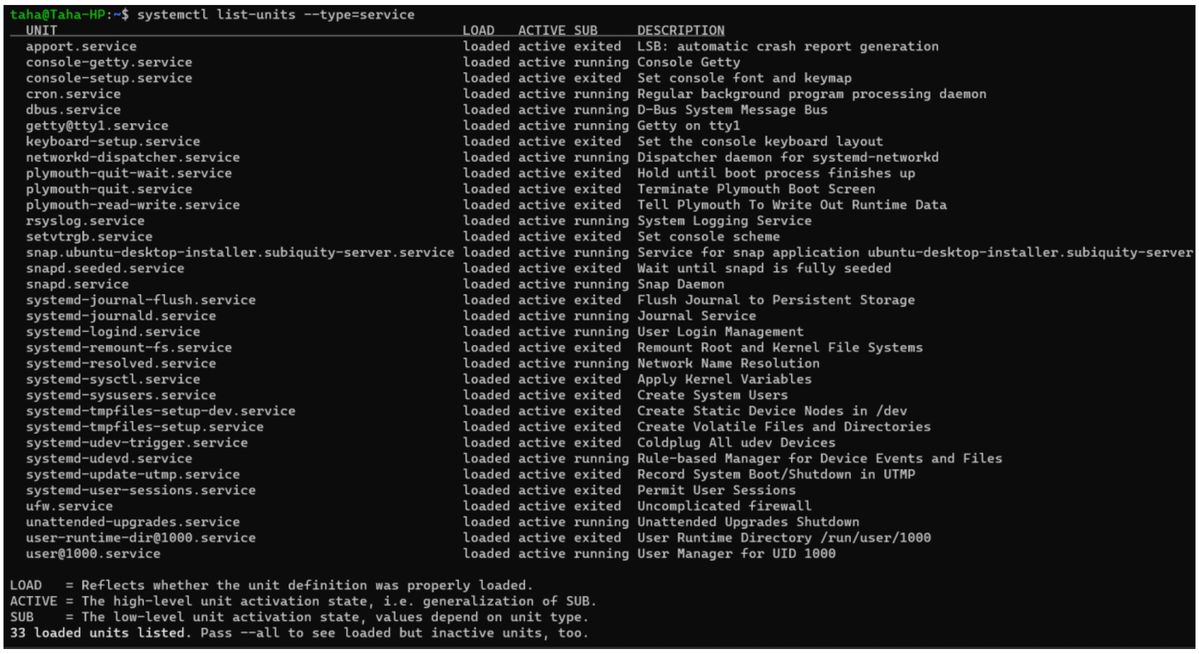
System Services

System services perform certain tasks while running in the background of your system. A service can have different functions, for example: managing I/O devices, security, error detection etc . The status of the service also varies, it’s not necessary that a service is always in running state.

Below is a table showing the various commands of systemctl

Command	**Description**
systemctl --type=service	Lists all loaded services in the system
systemctl --type=service --state=active	Lists only the active loaded services
systemctl --type=service --state=running	Lists only the running services
systemctl start *[service_name]*	Starts the given service
systemctl stop *[service_name]*	Stops the given service
systemctl restart *[service_name]*	Restarts the given service
systemctl enable*[service_name]*	Enables the service to start automatically on startup
systemctl disable *[service_name]*	Disables the service from starting automatically on startup
systemctl status *[service_name]*	Displays the status of the service
systemctl is-active *[service_name]*	Displays whether the service is active or not
systemctl is-enabled *[service_name]*	Displays whether the service is active or not
systemctl is-failed *[service_name]*	Displays whether the service is active or not

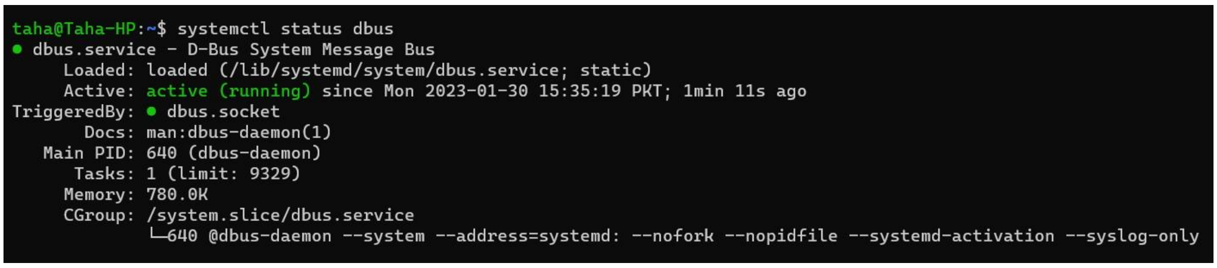
You can use GUI to monitor your system services, or monitor them through your terminal. Using systemctl, we can manage the system services through the terminal. Use the command **systemctl list-units –type=service** to view the services. The output should display the system services.



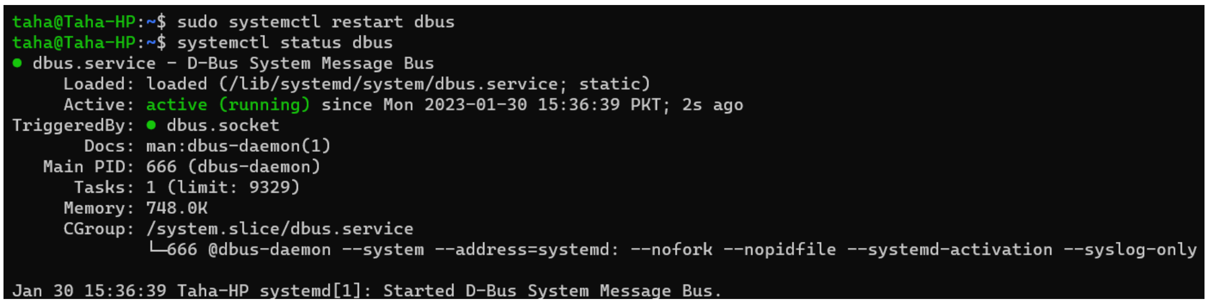
```
taha@Taha-HP:~$ systemctl list-units --type=service
  UNIT                                                LOAD   ACTIVE SUB     DESCRIPTION
  apport.service                                      loaded active exited  LSB: automatic crash report generation
  console-getty.service                               loaded active running Console Getty
  console-setup.service                               loaded active exited  Set console font and keymap
  cron.service                                        loaded active running Regular background program processing daemon
  dbus.service                                        loaded active running D-Bus System Message Bus
  getty@tty1.service                                  loaded active running Getty on tty1
  keyboard-setup.service                              loaded active exited  Set the console keyboard layout
  networkd-dispatcher.service                         loaded active running Dispatcher daemon for systemd-networkd
  plymouth-quit-wait.service                          loaded active exited  Hold until boot process finishes up
  plymouth-quit.service                               loaded active exited  Terminate Plymouth Boot Screen
  plymouth-read-write.service                         loaded active exited  Tell Plymouth To Write Out Runtime Data
  rsyslog.service                                     loaded active running System Logging Service
  setvtrgb.service                                    loaded active exited  Set console scheme
  snap.ubuntu-desktop-installer.subiquity-server.service loaded active running Service for snap application ubuntu-desktop-installer.subiquity-server
  snapd.seeded.service                                loaded active exited  Wait until snapd is fully seeded
  snapd.service                                       loaded active running Snap Daemon
  systemd-journal-flush.service                       loaded active exited  Flush Journal to Persistent Storage
  systemd-journald.service                            loaded active running Journal Service
  systemd-logind.service                              loaded active running User Login Management
  systemd-remount-fs.service                          loaded active exited  Remount Root and Kernel File Systems
  systemd-resolved.service                            loaded active running Network Name Resolution
  systemd-sysctl.service                              loaded active exited  Apply Kernel Variables
  systemd-sysusers.service                            loaded active exited  Create System Users
  systemd-tmpfiles-setup-dev.service                  loaded active exited  Create Static Device Nodes in /dev
  systemd-tmpfiles-setup.service                      loaded active exited  Create Volatile Files and Directories
  systemd-udev-trigger.service                        loaded active exited  Coldplug All udev Devices
  systemd-udevd.service                               loaded active running Rule-based Manager for Device Events and Files
  systemd-update-utmp.service                         loaded active exited  Record System Boot/Shutdown in UTMP
  systemd-user-sessions.service                       loaded active exited  Permit User Sessions
  ufw.service                                         loaded active exited  Uncomplicated firewall
  unattended-upgrades.service                         loaded active running Unattended Upgrades Shutdown
  user-runtime-dir@1000.service                       loaded active exited  User Runtime Directory /run/user/1000
  user@1000.service                                   loaded active running User Manager for UID 1000

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
33 loaded units listed. Pass --all to see loaded but inactive units, too.
```

As you can see, each service file ends with ".service". When using systemctl commands to manage these services, we don't need to add ".service" since the commands are only used for services. So for example, we want to check the status of "dbus.service". There are 2 equivalent commands:

1. **systemctl status dbus**
2. **systemctl status dbus.service**



```
taha@Taha-HP:~$ systemctl status dbus
● dbus.service - D-Bus System Message Bus
     Loaded: loaded (/lib/systemd/system/dbus.service; static)
     Active: active (running) since Mon 2023-01-30 15:35:19 PKT; 1min 11s ago
TriggeredBy: ● dbus.socket
       Docs: man:dbus-daemon(1)
   Main PID: 640 (dbus-daemon)
      Tasks: 1 (limit: 9329)
     Memory: 780.0K
     CGroup: /system.slice/dbus.service
             └─640 @dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
```



```
taha@Taha-HP:~$ sudo systemctl restart dbus
taha@Taha-HP:~$ systemctl status dbus
● dbus.service - D-Bus System Message Bus
     Loaded: loaded (/lib/systemd/system/dbus.service; static)
     Active: active (running) since Mon 2023-01-30 15:36:39 PKT; 2s ago
TriggeredBy: ● dbus.socket
       Docs: man:dbus-daemon(1)
   Main PID: 666 (dbus-daemon)
      Tasks: 1 (limit: 9329)
     Memory: 748.0K
     CGroup: /system.slice/dbus.service
             └─666 @dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only

Jan 30 15:36:39 Taha-HP systemd[1]: Started D-Bus System Message Bus.
```

As shown above, the restart command restarted the service, which is shown by the change in the time displayed. This time is the time when the service started, which resets when the service is restarted.