

Software Configuration management:

It is the ability to control and manage the changes in a software project. (Software Development Life Cycle).

Q: What is the Version Control? Source Code Management tool.
management of changes to documents or source code or other information.

Q: Why we use version control system?

- ✓ Parallel Development
- ✓ To track the version of files & documents.
- ✓ To track the changes that are done for files.

→ Different types of version control tools are:

1. CVS
2. Subversion (SVN) - from Apache
3. GITHUB
4. Mercurial
5. ClearCase - from IBM
6. MONOTON
7. Bazaar etc..

linode.com

Parallel Development:

It is nothing but a no. of users will work on a same file at a same time.

Diff. types of changes in SDLC:

- Code changing
- Requirement changing
- Design changing
- Server infrastructure may change
- Software version etc..

Version control system is, for a project there are so many developers. They are present at different geographical locations. In order to join i.e., to group the work done by the developers we will use version control system. That means some developers may change code & write a new code, this will be accessed by other developers at different location, all these modifications, updates and codes will be present in version control system.

Repository:

It is a centralized location or place where all the data is stored.

(or)

It is an interface of the data base to get the data. It contains all the data i.e., kept in the SVN.

In Three ways we can access the Repository.

1. HTTP [Apache + SVN] - for Windows
2. SVN + ssh - for Linux
3. SvnServe

- At the time of installation of SVN, we have to decide in which way we want to access the repository to get the data.

Ex:

for Apache SVN: `http:// <svn ip Add..> / repo / FB-V1.0`

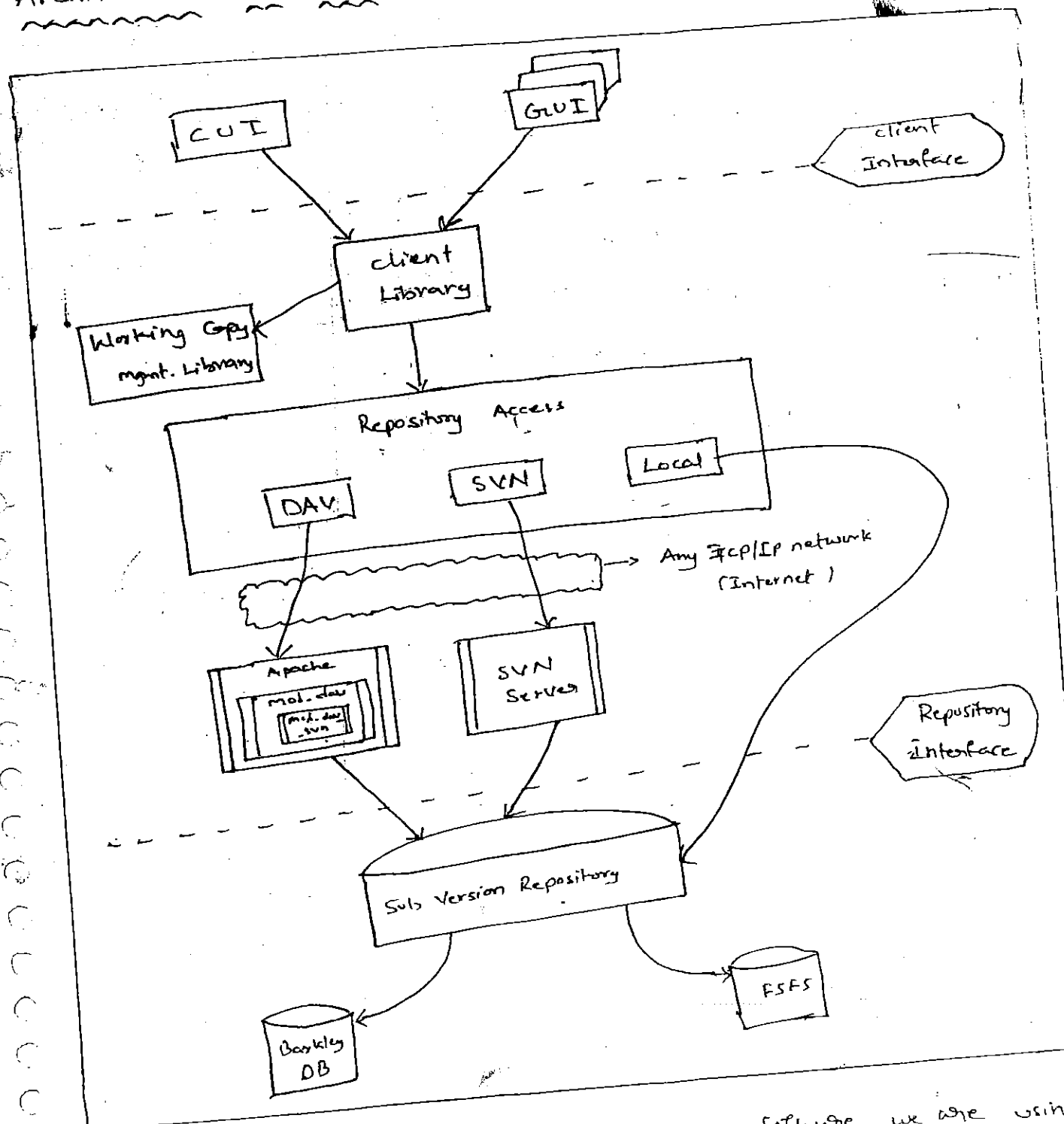
for SVN + ssh : `svn+ssh://username@svnip.../home/1SVN/repo / FB-V1.0`

In order to access the data using svn there are two types

1. CUI (character User Interface) :- Command-Mode
2. GUI (Graphical User Interface) .

In GUI we will install a s/w called SVN Tortoise client.

Architecture of SVN :



- Here the client library is nothing but a software we are using to access the repository. For example in Apache SVN the client library nothing but a SVN Tortoise client s/w.

Accessing Access:

- DAV - For Apache svn
- SVN - for svn+ssh or svnserve
- Local - It is nothing but a our own system will act as a server i.e., svn is installed in our system and in the same system we are accessing repository.

↳ For DAV & SVN, the server is in some where location.

We have to access the using internet (Any network), where as in Local No network is needed.

* Note: ↳ We cannot retrieve a single file using SVN. we have to retrieve the whole directory, where that file exists.

↳ To know the version of svn i.e., installed in our

system:

```
C:\> svn --version
```

```
o/p: svn version 1.7.3
```

```
compiled Feb 14 2012 12:...
```

Note: ↳ In Interview point of view, we can tell, we are using svn v1.6.

Installation of Apache SVN:

- Install Apache,
- Install SVN
- Integrate Apache & SVN

Note: ↳ To know the Apache is installed properly or not, we use the following

In browser: <http://localhost:portName> Enter

o/p: It Works!

Integration of Apache & svn:

↳ Copy 2 modules (mod_authz_svn, mod_dav_svn) from
svn\bin folder

↳ Paste these 2 modules in c:\Apache\modules folder.

↳ In Apache → httpd file write on that following.

```
LoadModule dav_svn_module modules/mod_dav_svn.so
LoadModule dav_module modules/mod_dav.so
LoadModule authz_svn_module modules/mod_authz_svn.so
```

And Save the file.

Creating Repository:

↳ We can create the Repository in any drive (C: or D: or E)

↳ In Command Prompt

c:\> svnadmin create <Repository name>

Then repository will be created.

Structure of Repository:

```
Repo/
  /conf/..
  /db/..
  /hooks/..
  /locks/..
  /format/..
  /Readme
```

} Folders

} Files

↳ Conf Directory:

- It contains all the configuration details of svn.
- It contains authz, passwd & svnserve files.

db directory

- Contains all the database related files.

Notes: SVN maintains the data in a different manner.

This db directory contains that logical manner.

hooks:

- Different types of Conditions will be stored in this directory.

Locks:

It contains database Locks.

Integrating SVN with Repository:

[c:\Apache\conf\httpd.conf]

→ In Apache folder we have httpd file. At the end write the following code:

```
<Location /Repo>
  DAV svn
  SVNPATH C:\Repository\location>
</Location>
```

```
<Location /svn>
  DAV svn
  SVNPATH c:\Desktop\repo
</Location>
```

→ Now in Web browser type the following url

http://localhost/repo

dp: Revision - 0 (It means there is no data in repository)

CHECK OUT:

→ It means downloading the data from SVN Server repository to our local system.

Note: It will be applicable for entire directory.

CHECK IN:

→ It means uploading or sending the data from local system to the SVN repository.

Note: It will be applicable for a single file or folder.

Basic commands :

- svn checkout or svn co
- svn checkout or svn ci
- svn Commit
- svn add
- svn import
- svn export
- svn info
- svn log
- svn copy
- svn merge
- svn update
- svn delete
- svn diff

- svnadmin load
- svnadmin dump
- svn status
- svn mkdir
- svnadmin hotcopy
- svn lock
- svn cleanup
- svn cat
- svn switch
- svn move
- svn blame
- svn info
- svn-revert
- svn unlock

Note :

- In order to know that our svn server is empty or not, we have to checkout a directory. Then if there is data in repository it will be downloaded to our directory otherwise, if there is no data in repository only .svn folder will get in our directory.

→ After -Creating Repository, we have to create 3 important folders

1. Tags
2. Trunk
3. Branches

Revision: It describes the state of SVN at that particular point of time

→ Revision is for whole SVN, not for a single file or folder.

→ Revision '0' means that, it is the starting point of svn. i.e., if repository is empty.

Q: What is a Revision Created in svn?

Whenever there is a change in the state of svn, i.e., from previous state to present state then a new revision will be created.

Note:

- Revision is always incremented by '+1'
- Revision is created for checkin only not for a checkout.
- Whenever we use the following commands a new revision will be created.

svn ci

svn copy

svn commit

svn import

svn delete

} for these commands we have to write a message in the Description column.

Note:

→ When we checkout some data and if we didn't do any modifications for that, and we are trying to checkin that data, the svn will not accept because there is no modifications made to that data.

→ In svn HEAD Revision means, it is the latest Revision.

Svn Merge:

It is used to merge two revisions.

Ex: svn merge -r 4:5 http://localhost/Repo

Svn Update:

This command gives the updated version data or latest data i.e., present in svn.

svn log:

This command gives the history of logs.

Ex: svn log http://localhost/Repo

Access Permission for svn Repository:

- In svn we can give access permission to groups and particular users.

→ Here we can give permissions to folder only not files.

- First we need to know the path of the file. Then go to the following process:

→ Login to svn server

→ go to the location where repository created.

→ go inside that repository → conf → auth3.conf &

write syntax:

groupName =	name1, name2, ... etc
↓	↓
groupname	username,

Ex:

DEV_FB_2.0 = Sudhakar, Anji, Ramo, Vasu

[/facebook/branches/FB-2.0]

@ DEV_FB_2.0 = rw.

Ex:

[/facebook/branches/FB-2.0]

& username = rw

→ In the above r is for checkout (read)
w is for checkin (write)

Ex:

[/facebook/branches/FB-2.0]

@ xyz = r

& Sudhakar = w

→ In the above example

DEV_FB_2.0 group having read & write permissions

xyz group having only read permissions

Sudhakar having only write permissions.

Branching Strategies:

There are two types of branching strategies

1. Stable Trunk Branching - strategy

or

Early Branching strategy.

2. Unstable Trunk Branching strategy

or

Late Branching strategy

or

Deferred Branch strategy.

Branching strategy:

It shows the how the data is aligned in SVN.

→ In any organisation, they can use either any one of this strategy.

→ Most of the organizations follow stable trunk branching strategy.

Stable Trunk Branching Strategy:

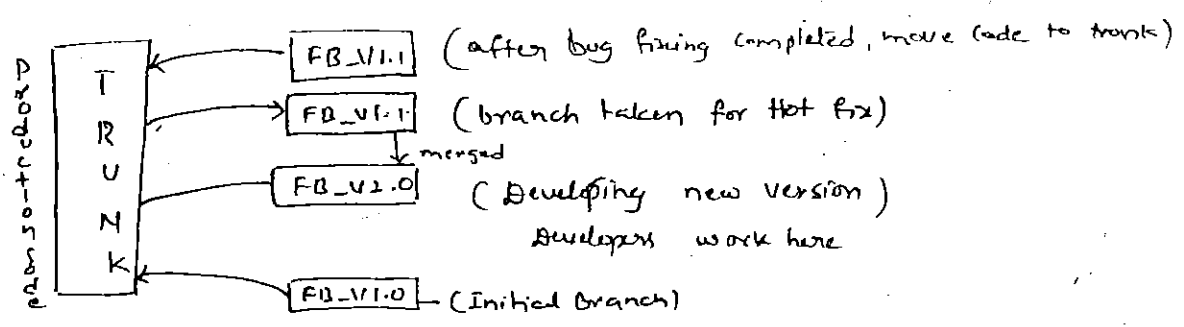
→ In this branches are the place where main line development happens. i.e., Developers checkin and checkout their code in branches.

→ Trunk will always contain production level code only.

→ In this strategy branches are created during the starting of your release. so, it is called Early Branching strategy.

→ Tags contains snapshot your branches

→ After a release the code will go to production. The branch code will merged back to trunk.



Advantages:

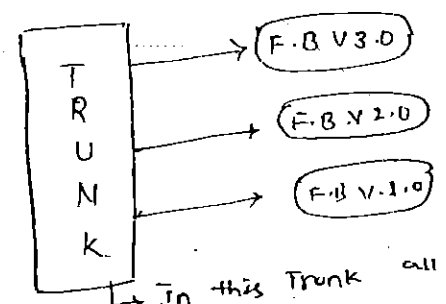
- This strategy used for complex project with more no. of development team.
- This strategy facilitates parallel development.

DisAdvantages:

- Merging is overhead in this process (frequently we need to merge your code to trunk).

Unstable Trunk Strategy:

- In this Branching strategy trunk is the place where mainline development happens. i.e., Developers did checkout & checkout from trunk
- Branches contain production level code
- Tags contain Snapshot of Trunk, because development happens in trunk.
- In this strategy branches are created during the end of release so, it is called late branching strategy.
- Once a release goes to production a new branch will create.
- There is no need of merging in this strategy.



- In this Trunk all the developers will work here.
- In this strategy all the checkins are very very important.

New Version = Previous Version + some new code

Advantages:

- This strategy is used in smaller teams and less complex project.
- There is no bother about merging.

Disadvantages:

- It doesnot facilitate parallel development
- Checkins

Release:

Delivering the product to the client. The product needs to change depending upon the client requirement.

→ We have different kinds of releases.

major Release → Having so many changes

Minor Release → Having small changes

Maintenance Release → Security & Infrastructure

Hotfix Release → fix the code at development level

Ex: Jdk 1.7.0.3

1 - major

7 - minor

0 - maintenance

3 - Hotfix.

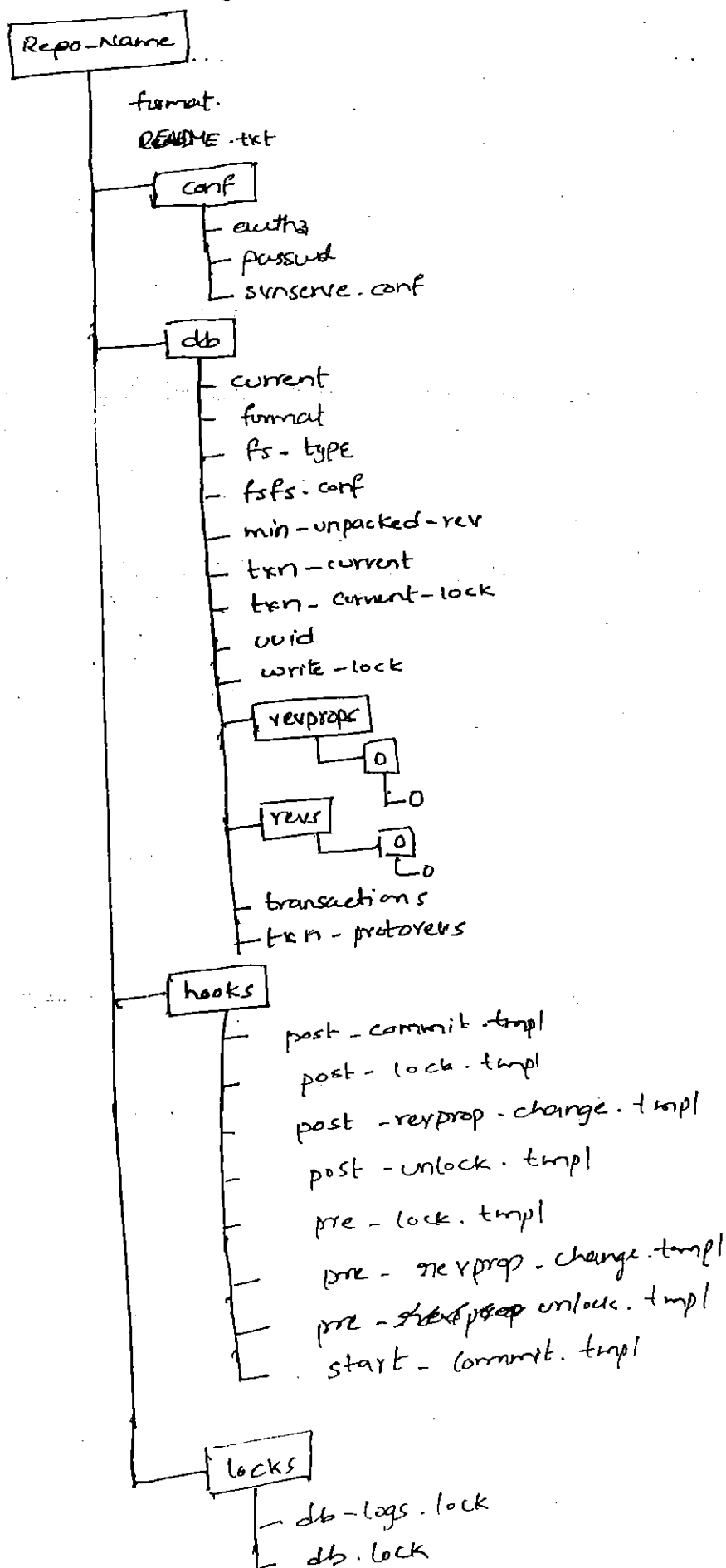
Note:

- We have another branching strategy, i.e., Feature strategy.
- This is used in Agile process model.

Q: How to create branches in SVN?

svn copy < source url > < destination > -m "message"

Repository Directory Structure :



svnadmin hotcopy:

- Make a hot-copy of a repository.

Syni `svnadmin hotcopy REPOS_PATH NEW-REPOS-PATH`

Desc The subcommand makes a full "hot" backup of your repository.

Including all hooks, Configuration files, and of course, database files.

- If you pass the `--clean-logs` Switch, `svnadmin` will perform a hotcopy of your repository, and then remove unused Berkeley DB logs from the original repository.

- You can run this command at any time and make a safe copy of the repository. Regardless of whether other processes are using the repository.

Value 1000

1000

Q: How do you create a svn repository backup?

A: `svnadmin dump <repository name> > E:\svn.bkp`

Q: How to restore svn from backup file?

`svnadmin load new-repo < E:\svn.bkp`

Tags:

- Tags are nothing but a snapshot.
- In stable Trunk strategy, The Snapshot of branch
- In Unstable Trunk strategy, The Snapshot of Trunk
- Tags are meant to reproduce our old builds
- We can create Tags by using svn copy command.

Naming Conventions:

Convention is not a rule. we cannot follow or not. It is not a mandatory for project. But the more convenient we can use it.

Ex: FB-R2.0

FB-R2.0.0.1

Syntax: <projectname>_<Release>

Base Line:

It serves as a reference to the future development.

- We can maintain Base line by using Tags, trunk.

.svn folder :

→ The .svn directory contains all the metadata about the working copy, including pristine copies of all the files & the properties of the files and references to the repository.
(or)

→ .svn folder is the administrative folder of your working copy.

→ Subversion calls the .svn directory as an administrative directory.

Working Copy :

→ A working copy is a folder that is checked out from a svn server, which contains .svn folder inside.

→ Having a hidden .svn directory is what makes an ordinary file system directory into a Subversion working copy directory, a directory that Subversion can recognize and manage.

→ The files in each administrative directory help Subversion to recognize which files contain unpublished changes and which files are out of date with respect to others work.

→ .svn directory contains pristine (unchanged) copies of files that are downloaded from the repository.

Note :

If we delete the .svn folder in our working folder, we cannot commit or checkin our folder.

Hotfix:

- When a bug is arised in the production code ~~and~~ that need to be fixed then the Hotfix will be created.
- Hotfix is nothing but a fixing the bug that is arised in the production code.

Q: How do you handle a Hotfix release?

we can handle a HF release in two ways.

1. By creating branch from the trunk.

- The HF Release branch is merged with next release branch such that HF code fix is also present in the next release branch.

- After HF goes to production we should merge HF branch with trunk.

2. A Hotfix branch can be created from a Branch:

- The HF release branch is merged with next release branch. Such that HF Code Fix is also present in next release.

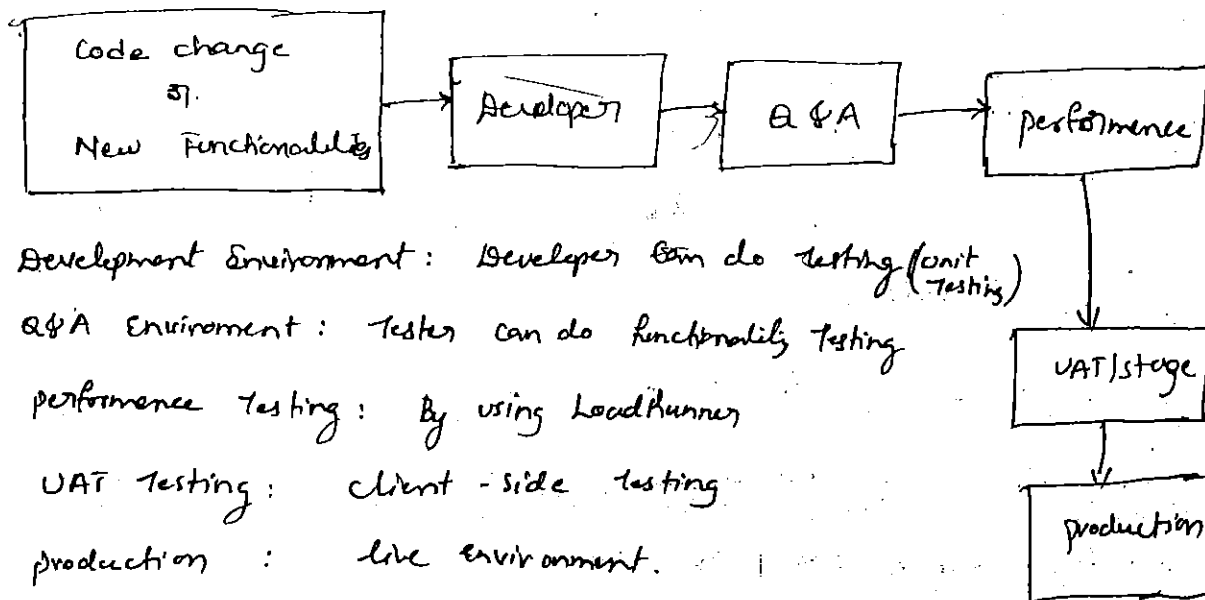
- After HF goes to production we should merge HF branch with trunk.

Environment:

It is the combination of H/W and Software and business logic that makes an application work.

We have different kinds of environments:

- ✓ Development Environment
- ✓ QA Environment
- ✓ Performance Environment
- ✓ UAT Environment
- ✓ Production Environment.



Build:

- Build is the process of converting our source code in to software artifacts

Buids are two types

✓ Nightly Builds:

- Nightly Builds are the ones that get Deployed in development Environment.

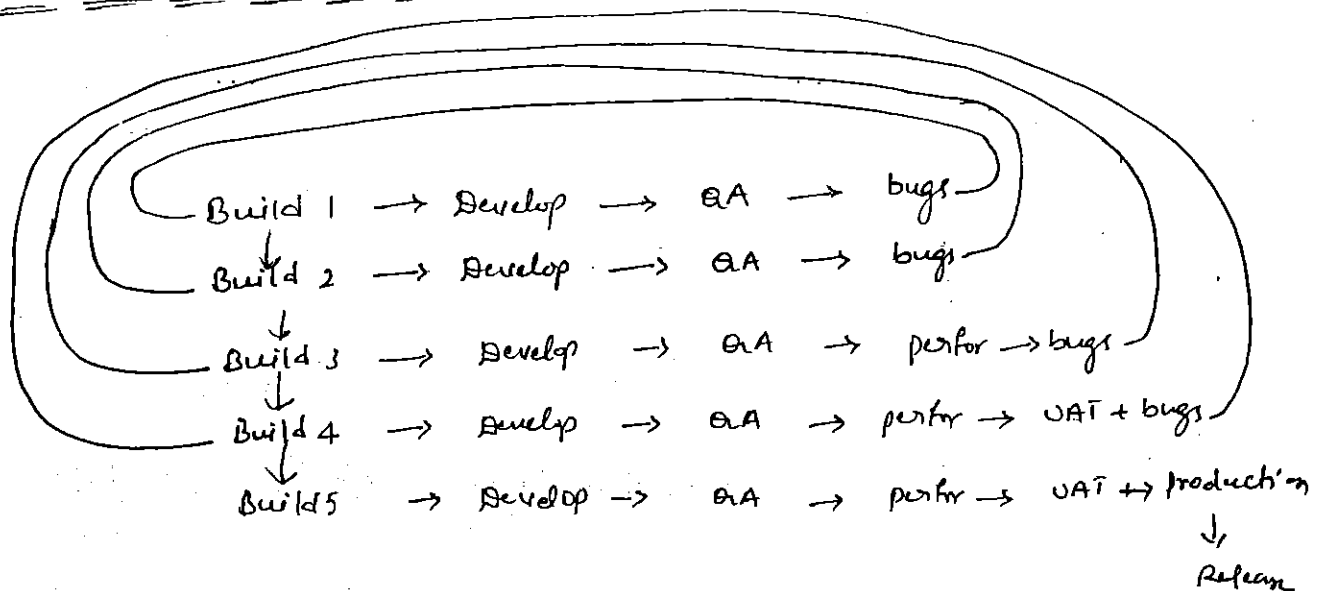
- They are meant to check the Surity of your code only

- This scope is with in the Development Environment only.

✓ Milestone Builds:

- Milestone Builds are the ones that get Deployed in Higher Environments.

Build Process Life-Cycle :



Ex: F.B V1.0 = F.B V1.0 + New change
+ (or)
production code + New changes

Build 1 = Build + N.C

Build 2 = Build 1 + N.C

Build 3 = Build 2 + N.C

Build 4 = Build 3 + N.C

Build 5 = Build 4 + N.C

Note:

QA - Quality Assurance

UAT - User Acceptance Testing

Q: Difference between Branch & Tags:

As we are following stable Trunk strategy in our organization.

- The snapshot taken from the branches, because in branches only the code modifications takes place. The code in trunk is stable.

- Tag is a cheap copy (logical copy). Its not physical copy.

- No one can make changes to tags.

- Tags are used to reproduce old builds or to get the old builds source code.

- In unstable Trunk strategy, tags are created from Trunk, because the modifications takes place in trunk.

Q: Will developers checkin tags?

If the developers have the access permissions in authz file then they can checkin otherwise No.

↳ Tags are only meant for checkouts, not for checkin.

↳ If developers start checkin tags, then there is no difference b/w tags & branches.

↳ When we try to checkin tags using Tortoise client.

It gives a warning message. (But we can checkin).

Q: How will you create a branch?

Using svn copy command we can create a branch.

In order to create a branch we must have 2 things

1. From where we want to create a branch (Source).

2. That the new name should be kept for that.

Syn: `svn copy <source-url> <dest-url> -m <message>`

Q: How do you create tags?

using svn copy command

svn: `svn copy <source-uri> <dest-uri> -m "message"`

* Naming Convention for tags:

→ Tag is for Each and Every module. It is not for a whole branch

Ex: chat-build

FriendRequest-build

here for chat & FriendRequest modules were created a tags with names chat-build & FriendRequest-build.

Q: Difference b/w svn export and svn checkout?

- If we use svn checkout option then we will get a .svn folder inside our working copy.
- If we use svn export option then we will not get .svn folder inside our working copy. But the remaining data present in the repository will be downloaded.
- The directory that used svn export option will not be checkin to the svn server and it is not under the control of svn server because there is no .svn folder.

Q: Difference b/w svn import and svn checkin?

- We know that a directory that contains .svn folder will be checkin to the svn server by using svn checkin option.
- But svn import command is also used to checkin a directory that doesnot contain .svn folder and that directory should not present in the svn server

Q: When do you create branches in svn?

As we are following stable trunk branching strategies, Branches are created during the starting of the release. Then the developers work on that code for further release.

Q: Why do we need tags?

To reproduce the old builds source code we will create tags.

Q: What is meant by reproduce?

Q: When do you create a tag?

Whenever some build is going to be higher

Build & Release

- Linux
- SVN - Version Control tool - SCM tool
- ANTI - Build tool
- Hudson / Jenkins - CI tool
- Apache
- Tomcat - App. Server
- Deployments
- shell scripting Basics
- Jira [Bug tracking] - Ticketing tool

↳ Build

↳ Deployment

↳ Release

→ Build and Deployment (Standalone eg: windows 7, etc..)

→ Build and Release (Webbased eg: facebook, mete..)

↳ ~~Java~~ Executable Components

Java - Jar, war, ear

.Net - .msi, .dll

C, C++ - obj files, .rpm

Build Tools

• Java - ANT, MAVEN

• .Net - NANT, MSBUILD

• C, C++ - make files

- SVN - SubVersion - Version Control System

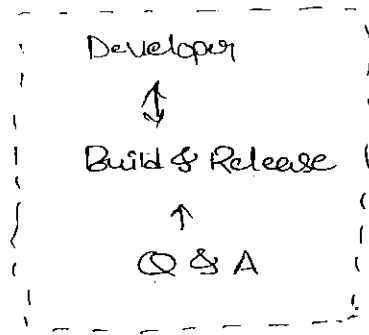
- Admin

- Installation

- Troubleshooting

- maintenance

1. In SDLC (Software Development Life Cycle) at implementation
(Deployment)
time Build & Release will come.



LINUX

Operating System:

- The Software that Controls the Hardware.
- Operating System is the set of programs that controls a computer so that makes hardware usable.
- Controlling Computer involves software at several levels.

✓ Kernel Services

✓ Library Services

✓ Application level Services.

} part of O/S.

- The core of O/S is kernel

* Kernel is a control program that functions in privileged state, reflecting to interrupts from external devices and to service requests and traps from process.

- The operating system is an interface between H/w & s/w.

Features:

- ✓ Multi-users
- ✓ Multi-tasking
- ✓ Portability
- ✓ Machine Independent
- ✓ Multi-programming
- ✓ Hierarchical File System.

- Operating system is a collection of system software programs which is coordinating the actions of the computer. This is also the resources of the system. & O/S is a software component.

Types of Software:

✓ Application Software :

The Software used by the users of the computer

✓ System Software :

The Software used by the computer.

Types of Operating Systems:

1. Single User & Single tasking OS
2. Single user & multi tasking OS
3. Multiuser & multi tasking OS.

→ The UNIX OS is made up of three parts:

1. kernel
2. Shell and
3. programs.

kernel:

The kernel of UNIX is the hub of the OS. It allocates time and memory to programs and handles the file store and communication's in response to system calls.

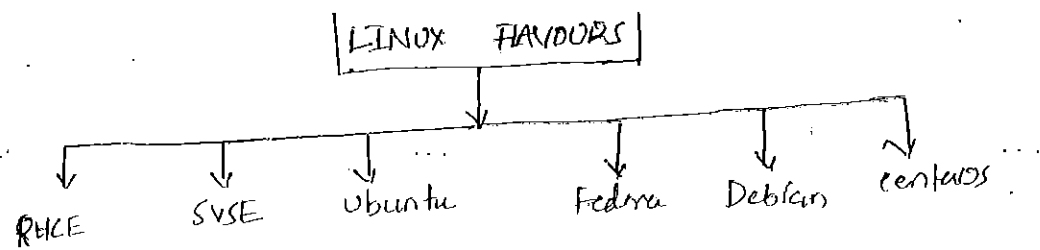
Shell:

The shell acts as an interface b/w user and the kernel.

Linux:

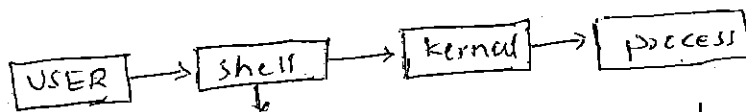
In 1969 → kernel of UNIX OS is developed in assembly language.

In 1992 → Linux is implemented by L.



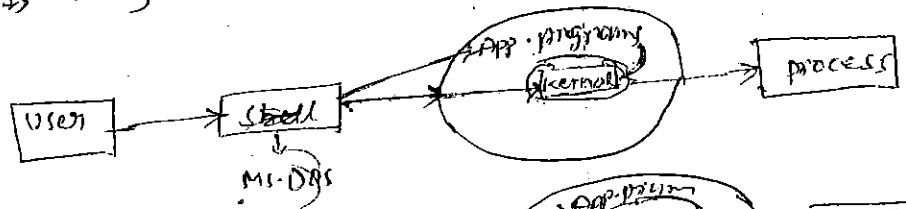
- To connect windows to linux server → use PTTY. (External slow)
- Linux to Linux → ssh → In built.
- windows to windows → MSTSC → In built.

UNIX ARCHITECTURE :

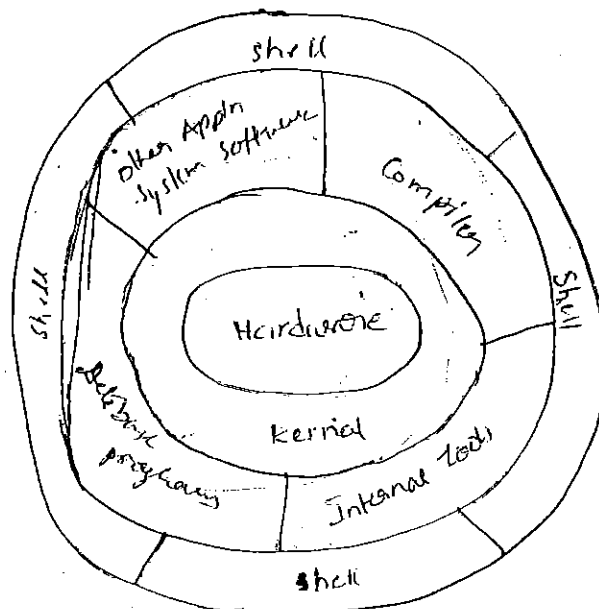
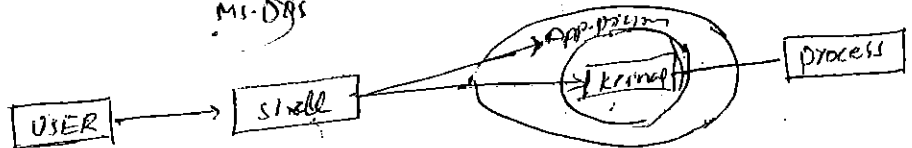


It's nothing but MS-DOS prompt.

In Windows:



In UNIX:



UNIX ARCHITECTURE

- Kernel is a part of the UNIX/LINUX OS loaded into the memory when the system is booted. It manages the system-resources, allocates the time b/w users & processes, decides process priorities and perform all the other tasks.

- Kernel is an interface b/w shell and hardware.
- Shell is an interface b/w user and kernel. And it is the "Command Interpreter". In Linux/Unix there exists different types of shells.

UNIX:

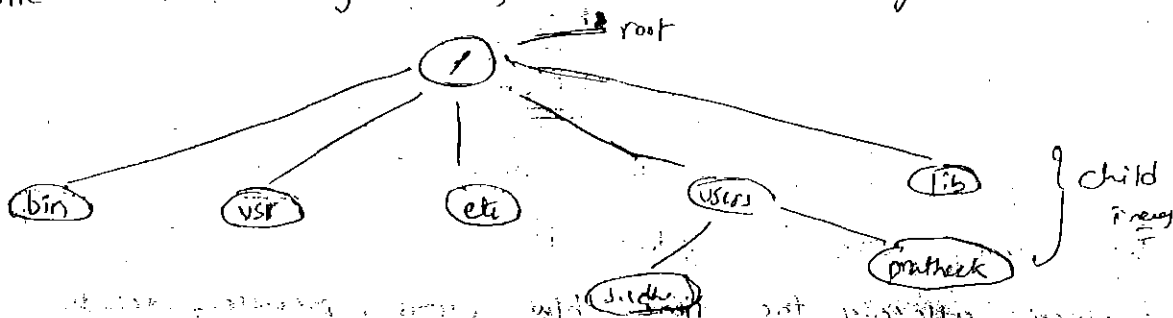
- ✓ Bourne shell developed by Stephen Bourne
- ✓ Korn shell developed by Keith Korn
- ✓ C shell developed by Jan Bell.

LINUX:

- ✓ Bash shell (Bourne again shell)
- ✓ tcsh shell (turbo C shell)

UNIX File System Structure:

- One of the features which make Unix especially attractive is the structure of its file system.
- UNIX makes use of an inverted branching tree file structure.
- The Tree trunk consists of a single large file "/" (root) which contains all the files on the system. Any number of sub-directories may branch from this main trunk.
- The UNIX file system is Hierarchical File System.



`/` : This is top level directory.

It is parent directory for all other directories.

It is called as root directory.

`/bin` : It contains commands used by all users.
Binary Files.

`/boot` : It contains OS boot related files.

`/dev` : It contains device related files [`/dev/hda` .. for hard disk]

`/etc` : It contains all configuration files [`/etc/passwd` ... `/etc/passwd`]

`/home` : It is home directory for other users.
It provides working environment for other users.

`/lib` : It contains library files which are used by OS.
It is similar to all files of windows.
Library files in linux are SO (shared object) files.

`/mnt` : It is default mount point for any partition.
It is ~~default~~ empty by default.
It contains external devices that connected to the OS.

`/opt` : It is optional directory for user.
It contains third party softwares.

`/usr` : by default softwares are installed in `/usr` directory.
(UNIX shareable resources)

`/sbin` : It contains commands used by only super user (root).
(super user's binary files).

`/root` : It is home directory for root user.
It provides working environment for root user.

`/var` : It is containing variable data like mails, log files.

tmp: It is entirely temporary files

/proc: It contains process files
It contents are not permanent, they keep changing
It is also called as Virtual Directory
It's file contain useful information used by o/s:

UNIX/LINUX COMMANDS:

Command Syntax:

CommandName -[options] [arguments]
 ↓ ↓ ↓
Mandatory optional optional

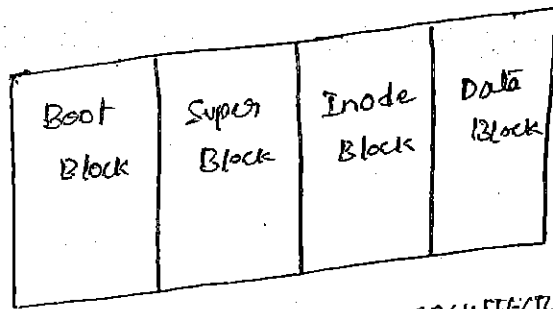
options: Change the behaviour of Command.

1. ls:

UNIX FILE SYSTEM ARCHITECTURE :

In Unix Environment a hard disk partitioned into several physical & logical blocks. These blocks contains the following important areas.

1. Boot Block
2. Super Block
3. Inode Block
4. Data Block



UNIX FILE SYSTEM ARCHITECTURE

Boot Block:

IT stores all bootable objects that are necessary for booting the system. It has completely bootable information.

Super Block:

- It stores all the information about the system like
- Size and status of the file system
 - Number of blocks allocated
 - Number of blocks unallocated
 - File system state

Inode block:

This block contains all the information about a file except its name. It contains the following details.

- Type of file.
- Mode of the file
- Owner of the file
- Date & time last modified
- No. of Hardlink to the file.

Data Block:

These blocks are storage blocks contains the rest of the space that is allocated to the file system.

UNIX File and Their Colours:

In UNIX Environment there are several categories which are related to the following colours.

<u>FILE NAME</u>	<u>SYM</u>	<u>Colours</u>
1. Normal files or Regular files	(-)	Black & white
2. Directories files	(/ or d)	Dark Blue
3. Linked Files	@	Blue
4. Compressed Files (or) Zip formatted (.gz or .z)		Red colour
5. Device Files		Yellow colour
6. Executable Files	(*)	Green colour

Relation between files \rightarrow

mediator the commands !

Wild Card characters or Meta characters :

<u>Meta character</u>	<u>Function</u>	<u>Example</u>	<u>Meaning</u>
- ^	Beginning of line anchor	/^love/	Matches all lines beginning with love.
- \$	End of line anchor	/love\$/	Matches all lines ending with love.
- .	Matches one character	/l.o.e/	Matches line containing l, followed by two characters, followed by e.
- []	Matches one in the list	/[LL]ove/	Matches lines containing love or Love
- [X-Z]	Matches one character within a range in the set	/[A-Z]ove/	Matches letters from A through Z followed by ove.
- [^]	Matches one character not in the set	/[^A-Z]/	Matches any character not in the range between A-Z
- \	used to escape meta-character	/love\. /	Matches lines containing love, followed by literal period; Normally the period matches one of any character.

Handwritten text along the right margin, possibly a page number or date.

Some basics of the boot process.

- ✓ When a Linux System boots, it must first locate and load the kernel.
- ✓ When the kernel finishes loading process, it loads and passes control to some initial process, usually the program init.
- ✓ The init process goes through its initialization procedures, part of which is to enter the currently defined run-level. Run-levels among other things determine if the system will display a text or graphical login prompt at the console.
- ✓ For text mode login, init spawns one or more getty-like processes and associates them with the appropriate terminals. In the case of actual serial terminals, a simpler getty-like process is usually invoked such as lcbt/mingetty.
- ✓ To provide for graphical logins, init spawns one of the graphical display managers such as xdm, gdm or kdm.
- ✓ The getty, agetty or mingetty process is the second of the three programs (init, getty and login) used by the system to allow users to login. getty or agetty is invoked by init to:
 - Open tty lines and set their modes.
 - Print the login banner, usually the contents of /etc/issue.
 - Spawn a login process for the user, usually /bin/login.
- ✓ The login program is what actually prompts the user to input their username and password. login will then validate the user and start the shell defined in the user's corresponding /etc/passwd entry.

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

Linux Commands

1. ls: Displaying directory contents

ls - List Directory Contents

- -a seeing hidden files
- -l long listings
- -d show directories not contents
- -h human readable file sizes
- -R recursively list sub directories
- -S Sort file list by size

- The ls Command is used to list the contents of a directory.

\$ ls

shows all files ~~(including "hidden" dot files)~~

\$

ls -a

show all files including "hidden" dot files

\$

ls -l

show long listing of files ~~with "human readable" file sizes~~

\$

ls -lah

show long listing of all files with human readable file sizes:

\$

ls -ls

show long listing sorted by file size.

\$

ls -lc

show long listing sorted by file's creation timestamp.

\$

ls -lc

show long listing sorted by file's access timestamp.

\$

ls -ld */

2. pwd: Print working Directory.

... pwd to see what the current directory

```
$ pwd
```

/home/Sudha

- path to current working directory.

3. cd:

The cd command changes the current directory.

```
$ cd .. - parent directory
```

Note: If you are logged in as root, you will be taken to /root - which is the user's home directory.

↳ The . character represents the current directory. Typing cd. at the command line has no effect because you will stay in the same directory.

↳ Typing cd .. will take you to the parent of the current directory.

↳ `cd ~root` will take you to username's home directory
↳ username

4. free:

The amount of physical memory in the system and usage can be obtained by running the free command.

```
$ free -m
```

	total	used	free	shared	buffers	cached
Mem:	1011	419	591	0	15	303
-lt buffers/cache:		100	910			
Swap:	2604	0	2604			

1) Uses: 5. df:

- The df command shows how much space each filesystem is using on the disk and where it is mounted.
- Using the -i options shows inode usage instead of free space.
- With the type (-T) and the human readable (-h) options df shows the filesystem type, and counts sizes in megabytes and gigabytes.

\$ df -hT

Filesystem	Type	Size	Used	Avail	Use%	Mounted on
/dev/sda2	ext3	3.8G	1.9G	1.8G	52%	/
/dev/sda1	ext3	46M	9.2M	36M	22%	/boot
tmpfs	tmpfs	506M	0	506M	0%	/dev/shm
/dev/sda3	ext3	3.4G	73M	3.2G	3%	/home

\$ df -h

This command is used to know the utilization of hard disk in human readable format.

\$ 6. du

- The du command shows the estimated disk space used by files and directories.
- If you execute du without any options it will output the sizes of all files starting in your current directory and all sub-directories of your current directory.

-h human readable sizes

-s Summarize, display total only for each argument

\$ du -sh /home

1.7M /home

\$ du -sh /home

\$ du -Ssh /home

8.0K /home.

7. uname

It describes what type of system is this?

- It gives hostname of the Linux box.

- It gives us kernel version installed in our system

- It gives the info whether it is a 32 or 64 bit OS.

`# uname -l`

Linux

`# uname -a`

```
Linux localhost.localdomain 2.6.18-8.el5 #1 SMP Fri Jan 26 14:15:21  
EST 2007 i686 i686 i386 GNU/Linux
```

-n - prints the node name

-r - prints the kernel release

-p - prints the processor type

-m - prints the hardware platform

8. uptime

- It gives information when the system reboot.

- It gives no. of users logged in to Linux box.

- It gives load on our system.

`# uptime`

07:07:39 up 2min 4 users, load average: 1.67, 0.84, 0.34

Q: How to see the version of kernel use

`# uname -r`

2.6.18-8.el5

Q: The same information can be seen in /boot/grub/grub.conf

Networking Details:

To determine network details, the `hostname` command can be used to determine the hostname as well as the domain name.

9. hostname

```
# hostname <
localhost.localdomain
```

```
# hostname -s <
localhost
```

```
# hostname -d
localhost.localdomain
```

```
# hostname -i
127.0.0.1
```

10. ifconfig

→ know ip addresses

```
# lsbin/ifconfig
```

cat /proc/cpuinfo : used to know the cpe info of our system.

```
# cat /proc/cpuinfo
```

processor : 0

vendor_id : GenuineIntel

cpu family : 6

model : 37

model name : Intel(R) Core(TM) i3 CPU M 350 @ 2.27GHz

stepping : 2

cpu MHz : 2259.952

cache size : 3072 KB

fdiv-bug : no

hlt-bug :

~~Deleting and creating files:~~

Touch: Creates an empty file or updates mtime and atime on existing files.

- Touch command will update the access (atime) and modification (mtime) times of each specified file to the current time or to a specified time, or it will create a new, empty file for each specified file that does not already exist.

\$ touch filename

options: -c: to avoid creating new file.

-a: change only the access time

-m: change only the modification time

-t: set a specific time.

\$ touch file

\$ ls -l file

-rw-r--r-- 1 root root 0 sep 1 9:44 file

Cat: displays entire files.

~~cat~~ cat dumps a file to STDOUT in its entirety and without stopping.

- It is read only.

More: more displays a file one screen at a time. It loads an entire file before it is displayed. So it is often unsuitable for very large files.

\$ more filename

- for scrolling press **[Enter]**

less: \$ less +50 filename

- open the file for paging starting at line 50

\$ more +/sudha filename

- open the file for paging searching for occurrence of the word sudha and starting at the first occurrence.

[space] - scrolls the display, one screenful of data at a time

[Enter] - scrolls the display one line

[d] - scrolls the display backwards one screenful of data

[/] - lets you search the text,

less:

- less displays files one screen at a time, less does not need to load entire file to begin displaying it.

\$ less filename ↵

for scrolling ↑ or ↓ button.

for Exit press q.

head:

- Head Command by default will print the first 10 lines of each file to standard output. if more than one file is specified. each file will

\$ head file ↵

shows first 10 lines of ~~the~~ file

\$ head -n 25 example

shows first 25 lines of the example file

\$ head -c 200 file

shows

\$ head -c 2b file

tail:

- The tail Command is very similar to head Command. except it will print the last 10 lines by default

\$ tail file

shows last 10 lines of file

\$ tail -n 25 file

shows last 25 lines of the file

\$ tail -n +25 file

shows from line 25 to the end of the file.

\$ tail -f file

continuously display the end of a file in real time (1 second default polling interval)

To stop press

ctrl + C

Vi Editor:

Vi is Visual Editor used to enter and edit text files containing data and programs. It is used to create new files, if file already exists it opens the file.

There are two modes in Vi Editor.

↳ Insert Mode:

This mode used to insert some data in to the file by just press 'i' we will go to this mode. It is also called as input-text mode.

↳ Escape Mode:

This mode used to perform some operations like save, save & quit etc. The bottom line of the Vi editor is known as command line. (~~All commands entered in the E~~)

By just press "escape" key we will go to this mode.

Options:

- :w - Save all changes made so far
- :wq - Save all changes and quit
- :q - quit Vi
- :q! - quit without saving changes

- w - move the first character of the next word
- b - move the first character of the previous word
- e - move end of the current word

- yy - copies the current line
- nyy - copies n number of lines
- p - paste copied line after cursor position
- P - paste copied text before current position

- dw - Deleting current word
- dd - Delete a line
- ew : change word
- cc : change line
- X : Delete character at cursor position
- x : Delete character before cursor position
- h : Moves the cursor one position to left
- l : moves the cursor one character to right
- J : moves down another
- K : moves up.
- : \$ - End of the line of the file
- : ^ - Beginning of the line of the file.
- A - Append after the character
- I - Insert before the current character

Q1 How to search for a pattern after opening a file?

↳ first go to escape mode

- Then press / patternname

- To find out from top / patternname

- To find out from bottom ^ ? patternname

↳ To replace a pattern in whole the file

Syntax [: / . s / old pattern name / new pattern name]

EX: %/s /sda /sdhakar.

↳ To delete 5 lines → 5dd

↳ To cut 2 lines → 2LC

↳ To paste → P

File Comparison Commands:

\$cmp: This command compares the given two files, byte by byte and display the differed lines with their filenames.

Syn: \$cmp [options] file1 file2

options:
-l : report on each difference
-s : report exit status only

\$sdiff : This command compare two files

Syn: \$sdif file1 file2

who

This command will gives the info about no. of users logged in to the system and their terminals, Ip address currently.

Ex: # who ↵

who am i:

- Gives the info which user we have logged in as

Ex: # who am i ↵

su:

- To switch the user from ^{Admin} root to user viceversa.

Ex: so # su sudha ↵

Note: If you are already logged in as Admin, you are not need to enter the password to change to user.
If you are logged in as normal user you want to

Sort:

It is used to sort the output in numeric or alphabetic order.

sort filename

Q: To sort the file according to numbers

sort -n kfile

(B)

sort -h kfile

Q: To remove the duplicate entries from the output

sort -u filename

cut:

The cut command is used to pick the given expression (in columns) and display the output.

cut -d -f filename (where d stands for delimiter ex. :, " ; and f stands for field.

cut -d : -f1 /etc/passwd

root

bin

daemon

adm

Q: To delimit spaces and print the field

cut -d " " -f1 filename

Q: To delimit commas and print the field

cut -d , -f1 filename

Sed Command

Sed stands for Stream Editor, which is used to search and find word in the file and replace it with the word required to be in the output.

Note: It will only modify the output, but there will be no change in the original.

Q: How to replace a word inside the file and save?

```
# sed -i 's/sudha/sudhakar/g' sss.txt
```

→ sed 's/sudha/sudhakar/g' file

Q: How to print lines b/w 1 to 50.

```
# sed -n '1,50p' sss.txt
```

Q: How to print 5th line of a file

```
# sed -n '5p' sss.txt
```

Q: How to delete lines b/w 1 to 5.

```
# sed -n '1,5d' sss.txt
```

Q: How to know the CPU information?

```
# cat /proc/cpuinfo
```

processor : 0

vendor-id : GenuineIntel

cpu family : 6

model : 37

model name : Intel(R) Core(TM) i3 CPU

M3500 2.27 GHz

stepping : 2

cpu MHz : 2267.533

cache size : 3072 KB

cpu dlevel : 11

bogomips : 4526.17

How to delete special characters in a file.

```
sed 's/[!:@%#&*.^&#92]'
```

(or) '1/g' spe.txt

```
awk 'gsub(/[!:@%#&*.^&#92],  
"")' spe.txt
```


Q: How to know the process id of particular process?

```
# ps -ef | grep 'processname.'
```

```
# ps -ef | grep 'init'
```

Q: How to see the Runlevels in our Linux box?

```
# cat /etc/passwd
```

init - This file describes how the init process should set up the system in a certain run-level

The following are the ~~def~~ runlevels

0: halt - shutdown

1: Single-user mode

2: Multi-user, without NFS

3: Full multi-user mode

4: unused - research

5: X11 - Graphical

6: reboot

Q: How to identify the default runlevel on your system

```
# grep initdefault /etc/passwd
```

To check the default run level in Linux the command is

```
# who -r
```

```
run-level 5 2013-09-13 21:04
```

Q: How to change the run level 5 to 3

do modification on the file

```
# vi /etc/passwd
```

reboot the system.

After that to start the graphical interface when your level 3, use the following command.

```
# startx
```

change it back to runlevel 5. and reboot the system.

Q: How to check whether OS is 32 bit 64 bit /

```
# arch <# uname -m
i686 (32 bit) | i686 (32 bit)
```

Q: How to check the version of the OS in the system

```
# cat /etc/redhat-release
```

Red Hat Enterprise Linux Server release 5 (Tikanga)

Red Hat Enterprise Linux Server release 6 (Santiago)

Q: How to show the list of all the currently loaded modules.

```
# lsmod
```

Q: How to see the particular module we

```
# lsmod | grep -i modulename.
```

```
# lsmod | grep -i cdrom
```

```
cdrom      36705  1 ide-cd
```

Q: How to remove the loaded module

```
# modprobe -r module-name <
```

Q: How to install/re-install a module

```
# modprobe module-name <
```

Q: How to see the information about the module

```
# modinfo mod-name <
```

Runlevel Programs:

- when the Linux system is booting up, you might see various services getting started.

- Depending on your default init setting, the system will execute the programs from one of the following directories.

• Run level 0 - /etc/rc.d/rc0.d/

• Run level 1 - /etc/rc.d/rc1.d/

• Run level 2 - /etc/rc.d/rc2.d/

• Run level 3 - /etc/rc.d/rc3.d/

• Run level 4 - /etc/rc.d/rc4.d/

• Run level 5 - /etc/rc.d/rc5.d/

• Run level 6 - /etc/rc.d/rc6.d/

Crontab:

- In any operating system, it is possible to create jobs that you want to occur. This process known as job scheduling, is usually done based on user-defined jobs.
- For Red Hat or Any Linux, this process is handled by the cron service or a daemon called crond.
- Which can be used to scheduled tasks.
- The importance of the job scheduling is that the critical tasks like taking backups,

Crontab Format:

MIN	HOUR	DOM	MON	DOW	CMD
MIN - Minute Field	0 to 59				
HOUR - HOUR Field	0 to 23				
DOM - Day of Month	1-31				
MON - Month field	1-12				
DOW - Day of Week	0-6				
CMD - Command	Any Command to be Executed.				

Options

- * - is treated as a wildcard. Meaning any possible value
- */5 - is treated as every 5 minutes, hours, days or months. Replacing the 5 with another numerical value will change this option.
- 2,4,6 - Treated as an OR, so if placed in the hours, this could mean at 2, 4, or 6 o'clock.
- 9-17 - Treats for any value between 9 and 17. So if placed in day of month this would be days 9 through 17. Or if put in hours it would be between 9 and 5.

Crontab Commands :

Command

- crontab -e - Edit your crontab file, or create one if doesn't already exist.
- crontab -l - Display your crontab file.
- crontab -r - Remove your crontab file.
- crontab -u - If combined with -e, edit a particular user's crontab file.
- If combined with -l, display a particular user's u file.
- If combined with -r, delete a particular user's u file.

Q: To check the assigned cron jobs of currently logged in

User,

crontab -l.

o/p: no crontab for root

Q: to check the cron jobs for a particular user.

crontab -l -u username

crontab -lu username

o/p: no crontab for username.

problem: Setting a job to display the current date for every minute on present console.

steps: check the console on which you are working by following

Command

tty

/dev/pts/0 - is the console address.

Scheduled the task:

crontab -e

* /1 * * * * date > /dev/pts/0

Q: How to delete a file for every ~~5 minutes~~ 5 minutes every month, every day of the week?

```
* /5 * * * * "rm -rf /tmp/sudha.txt"
```

Q: How to delete a file for every 5 minutes for every month, Monday to Friday.

```
* /5 * * * * [1-5] "rm -rf /tmp/sudha.txt"
```

Q: How to check whether the service is running or not?

service cond status.

- We can start a service by # service cond start.

- In /etc/init.d directory the services will display.

- We can't stop a service. It can be done by Admin only.

To start, stop and status checking of services:

```
service <servicename> start  
service <servicename> stop  
service <servicename> status.
```

- To restart cron service:

```
# service cond -restart &
```

Q: Schedule a cron job to create a directory "sudha" under "/root" on "Sunday 22 September at 12:30 PM".

```
# crontab -e
```

```
30 12 22 09 . 0 mkdir /root/sudha
```

Note: you can use 0 or 7 for Sunday

mkdir: (making) creating directory

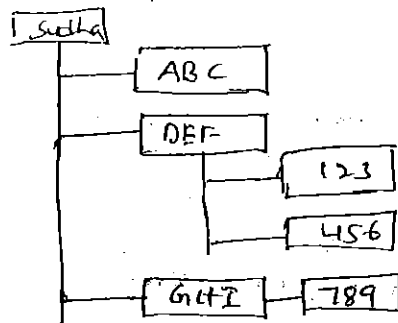
- This command is used to create a directory.

Syn: # mkdir <dir name>

mkdir sudha &

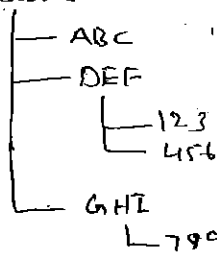
- How to make a multiple directories inside a directory

Ex:



Syn: # mkdir ~~cmd~~ -p Sudha/{ABC, DEF/{123, 456}, GHI/{789}}

tree Sudha/



CP:

- copy file into directory

cp filename directoryName

- copy directory from one location to another

cp -rvfp ABC DEF

- moving file from one location to other.

mv filename directoryName

- Renaming filename

mv ^{old} filename1 ^{new} filename2

Removing empty directory

rmdir dir-name

- Removing directory structure

rm -rf dir-name &

-f -force

-i - prompt before overwrite

-l → link files instead of copying

-r - copy directories recursively

-P never following symbolic links in source

Q: How to know whether the port number working or not?

S: `# netstat -anp | grep 22`

Q: How to know the current processes currently running in Linux box:

`# ps -cf`

- lists the processes that are running in our Linux box.

Grep:

Grep stands for Global Regular Expression Print. It is used to pick out the required expression from the file and print the output.

- It will search for a pattern without opening the file.
- If grep is combined with another command it can be used to pick out the selected word, phrase from the output of first command and print it.

Syntax:

`commandname -<option> "pattern" filename`

Ex: `grep "abc" sudha.txt`

- Let us pick the information about root from the file `/etc/passwd`

`# grep root /etc/passwd`

or:

- To avoid case sensitivity of the word (i.e., word may be uppercase or lowercase) use `-i`.

Ex: `# grep -i ABC "sudha.txt"`

- To Display a word and 2 lines after the word:

`# grep -nA2 abc sudha.txt`

- To Display a word and 2 lines before the word:

`# grep -nB2 abc sudha.txt`

- To Display the things except the given word:

```
# grep -v abc sudha.txt
```

- To Display the Searched word in Color

```
# grep -color root /etc/passwd
```

- To Display the no. of occurrences of a pattern in that file.

If the file consists

abc
def
123
abc abc
abc

```
# grep abc sudha.txt
```

abc
abc
abc

```
# grep -o abc sudha.txt
```

abc
abc
abc
abc

```
# grep -o -w 'Report' IS.txt
```

It will display character position of matched string

```
# grep -n 'Report' IS.txt
```

It will display lines of string found

```
# grep -v '|||' IS.txt
```

It will remove empty lines and display content of a file

-r - recursive search

-Bn - ~~Before~~ print n number of times before the pattern

-An - print n root lines after match pattern

-en - print n root lines before and after match found

-c - display no. of occurrences

-n - display lines number and lines where pattern found

-D - it will display only matched words

-w - it will search of whole words

-i - it will ignore case sensitive search

-b - it will display position of matched string

Pipe Command:

It is used to club two commands.

- It should be used when the o/p of first command is used to next command obviously.

Ex: `# cat sudha.txt | grep -l abc` - Display content and search for file

`# ls -l | grep -l sudha`

`# ifconfig | grep -l eth0`

Ex: `ls /tmp | ls /home`

- In this case pipe command not be used.

- This command is used to transfer o/p of one command to other command.

Ex: `# cat sudha.txt | grep abc`

↳ It will takes the o/p of cat command as the argument to 'grep' command.

- The Pipe Command used to use two or more commands at the same time and run them consecutively.

- It is denoted by "|" Symbol.

Ex: When we use cat command to view a file, which spans multiple pages, the prompt quickly jumps to the last page of the file and we do not see the content in middle.

To avoid this we can 'pipe' the o/p of the file 'cat' and to "less" which will show us only one scroll length of content at a time.

`cat filename | less`

wordcount (wc):

Syn: `wc -<options> <what to count>`

- It counts lines, words, letters/characters.

Q: How to learn the number of lines present in

1/tmp/abc.txt file?

A: # cat 1/tmp/abc.txt | grep Sudha | wc -l.

SSH:

There are a couple of ways that you can access a shell remotely on most Linux/Unix systems.

- By using SSH (Secure Shell) Command you can connect from one Linux box to another Linux box.

- SSH Configuration file is /etc/ssh/sshd_config

- SSH daemon or service is sshd

- To access the remote machine using ssh, the syntax is

#ssh <ip address / host name of remote machine>

Note: hostname can only be used when the hostname is saved in /etc/hosts file or if DNS is configured.

Ex: #ssh username@IPaddress

- now it will ask the password

Sudhakar@107.63.11.20's password: nxxx

- Enter the password of the remote system correctly, once logged in check hostname and ip address to confirm log.

- To leave the session, just type exit or logout command

and you will be back to your own machine through which you logged in.

Q: How to transfer files from windows to Linux box or vice versa?

A: using winscp cmd.

→ We can't give plain text 'password' to ssh command by input.

→ We can do the password less ssh using key exchange. 2 types?

Exchanges → RSA
→ DSA

SSIN000685x

SCP: (Secure Copy)

- scp stands for Secure Copy, which means that you can copy files across an ssh connection that will be encrypted, and therefore secured.
- An scp will be using ssh protocol to transfer the data, hence it's termed as the safest method of transferring data from one location to another.

Syntax

scp <filename> <remote host:IP> : / <location to copy the file>

Ex: # scp sudha.txt sudha@107.63.13.20 : /home

- To copy a file using scp from a remote machine being in destination's location:

✓ Let's reverse the previous task, login to sudha's machine whose IP address 107.63.13.20.

✓ Let's first remove the file sudha.txt, then copy it again from destination's location.

Syntax: scp 107.63.13.18 : /sudha.txt /root/

Note: Password will be asked for every transfer if public key is not saved on both locations, in our case we have already generated and copied the key.

- To transfer directories from one linux box to another linux box

scp -r /sudhadir 107.62.13.20 : /root/ <

- To copy a directory using scp from a remote machine being destination's location

scp -r 107.62.13.18 : /ktedir /root/ <

- To use scp cmd w/o password 'password less Copying'
- scp tmp / sudha.txt sudha:password@107.62.13.20 : /home <

wc:

- stands for word count
- using this command we can display no. of new lines, words and bytes in a file.

options: # wc [option] filename <

- ↳ -c or --bytes
print the byte counts
- ↳ -m or --char
print the character counts
- ↳ -l or --lines
print the new lines count
- ↳ -L or --max-line-length
print the length of longest line
- ↳ -w or --words
print the word counts

Ex: # cat > sudha.txt

wc sudha.txt

3 18 90 sudha — filename
↓ ↓ ↓
No. of words No. of characters
lines

INSPIRON N5110

Service TAG(S/N): 553TVQ1

Express Service code: 1280947829
12606107829

finger:

- Gives the info about a particular user
- It will display login, username, home directory, shell information about a particular user as shown below.

finger sudha <

o/p: Login: sudha Name: (null)

Directory: /home/sudha shell: /bin/bash

tar Command

- In information technology, a backup or the process of backing up is making copies of data which may be used to restore the original after a delete loss event.

tar:

- creates a new tar archive. Compress the file or directories.

Ex.

- To backup the file using tar syntax is

```
# tar -cvf <destination and name to be> <source file>
```

```
# tar -cvf /opt/etc.tar /etc
```

- To check the size of tar by using du -h <filename> Command

```
# du -h /opt/etc.tar
```

```
29M /opt/etc.tar
```

- Apply gzip on tar file and check the size

```
# gzip /opt/etc.tar
```

- Now check the size of the file

```
# du etc.tar.gz
```

```
7544 etc.tar.gz
```

- Transfer the file to other system and gzip and tar from it check the size.

```
# scp /opt/etc.tar.gz 192.168.10.95: /root/
```

```
etc.tar.gz 100% 7544 7.4 MB/s 00:01
```

- Login to remote system ^{using ssh}, remove gzip it and check the size

```
# gunzip etc.tar.gz
```

```
# du -h etc.tar
```

```
29M etc.tar
```

- To untar a file & ~~check~~ or

```
# tar -xvf etc.tar
```

```
# du -h etc
```

Shutdown:

- Reboot the system using shutdown command.

- ~~to~~ To shutdown the Unix-system

shutdown -h now - shutdown immediately.

shutdown -h +10 - shutdown after 10 min's

shutdown -r now - Reboot the system immediately

shutdown -fr now - ~~for~~ Force the file system checking
during Reboot.

Ping:

- By using this command we can know that the other system
up or down for Linux/windows.

Syntax: ping ip address

RPM & YUM

To Manage the software in Linux, two utilities are used.

1. RPM - Redhat Package Manager

2. YUM - Yellowdog Updater Modified.

RPM:

RPM is a package managing system (collection of tools to manage software packages).

RPM is a powerful software management tool for installing, uninstalling, verifying, querying and updating software packages.

- To check all the installed packages in the system

rpm -qa (q - query a - all)

Note: The output of the above command will be very lengthy.

- To check whether a package is installed or not

rpm -qa packagename

Ex: # rpm -qa httpd

- One more method of checking the installed package, when you are not sure about the package name

rpm -qa | grep -i httpd

How to know the package consistency?

```
# rpm -ivh --test <packagename>
```

where i = install, v = verbose view, h = hash progress.

If the installation status shows 100%, then the package good or consistent.

- To install the package the syntax is

```
# rpm -ivh <packagename>
```

- To remove a package (or) uninstall the package

```
# rpm -e <packagename>
```

- To see the info of a package, the syntax is

```
# rpm -qip <packagename>
```

- To see the info about a installed package.

```
# rpm -qi <packagename>
```

- To know installed location of a particular command is

```
# which cat
```

```
/bin/cat
```

- To check the package of a particular command

```
# rpm -qf /bin/cat
```

- To see the configuration files of the installed package.

```
# rpm -qlc <packagename>
```

- To see the directories with which a particular package is associated

```
# rpm -qld <packagename>
```

- To install a package without installing dependencies

```
# rpm -ivh <packagename> --nodeps
```

- To update a package

```
# rpm -Uvh <packagename>
```

- To check the changes are made after installation of package.

```
# rpm -V <packagename>
```

Yum:

The Yellow dog Updater Modified (yum) is a package management application for computers running Linux operating system.

- To install the rpm file automatically without dependency (resolves the dependencies automatically)

Syn: `yum install <packageName>`

- To install a package

`yum install Apache` (httpd)

- To update package

`yum update <packageName>`

- To remove package

`yum remove <packageName>`

- To install a package locally from a folder; pendrive or cd.

`yum localinstall <packageName> -y`

- To see the information about the package

`yum info packageName`

Listing installed packages in system:

~~4~~ `yum list installed` ↴
(or)

~~find~~ `find-repos-of-install` ↴

find Command:

- find Command is used to find the files or directory's path.

Syn: find / (under root) -option filename

Options:

- name : For Searching a file with its name
- inum : For Searching a file with particular inode inode number
- type : For Searching a particular type of file
- user : For Files whose owner is a particular user
- group : For Files belonging to particular group.
- mtime : ~~For~~ for files based on modified time

Ex: Finding with file name.

```
# find / -name sudha.txt &
```

Ex: Finding with inode number

```
# find / -inum 5934 &
```

Ex: Find with username

```
# find / -user sudha &
```

Ex: Find with group

```
# find / -group kgroup &
```

Diff b/w exec & pipe :

pipe : Gives output as a group

-exec : Gives the output as individual
(process the cmd individually)

Eg: find /tmp -name "sudha.txt" -exec rm -rf {} \;

O/p: /tmp/abc/sudha.txt

/tmp/123/sudha.txt

/tmp/sudha.txt

-ctime — change the status
(modify the file or its
attributes)

-atime -atime (read the file's
contents).

-mtime +60 \Rightarrow means you are looking
for a file modified 60 days

-mtime -60 \Rightarrow means you are looking
for a file modified 60
days before

-mtime 60 \Rightarrow means you are looking
for a file modified on 60th
days exactly

-newer — finds files

-size — finding files based on size

Eg:-

Ex: #find /tmp -perm. -g=r -type f

- It will find the file which has the group permission read only in /tmp directory based on type

Ex: To search for a empty file in home directory.

#find ~ -empty
↳ home directory

Ex: #find . -not -empty -type f -exec ls -l {} \; | sort -nr | head -5
↓
list the files by size

- It will display the first 5 top files that are not '0' size.

- -r is used to display 5 least files.

Ex:-

#find . -size +20m

→ It will display all files over 200MB in the current directory

Locate :-

The locate command is used to find files by their filename

→ locate does not search the files on disk rather it searches for file paths in a database.

→ the database is a file that contains information about the files and their path in your system. The locate database file is located at:

/var/lib/locate/locate.db.

Ex:- #locate -l 5 java

↳ restricting output to 5 lines

#locate -i JAVA

↳ searching in case insensitive

#locate -c java

RS2

→ display number paths matched.

-o - display all the output in one line

-c - display the count of number of matching entry

-i - ignore case sensitive search

-l - ~~restricting~~ restricting the output

File Permissions :

- permissions are applied on three levels:

1. owner/user level
2. Group level
3. Others level

- Access modes are of three types:

- r - read only
- w - write / edit / delete / append
- x - execute / run a command

- Permission can be set on any file/dir by two methods:

1. Symbolic method (ugo)
2. Absolute methods (numbers)

- If we create any file by default the following group of details available.

rw -	rw -	r - -
↓	↓	↓
owner/	group	other
group		

chmod :

- It allows us to change permissions on a file. The basic usage is

chmod permissions file

- where permissions are either numeric or the alpha equivalent of the file we want to assign and FILE is file or directory we want to effect.

- The file permissions are in the form

owner | group | All others

- Each of this section include

Read | write | Execute.

- Each permission (read, write, execute) is represented with the binary representation of the initial letter.

r - 4, w - 2, x - 1.

Symbolic mode

In this mode we can access the permissions with the help of following symbols:

- + → adding permissions
- → removing permissions
- = → assigning permissions

Ex: Adding execute permission to all groups and removing write permissions to others to a file sudha.txt

```
# chmod ugo+x, o-w sudha.txt
```

Absolute mode:

In this mode we can access file permissions with the help of octal numbers system principle.

- Here 0 (zero) indicates absence the permission

- 1 - permission present

0 - 000 - rwx - rwx

1 - 001 - rwx - execute

2 - 010 - rwx - write

3 - 011 - rwx - write, execute

4 - 100 - rwx - Read

5 - 101 - rwx - Read, execute

6 - 110 - rwx - Read, write

7 - 111 - rwx - Read, write, execute

Ex: Same above question.

```
# chmod 115 sudha.txt
```

chown:

- This command changes the user and/or group ownership of the file for given file.

options:

```
chown owner-user file;  
chown owner-user:owner-group file  
chown owner-user:owner-group directory  
chown options owner-user:owner-group file
```

Ex:

- First list the permissions for the file.

```
# ls -l sudha.txt
```

```
o/p: -rw-r--r-- 1 root May 31 7:48 sudha.txt
```

→ Now, here we want to change the file ownership to sudha.txt user and list the permissions, then:

```
# chown dinesh: sudha.txt
```

```
# ls -l
```

```
o/p: -rw-r--r-- 1 dinesh root 0 May 31 7:48 sudha.txt
```

converting from uppercase to lowercase

```
tr '[:upper:]' '[:lower:]' ABC <↵  
(or)
```

```
cat  
y=${x,,}
```

(or)

```
tr '[:upper:]' '[:lower:]' ABC
```

lower case to upper case

```
y='hello'
```

```
tr '[:lower:]' '[:upper:]' $y
```

(or)

```
x=${y^^}
```

(or)

```
tr '[:lower:]' '[:upper:]' y
```

(or)

```
echo $y | tr '[:lower:]' '[:upper:]'
```

y

Umask:

Using this command we can assign file permissions at the user level.

Syntax: #umask [options] mode filename

Generally Unix operating system default nature is file having 666 permissions, dir having 777 permissions.

Ex: #umask 242

File	
666	
<u>242</u>	
424	
/	/
owner	group other
100	010 100
<u>rx</u>	<u>w</u> <u>rx</u>
r--	-w- r--

file: #touch sudha

#ls -l

-rw-r--r-- 1 sudha 20 May 15 01:29 sudha.

Ex: # directory

777
<u>242</u>
535
owner group other
101 011 101
<u>rx</u> <u>w</u> <u>rx</u>
r-x -wx r-x

Processes

- A Linux process is a program running in the Linux system. Depending on Linux distributions, it is also known as service.
- When you start a program or running an application in Linux, you actually execute that program. A Linux process, running in foreground or in the background, uses memory and CPU resources. That's why we need to manage Linux process. Keeping unused Linux process running in the system is a waste and also exposes your system to security threat.
- In Linux, every process or daemon is given an identity number called PID (Process ID). The process id is unique.

→ # To Monitor the process using ps command.

```
# ps
```

PID	TTY	TIME	CMD
3568	pts/0	00:00:00	bash
3579	pts/0	00:00:00	ps

- To see total number of process running in the terminal.

```
# ps -a
```

3582	pts/0	00:00:00	ps
------	-------	----------	----

- To see the processes running by the logged in user

```
# ps -u <username>
```

```
# ps -u sudhakar
```

```
# ps -u (if no username is given it will show the process of the logged in user).
```

- To see which process are attached with some terminals (tty) and which are not.

```
# ps -x
```

→ The process which are showing "?" are not attached to any tty.

- To see which process was running by a particular group

ps -G <group name> or # pgrep -G

- To see offline process of the system, already executed.

ps -au.

Signals:

- Signals are a way of sending simple messages to processes. Most of these messages are already defined and can be found in <linux/signal.h>
- Every signal has a unique signal name, an abbreviation that begins with SIG (SIGINT for interrupt signal)
- Signals can be generated by the process itself, or they can be sent from one process to another.
- There are 64 signals in Linux. the list of all the signals can be seen by # kill -l.

The most common signals used are

- ✓ 1 for reloading the process
- ✓ 9 for killing the process
- ✓ 15 for terminating the process
- ✓ 20 for stopping the process

Restart the process continue working

fg <pid>

fg %1

Monitoring the process using top Command:

- when you need to see the running process on your linux in real time, you have top as your tool for that.
 - top also display other info besides the running processes, like free memory both physical and swap.
- Monitoring all process using top command:

top ↵

top - 23:42:59 up 1:08, 1 user, load average: 0.11, 0.11, 0.04

Tasks: 106 total, 2 running, 103 sleeping, 0 stopped, 1 zombie.

Cpu(s): 0.3%us, 0.3%sy, 0.0%ni, 99.0%id 0.0%wa,
0.0%hi, 0.0%si, 0.0%st

Mem: 1035400k total, 468304k used, 567072k free, 34424k buffs

Swap: 2666748k total, 0k used, 2666748k free, 318420k cache

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	Com
3024	--										
4014											
1											
2											
3											
...											

The first line in top:

" 23:42:59 " is the current time; " up 1:08 " shows how long the system has been up for; " 1 user " how many users logged in; " load average: 0.11, 0.11, 0.04 " the load average of the system.

Second line in top:

- shows the number of processes and their current state

Third line in top:

- shows cpu utilization details.

The fourth & fifth lines in top:

" 1035400 - total memory in the system. is the part of the RAM that currently contains information.

PID - process ID

USER - Effective UID

PR - Dynamic priority

NI - Nice Value, also known as base priority

VR - Virtual size of the Task.

RES - The size of RAM currently consumed by the task.

SHR -

S - task status

%CPU - The percentage of CPU time dedicated.

%MEM - The percentage of RAM currently consumed by ~~RAM~~ task.

TIME+ -

Command - showing program names.

Interacting with TOP:

- Now that we are able to understand the output from TOP let's learn how to change the way the output is displayed.

- Just press the following key while running top and output will be

sorted in real time.

✓ M - Sort by memory size

✓ P - Sort by CPU usage

✓ T - Sort by ~~accumulation~~ time

✓ Z - Color display

✓ k - Kill a process

✓ q - quit.

✓ r - to rename a process

✓ h - help

Q: How to start Tomcat and Apache at Boot-time?

- Give permissions to the user for the below directories.

CATALINA_HOME/conf

CATALINA_HOME/logs

- Set JAVA_HOME Environment variable

- Save the following scripts as

/etc/init.d/tomcat

/etc/init.d/apache

They will automatically be read & run at boot time

- make a link to it from /etc/rcs.d

- cd /etc/rcs.d

sudo ln -s ../init.d/tomcat 571 tomcat

sudo ln -s ../init.d/apache 572 apache

tomcat

```
#!/bin/bash
```

```
./etc/init.d/functions
```

```
RETVAL=$?
```

```
CATALINA_HOME="/usr/local/apache/tomcat/tomcat 6.0"
```

```
case "$1" in
```

```
start) if [ -f $CATALINA_HOME/bin/startup.sh ];
```

```
then
```

```
echo $"Starting tomcat"
```

```
/bin/su tomcat & CATALINA_HOME/bin/startup.sh
```

```
fi
```

```
;;
```

```
stop) if [ -f $CATALINA_HOME/bin/shutdown.sh ];
```

```
then
```

```
echo $"Stopping Tomcat"
```

```
/bin/su tomcat & CATALINA_HOME/bin/shutdown.sh
```

```
fi
```

```
*) echo & "usage: $0 {start|stop}"
```

```
exit 1
```

```
esac  
exit $RETVAL
```

- There are two important way a user administrator should be aware of.

1. "etc/passwd"
2. "etc/shadow"

creating a user:

useradd:

This command is used to create an user.

```
# useradd <username>
```

options:

- u uid
- G Secondary group id
- g primary group id
- d home directory
- c comment
- s shell

- when no option is used with useradd command the options like UID, GID, home dir and shell will be assigned default.

```
# useradd sudhakar &
```

Assigning password to the user:

- As a root user we can assign any password to any user.

```
# passwd sudhakar &
```

Deleting a User:

- To delete a user the system used is

```
# userdel
```

Q: How to check Hard disk utilization using shell

```
#!/bin/sh → shebang.
```

```
df -h [group vda 1] > /tmp/disk.txt
```

```
Disk = $(awk -F ' ' 'NR==1 {print $5}' /tmp/disk.txt)
```

```
echo "Disk string value is $Disk"
```

```
result = $(echo $Disk | cut -c 1-2)
```

```
echo $result
```

```
if [ $result -gt 50 ]
```

```
then
```

```
echo "Hard disk utilization is more than 50%."
```

```
else
```

```
echo "HD utilization is ok"
```

```
fi
```

Q: How to install apache in 10/more servers at a time

```
#!/bin/sh
```

```
array = (10.10.10.100, 10.10.10.101, 10.10.10.102, ... 10.10.10.110)
```

```
${#array}
```

```
for ((i=0; i<${#array}; i++))
```

```
do
```

```
ssh username@${array[i]} "rpm -ivh /tmp/apache.rpm"
```

```
done
```

→ How to find out user information in Linux?

we can find out user information in three ways which are

\$ id username ↙

\$ groups username ↙

\$ finger username ↙

Managing installed services

- Services are programs that once started run continuously in the background and are ready for input or monitor changes in your computer and respond to them.
- Many Services are required to run all the time however many can be safely turned off for both security reasons as running unnecessary services opens more doors into your computer, but also for performance reasons. It may not make much difference but your computer should boot slightly faster with less services it has to start on boot.
- One of the techniques in every Linux administrator's toolbox to improve security of a box is turn off unneeded services.
- There are 2 commands used to control services:

service - This controls the starting and stopping of services during a session, these settings are not saved.

chkconfig - This controls which services are set to start on boot. By their nature these settings are saved and are applied at next boot.

```
# service <name of service> status
# service      "      start
# service      "      stop
# service      "      reload
# service      "      restart
```

```
# chkconfig --list
```

```
# chkconfig <service> on
```

```
# chkconfig < u > off
```

- To check the status of the service

service <servicename> status

- To start the service

service <servicename> start

- To reload the service

service <servicename> reload

- To restart the service

service <servicename> restart

- To check the status of all the service availability

chkconfig --list

- To check the status of particular service

chkconfig --list <service-name>

- To make the service availability on for service.

chkconfig <service-name> on

- To make the service availability off for service

chkconfig <service-name> off

- To make the service availability ~~off~~ on, on a particular run.

chkconfig --level 5 <servicename> on

Setting up the priority to a process.

- When talking about process priority is all about managing processor time. The processor or CPU is like a human juggling multiple tasks at the same time.

- In Linux we can set guidelines for the CPU to follow when it is looking at all the tasks it has to do. These guidelines are called niceness or nice value.

- The Linux niceness scale goes from -20 to 19. The lower the number the more priority that task gets. If the niceness value is high number like 19 the task will be set to the lowest priority and the CPU will process it whenever it gets a chance. The default nice value is zero.

- There are two options to increase value of process. You can either do it using the nice command or the renice command.

- To set a priority to a process before starting it.

```
# nice -n <nice value range (-20 to 19)> <command>
```

- To check nice value

```
# ps -elf
```

- To change the nice value of any process while it is running

```
# ps -elf -to find out the pid
```

```
# renice <nice value (-20 to 19)> <PID>
```


AT JOBS:

- "at" is used to schedule the job for a particular time or interval, in other words it is used only for one time or only for one interval.

The disadvantages of at jobs are:

- It can be modified like cron jobs
- It cannot be reused
- The content cannot be viewed in normal human readable format.

mysql:

- mysql is probably the most widely used open source database on Linux. Even if you don't run a mysql database on your server, you might end-up using the mysql command (client) to connect to a mysql database running on the remote server.

- To connect to a remote mysql database. This will prompt for a password.

```
$ mysql -u root -p -h 192.168.10.12
```

- To connect to a local mysql database

```
$ mysql -u root -p
```

Note: If you want to specify the mysql root password in the command line itself, enter it immediately after -p (without any space)

ftp: file transfer protocol

- Both ftp and secure ftp has similar commands, To connect to a remote server and download multiple files, do the following:

```
$ ftp IP/hostname
```

```
ftp > mget *.html
```

- To view the file names located on the remote server before downloading, use ftp command as shown below

```
ftp > ls *.html
```

wget:

The non-interactive network downloader

- The quick and effective method to download software, music, video from internet is using wget command.

```
$ wget http://www.somenet.com/some/source/file/ver 1.2.3.tar.gz
```

- Download and store it with a different name.

```
$ wget -O target.zip <url>
```

awk:

awk command is used to manipulate the text. This command checks each line of a file, looking for pattern that match those given on the command line.

Syntax:

awk '{ pattern + action }' { filenames }

Options:

- V version - Display Version information and exit
- F - print help message and exit.

Ex:

Lets create a file with data and name as sudha.txt

sudhakar	bhanu	Raghava
20	15	16
15	20	11

- To print the second column data

```
# awk '{ print $2 }' sudha.txt
```

- Redirect the output to bhanu.txt

```
# awk '{ print $1, $2 }' sudha.txt > bhanu.txt
```

- Remove Duplicate lines using awk

```
# awk '!($0 in array) { array[$0]; print }' temp
```

- print all lines from /etc/passwd that has the same uid & gid

```
# awk -F ':' '$3 == $4' passwd.txt
```

- There are three types of accounts on a Unix System:

Root Account

System Account

User Account

- Managing users & groups: There are three main user administration files.

1. /etc/passwd: keeps user account and password information.

This file holds the majority of info. about accounts on the Unix system.

2. /etc/shadow: Holds the encrypted password of the corresponding account.

3. /etc/group: This file contains the group info. for each account.

4. /etc/gshadow: This file contains secure group account information.

STICKY BIT:

If sticky bit is applied on a file or directory, then only root and owner of that file or directory can delete it. Even if others are having full permissions they cannot delete the file or directory.

```
#chmod o+t /etc/dir
```

Sudo:

Execute a command as another user.

- Sudo allows a permitted user to execute a command as the supervisor or another user, as specified in the sudoers file.
- Sudo determines who is an authorized user by consulting the file `/etc/sudoers`.
- If a user who is not listed in the sudoers file tries to run a command via sudo, mail is sent to the proper authorities, as defined at configure time or in the sudoers file.
- If sudo is run by root and the `SUDO_USER` environment variable is set, sudo will use this value to determine who the actual user is. This can be used by a user to log commands through sudo even when a root shell has been invoked.

Options:

Basic Command Line Editing

ctrl+L : clear the screen

ctrl+W : Delete the word starting at cursor.

ctrl+U : clear the line i.e., Delete the all words from command line.

up & Down arrow keys : Recall commands.

Tab : Auto-complete files, directory, command names and much more.

ctrl+A : Search through previously used commands

ctrl+C : Cancel currently running commands.

ctrl+T : Swap the last two characters before the cursor.

ESC+T : Swap the last two words before the cursor.

vi most cuts

i - insert mode

I - insert beginning of the line

ce - insert next word

A - insert ending of the line

O - append new line below the cursor position

O - append new line above the cursor position

u - undo changes

U - undo changes in entire line

/ - for searching for patterns.

cw - cut the current word

n cw - cut the n number of words

CC - cut the current line

n CC - cut the n number of current lines.

dd - delete the current line

n dd - delete the n number of current lines.

dW - delete the current word (backspace).

yy - copy the current line

n yy - copy the n number of lines

P - paste the line below the cursor position

P - paste the line above the cursor position

K - works like up arrow (↑)

X - works like right arrow (→)

J - works like down arrow (↓)

H - works like left arrow (←)

gg - go to the ~~start~~ starting of file

G - go to the end of the file

w - move word by word forward

b - move word by word backward

nw - move n number of words forward

nb - move n number of words backward.

ESC+:w → save the changes

ESC+:w! → save changes forcefully

ESC+:wq → save the changes and quit

ESC+:q → quit without changes

ESC+:q! → quit forcefully without changes.

ESC+:wq! → save and quit forcefully

ESC+:X → save and quit

ESC+X → set ex) delete the password for a file

ESC+:10 → go to 10 line

ESC+:n → go to nth line

ESC+:se n → setting line number

ESC+:se non - Removing the line numbers.

HELL/ PERL

- Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development and more.
- Perl is a stable, cross platform programming language.
- Perl stands for Practical Extraction and Report Language.
- Perl is a scripting language developed by Larry Wall.
- Perl takes the best features from C, awk, sed, sh and BASIC, etc..
- Perl works with HTML, XML and other program make-up languages.
- Perl supports both procedural & object-oriented programming.
- Perl is extensible. There are over 500 third party modules available.
- Perl is interpreted language.
- First line of Perl script is `#!/usr/bin/perl`
- Extension of Perl script is `.pl` (PL)
- Executing Perl script is `$ perl <filename.pl>`
- `$ perl -v` [command line flags affect how Perl runs your program] will give the following result

This is perl, v 5.001

- `$ perl -e 'print 4; \n'` - use `-e` option will execute Perl statements from command line.

- Perl has three built-in variable (Data types) types:

- Scalar

- Array

- Hash

- Scalar represents a single value

```
$a = 10;
```

- Scalar value can be string, integer or floating point number and Perl will automatically convert between them as required.

Ex: `#!/usr/bin/perl`
`$a = 10;`
`print "$a";` } `a.pl`

```
#!/usr/bin/perl
```

```
010 = 10;
```

- Array represents a list of values:

```
@a = (10, 20, 30);
```

```
@b = ("sudha", "bhanu", "satya");
```

```
@mixed = ("sudha", 143, "bhanu");
```

```
print $a[0]; o/p: 10
```

```
print $mixed[1]; o/p: 143
```

```
print $mixed[2]; o/p: bhanu;
```

- Array are zero indexed but we can change this setting by changing default variable `$[` or `$ARRAY_BASE`.

↳ The special variable `$#array` tells you the index of the last element of an array:

```
print $mixed[$#mixed]; o/p: bhanu.
```

- There are a couple of special arrays too. such as `@ARGV` (the command line arguments to your script) & `@_` (the arguments passed to a subroutine).

- A Hash represents a set of key/value pairs. Actually hash are type of array with the exception that hash index could be a number or string. They are prefixed by `%` sign follows.

```
%fruit-colors = ("apple", "red", "banana", "yellow");
```

(*)

```
%fruit-color = ( apple => "red",  
                banana => "yellow" );
```

```
$fruit-color ("apple"); o/p: red.
```

- To know the size of array we use 2 ways

1:

```
@a = (10, 20, 30)  
$size = $#a + 1;  
print "$size";
```

2:

```
@a = (10, 20, 30);  
$size = length(@a);  
print "$size";
```


- variable names are case sensitive: \$foo, \$FOO are all separate variables as far as perl is concerned.

- Adding and Removing Elements in Array

push(): adds an element to the end of an array

unshift(): adds an element to the beginning of an array

pop(): removes the last element of an array

shift(): removes the first element of an array

Ex: @amp = ("half", "full", "quarter");

push (@amp, "empty");

unshift (@amp, "empty");

pop (@amp);

shift (@amp);

- Replacing Array Elements

Replacing is possible with splice() Function.

splice (@array, first-element, sequential length, new elements)

Ex: @nums = (1...20); [print "nums o/p: 1,2,3,4,5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

splice (@nums, 5, 5, 21...25);

print "@nums";

- Here actual replacement begins after 5th element, starting with the number 6. Five elements are then replaced from 6-10 with the num 21-25.

- Transform String to Arrays:

- It is possible with split() Function.

\$string1 = "This is perl example";

\$string2 = "This is done by me";

@array1 = split('-', \$string1);

@array2 = split('-', \$string2);

print \$array1[3] o/p: example

print \$array2[3] o/p: is

\$string3 = join(" ", @array1);

\$string4 = join(" ", @array2);

print "\$string3"

print "\$string4"

Transform Array to String

- It is possible with join

← Same Example

Sorting arrays:

- sort() method.

```
@array = ("pizza", "steak",  
          "chicken", "burger");  
print "@array";
```

```
@foods = sort(@array);  
print "after sorting @foods is";
```

Before: pizza, steak, chicken, burger

After sorting: burger, chicken, pizza, steak

Selecting Elements from list:

merging list:

```
@numbers = (1, 3, (4, 5, 6));
```

```
@numbers = (primes(1, 5), 4);
```

```
$one = (5, 4, 3, 2, 1)[4];
```

```
print "$one"; o/p: 1
```

```
@newlist = (5, 4, 3, 2, 1)[1..3];
```

```
print "@newlist"; o/p: 4 3 2
```

```
$.hash = ('new' => "Hero",  
          'old' => "Heroin");
```

```
print $.hash('new'); o/p: Hero.
```

✓ if

✓ unless

if having five different formats

→ if (EXPR)

→ if (EXPR) BLOCK

→ if (EXPR) BLOCK else BLOCK

→ if (EXPR) BLOCK elsif (EXPR) BLOCK...

→ if (EXPR) BLOCK elsif (EXPR) BLOCK...

else BLOCK

→ The final condition is actually

an operator the conditional operator.

→ It is synonymous with the if-else

condition statement. but is shorter

and more compact.

(EXPR) ? (statement if true) : (statement if false)

Ex: (\$date == \$today) ? print "Happy

Day! In";

print "Happy

Day! In";

→ perl supports four main loops:

1. while

2. for

3. until

4. foreach

1. Numerical Comparison:

== equality

!= inequality

< → less than

> → greater than

<= less than or equal

>= greater than or equal

2. String Comparison

eq - equality

ne - inequality

lt - less than

gt - greater than

le - less than or equal

ge - greater than or equal

Boolean logic operators:

&& - and

|| - or

! - not

Miscellaneous operators

= assignment

* string concatenation

* string multiplication

.. range operator

SHELL

- The 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 output.
- A shell is an environment in which we can run our commands, programs and shell scripts.

Shell Types:

In Unix there are two major types of shells:

1. The Bourne shell - The default prompt '\$'
2. The C shell - The default prompt '%'

- It is a scripting language i.e., usually it is like a programming language.
- In shell scripting a group of Linux commands are written in a file and executing those commands in a sequence.
- Scripting is mainly used to automate the manual process.
- Shell scripts are written in vi Editors.

Syntax: vi filename.sh &

The Extension for a shell script should be ".sh".

- The first line of shell script contains...

#!/bin/sh - This is known as shebang.

- It states the executable path of our shell script
- It gives the path of the interpreter i.e., The executable file of shell is installed in bin.

Executing shell scripts:

1st way: \$ sh filename.sh

2nd way: \$ compile/compile/path/ filename.sh

3rd way: \$./filename.sh

- A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device or any other type of data.

- A variable is nothing more than a point to the actual data. The shell enables you to create, assign, and delete variables.

- The name of variable can contain only letters (a to z or A to Z), numbers (0 to 9) or underscore character (-).

- Variables defined as

variable-name = variable-value.

NAME = "Sudhakar"

- The shell enables you to store any value you want to in a

variable.

VAR1 = "Sudha"

VAR2 = 100

Accessing Values of Variables:

- To access the value stored in a variable, prefix its name with the dollar sign "\$".

Exo: #!/bin/sh

NAME = "Sudhakar"

echo \$NAME

Readonly Variable:

- The shell provides a way to mark variable as read-only by using the readonly command. After a variable is marked readonly, its value cannot be changed.

Ex:

#!/bin/sh

NAME = "Sudha"

readonly NAME

echo \$NAME

NAME = "Sudha"

• Execute this program.

o/p:

\$ sh filename.sh

/bin/sh: NAME: This variable is read only.

UnSetting Variables:

Unsetting or deleting a variable tells the shell to remove the variable from the list of variable that it tracks. Once you unset a variable, you would not be able to access stored value in the variable.

Syn: unset variable_Name

Exs: `#!/bin/sh
Name="Sudha"
unset Name
echo $Name`

This Example would not print anything.
You cannot use the unset command to unset variable that are marked readonly.

Variable Types:

✓ Local variable:

A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at command prompt.

✓ Environment Variable:

An Environment Variable that is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually a shell script defines only those environment variables that are needed by the programs that it runs.

✓ Shell Variables:

A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

Unix-Special variables:

Variable

- $\$0$ - The filename of the current script.
- $\$n$ - These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument. (the first arg. $\$1$, 2nd arg. $\$2$, ...)
- $\#\#$ - The number of argument supplied to a script.
- $\$*$ - All the arguments are double quoted. If a script receives two arguments, $\$*$ is equivalent to $\$1 \2 .
- $\$@$ - All the arguments are individually double quoted. If a script receives two args. $\$@$ is equivalent to $\$1 \2 .
- $\$?$ - The exit status of last command executed.
- $\$\$$ - The process-number of current shell. For shell scripts, this is the process ID under which they are executing.
- $\$!$ - The process number of the last background command.

Reading positional parameter

- | | |
|-------------------------|-----------------------------|
| $\$1$ - 1 st | $\${10}$ - 10 th |
| $\$2$ - 2 nd | $\${11}$ - 11 th |
| $\$3$ - 3 rd | $\${12}$ - 12 th |
| $\$4$ - 4 th | $\${13}$ - 13 th |
| $\$5$ - 5 th | |
| $\$6$ - 6 th | |
| $\$7$ - 7 th | |
| $\$8$ - 8 th | |
| $\$9$ - 9 th | |

Unix - Using Arrays:

- A shell variable is capable enough to hold a single value. This type of variable are called scalar variable.
- Shell support a different type of variable called an array variable that can hold multiple values at the same time.
- Array provide a method of grouping a set of variables. Instead of creating a new name for each variable that is required, you can use a single array variable that stores all the other ~~variables~~ variables.

Defining Array Value:

- We can use a single array to store all the values.

Ex: `array-name = (value1, ..., valueN)`

`array-name [index] = value`

Accessing Array Value:

- After you have set any array variable, you access it as follows:

`${array-name [index]}`

- You can access all the items in an array in one of the following ways

`${array-name[*]}`

`${array-name[@]}`

Linux - Basic Operators

There are various operators supported by each shell. Our tutorial is based on default shell (Bash) so we are going to cover all the important Bash shell operators in the tutorial.

Operators :

- Arithmetic
- Relational
- Boolean
- String
- File Test

Arithmetic : $a=10, b=20$

- + - Addition - Adds value on either side of operator - 'expr \$a+\$b' - 30
- - Subtraction - Subtracts right hand operand from left - 'expr \$a-\$b' - 10
- * - Multiplication - multiplies values on either side of op - 'expr \$a*\$b' - 200
- / - Division - $\$a / \b -
- % - Modulus - $\$b \% \a
- = - Assignment - $\$a = \b
- == - Equality - $[\$a == \$b]$
- != - Not Equality - $[\$a != \$b]$

Relational :

- eq - Checks if the value of two operands are equal or not.
if yes then condition becomes true.
 $[\$a -eq \$b]$ - true
- ne - checks - $[\$a -ne \$b]$ - true
- gt - $[\$a -gt \$b]$ - not true
- lt - $[\$a -lt \$b]$ - true
- ge - $[\$a -ge \$b]$ - Not true
- le - $[\$a -le \$b]$ - true

Boolean:

! - This is logical negation. This inverts a true condition into false and vice versa. [! false] - true - [\$a != \$b] -

-o - This is logical OR. [\$a -lt 20 -o \$b -gt 100] - true

-a - This is logical AND [\$a -lt 20 -a \$b -gt 100] - false

String:

= - checks if the two operands are equal or not. [\$a = \$b] - true

!= - [\$a != \$b]

-z - [-z \$a] - checks if the given string operand size is zero.

-n - [-n \$a] - checks if the given string operand size is Non-Zero

str - [\$a] - checks if str is not the empty string.

File Test Operators:

-r file - checks if file is readable if yes then condition becomes true

-w file - file is writable

-x file - file is Executable.

-f file - file is an ordinary file or Special file

-s file - Non Empty file

-d file - check file is a directory

-e file - check file exist.

Unix - Decision Making

While writing a shell script, there may be a situation when you need to adopt one path out of the given two paths. So you need to make use of Conditional statements that allow your program to make correct decision and perform right action.

- shell supports Conditional statements:

- if... else statements
- case... esac statement

if ... fi statement

```
if [ expression ]  
then  
    statement(s) ...  
fi
```

if .. else ... fi statement

```
if [ expression ]  
then  
    statements ...  
else  
    statements  
fi
```

if ... elif ... else ... fi statement

```
if [ expression 1 ]  
then  
    statements  
elif [ expression 2 ]  
then  
    statements  
:  
else  
    statements  
fi
```

How to debug shell script in Unix or Linux

1) Using the set built-in ~~set~~ Command

you can use the set Command to enable the debugging of the shell script.

-x - prints Command and their arguments as they are executed

-v - prints shell input lines as they are read

2) #!/bin/bash

set -x # enabling debugging mode

```
for i in $(seq 1 5)  
do  
    echo $i
```

done

set +x # disabling debugging mode

ls

2) Debug when running the script

another way of debugging a shell script is to specify the debug options when executing the script

> bash -xv script.sh

The case... esac statement

case word in

Pattern 1)

Statement...

;;

Pattern 2)

Statement...

;;

:

esac

Ex:

FRUIT="kiwi"

case "\$FRUIT" in

"apple") echo "Apple pie is quite tasty"

;;

"banana") echo "I like banana not bread"

;;

"kiwi") echo "New Zealand is famous for kiwi"

;;

esac

3) Debug function:- we can use the debug function for debugging specific statement in the shell script.

#!/bin/bash

_DEBUG="on"

function DEBUG() {

["\$_DEBUG" == "on"] && {

{

DEBUG echo "printing numbers"

for i in {1..3}

do echo \$i

done

How to pass external shell variables to awk

space.sh

#!/bin/sh

var="hello this is Venkatesh Rao"

awk 'BEGIN {print "||| \$var |||"}'

Steupe:

sh space.sh

hello this is

Venkatesh Rao

\$ bash script.sh

1
2
3

\$ export _DEBUG=on

Printing numbers

1
2
3

Unix - Shell scripts

- Loops are powerful programming tool that enable you to execute a set of commands repeatedly.

- The While loop:

Syn: while condition

do
statements
done

Ex: #!/bin/sh

```
#!/bin/sh
while [ $a -lt 10 ]
do
    echo $a
    a=$((a+1))
done
```

- The for loop:

Syn: for var in word1 word2 ... wordN

do
statements
done

Ex: #!/bin/sh

```
#!/bin/sh
for var in 0 1 2 3 4 5 6 7 8 9
do
    echo $var
done
```

Ex: #!/bin/sh

```
#!/bin/sh
for FILE in $HOME/.bash*
do
    echo $FILE
done
```

- The Until loop:

The While loop is perfect for a situation where you need to execute a set of commands while some condition is true. Sometimes you need to execute a set of commands until a condition is true.

Syn: until command

do
statements
done

The select loop:

The select loop provides an easy way to create numbered menu from which users can select options. It is useful when you need to ask the user to choose one or more items from a list of choices.

Syn: select var in word1, word2, ... wordN
 do
 statements
 done

Ex: ~~#!/bin/sh~~ #!/bin/ksh
 select Drink in tea coffee water juice apple all none
 do
 case \$Drink in
 tea|coffee|water|all)
 echo "Go to canteen"
 ;;
 juice|apple)
 echo "Available at home"
 ;;
 none)
 echo "ERROR: Invalid selection"
 ;;
 esac
 done

#

Unix shell functions:

Functions enable you to break down the overall functionality of a script into smaller, logical subsections, which can then be called upon to perform their individual task when it is needed.

- Using functions to perform repetitive tasks is an excellent way to create code reuse.

Creating functions:

Syn: function_name () {
 list of commands
 }

Ex: #!/bin/sh

 # Defining function

```
    Hello () {  
        echo "Hello world"  
    }
```

 # Invoking function

 Hello

Pass Parameters to a function:

you can define a function which accept parameters while calling those functions. The parameters represented by \$1 \$2 ... etc

Ex: #!/bin/sh

```
    Hello () {
```

```
        echo "Hello world $1 $2" &
```

```
    }      $1      $2
```

```
    Hello Sudha Bhanu
```

o/p: Hello world Sudha Bhanu

Returning values from Functions:

Deployment scripts for Different Environments:

Startup Script / Shutdown script Example:

```
#!/bin/sh
```

```
Env = $1
```

```
echo "we are shutdown $1 servers"
```

```
if [ $env -eq "DEV" ]
```

```
then
```

```
chat-ip = 10.10.10.101;
```

```
security-ip = 10.10.10.102;
```

```
notification-ip = 10.10.10.103;
```

```
;
```

```
elif [ $env -eq "QA" ]
```

```
then
```

```
chat-ip = 20.20.20.201
```

```
security-ip = 20.20.20.202;
```

```
notification-ip = 20.20.20.203;
```

```
;
```

```
elif [ ...
```

```
fi
```

```
ssh username@ $chat-ip " /home/weuser /chat /bin /shutdown.s
```

```
ssh username@ $chat-ip " rm -rf weuser /chat /webapps /chat.s
```

```
if [ $? -eq 0 ]
```

```
then "removing chat.war file success"
```

```
scp /home/otay/e2.0/build/int.war username@ $chat-ip /home  
ls weuser /chat /webapps /
```

```
ssh username@ $chat-ip /home/weuser /chat /bin /startup.sh
```

SW Administrators

- Installing Apache, svn
- Integrating svn, apache
- creating branches
- creating tags
- giving access permissions to the users
- taking daily backups
- trouble shooting.

Jenkins

- Installing
- Configuring
- troubleshooting
- Creating Jobs
- Scheduling Jobs
- Setting up automated builds
- Deployment builds
- Manage the highly builds
- Install plugins

Release notes:

- Release Notes contains Everything that is necessary to build your files successfully

- Release Notes always cumulative

Ex: ReleaseNotes 14 = ReleaseNotes 13 + New changes

1. Introduction

1.1 changes to the release notes

1.2 scope

2. New Instruction & System Requirements

2.1 New Hardware requirements

2.2 New partner connectivity / changes

2.3 changes in service deployment location

2.4 IP white listing & VPN Setups

2.5 Certificate changes

2.6 New s/w installation

2.7 s/w version upgrade

3. New features / functional changes

4. Resolved Issues

4.1 Included OAT Defects

4.2 " production incident fixes

4.3 " Internal QA defects fixes

4.4 " Platform & Tools improvements

5. Dropped features / functional changes

6. Known issues, Limitations & Restrictions

6.1. open defects

6.2. Limitation & Risk Assessment

6.3. Restrictions

6.4 Exceptions

7. SVN Tag Number for Revision

8. Change Details

8.1 Application changes

8.2 Database changes

8.3 Property Value changes

8.4 Batch Job / Cron Job changes

8.5 Mail template

9. Any Additional Deployment Instructions

10. Related Documentations

Maven

Common Usage of maven is :

- Build Tool (similar to ant) ✓
- Project management tool ✓

It helps to generate reports and Dependency management ✓

Common Problems and activities while developing applications :

- Multiple jars (using F/w in my app i.e., Spring, Hibernate)
- Dependencies and versions (jar dependency (one jar file depend on another jar))
- project structure (web app maintain project structure)
- Building, publishing and deploying.

Download from:

maven.apache.org/download.html

How to set PATH:

```
$ export M2_HOME = /home/sudha/java/apache-maven-3.0.3
```

```
$ export PATH = /home/sudha/java/apache-maven-3.0.3/bin : $PATH
```

How to know the version of maven:

```
$ mvn --version
```

Note: We need to connect internet while using maven because maven can download all the jars and lib's at runtime.

- Maven is a project management and comprehension tool. Maven provides developers a complete build life cycle framework.
- Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build life cycle.

- Maven automatically creates directory structure!

Creating a Simple jar:

\$ mvn archetype:generate ↵

choose a Number: 106 ↵ default. ↵

It displays Archetype (model)

choose version:

1 1.0-alpha-1

2: ...

check a number: 6: 6 ↵

- download more plug-in's related to app'n.

- Define value for property "groupId": (package name)
org.sudha.java ↵

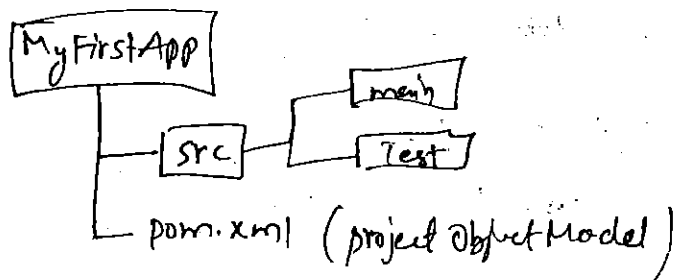
- Define value for property "artifactId": (Name of our jar file)
MyFirstApp ↵

"version": 1.0-SNAPSHOT ↵

"package": org.sudha.java ↵

It asks for Confirmation: Yes ↵

- Now It creates directory MyFirstApp with one directory & (src)
pom.xml file.



POM.XML:

```
< project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                              http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion> 4.0.0 </modelVersion>
```

```
<groupId> org.sudha.java </groupId>
```

```
<artifactId> MyFirstApp </artifactId>
```

```
<version> 1.0-SNAPSHOT </version>
```

```
<packaging> jar </packaging>
```

```
<name> MyFirstApp </name>
```

```
<url> http://maven.apache.org </url>
```

```
<properties>
```

```
</properties>
```

```
<dependencies>
```

```
<dependency>
```

```
<groupId> junit </groupId>
```

```
<artifactId> junit </artifactId>
```

```
<version> 3.0.1 </version>
```

```
<scope> test </scope>
```

```
</dependency>
```

```
</dependencies>
```

```
</project>
```

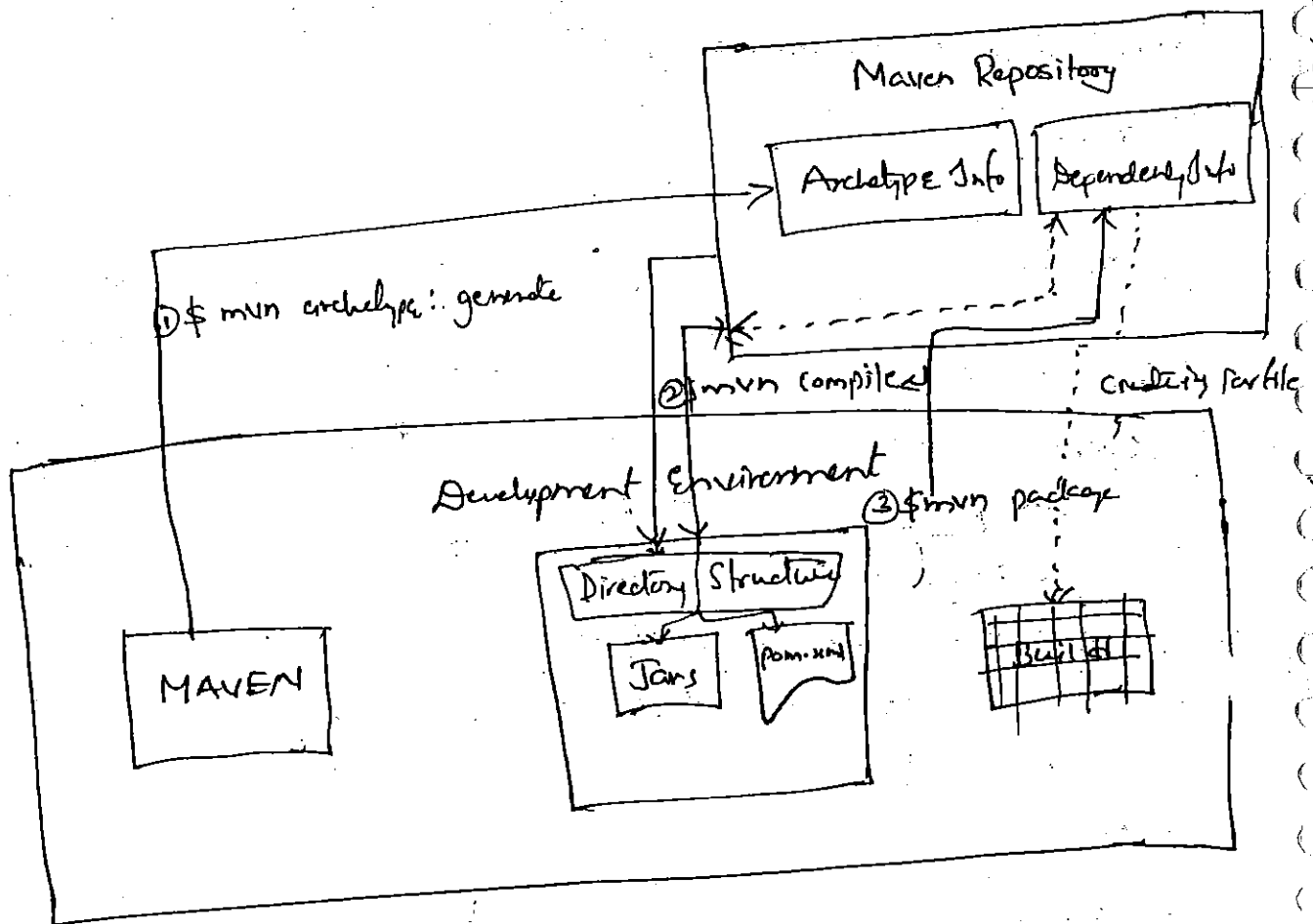
→ \$ mvn compile <
downloading all the plugins for our app.

→ \$ mvn package <
It's download more plugin's for packaging our app.

→ \$ java -cp target/MaytenBtApp.jar
org.sudha.java.App <

op: Helloworld.

Maven
Maven



Architecture

Maven plugins.

- plugins extend the functionalities of the maven.

eg.

<build>

- Compiler plugin

<plugins>

<plugin>

main

{ <groupId> org.apache.maven.plugins </groupId>
<artifactId> maven-compiler-plugin </artifactId>
<version> 2.0.2 </version>

<configuration>

<source> 1.4 </source> - JVM version (Javac 1.4 version)

<target> 1.4 </target> - at runtime

</configuration>

</plugin>

</plugins>

</build>

- The maven architecture is an assembly of different plugins.
- For example if we use generics in our web application we need to change the source and target values.

* Adding Jetty plugin (Servlet lightweight plugin)

<build>

<plugins>

<plugin>

<groupId> org.mortbay.jetty </groupId>

<artifactId> maven-jetty-plugin </artifactId>

<version> 6.1.10 </version>

</plugin>

</plugins>

</build>

\$ mvn jetty:run

- It download all the required jar files and

Note: - Maven has a modular architecture.

- maven has a Eclipse plugin
- Eclipse has a maven plugin.

Adding maven eclipse plugin into maven:

\$mvn eclipse:eclipse

- Now the maven is looking for repository and download the required files.

Adding maven plugin into Eclipse:

→ start Eclipse

go to File

↳ import existing project

↳ General

↳ choose existing project into workspace (click next)

② select root directory

path of Existing file

Browse

projects

☒ mywebapp

click Finish

Creating Java webapp using maven:

\$ mvn archetype:generate

choose a number: 269

choose version:

choose a number: 4

"groupId": org.scler.java

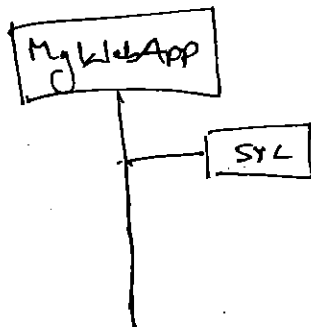
artifactId: MyWebApp

version: 1.0-SNAPSHOT

package: org.scler.java

confirm: yes

BUILD SUCCESS



\$ mvn compile

\$ mvn package

pom.xml: tr xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    "http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion> 4.0.0 </modelVersion>
<groupId> org.sudheer.java </groupId>
<artifactId> mywebapp </artifactId>
<version> 1.0-SNAPSHOT </version>
```

```
<name> mywebapp </name>
<url> http://maven.apache.org </url>
```

```
<dependencies>
```

```
<dependency>
```

```
<groupId> javax.servlet </groupId>
<artifactId> servlet-api </artifactId>
<version> 2.4 </version>
<scope> provided </scope>
```

```
</dependency>
```

```
<dependency>
```

```
<groupId> javax.servlet.jsp </groupId>
<artifactId> jsp-api </artifactId>
<version> 2.1 </version>
<scope> provided </scope>
```

```
</dependency>
```

```
<dependency>
```

```
:: Test
```

```
<build>
```

```
<plugins>
```

```
<plugin>
```

```
<groupId> org.apache.maven.plugins </groupId>
<artifactId> maven-compiler-plugin </artifactId>
<version> 2.0.2 </version>
```

```
<configuration>
```

```
<source> 1.4 </source> - Jvm version 1.4
<target> 1.4 </target>
```

```
</configuration>
```

```
</plugin>
```

```
</plugins>
```

```
</build>
```

Maven Advantages:

- Source Code directory
 - Configuring the dependencies.
 - Compiling & packaging
- } Automatically
- Maven Create Project Template
 - Build

Project Template

mvn archetype:generate

- Folder structure
- pom.xml.

- ✓ Archetype
- ✓ groupId
- ✓ ArtifactId
- ✓ Version
- ✓ package

These are all
Effect on pom.xml

pom.xml:

- maven co-ordinates - Identifies of the artifact
- Metadata - about the project
- Build Information - project is jar, war, etc.
- Resources and dependencies - required for the successfully build a project

Maven Build:

- Build Lifecycle
 - ↳ consists of phases
- Default behaviour of phases
- Specify the build phase you need, previous phases automatically run.

Some maven

- ✓ Validate - It checks everything in a order
 - Configuration is proper
- ✓ Compile - .java to .class file
- ✓ test
- ✓ package
- ✓ install - (into local maven repository)
- ✓ deploy - (to other remote repository)

Dependency:

Note: mvn clean

\$ mvn clean ↵

it will delete the own target folder

Adding Dependency:

```
{ Logger log = LoggerFactory.getLogger (App.class);  
  log.info ("Hello world");
```

```
import org.slf4j.*;
```

Add the above code into App.java in main folder.

\$ mvn compile ↵

It is showing Error message.

- Note: Here we need to add the dependency in pom.xml
↳ go to maven dependency search engine

```
<dependency>
```

```
<groupId> org.slf4j </>
```

```
<artifactId> slf4j-api </>
```

```
<version> 1.6.1 </>
```

```
</dependency>
```

- Continuous Integration:

Continuous Integration, also known as CI, when Continuous Integration is introduced into an organization, it radically alters the way teams think about the whole development process.

A good CI infrastructure can streamline the development process right through to deployment, help detect and fix bugs faster, provide a useful project dashboard for both developers and non-developers, and ultimately help teams deliver more real business value to end-users.

Continuous Integration, in its simplest form, involves a tool that monitors your version control system for changes. Whenever changes are detected, this tool automatically compiles and tests your applications. If something goes wrong, the tool immediately notifies the developers so that they can fix the issue immediately.

Continuous Integration can also help you keep tabs on the health of your code base, automatically monitoring code quality and code coverage metrics, and helping keep technical debt down and maintenance costs low.

CI can also act as a communication tool, publishing a clear picture of current state of development efforts. And it can simplify and accelerate delivery by helping you automate the deployment process, letting you deploy the latest version of your app either automatically or as a one-click process.

The practice of automatically deploying every successful build directly into production is generally known as continuous deployment.

java -jar jenkins.war

Introducing Jenkins.

- Jenkins, originally called Hudson, is an open source Continuous Integration tool written in Java.
- Jenkins is used by teams of all sizes, for projects in a wide variety of languages & technologies, including .NET, Ruby, Groovy, etc.
- First, Jenkins is easy to use. The user interface is simple, intuitive, and visually appealing, and Jenkins as a whole has a very low learning curve.
- Jenkins does not sacrifice power or extensibility: it is also extremely flexible and easy to adapt to your own purpose. Hundreds of open source plugins are available.
- These plugins cover everything from version control systems, build tools, code quality metrics, build notifiers, integration with external systems, UI customization, games, and much more.

Hudson to Jenkins:

- Downloading & Installing Jenkins.
- Preparing a Build Server for Jenkins
- Running Jenkins as a Stand-Alone Application
- Running Jenkins behind an Apache Server
- Running Jenkins on an Applet Server.
- Memory Considerations
- Installing Jenkins as a Windows Service.

Configuring the Tools:

Before we get started, we need to do a little configuration. More precisely, we need to tell Jenkins about the build tools and JDK version we will be using for our builds.

- ✓ configuring JDK setup
- ✓ configuring Ant setup
- ✓ configuring Maven Setup
- ✓ First Jenkins Build Job.

New job → Name
@ Build free-style project
☒

Project name

Description

- ☐ Discard old builds
- ☐ This build is parametrized
- ☐ Obsolete builds
- ☐ Execute Concurrent builds if necessary

Source code management:

☒ SVN

Repository

Build Triggers:

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Poll SCM

Build

- ✓ Execute windows batch cmd
- ✓ Execute shell
- ✓ Invoke Ant
- ✓ Invoke top-level maven targets

Post Build Actions

- ✓ Archive the artifacts
- ✓ publish javadoc
- ✓ E-mail Notification
- ✓ Build other projects.

☒ ☒

Manage Jenkins → Manage Nodes → New Node

NucleName: kinderess

⑥ Burnt Skene

ot ↖


Name Kendrew

Description	

No of executors

Remote FS Root

Labels

usage 

Launch method

Host

Credentials ~~_____~~

Availability

Hide property

□ Environment variables

☐ TOOL Location

Signature

Manage Jenkins → Configure global Security

☐ Enable Security

Access Control Security Reason

○ Delegate to Serial of Containers

- Jentich's own user database

○ LDAP

Authorization

0. Anyone can do anything

0 Legally made

0 Legged - it user can do any thing

0 Matrix-based security

④ project-based Matrix Authorization Strategy.

Answer -

Sudhe

Jenkins plugins:

- plugin's are extends the functionality of the Jenkins. Depending upon the requirement we can add the plugin's.

go to Jenkins home

<http://localhost:8080/Jenkins> ↵

click on Manage Jenkins

↳ Manage plugins

update | Available | Installed | Advanced

Installed

- Ant plugin
- Credentials plugin
- External Monitor Job type plugin
- Javadoc plugin
- Jenkins cvs plug-in
- Jenkins Mailer plug-in
- Jenkins SSH slaves plug-in
- Jenkins Subversion plugin
- Jenkins translation Assistance plugin
- LIDAP plugin
- Maven Integration plugin
- pam-auth
- SSH Credentials plugin.

✓ Source Code management plugins - clearcase, blame subscription

✓ Build trigger plugins - FilesFoundDigger,

✓ Build tools plugins - (Ans), BuildBuilderPlugin, CopyArtifactPlugin

✓ Build wrappers plugins

Credential plugin crs plugin

✓ Build Notifications

Mailer plugin

✓ Slave launchers and controllers SSH slaves plugin

Stack

✓ Build reports plugins

Monitoring External Jobs

✓ Artifact uploader plugins - BuildPublisherPlugin - This plugin allows records from one Jenkins to be published on another Jenkins.

✓ Other Post-build actions plugins Deploy Plugin - This plugin takes a war/car file and deploy that to a running remote application server at the end of build.

HTML Publisher Plugin:

Gerrit plugin - This plugin integrates Gerrit Code Review to Jenkins.

weblogic Deployer plugin

✓ UI plugins

✓ List View Column plugins

✓ Page decorators

✓ Authentication and user Management plugins FTP publisher plugin

✓ Cluster management and distributed build plugins

✓ CLI extensions plugins

SCP plugin

✓ Parameters plugins

Deploy to Docker plugin

Screenshot plugin

Build with Parameters plugin: Allows the user to provide parameters for a build in the url, prompting for confirmation before triggering the job.

JIRA plugin: This plugin integrates Atlassian JIRA to Jenkins.

Apache

- The Apache webserver is arguably the most popular web server in use on the Internet today.

- You cannot expose a source code to external world.

- It can be for static content.

- Here we can route control from apache & Tomcat.

- Apache is most widely used webserver in java environment.

- We can install Apache under linux. The three methods:

- Binary installation

- Using an RPM (Redhat Package Manager)

- Building from source

- Default Directory Structure :

Apache :

- bin - binary code, that makes your apache run
- cgi-bin - It contains scripts like shell & perl for redirection
- conf - All the configuration details about your apache
- error
- htdocs
- icons
- logs
- manual
- module

The main configuration file in apache is httpd.conf.

ANT

- ANT is a build tool. stands for Another Next Tool, developed by Apache Org.
- A Small project designed to help Software teams develop big programs by automating the drudge-work of compiling code, running tests and packaging the results for redistribution.

ANT Core Concepts

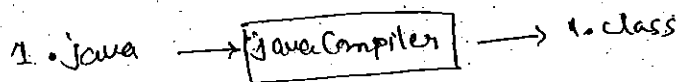
- ANT was designed to be an extensible tool to automate the build process of Java development project.
- ANT uses xml file called build files to describe how to build the targets and action to take on achieve the goal.

Builds

The process of converting our source code into Software Artifacts.

In Java the software artifacts are

1. jar - Java Archive - Collection of .class files
2. war - web Archive - JAR + class + jsp/servlets + web.xml
3. ear - Enterprise Archive - war + EJBs + App.xml



- In ANT the main configuration file is build.xml.

Installation:

- Ant comes in two editions:

1. Binary release - just download, uncompress & go
2. Source release

Q: How to execute ANT script if your configuration file build.xml?

C:\> ant

other than build.xml:

C:\> ant -f some.xml

ant -file some.xml

ant -buildfile some.xml

Q: How to know Ant is installed in your system?

C:\> ant --version.

Q: What is starting tag in build.xml?

<project>

</project>

Building blocks

- Building blocks nothing but a main tags in build.xml file. we can define inside of these blocks.

1. project - name, default, basedir

2. target

3. tasks

- Other important tags:

- java

- javac

- war - warfile, webxml/file, destfile

- ear

- exec - Executable

- scp

- ssh

- copy - file, todir

- delete

- filelist

- fileset, dir

- includes

- excludes

- property - name, value

- antcall - target

- jar - destfile, basedir, includes, excludes

- env - key, value

- ftp

- classpath

✓ Project:

- The Project element is the main tag in build.xml
 - All the buildfile requires the project element and at least one target element.
- Project attributes:

- name: Name of the project
- default: The default target for the build script. A project may contain any number of targets. This attribute specifies which target should be considered as the default.
- basedir: The base directory (or) the root folder for the project.

✓ Target:

A target is a collection of tasks that you want to run as one unit.

Target attributes:

- name: The name of the target.
- depends: Comma separated list of all targets that this target depends on.
- description: A short description of the target.
- if: Allows the execution of a target based on the truthiness of a conditional attribute.
- unless: Adds the target to the dependency list of the specified extension point. An extension point is similar to a target, but it does not have any tasks.

FileSet:

Fileset data type represents a collection of files. The fileset data type is usually used as a filter to include and exclude files that match a particular pattern.

Filelist:

Filelist data type is similar to the file set that the file list contains explicitly named lists of files and do not support wild cards.

On The major difference between the filelist and fileset data type is that the file list can be applied - for files may or may not exist yet.

javac:

- It is used to compile our source code.

Usage: `< javac srcsourcedir = " |path| of | java files |" includeantruntime = "false"
destdir = " |place| own | .class files |" />` classpath

java

- It is used to execute our .class files.

Usage: `< java src classpath = " |where | .class |"
classname = " src filename " />`

jar:

- It is used to create our jar (java archive) files.

`< jar basedir = " |path| of | .class |"
destfile = " |path| to | create | jar | .jar |" />`

exec:

- Execute a System Command. When the os attribute is specified, then the command is only executed when Apache Ant is run on one of the specified os.

Attribute:

command, }
executable }

`< exec executable = " |path| to | start |"`

classpath: Attributes - path element

Build Automation using Apache ANT:

Ant Predefined Properties:

ant.file
ant.version
basedir
ant.java.version
ant.project.name
ant.project.default-target
ant.project.invoked-targets
ant.core.lib
ant.home
ant.library.dir

File^{list}Set:

- The Filelist datatype is similar to the FileSet except that the Filelist contains explicitly named lists of files and do not support wildcards.
- The Major diff b/w Filelist & FileSet is that the Filelist datatype can be applied for files that may or may not exist yet.

FileSet:

The FileSet datatype represents a collection of files. The FileSet datatype is usually as a filter to include & exclude files that match a particular pattern.

path:

The path data type is commonly used to represent a classpath.

```
<path id="build.classpath.jar">
```

```
<pathelement path="{env.J2SE_HOME}/lib/j2ee.jar"/>
```

```
</path>
```

```
<target name="build" ...>
```

```
<classpath refid="build.classpath.jar">
```

- ANT Having mainly 5 targets like

- compile

- build

- test

- deploy-war

- java-doc

- clean

```
<exec executable="{env.EXECUTABLE_FULL_PATH}"
```

```
<scp file="myfile.txt" todir="os@somehost:home/chunk"
```

```
password="password"/>
```

Builds:

Build is nothing but a converting our source code into software artifacts. In Any organization we can maintain the builds as two kinds.

Nightly builds

- These are the scheduled builds. They run at a interval of time.
- These are meant to check the sanity of your code only i.e., They are deployed to development Environment only.

Milestone builds

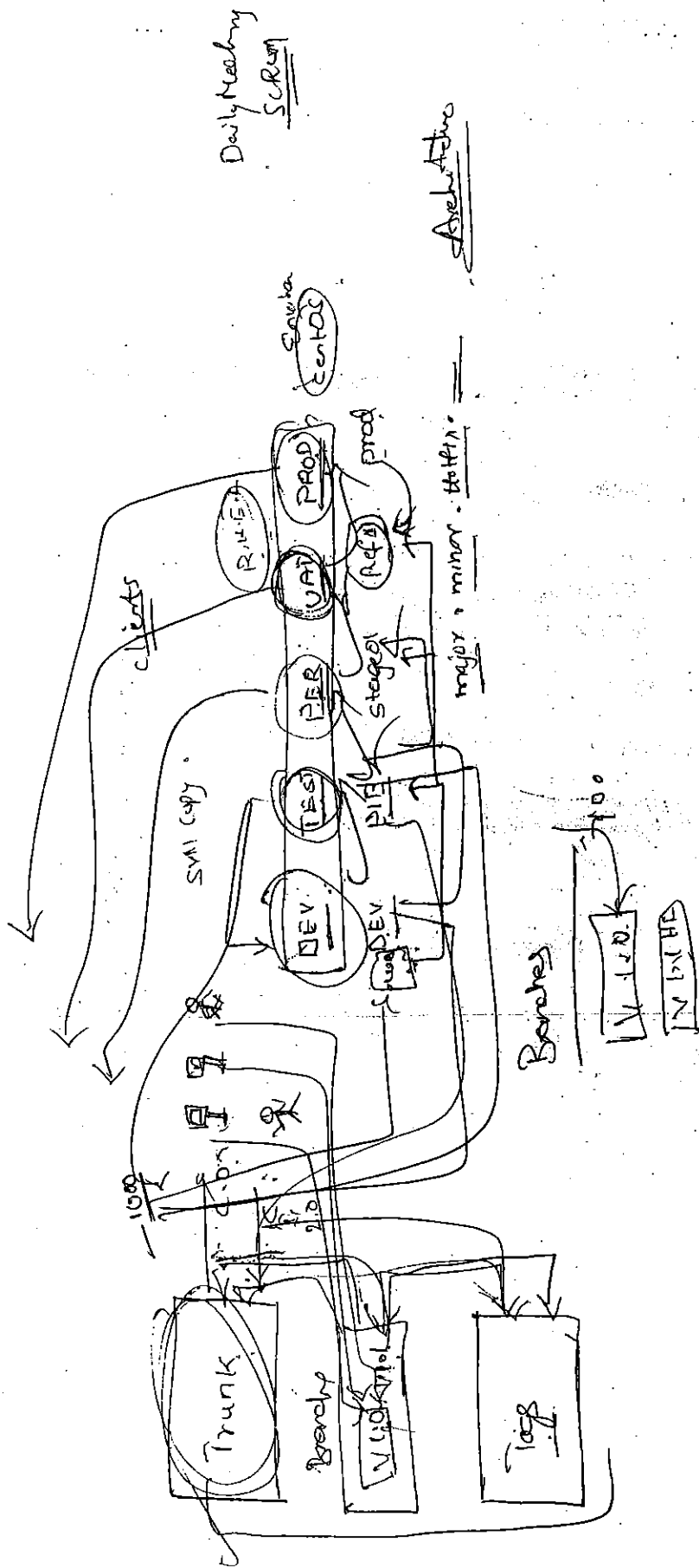
- These are manual builds. i.e., they are Executed manually
- They are not scheduled builds.
- These builds are deployed to higher Environments like QA, perf. &

Best Practices for doing builds

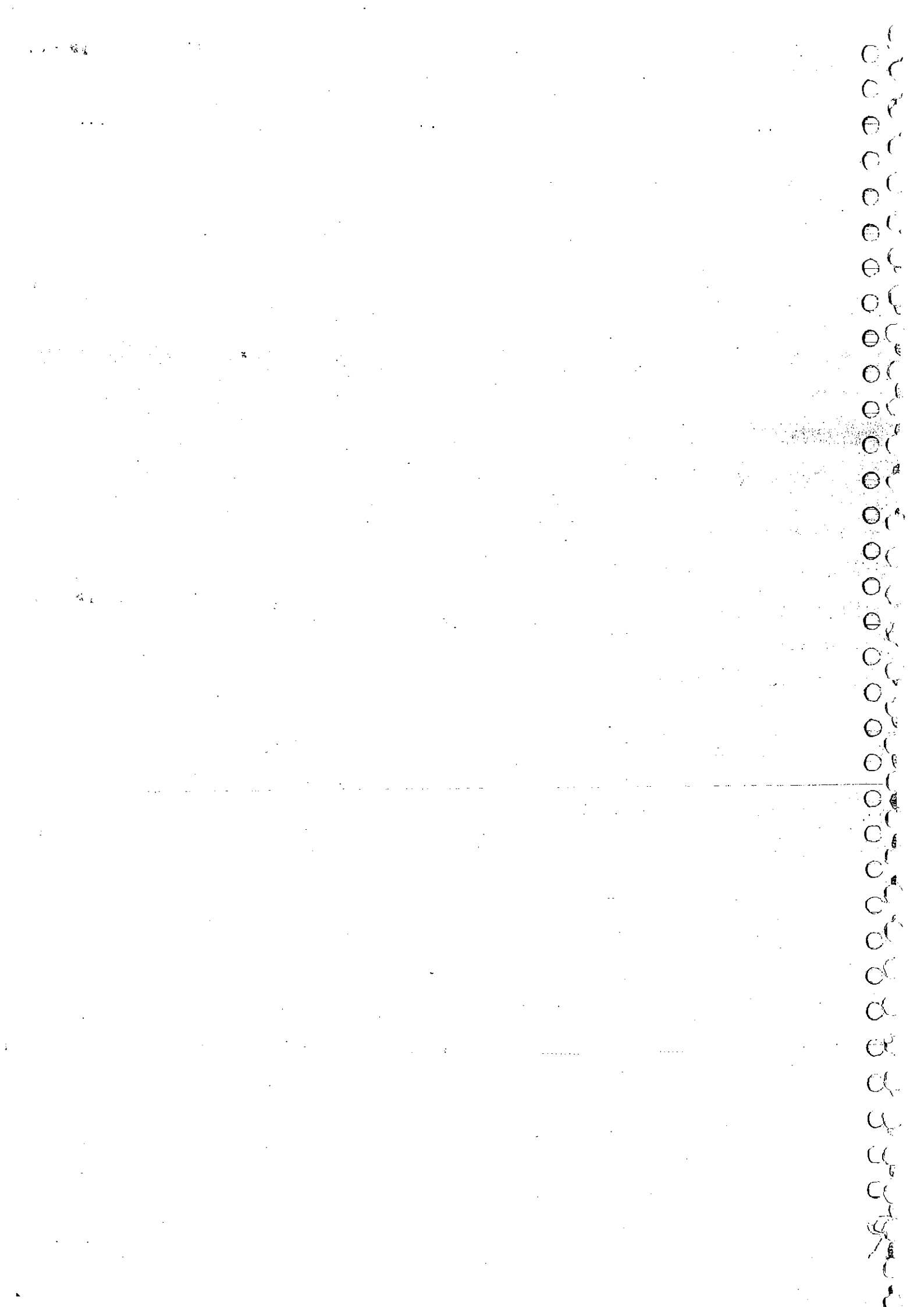
- Always use a Secure and dedicated Server as a build server.
- Checkin all the necessary files prior to build.
- Fully automate the build process.
- Create a build MANIFEST of a build artifact.
- Create & keep your build logs.
- Send the build status via emails.
- Build in a clean Work Space.

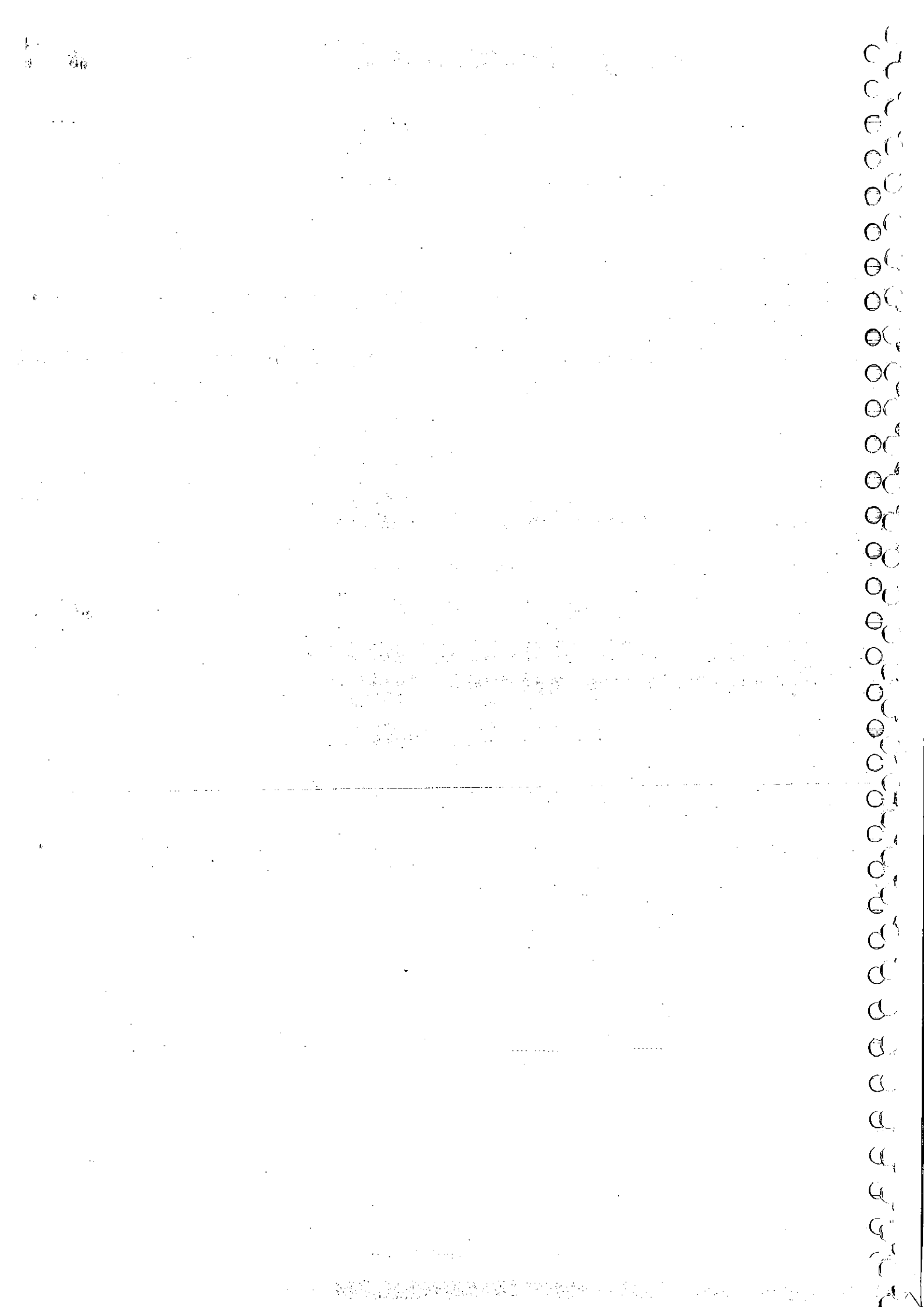
Branching Strategies

1/4

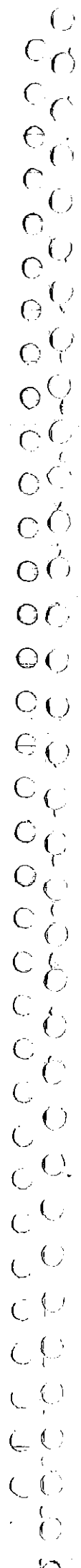








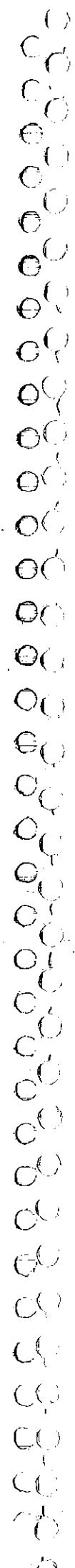
CONFIDENTIAL





[Faint, illegible text covering the majority of the page, possibly bleed-through from the reverse side.]

Page 1 of 1



SCM - Software Configuration Management

ITIL: Information Technology Infrastructure Library

ITIL Vision:

Implement a strategic framework (ITIL) that brings together the technology, people and processes to increase effectiveness, reduce costs, increase productivity, and optimize current services.

Provide a roadmap to improved customers' satisfaction with IT services, better communication and information flows between IT staff and customers, and reduced costs in developing procedures and practice within ISD.

ISD - Information Service Division

ITIL V3 Library / ITIL Service Lifecycle

Service Strategy

Service Design

Service Transition

Service Operation

Continual Service Improvement

ITIL

- Systematic approach to high quality IT service delivery.
- Provides common language .. with well-defined terms.

Service

Delivers value to customer by facilitating outcomes customers want to achieve without ownership of the specific costs and risks.

Service Level:

Measured and reported achievement against one or more service level targets.

eg: Red = 1 hour response 24/7

Amber = 4 hour response 24/7

Green = Next Business Day

Service Level Agreement:

Written and negotiated agreements b/w Service Provider and Customer documenting agreed service levels & costs.

Incidents:

Unplanned interruption to an IT service or an unplanned reduction in its quality.

Problem:

Unknown underlying cause of one or more incidents.

The Service Life Cycle :

Service Strategy:

- strategy generation
- Financial management
- Service portfolio management
- Demand management

Service Design:

- Capacity, Availability, Security, management
- Service level & supplier, mgmt

Service Transition :

- planning & support
- Release & Deployment
- Asset & Config management
- change management
- Knowledge management

Service Operation:

- problem & incident management
- Request fulfillment
- Event & Access management

Continual Service Improvement

- Service measurement & reporting
- 7-step improvement process

- ✓ Incident Management
- ✓ Problem Management
- ✓ Change Management
- ✓ Release Management
- ✓ Configuration Management Database
- ✓ Implementing 24/7 Service Support with Service Desk Plus.

shell

Skills:-

Sun, Maven, Ant Jenkins,
TortoiseSVN, Visual SVN,

ADALMA JF
Talent
walllogic
JBOSS

Source code management

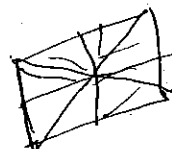
Constr, Maven, Jenkins

Build,
Deployment,

Release,

release, ~~Java~~ Shell scripting
JSP,

DevOps,
Puppet,
Linux,



514489383738 ←