

Kernel

Kernel is the important part of an Operating System. The kernel is the first program that is loaded after the boot loader whenever we start a system. The Kernel is present in the memory until the Operating System is shut-down.

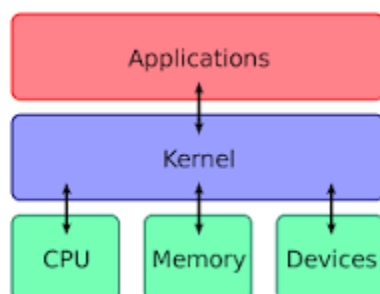
Kernel provides an interface between the user and the hardware components of the system. When a process makes a request to the Kernel, then it is called System Call.

Functions

The functions of the kernel are as follows –

- Process management
 - Access computer resources
 - Device management
 - Memory management
 - Interrupt handling
 - I/O communication
 - File system...etc.
1. **Access Computer resource** – A Kernel accesses various computer resources like the CPU, I/O devices and other resources. Kernel is present in between the user and the resources of the system to establish the communication.
 2. **Resource Management** – Kernel shares the resources between various processes in a way that there is uniform access to the resources by every process.
 3. **Memory Management** – Generally memory management is done by the kernel because every process needs some memory space and memory has to be allocated and deallocated for its execution.
 4. **Device Management** – The allocation of peripheral devices connected in the system used by the processes is managed by the kernel.

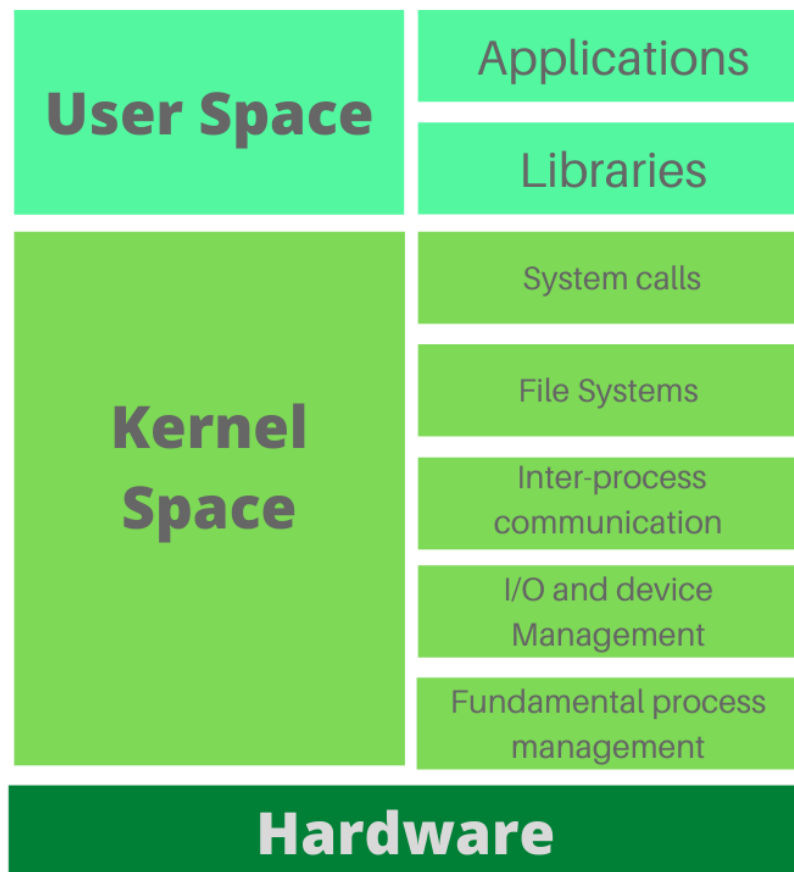
How does Kernel interact with hardware



The kernel is a computer program at the core of a computer's operating system and generally has complete control over everything in the system. It is the portion of the operating system code

that is always resident in memory, and facilitates interactions between **hardware and software components**.

How does the user communicate with the kernel



What are the Shell types

- Bourne shell (sh)
- C shell (csh)
- TC shell (tcsh)
- Korn shell (ksh)
- Bourne Again SHell (bash)

Why Bash Shell is familiar

Shell scripts have several required constructs that tell the shell environment what to do and when to do it. Of course, most scripts are more complex than the above one.

The shell is, after all, a real programming language, complete with variables, control structures, and so forth. No matter how complicated a script gets, it is still just a list of commands executed sequentially.

The following script uses the **read** command which takes the input from the keyboard and assigns it as the value of the variable **PERSON** and finally prints it on **STDOUT**.

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

```
#!/bin/sh

# Author : Zara Ali
# Copyright (c) Tutorialspoint.com
# Script follows here:

echo "What is your name?"
read PERSON
echo "Hello, $PERSON"
```

Here is a sample run of the script –

```
$/test.sh
What is your name?
Zara Ali
Hello, Zara Ali
$
```

A **Shell** provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

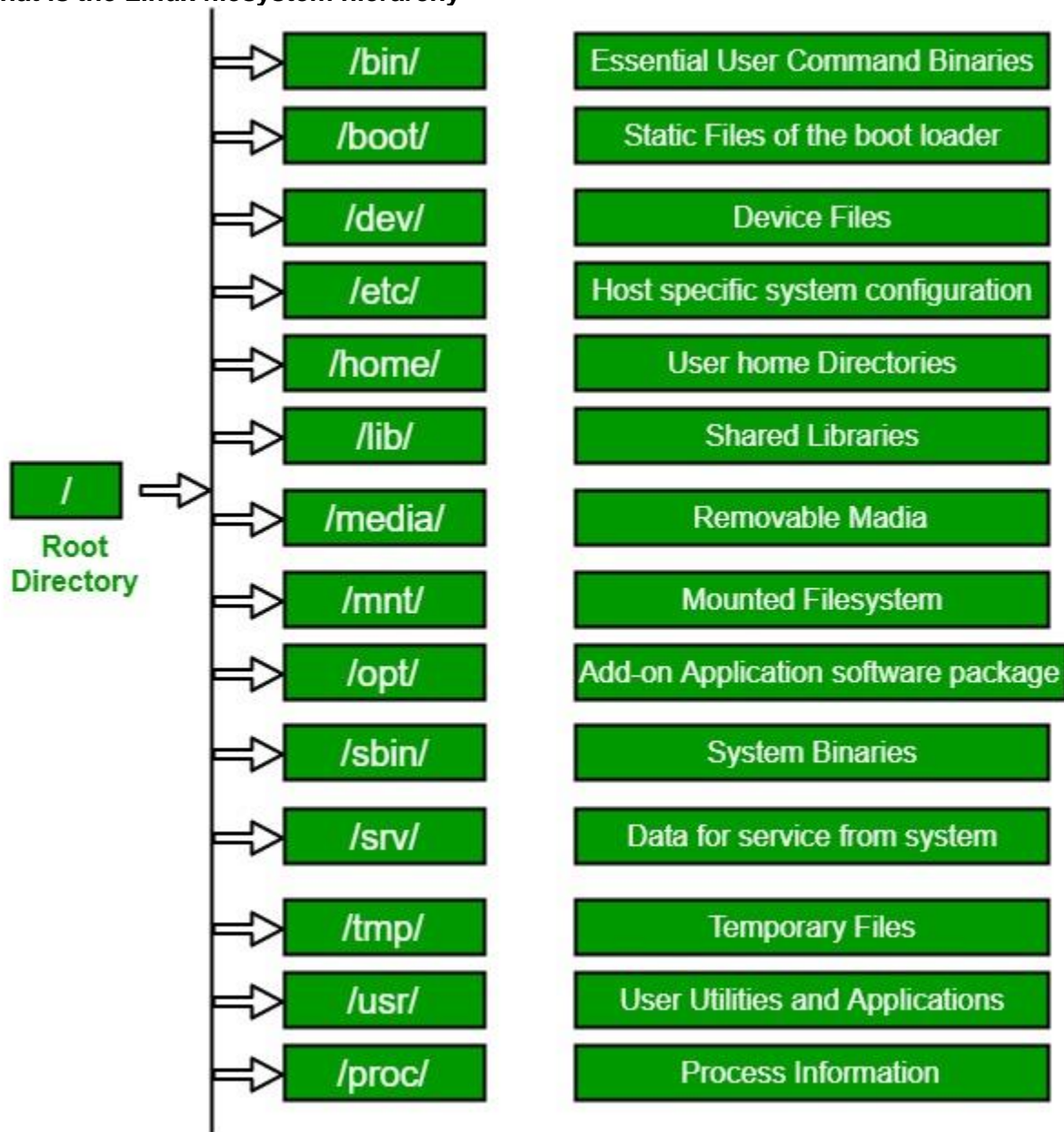
What is a terminal

The Linux terminal is **a text-based interface used to control a Linux computer**. It's just one of the many tools provided to Linux users for accomplishing any given task, but it's widely considered the most efficient method available

How many terminals are there in Linux

One of them is graphics terminal, the other six is character terminal. The **7 virtual terminals** are more commonly known as virtual consoles and they use the same keyboard and monitor. Physical console is the combination of your monitor and keyboard

What is the Linux filesystem hierarchy



What are Run Levels in Linux

A run level is a state of init and the whole system that defines what system services are operating. Run levels are identified by numbers. Some system administrators use run levels to define which subsystems are working, e.g., whether X is running, whether the network is operational, and so on.

- Whenever a LINUX system boots, firstly the **init** process is started which is actually responsible for running other start scripts which mainly involves initialization of you hardware, bringing up the network, starting the graphical interface.

- Now, the **init** first finds the default **runlevel** of the system so that it could run the start scripts corresponding to the default run level.
- A **runlevel** can simply be thought of as the state your system enters like if a system is in a single-user mode it will have a **runlevel 1** while if the system is in a multi-user mode it will have a **runlevel 5**.
- A **runlevel** in other words can be defined as a **preset single digit integer** for defining the operating state of your LINUX or UNIX-based operating system. Each runlevel designates a different system configuration and allows access to different combination of **processes**.

The important thing to note here is that there are differences in the runlevels according to the operating system. The standard **LINUX kernel** supports these seven different runlevels :

- 0 – System halt *i.e* the system can be safely powered off with no activity.
- 1 – Single user mode.
- 2 – Multiple user mode with no NFS(network file system).
- 3 – Multiple user mode under the command line interface and not under the graphical user interface.
- 4 – User-definable.
- 5 – Multiple user mode under GUI (graphical user interface) and this is the standard runlevel for most of the LINUX based systems.
- 6 – Reboot which is used to restart the system.

By default most of the LINUX based system boots to runlevel 3 or runlevel 5.

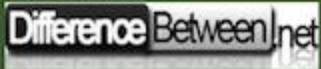
Diff b/w Redhat 6,7,8 booting process

S.No	Description	RHEL 8	RHEL 7	RHEL 6
1	Boot Loader: The GRUB2 looks very similar to GRUB but there are many features added.	GRUB 2	GRUB 2	Legacy GRUB
2	Runlevel: Runlevels are referred as target but there is no difference but they merged runlevel 2,3,4 into one.	runlevel0.target -> poweroff.target runlevel1.target -> rescue.target runlevel2.target -> multi-user.target runlevel3.target -> multi-user.target runlevel4.target -> multi-user.target runlevel5.target -> graphical.target runlevel6.target -> reboot.target	runlevel0.target -> poweroff.target runlevel1.target -> rescue.target runlevel2.target -> multi-user.target runlevel3.target -> multi-user.target runlevel4.target -> multi-user.target runlevel5.target -> graphical.target runlevel6.target -> reboot.target	runlevel 0 runlevel 1 runlevel 2 runlevel 3 runlevel 4 runlevel 5 runlevel 6
3	To view runlevel/target	systemctl get-default	systemctl get-default	runlevel
4	To change runlevel/target	systemctl isolate [Name.target]	systemctl isolate [Name.target]	init [runlevel]
5	To configure default runlevel/target	systemctl set-default [Name.target]	systemctl set-default [Name.target]	/etc/inittab

S.No	Description	RHEL 8	RHEL 7	RHEL 6
6	To break root password or Boot into single user mode	Append rd.break or init=/bin/bash to kernel cmdline	Append rd.break or init=/bin/bash to kernel cmdline	Append 1 or s or init=/bin/bash to kernel cmdline
7	KDUMP	Kdump is enabled by default and will run without any problems if the system has too much RAM.	Kdump is enabled by default and will run without any problems if the system has too much RAM (up to 3 TB).	Kdump is enabled by default and will run without any problems if the system has too much RAM.

Difference between absolute & relative path

ABSOLUTE PATH VERSUS RELATIVE PATH

Absolute Path	Relative Path
It points to a specific location in the file system, irrespective of the current working directory.	It points to the location of a directory using current directory as a reference.
It is also referred to as full path or file path.	It is also referred to as non-absolute path.
It refers to the location of a file or directory (filesystem) relative to the root directory in Linux.	It refers to the location of a file or directory (filesystem) relative to the current directory.
Absolute URLs are used to link to other websites that are not located on the same domain.	Relative URLs are used to link to other websites that are located on the same domain.
For example: If your pictures are in C:\Sample\Pictures and index in C:\Sample\Index, then the absolute path for pictures is C:\Sample\Pictures.	For example: If your pictures are in C:\Sample\Pictures and index in C:\Sample\Index, the relative path is "..\Pictures". 

Absolute path vs. relative path

- Absolute path is the exact address of the file in the filesystem, starting from the root.

C:\Geoinformatics\myFile.txt

A:\Courses\Python\Programming.html

- Relative path writes it with respect to another point in the file system

Python\Functions\map.txt

- We can either append a relative path to an existing absolute path.
- Or, we can implicitly reference the current working directory where a Python program may consider to be during an execution.

Directory Creation and Parent Dir

For instance, let's say I'm in an empty folder. Now I can create parent directories if they don't exist when creating a folder

```
mkdir -p nested/folder
```

I can create files in existing directories

```
touch nested/folder/something.txt
```

But I can't create a file in a directory that doesn't exist yet

```
touch nested/folder/deep/more.txt
```