<http://solarisfacts.blogspot.com/p/drilling-down-file-system.html#:~:text=UFS(Unix%20File%20System)%20is,or%20as%20primary%20file%20system>.

Solaris ZFS: <https://www.youtube.com/watch?v=n3kjqaVdZ-E>

**POSIX – portable operating system interface**

**‘GNU’s Not Unix’**

**Alias c = clear**

**Cat /etc/shells**

**Echo $SHEL**

**Ps –ef = to find deamons**

**Free –m / -g // /proc/meminfo**

**Vmstat 2 6**

**Sar –u = cpu usage (system activity report)**

**Nagios, nrpe**

**Sas and scsci drive names start with ‘s’. ide drives starts with ‘h’. floppy’s as ‘f’**

**/dev/tty serial ports are communication ports**

**/dev/lp parallel ports – printer**

**Hard links can be created only for files**

**Env = command for environment variable**

The Global **environment variables** of your system are **stored** in /etc/**environment** . Any changes here will get reflected throughout the system and will affect all users of the system.

You can set your own persistent **environment variables** in your **shell** configuration file, the most common of which is ~/. bashrc. If you're a system administrator managing several users, you can also set **environment variables** in a script placed in the /etc/profile. d directory.

Ps – aux = status of a process

Nohup = command to run process in the background

**36) What Are Performance Tool Used?**

**Ans:**

Iostat ,vmstat , prstat , sar ,netstat, top

**40) Tell Me The Port For Telnet, Ftp,nis,ssh,nfs?**

**Ans:**

ftp = 21ssh = 22Telnet = 23nfs = 2049nis = 749

**41) In Which File Port No’s Are Defined?**

**Ans:**

# /etc/service

**43) How Do You Check The Run Level?**

**Ans:**

# who –r

When we login as root user on the network, then both **whoami and who am** i commands will show the user as root. But when any other user let say john logs in remotely and runs su –root, **whoami** will show root, but who **am** i will show the original user john.

**49) What Is The Difference Between Dsk And Rdsk?**

**Ans:**

DSK: Block level devices  
RDSK: Raw level device or character level device

**77) What are LDOMS?**

**Ans:**

Sun Logical Domains or LDoms is a full virtual machine that runs an independent operating system instance and contains virtualized CPU, memory, storage, console, and cryptographic devices. This technology allows you to allocate a system resources into logical groupings and create multiple, discrete systems, each with their own operating system, resources, and identity within a single computer system.

We can run a variety of applications software in different logical domains and keep them independent of performance and security purposes. The LDoms environment can help to achieve greater resource usage, better scaling, and increased security and isolation.

**79) What is the difference between container and zones?**

**Ans:**

Zones: A zone is a virtual operating system abstraction that provides a secured environment where applications run.The applications are protected from each other to provide software fault isolation.

Container: zone + resource management The ability to control resource usage for processes,task and zones.Resources can be CPU level,RAM,virtual memory,Kernel level tables etc.

**88) What is /etc/system**

**Ans:**

Alternative for /etc/sysctl.conf ( linux)

The /etc/system file provides a static mechanism for adjusting the values of kernel variables. Values specified in this file are read at boot time and are applied. Any changes made to the file are not applied to the operating system until the system is rebooted.

Sdtwsinfo = for knowing RM

1. The ipconfig command tells you the IP for the computer. (linux)

ZFS is the root file system of Solaris 11

Echo | format = to display the hard disks installed

In order to use the zfs file system first create a pool of devices.

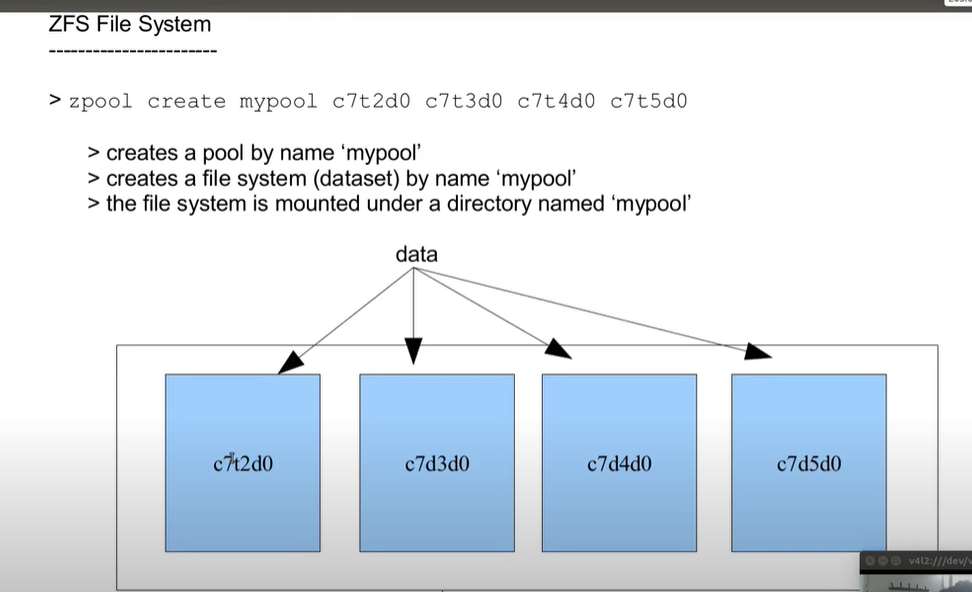
In solaris 11 zfs, with one command can create and mount the file system.

Zpool create mypool disk1 disk2

The devcices may consist of disks, large files etc. means, not only disk

Df –j or df –h = command to find the file systems currently mounted

Rpool is the default pool name // zpool list



The data is striped across each disks, so a loss of 1 disk causes data loss. No redundancy.

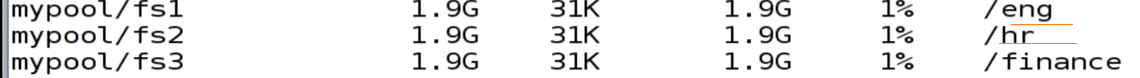
Zfs create mypool/fs1

Zfs list = to list the file system.

Zfs is a property centric file-system. Manage it by setting and getting properties.

Zfs get all mypool

Zfs set mountpoint=/eng mypool/fs1 = to set the mountpoint for the mypool/fs1



Zpool add mypool disk3 = adding another disk to ‘mypool’

Zfs set quota=500m mypool/fs1 = setting quota on the filesystem/dataset of ‘fs1’

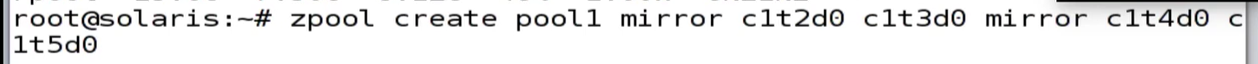
Setting a quota is not a guarantee, but it is only an upper limit.

Zfs set reservation=500m mypool/fs1 = is now a guarantee

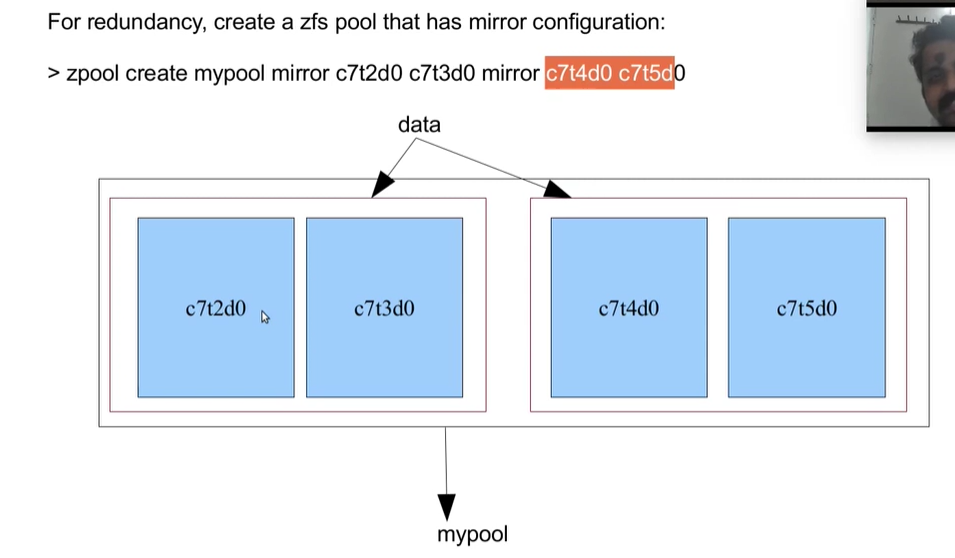
Zpool status mypool = shows devices in the pool

Zpool destroy mypool

Mirror pool:



Data will be striped across virtual devices.



Create a pool with parity 1 disk:



Create large file: mkfile



Snapshot can be taken on zfs file system.

Du –sh <filename> = to find the file size



List the snapshot taken:



When the original files are removed from the file system, it is added to the snapshot. Not when it exists on the file system.

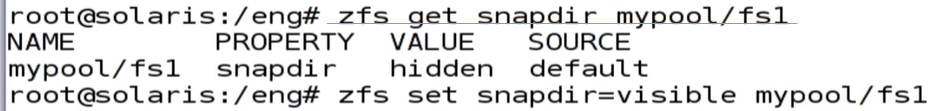
Restore/rollback the snapshot:



Snapshot is hidden under a hidden directory:



Showing a hidden directory in the zfs:



Share a file system: dfshares



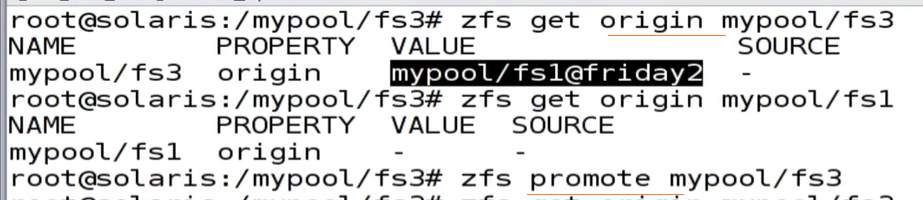
No etc/dfstab entry required

A clone of a filesystem can be created only from its snapshot.

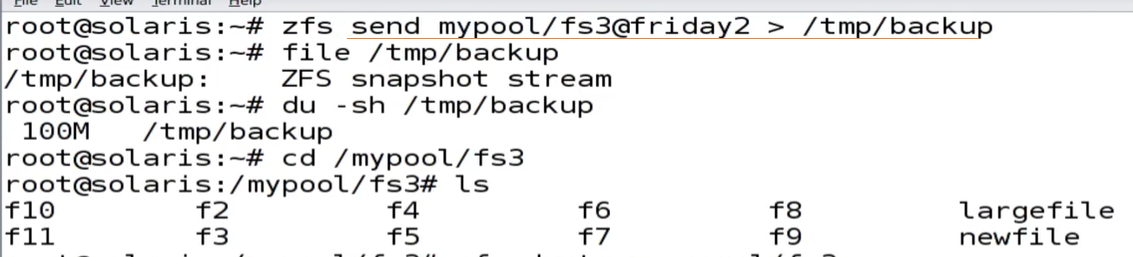


A clone is a writable copy.

Promoting a clone to a file system as original:



Sending a snapshot to an nfs share or some other place as a backup



Restoring the file from backup:



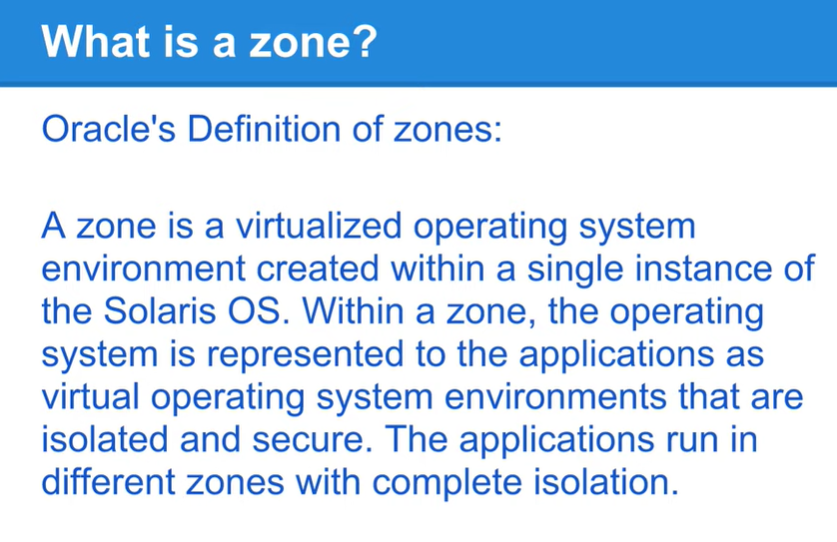
Sending a snapshot from one machine to another using ssh:

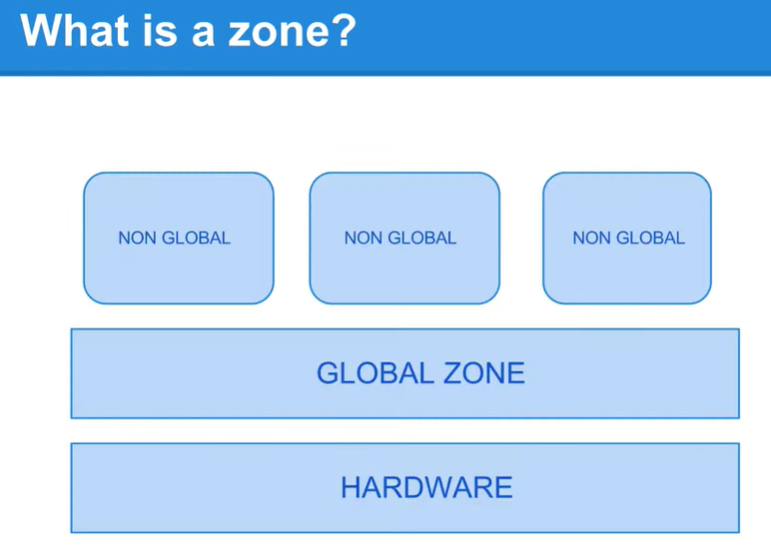
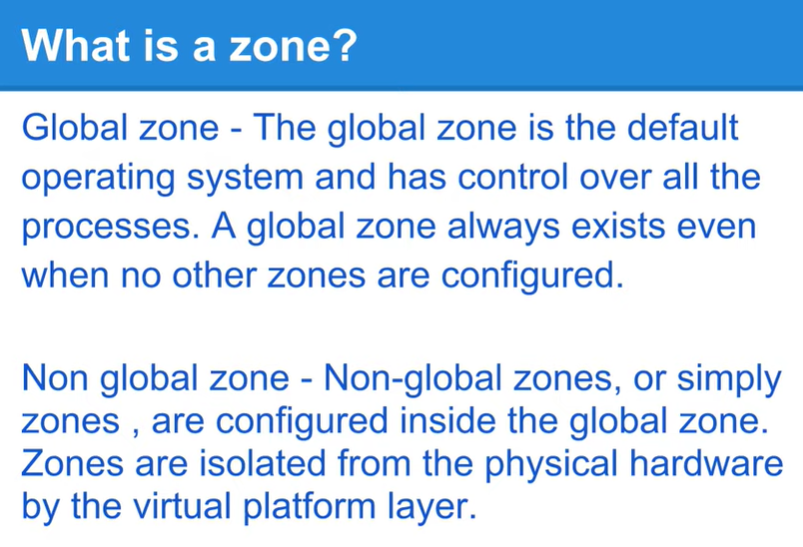


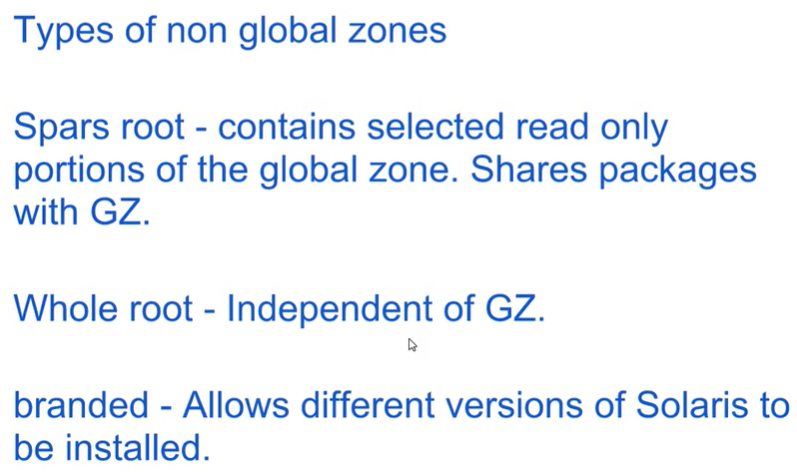
A destroyed pool can be recovered:

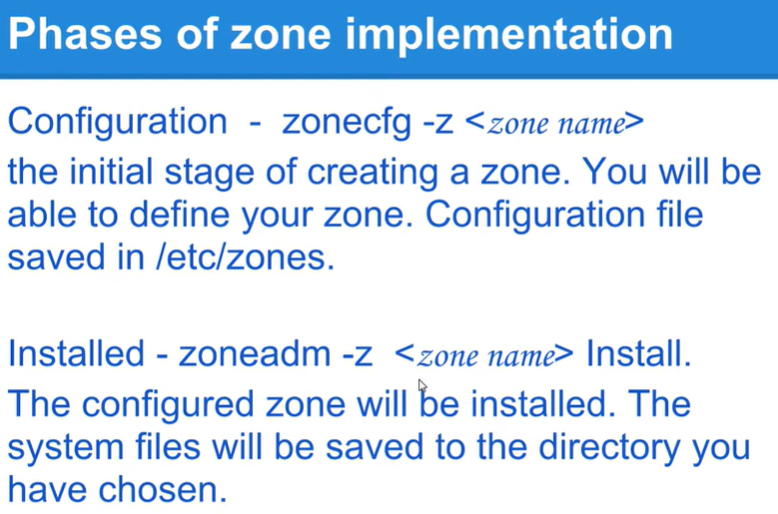


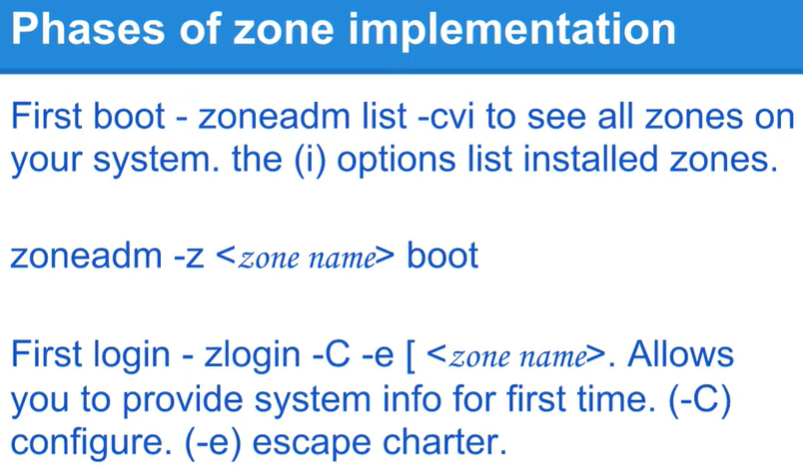
Zone: <https://www.youtube.com/watch?v=eQ4z-65zdKw>

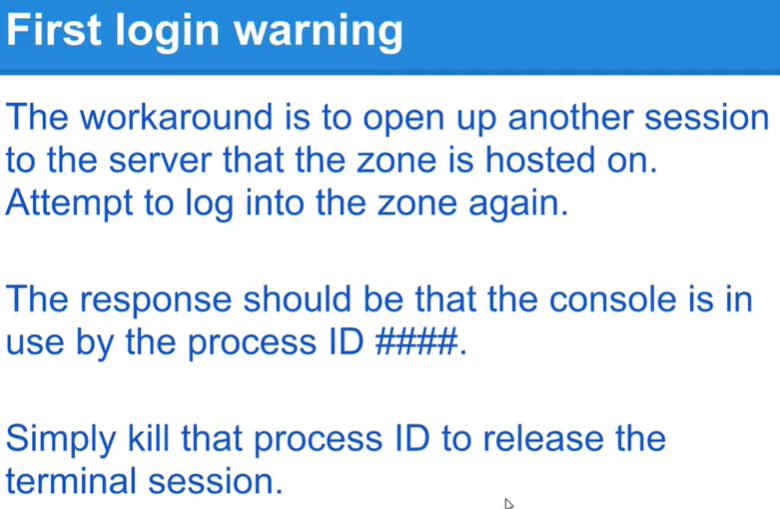


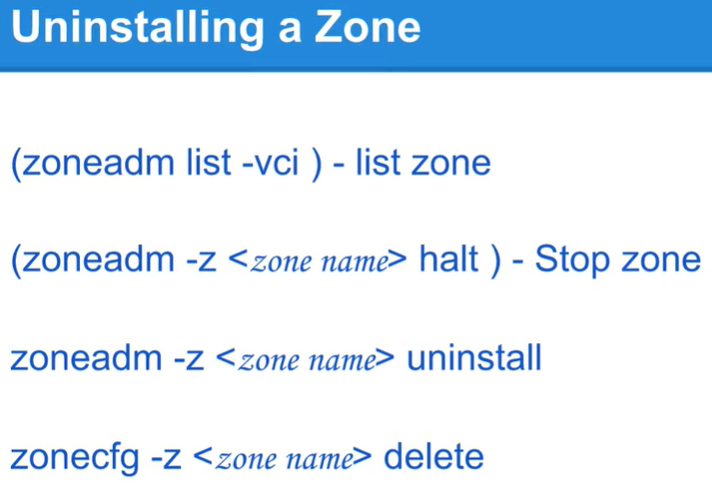












With Solaris zones introduced in Solaris 10, we can install multiple **instances** of the same OS installed. One base installation (one kernel) and many instances of it.

No additional layer of hypervisors.

Most of the applications that run on solaris 11 are certified to run on solaris 11 zone also.

Zoneadm list –cv = to see howmany zones are currently running

Global zone is the base zone.

Zonecfg = for zone confirguration

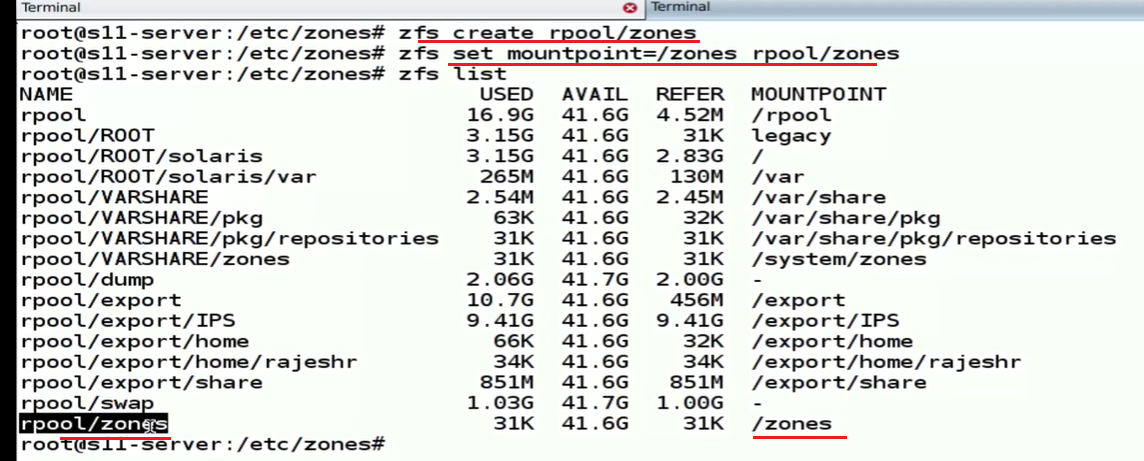
/etc/zones = when configuring a new zone, SYSdefault.xml template is used to set the path. Ip-type can be either ‘shared’ or ‘exclusive’

Ipadm showaddr = to show the ip address

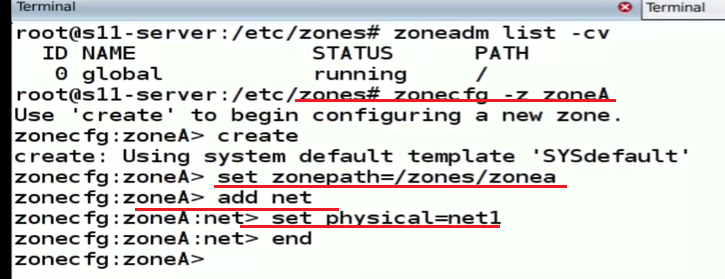
In solaris 11 any number of ‘vnic’s can be created (virtual nic) for virtualization.

Zfs is the root file system, that is the zonepath

Create a zfs filesystm and mount:



Configure a zone and set zonepath to the zfs file system and add network port:

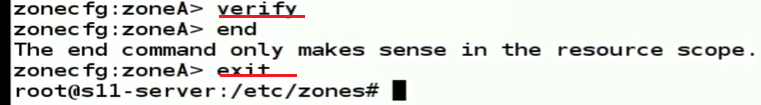


If high availability is needed on a zone, add 2 ports to the zone. No need to assign a physical port, can assign a vnic built on top of the physical device.

A zone is a virtual instance of an OS

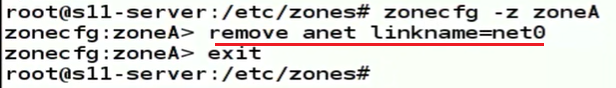
Add a resource in the zone and set value for its attribute.

Verify the configuration and exit



After creating a zone, a file is created inside the /etc/zones folder as <zonename>.xml with all the configurations for that zone.

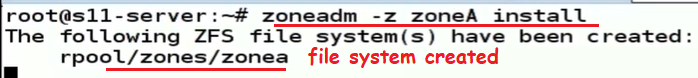
Removing a resource from the zone:



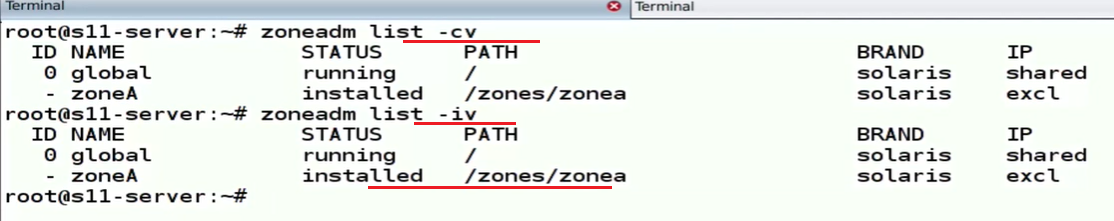
Anet is automatic net

Ips = image packaging system in solaris 11. To install solaris, need access to the repository

Install the zone:



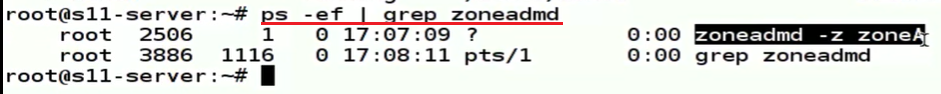
Zone creation status: configured, incomplete, installed, ready, running



Zoneadm –z ZoneA boot = to boot the zone

A zone is known by its name, not by id

Deamon zoneadmd is responsible for booting the zone, assign an ip address, assign an id. Zonestatd



Zonename = to find the zone name

Svcs ssh

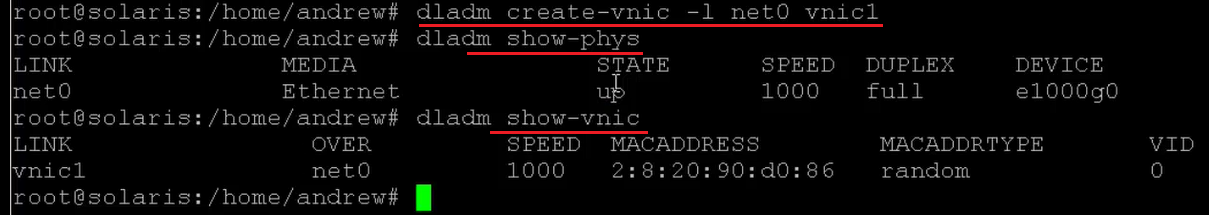
Zoneadm –z ZoneA shutdown = for shutting down

Zlogin –C <Zonename> = to access the zone console.

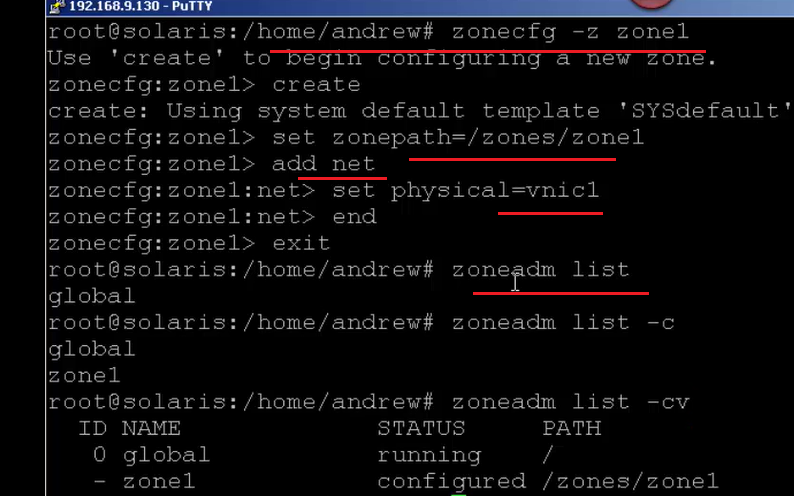
#pkg info entire

#pkg publisher solaris

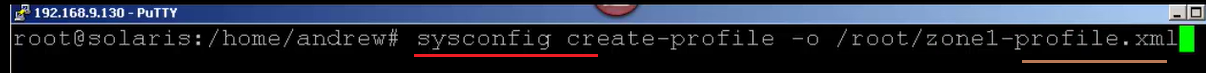
#dladm = create virtual network create-vnic



Create the zone:



Sysconfig: this is the answer file for the installation



Installing:



#pkg install <packagename>

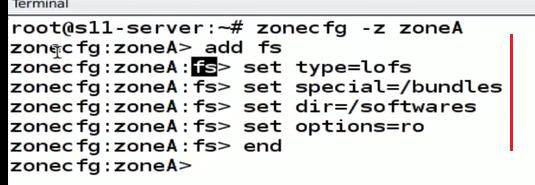
#svcadm enable <servicename>

#zonestate 5 = retrieving the zone status after 5 seconds

#zonecfg –z zoneA info = to get more info about the zone

#zonename

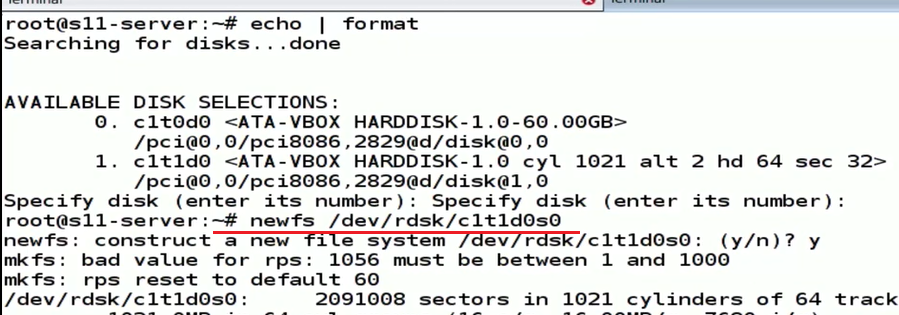
Adding a resource from the global zone:



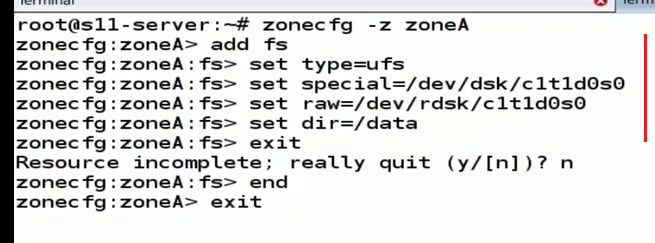
Only the global id will be stable, which is 0

#echo | format = to show the disks we have

#newfs = to create a ufs file system



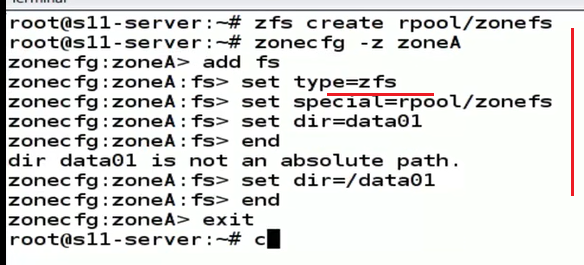
Adding a ufs resource into the zone:



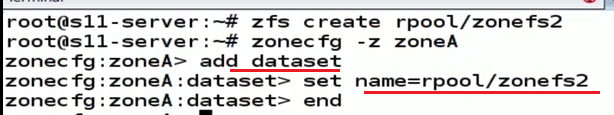
Mount the filesystem in the non-global zone from the global zone:



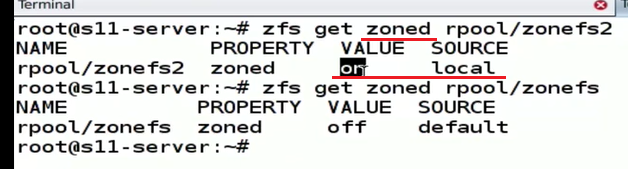
Added a zfs file system resource:



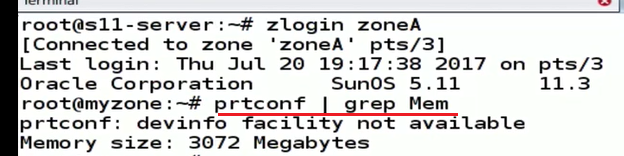
Adding a dataset:



Zoned:



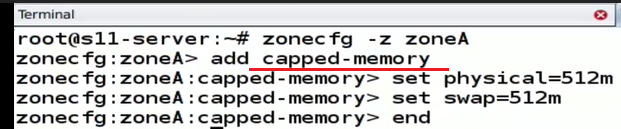
Prtconf: checking memory



**grep**, **egrep**, **fgrep**, **rgrep**, **pgrep** - are commands in Unix-like operating systems that print lines matching a pattern. The grep searches the named input FILEs for lines containing a match to the given PATTERN. By default, it prints the matching lines. In addition, the variant programs **egrep**, **fgrep**, and rgrep are the same as **grep -E**, **grep -F**, and **grep -r**, respectively.

|  |  |
| --- | --- |
| ^ | the circumflex is used to match the beginning of a line. |
| $ | used to match the end of a line. |
| . | matches any character except a new line. |
| [] | matches single character inside the brackets. If there's a ^ inside, it would match anything but the characters in the bracket. |
| \ | before any of the non-alphanumeric characters quotes them. |
| \* | symbol matches the preceding character or subexpression zero, one or more times. |
| \1 | backreferences 1-9 match the exact text by the corresponding group. |
| \{m,n\} | matches the preceding elements at least *m* and no more than *n* times. |
| \| | foo\|bar matches foo or bar. |
| \? | short for {0,1} |
| \+ | (short for **{1,}**) match the preceding character or subexpression at most 1 time, or at least 1 time respectively. |
| \n | matches a newline, \t matches a tab, etc. |
| \w | matches any word constituent and \W matches any character that isn't a word constituent. |
| \<\> | match the empty string only at the beginning or end of a word |
| \b | matches either and **\B** matches where \b doesn't. |
| **Symbol** | **Details Extended Regular Expressions (ERE)** |
| ^ | match only at the beginning |
| $ | match only at the end of a line. |
| . | matches any character (or any character except a newline). |
| […] | matches any one character listed inside the brackets (character set). Add an initial ^ and ranges work like in BRE (see above). |
| (…) | syntactic group, for use with \* or \DIGIT replacements. |
| \| | for alternation: foo|bar matches foo or bar. |
| \* | matches the preceding character or subexpression a number of times: 0, 1 or more times |
| + | matches 1 or more times preceding character. |
| ? | matches preceding characters 0 or 1 times. |
| \ | Backslash quotes the next character if it is not alphanumeric. |
| {m,n} | matches the preceding character or subexpression between m and n times (missing from some i |

Capped memory:

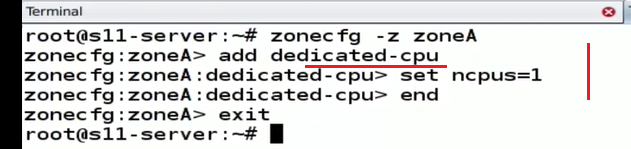


Enable rcap: rcapadm –E



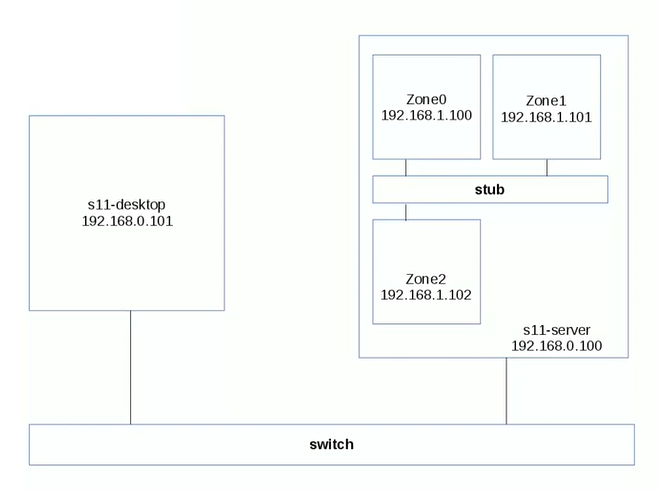
Pooladm, poolcfg poold daemon

Adding cpu:



Man zonecfg

A ‘stub’ is a private switch in solaris 11



Dladm show-etherstub = to show the stub

Dladm create-etherstub

Export a zone configuration:



Create a zone from an existing configuration



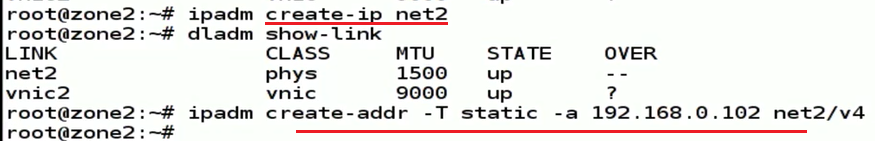
Create a vnic:



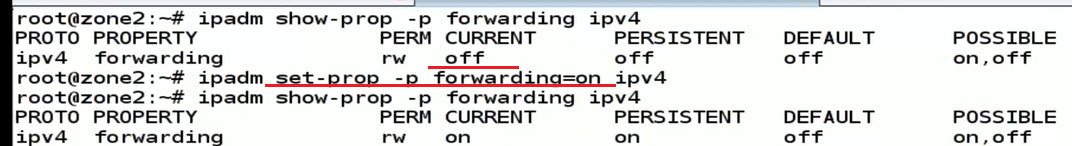
Router is a device that knows multiple networks

@dladm show-link = will show the ports

Create an ip address:



Ip forwarding:



Traceroute <ip addr> = to check the network route

Dladm delete-vnic vnic0

For automated installation of solaris, we need 3 servers plus client. 1)AI for automated client interaction 2) DHCP for providing ip address to client, 3) Image packaging repository

LDOM (Logical Domain) – VM Server

CMT is chip multi threading

Niagara processor and UltraSparc Hypervisor

Ldmd, vntsd = for terminal service - deamons

Th e Ul tr aS PARChy perv isor is a thi n layerof sof tw are stored withintheAL OMCMTfirm ware.

Hy perpriv ileg ed accessenablesthe hyper -vis or to eith er exp oseor hi de res ourcesfroman ins tanceof an op er atingsys -tem.Th is al low s res ourcesto be grouped intologicalpa rtitions or domains.

Re source s su ch as CPU threads, cryp tographic threads, an d memory are par -titioned intoa lo gi cal domain . Otherres ourcesare virtualiz ed and servicedthr oughthe useof Logica l Domain Channels, or LDCs.LDCsprovide securecommunicationandda ta pa thways between LDomsand th e hypervisor.Th is al low s an operati ng systemin oneLDomto make an I/Orequest, whichis servic ed by an other LDomthatha s privilegedacc ess to the underlyinghard ware

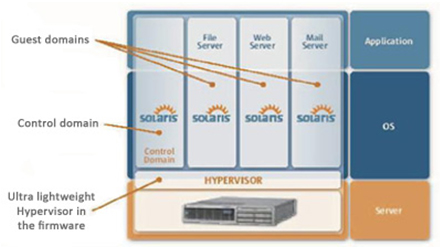
Log ic al DomainTypeDescri ption

1. Gue stDomain th at is a consumerof virtualize d devicesandserv ic es
2. I/ODomain th at hasprivi legedaccess to a PCI-Econtrollerbutdo es notprovide vi rtualizeddevi ces or se rvicesto guestdo mai ns
3. Serv ic eI/O domain thathasprivilegedaccess to oneor morePC I- Econtroll er s; provides vi rtu al iz ed devi ce s andservicesto guestdo mai ns
4. ControlService domain th at runs managementsoftwareto controlthehyp ervisor con figurati on of th e pl atform

Psrinfo –v | more for the processors info

Solaris LDoms is a server virtualization and partitioning technology that is used by many enterprises to host multiple instances of Solaris or Linux virtual machines (VMs) on a Sun server. The LDoms (or Logical Domains) technology allows administrators to allocate a system's various resources, such as memory, CPUs, and devices, into logical groupings and create multiple, discrete systems, each with their own operating system, resources, and identity within a single computer system.

A Solaris LDoms server has multiple guest domains and a single control domain for monitoring and reconfiguration of the guest domains. The guest domains are where the applications are hosted, and can be independently powered on and off without affecting other domains. The applications on the logical domains use the virtual CPU/disk/memory resources that the hypervisor allocates to every guest.

**The architecture of a Solaris LDoms server

# [Difference between LDOM and Zones](http://solsupport.blogspot.com/2010/10/difference-between-ldom-and-zones.html)

Sun Logical Domains or LDoms is a full virtual machine that  
runs an independent operating system instance and contains  
virtualized CPU, memory, storage, console, and cryptographic  
devices.  
This technology allows you to allocate a system resources  
into logical groupings and create multiple, discrete  
systems, each with their own operating system, resources,  
and identity within a single computer system.  
We can run a variety of applications software in different  
logical domains and keep them independent of performance and  
security purposes.  
  
A zone is a virtual operating system abstraction that  
provides a protected environment in which applications run.  
The applications are protected from each other to provide  
software fault isolation. To ease the labor of managing  
multiple applications and their environments, they co-exist  
within one operating system instance, and are usually  
managed as one entity.  
  
LDOMs allow you to assign your hardware to different virtual hosts for exclusive use. Say you have a T5220 with an 8-core CPU. You could create 8 LDOMs and assign a core to each. Each core will be for the exclusive use of the domain you assigned it to, regardless of the load on the other domains.

With zones/containers, all zones can use any CPU core/thread that they need when they need it. Limits can be set as far as how much CPU they use, but you cannot specify which CPU/core/thread they are to use.  
  
With LDOMs you get an OBP per domain that can be configured independently from the others. Zones don't have OBPs.  
  
You can "brand" a zone as a Solaris 8 or Solaris 9 zone (and I understand RHEL as well), running an instance of either OS on top of Solaris 10. This is useful in those cases when you are refreshing hardware but app vendors only support their apps on the older OSs.

You can't "brand" an LDOM, but you can create a branded zone inside an LDOM.  
  
Root on the global zone can see all of the file systems on each zone.  
Root on the primary domain cannot see the files systems on the domains.  
  
Solaris Containers / Zones  
------------------  
No special hardware required  
Single OS image  
Sub-CPU resource granularity  
Shared kernel, memory, file systems (configuration, resources and  
management)  
Solaris only (excluding Linux branded zone on x86)  
CPUs can be shared  
Works on all systems  
Virtually unlimited partitioning (max is 8191 non-global zones)  
Single system patch level  
Most admin operations can be applied to all containers in a single operation  
Very little performance overhead for zone infrastructure  
  
  
LDoms  
-----  
Sun4v systems only  
Multiple OS images  
Multiples of CPU granularity  
Dedicated kernel, memory, file systems  
Can support other OSes  
CPUs can not be shared (CPUs here refers to a strand/thread)  
Currently available on Tx000, T5xy0 only  
Partitioning limited to number of CPUs  
Multiple and different patch and release levels possible  
Each LDom must be fully managed separately

LDOM is a hardware virtualization which works on top of the hypervisor.

* Control Domain is the place where you are going to install the LDOM software and managing the complete logical domain environment. It used to configure the resources and guest domains

* Service domain provides the various virtual services to guest domains. Virtaul services can be virtual disks,network switches or virtual consoles. Mostly we will be configuring the control domain as service domain too.

<https://www.unixarena.com/2014/02/ldom-oracle-vm-sparc-tutorial-1.html/>

Man page sections:

The standard sections of the manual include:

1 User Commands

2 System Calls

3 C Library Functions

4 Devices and Special Files

5 File Formats and Conventions

6 Games et. al.

7 Miscellanea

8 System Administration tools and Daemons

Package administration:

From Solaris 11, IPS is tightly coupled with ZFS.

<http://pkg.oracle.com/>

‘Pkg’ is the main command to work with Solaris packages

SRU – support repository updates

FMRI – fault managed resource identifier

Pkg kernel info // pkg publisher

Pkg info –r apptrace = fetch package info remotely (-r) // pkg search “apptrace”

Pkg install –n apptrace = this will do a dry run, no actual install // pkg install apptrace

Pkg uninstall apptrace

Beadm = boot environment file system // beadm create solaris-1

Beadm activate solaris-1 // change the be

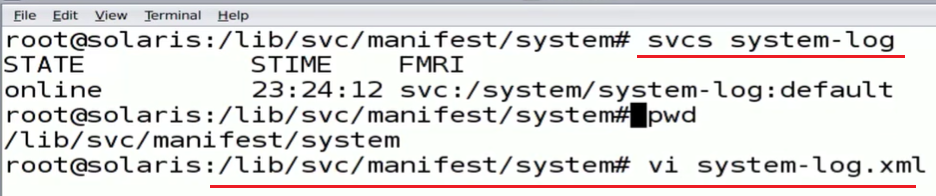
Mounting a BE onto a mountpoint : 

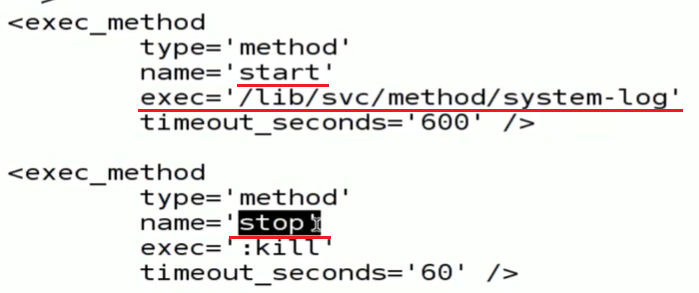
Installing on the mounted BE:



SMF – service management facility

Svcs –a = will show all the services

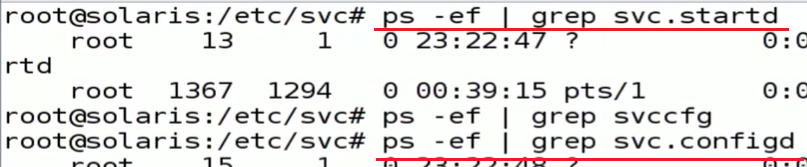




Every service has a manifest in xml.

/etc/svc has a depository.db it has all the service definitions. System uses this depository to start the services during the boot process.

Svccfg = Can interact with the depository using the



Svc.configd = interacts with the depository

Svc.startd = starts the services

Boot –m milestone = none –in a sparc machine when all services are down and want to switch to the repository.

Restore\_repository

LDOM: <https://docs.oracle.com/cd/E35434_01/pdf/E23807.pdf>

## Types of CPU scheduling Algorithm

<https://www.guru99.com/cpu-scheduling-algorithms.html>

There are mainly six types of process scheduling algorithms

1. First Come First Serve (FCFS)
2. Shortest-Job-First (SJF) Scheduling
3. Shortest Remaining Time
4. Priority Scheduling
5. Round Robin Scheduling
6. Multilevel Queue Scheduling

Pre-emptive, non-preemptive

* The number of processes that finish their execution per unit time is known Throughput.

-------------

Booting (boot strapping) cold boot, warm boot

Bootloader grub

<https://askubuntu.com/questions/173248/where-is-the-bootloader-stored-in-rom-ram-or-elsewhere>

49

ROM is a separate chip from the RAM. It does not require power to retain its contents, and originally could not be modified by any means, but came hard wired from the factory. Later PROM, or Programmable Read Only Memory replaced true ROM. These chips came blank from the factory, and could be written to once using a special procedure that essentially burned out bits of the chip causing their state to change. This was then replaced with EPROM, or Eraseable Programmable Memory. These chips had a little window on them and if you shined ultraviolet light into them, could be erased, allowing them to be programmed again. These were then replaced with EEPROM, or Electrically Erasable Programmable Memory. These chips have a special software procedure to erase them so they can be reprogrammed. ROM generally is still used to refer to all of these types generically.

The motherboard has some type of ROM chip that holds the firmware, which in PC parlance is usually called the BIOS, or Basic Input Output System, though it is being replaced these days with EFI firmware. This is the software that the CPU first starts executing at power on. All firmware performs initialization of the hardware, typically provides some diagnostic output, and provides a way for the user to configure the hardware, then locates and loads the boot loader, which in turn locates and loads the OS.

With PC BIOS, it simply loads and executes the first sector off the disk it decides to boot from, which typically is the first hard disk detected. By convention the first sector of a hard disk, called the Master Boot Record, contains a DOS partition table listing the locations of the partitions on the disk, and and leaves some space for the boot loader. Ubuntu uses the GRUB boot loader, which places enough code in the MBR to load and execute /boot/grub/core.img. Normally a copy of this file is placed in the sectors following the MBR, but before the first partition, and that is actually what the MBR loads, since finding the location of /boot/grub/core.img is too difficult to do properly in the very limited space available in the MBR.

The grub core image contains the base grub code, plus any modules needed to access /boot/grub so that it can load additional modules there, and the grub config file that describes what operating systems can be booted, and where they can be found.

The EFI firmware used on Intel Macs and becoming available as a replacement to BIOS on most recent PC motherboards requires a dedicated partition that holds boot loader files, and the firmware is smart enough to find those files and load one instead of just loading and executing whatever is in the first sector of the disk.

<https://www.oreilly.com/library/view/essential-system-administration/0596003439/ch04s01.html>

The normal Unix boot process has these main phases:

* Basic hardware detection (memory, disk, keyboard, mouse, and the like).
* Executing the firmware system initialization program (happens automatically).
* Locating and running the initial boot program (by the firmware boot program), usually from a predetermined location on disk. This program may perform additional hardware checks prior to loading the kernel.
* Locating and starting the Unix kernel (by the first-stage boot program). The kernel image file to execute may be determined automatically or via input to the boot program.
* The kernel initializes itself and then performs final, high-level hardware checks, loading device drivers and/or kernel modules as required.
* The kernel starts the init process, which in turn starts system processes (daemons) and initializes all active subsystems. When everything is ready, the system begins accepting user logins.

The configuration file ( /**boot**/**grub**/**grub**.**conf** ), which is used to create the list of operating systems to boot in GRUB's menu interface.

A **superblock** is a record of the characteristics of a filesystem, including its size, the block size, the empty and the filled blocks and their respective counts, the size and location of the inode tables, the disk block map and usage information, and the size of the block groups.

Superblock, inode, dentry

<https://unix.stackexchange.com/questions/4402/what-is-a-superblock-inode-dentry-and-a-file>

First and foremost, and I realize that it was not one of the terms from your question, you must understand *metadata*. Succinctly, and stolen from Wikipedia, metadata is data about data. That is to say that metadata contains information about a piece of data. For example, if I own a car then I have a set of information about the car but which is not part of the car itself. Information such as the registration number, make, model, year of manufacture, insurance information, and so on. All of that information is collectively referred to as the metadata. In Linux and UNIX file systems metadata exists at multiple levels of organization as you will see.

The **superblock** is essentially file system metadata and defines the file system type, size, status, and information about other metadata structures (metadata of metadata). The superblock is very critical to the file system and therefore is stored in multiple redundant copies for each file system. The superblock is a very "high level" metadata structure for the file system. For example, if the superblock of a partition, /var, becomes corrupt then the file system in question (/var) cannot be mounted by the operating system. Commonly in this event, you need to run fsck which will automatically select an alternate, backup copy of the superblock and attempt to recover the file system. The backup copies themselves are stored in block groups spread through the file system with the first stored at a 1 block offset from the start of the partition. This is important in the event that a manual recovery is necessary. You may view information about ext2/ext3/ext4 superblock backups with the command dumpe2fs /dev/foo | grep -i superblock which is useful in the event of a manual recovery attempt. Let us suppose that the dumpe2fs command outputs the line Backup superblock at 163840, Group descriptors at 163841-163841. We can use this information, and additional knowledge about the file system structure, to attempt to use this superblock backup: /sbin/fsck.ext3 -b 163840 -B 1024 /dev/foo. Please note that I have assumed a block size of 1024 bytes for this example.

An **inode** exists in, or on, a file system and represents metadata about a file. For clarity, all objects in a Linux or UNIX system are files; actual files, directories, devices, and so on. Please note that, among the metadata contained in an inode, there is no file name as humans think of it, this will be important later. An inode contains essentially information about ownership (user, group), access mode (read, write, execute permissions), file type, and the data blocks with the file's content.

A **dentry** is the glue that holds inodes and files together by relating inode numbers to file names. Dentries also play a role in directory caching which, ideally, keeps the most frequently used files on-hand for faster access. File system traversal is another aspect of the dentry as it maintains a relationship between directories and their files.

A **file**, in addition to being what humans typically think of when presented with the word, is really just a block of logically related arbitrary data. Comparatively very dull considering all of the work done (above) to keep track of them.

#### **What is an RPM?**

<https://rpm-packaging-guide.github.io/>

An RPM package is simply a file containing other files and information about them needed by the system. Specifically, an RPM package consists of the [cpio](https://en.wikipedia.org/wiki/Cpio) archive, which contains the files, and the RPM header, which contains metadata about the package. The rpm package manager uses this metadata to determine dependencies, where to install files, and other information.

There are two types of RPM packages:

* source RPM (SRPM)
* binary RPM

SRPMs and binary RPMs share the file format and tooling, but have different contents and serve different purposes. An SRPM contains source code, optionally patches to it, and a SPEC file, which describes how to build the source code into a binary RPM. A binary RPM contains the binaries built from the sources and patches.

<https://www.tecmint.com/20-practical-examples-of-rpm-commands-in-linux/>

\*starend