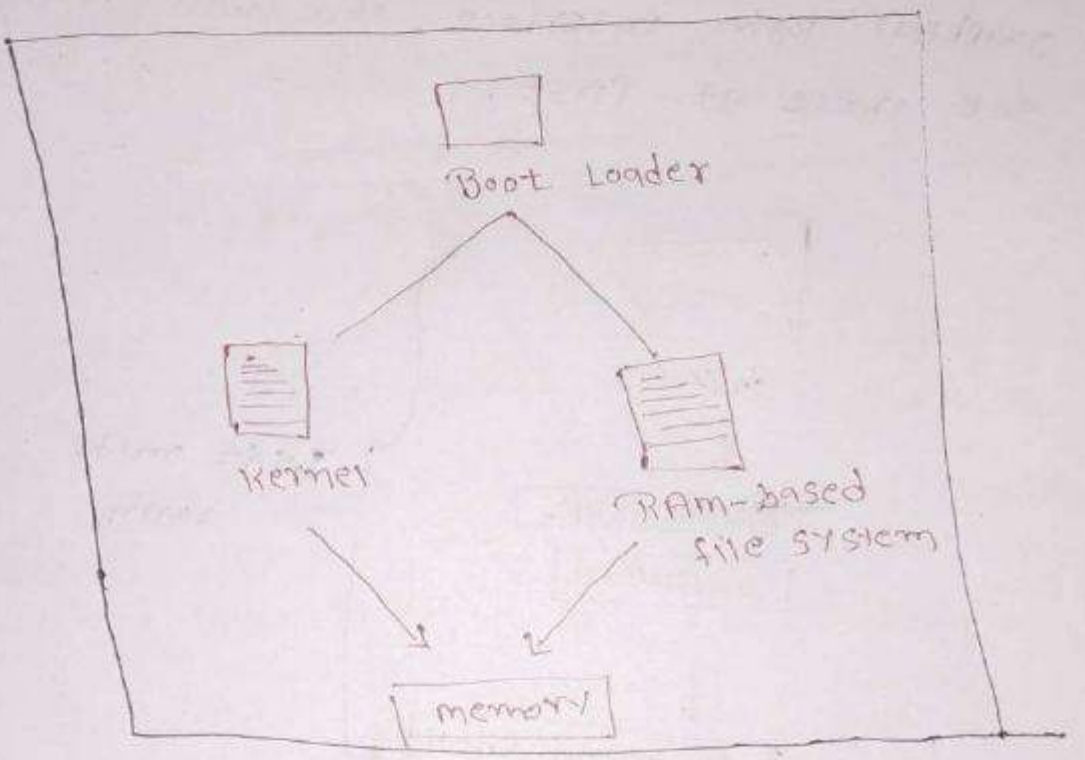
* Kernel, init and services

• The Linux Kernel

→ The boot loads both Kernel and an initial RAM-based file system (initramfs) into memory, so it can be used directly by the Kernel.



The Linux Kernel

→ When the Kernel is loaded in RAM, it immediately initialize and configure the computer's memory and also configures all the hardware attached to the system.

→ The include all processors, I/O subsystem, storage devices etc.

→ The Kernel also loads some necessary user space application.

* /sbin/init and services

—> once the kernel has set up all it's hardware and mounted the root file system, the kernel runs /sbin/init.

—> This then becoms the initial Process, which then starts other Processes to get the system running.

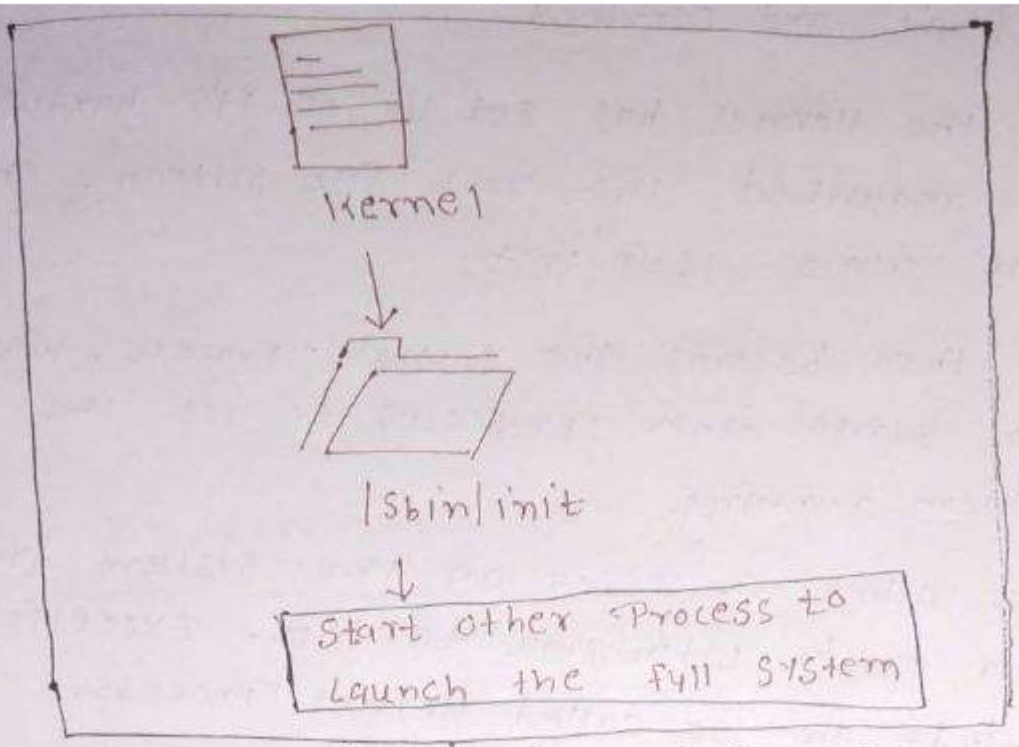
—> most other Processes on the system trace their origin ultimately to init. Exception include the so called Kernel Process.

—> These are started by kernel directly and their job is to manage internal operating system details.

—> Besides starting the system, init is responsible for keeping the system running and for shutting it down cleanly.

—> one of its responsibilities is to act when necessary as a manager for all known Kernel Process

—> It cleans up after them upone completion and restarts user Log in services as needed when user Log in and out and does the same for other background system services.



/sbin/init and Services

→ Traditionally, this Process startup was done using convention that date back to the 1980's and the System V variety of UNIX.

→ This Serial Process (called sysvinit) had the System Pass through a sequence of runlevels containing collection of scripts that start and stop services.

→ Each runlevel supported a different mode of running the System.

→ Within each runlevel, individual could be set to run, or to be shut down if running.

→ How ever, all major distribution have moved away from this sequential method of sytem initialization, although they usally can emulate many system V utilities for compatibility purpose.

* Startup Alternatives

→ SysVinit viewed things as seriel Process, divided into a series of sequential stages.

→ Each Stage required completion before next could Proceed.

→ Thus, start up did not easily take advantage of parallel Processing that could be done with multiple Processors or cores found on morden systems.

→ Furthermore, starting up and rebooting were seen as relatively rare events; Exactly how long they took was not considered important

→ This is no longer true, especially with mobile devices are embedded linux systems.

→ Some morden methods, such as the use of containers, can require almost instantaneous startup time.

→ Thus systems now require methods with faster and enhanced capabilities.

→ Finally the older methods require rather complicated startup scripts

→ which were ~~diffrent~~ difficult to keep Universal across distribution version, kernel versions, architectures, and the types of systems.

→ The two main alternatives developed were:

UPStart

- Developed by Ubuntu and first included in 2006
- Adopted in Fedora 9 (in 2008) and in RHEL 6 and its co clones

Systemd

- Adopted by Fedora first (in 2011)
- Adopted by RHEL 7 and SUSE
- Replace UPStart in Ubuntu 16.04

→ while the migration to systemd was rather controversial, it has been adopted by all major distributions.

→ Regardless of how one feels about the controversies of the technical method of systemd, almost universal adoption has made learning how to work on Linux systems simpler

* Systemd features

→ Systems with systemd start up faster than those with earlier init methods.

→ This is largely because it replaced a serialized set of steps with aggressive parallelization techniques

→ with permits multiple services to be initiated simultaneously.

→ Complicated startup shell scripts are replaced with simpler configuration files. which enumerate what has to be done before a service ~~start~~ started,

→ How to executer service startup.

→ And what condition the service should indicate have been accomplished when startup is finished.

→ one thing to note is that /sbin/init now just points to /lib/systemd/systemd.

i.e systemd takes over the init process

→ one systemd command (systemctl) is used for most basic task.

- starting, stoping, restarting a service (using httpd, the Apache web server as an example) on currently running system.

```
$ sudo systemctl
    start | stop | restart
        httpd.service
```

- Enabling or disabing a system service from starting up at system boot

```
$ sudo systemctl enable | disable
        httpd.service
```

- checking on the status of a service

```
$ sudo systemctl status
    httpd.service
```

→ In most cases, the .service can be omitted.

→ There are many technical diffrence with older methods that lie beyond the scope of our discussion.

Systemd

* Lab :- Apache we server status

-> Check the status of httpd (Apache web server) on your system

-> If it is running, stop it and check again.

-> Start the service and check the status

-> You probably want to stop it again when you are done.

1. $ sudo systemctl status httpd
2. $ sudo systemctl start httpd
3. $ sudo systemctl status httpd
4. $ sudo systemctl stop httpd

Solution

* Linux File systems

-> Libraries seprate books and other media into multiple section; this organization will depend on the subject matter, audience, media type, and frequency of retrieval.

-> The same concept applies to a file system which is the embodiment of method of storing and organizing arbitary collections of data in a human-usable form.

-> Different types of file systems supported by Linux:

- conventional disk file systems
  -> ext3, ext4, XFS, Btrfs, JFS, NTFS, vfat, exfat, etc.
- Flash storage file systems
  -> ubifs, jffs2, yaffs, etc
- special purpose file systems
  -> Procfs, sysfs, tmpfs, squashfs, debugfs, fuse, etc.

* Partitions and File systems

-> A partition is a dedicated subsection of physical storage.

-> Historically this meant a physically contiguous portion of hard disk

-> Today's storage devices can be more complicated but we still think of a partition as a fixed area to be treated as a whole.

-> A file system is just a method of storing and accessing files.

-> one can think of a Patition as a container in which a file system resides.

-> How ever in some circumstances, a file system can span more than one partition if one uses symbolic links.

	windows	Linux
Partition	DISK1	/dev/sda1
Filesystem TYPE	NTFS/VFAT	EXT3/EXT4/XFS/BTRS...
mounting Parameters	Drive Letter	mount Point
Base folder	C:\	/

* The Filesystem Hierachy Standard

-> Linux systems store their important files according to standard layout called the File system Hierarchy standard (FHS).

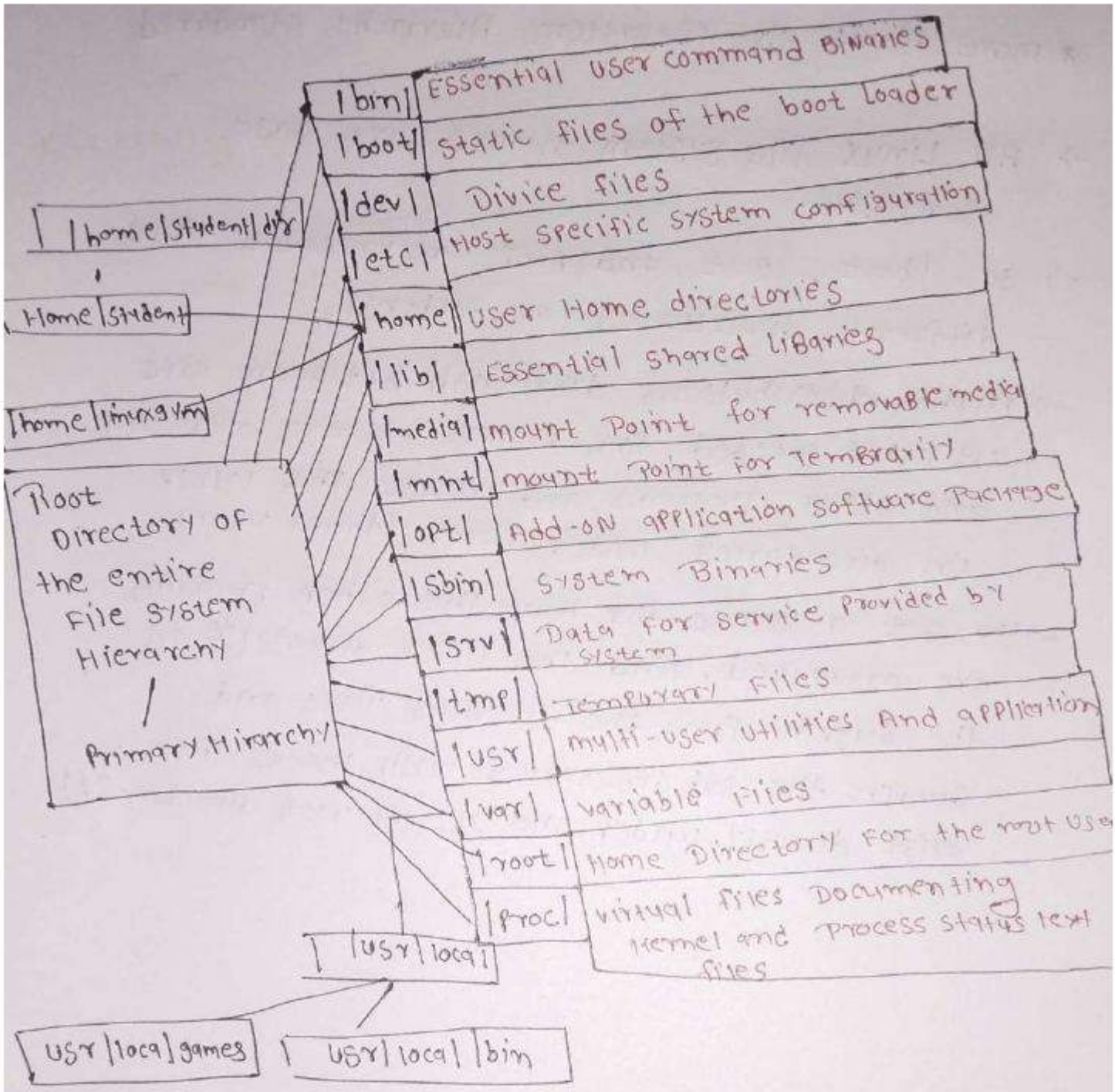
-> which has long been maintained by the Linux foundation.

-> Linux uses '/' character to seperate paths (as sis UNIX Unlike windows, which uses '\') and does not have drive letters.

-> mulliple drivers and/or Partitions are mounted as directories in the single file system.

-> Removable media such as USB drivers and CDs and DVDs will show up as mounted at /run/media/yourusername/disklabel for recent Linux systems or under /media for older distributions.

-> For Example, if your username is student, a USB Pen drive labeled FEDORA might end up being found at run/media/student/FEDORA, and a file README.txt on that disc would be at run/media/student/FEDORA/README.txt



File System Hierarchy Standard (FHS)

* more About the Filesystem Hierarchy Standared

-> All Linux file system names are case-sensitive.

-> So /boot, /Boot, and /BOOT represent three diffrent directories (or folders).

-> many distributions distiguish between core utilities needed for proper system operation and other Programs, and place the latter in directories under /usr (think user).

-> To get a sense for how the other programs are organized, find the /usr directory in the diagram from the Previous Page and compare the subdirectories with those that exist directly under the system root directory (/).