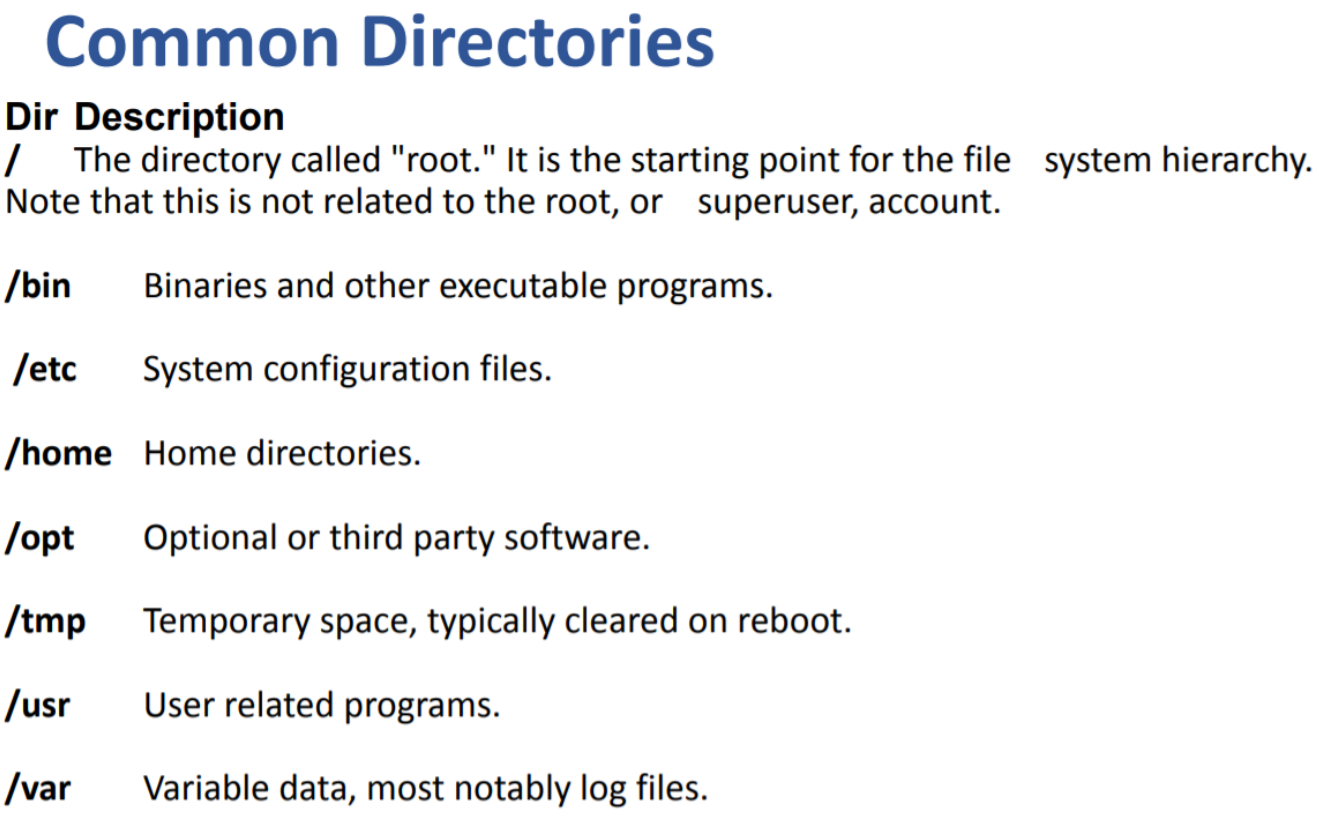
Table of contents

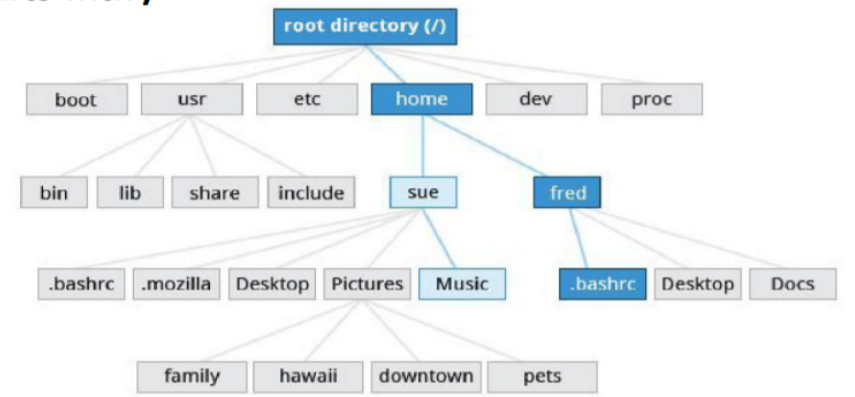
Common Directories



Absolute Path and Relative Path

Absolute Path and Relative Path

- **Absolute pathname**: begins with the root directory , it always starts with **/**
- **Relative pathname**: starts from the present working directory , it never starts with **/**



1.In the above example, we use the relative path method to list the files under Music from your current working directory (sue)
$ ls ../sue/Music

2.In the above example, we use the Absolute pathname method to edit the .bashrc file:
$ gedit /home/fred/.bashrc

Files and Directory Operations

- All input lines entered at the shell prompt have three basic elements:

```
Command
Options
Arguments
```

- The command is the name of the program you are executing.
- It may be followed by one or more options that modify what the command may do.
- Arguments specify on what the command will operate on.

File Operations

```
which ls
whereis ls
locate ls
cat - view a file or concatenate
$tac - view file from backwards
head - print first 10 lines
tail - print last 10 lines
touch - create an empty file
file <name> - will give type of file
mkdir – create a directory
rmdir – remove directory*
mv – move or rename a file
```

```
rm - remove a file (Use cautiously)
rm -f : forcefully remove (Use cautiously)
rm -i : interactively remove a file (Use cautiously)
rm -rf : recursively remove entire tree structure (Use cautiously)
```

File Permissions

Files have three kinds of permissions: read (r), write (w), execute (x). rwx. • These permissions affect three groups of owners: user/owner (u), group (g), and others (o).

`rwx: rwx: rwx u: g: o`

- Numbers for permissions
  - 4 if read permission is desired.
  - 2 if write permission is desired.
  - 1 if execute permission is desired

Symlinks

- Symbolic links are used in linux for primary 2 reasons
  - Creating a shortcut of a file i.e. soft link
  - Creating copy of a file i.e. hard link

```
#Symbolic Links#
#Creating Soft Link
mkdir test-symlinks
cd test-symlinks
echo "AWS Sample Text" > file.txt

cat file.txt

ln -s file.txt softlinkfile.txt

cat file.txt
cat softlinkfile.txt

ls -ail

the inode number and file permissions  are different

rm file.txt
cat softlinkfile.txt

#Creating Hard Link
echo "AWS Sample Text" >file.txt

cat file.txt

ln file.txt hardlinkfile.txt
```

```
cat hardlinkfile.txt
ls -lia

#both hardlinkfile.txt and file.txt have the same the inodes number  and file
permissions

#If we change the permissions on file.txt, the same permission will be applied to
the hardlinkfile.txt as well

```bash
rm file.txt

cat hardlinkfile.txt
```

File descriptors and redirections

- File descriptors are integers associated with an opened file or data stream
    - **0: stdin**
    - **1: stdout**
    - **2: stderr**
- Use the greater-than symbol to append text to a file:

```
echo "This is a sample text 1" > temp.txt
```

- This stores the echoed text in temp.txt. If temp.txt already exists, the single greater-than sign will delete any previous contents.
- Use double-greater-than to append text to a file:

```
echo "This is sample text 2" >> temp.txt
cat temp.txt
```

- A message is printed to the `stderr` stream when a command generates an error message.

```
ls 120
```

-ls: cannot access '120': No such file or directory``

- Here 120 is an invalid argument for `ls` command and hence an error is returned.

> Successful and unsuccessful commands. When a command exits because of an error, it returns a nonzero exit status. The command returns zero when it terminates after successful completion. The
```

> return status is available in the special variable `$?` (run echo $? immediately after the command execution statement to print the exit status).

- Redirect `stderr` to err.txt

```
$ ls + 2> err.txt
```

- Redirect `stdout` to out.txt file

```
$ ls 1> out.txt
```

- We can redirect stderr to one file and stdout to another file. `cmd 2>stderr.txt 1>stdout.txt`

Firewall

- Similar to aws security group , we also have a OS level firewall which we can use to block traffic
- This firewall comes as a package and can used to add or remove flow of traffic

Firewall Commands

- Launch 2 EC2 instances (`EC2-A and EC2-B`) with Amazon Linux AMI and allow private ip connection on both in AWS Security Group.
- Change the hostname of the each machine:
- On `EC2-A`

```
`sudo hostnamectl set-hostname a-ec2.example.com && bash`
```

- On `EC2-B`

```
`sudo hostnamectl set-hostname b-ec2.example.com && bash`
```

- Edit the /etc/hosts file on both the machines
- Install httpd, start httpd, write some content to /var/www/html/index.html
- Test the request of httpd to the EC2-A instance

```
curl EC2-A-private-ip
```

- The output for above command will be content of the /var/www/html/index.html

- On EC2-A

```
#install firewalld

yum install -y firewalld


 #Enable the service at boot time

systemctl enable firewalld


#Start the service

systemctl start firewalld

firewall-cmd --state


#After the firewalld service is started, test the curl from EC2-B to EC2-A again

curl EC2-A-private-ip


#By default, firewalld will be active and will reject all incoming traffic with a
couple of exceptions, like SSH.

#List information for all zones


firewall-cmd --list-all-zones


#To check which is the default zone

firewall-cmd --get-default-zone


#To Enable all the incoming ports for a service

firewall-cmd --zone=public --add-service=http

firewall-cmd --list-services



#Test the curl from EC2-B to EC2-A again

#To List the services that are allowed for the public zone

firewall-cmd --zone=public --list-services
```

```
# Here only runtime configuration is updated, it is lost if firewalld service is
restarted.


systemctl restart firewalld
firewall-cmd --zone=public --list-services


#Use below command to make this changes permanent

firewall-cmd --permanent --zone=public --add-service=http
firewall-cmd --zone=public --list-services


#Remove a service from a zone

firewall-cmd --permanent --zone=public --remove-service=http

#Test the curl from EC2-B to EC2-A again

#Traffic can be allowed on specific port
#firewall-cmd --add-port=[YOUR PORT]/tcp

firewall-cmd --add-port=80/tcp


# To add above permanently
#firewall-cmd --permanent --add-port=[YOUR PORT]/tcp


firewall-cmd --permanent --add-port=80/tcp

#test the curl from EC2-B to EC2-A again

# To list what ports are open use below

firewall-cmd –-list-ports


#Add/Remove a specific source IP

firewall-cmd --permanent --add-source=<PRIVATE_IP>
firewall-cmd --permanent --remove-source=<PRIVATE_IP>

firewall-cmd --zone=public --list-all
```

httpd configuration port change

```
cat /etc/httpd/conf/httpd.conf | grep -i "80"
```

- Modify the Port Number for httpd in above config file and restart the service.
- Check for particular service running on a particular port with below command.

```
netstat -nltp
```

- Use the above `--add-port=80/tcp` command above to allow connections on a specific port.

```
## Debugging Shell scripts
- Debugging helps you troubleshoot and resolve such errors, and is one of the most
important tasks a system administrator performs.
```

bash –x ./script_file

```
## User Operations
- Only root (i.e. system administrator)can use adduser command
to create new users. It is not allow to other users.
- You can see list of all users in /etc/passwd file
- The list will contain both system users as well as users created later
```bash
$who : list of users currently logged in (-a option for detailed info)
$whoami : current user
$adduser : adding a user
$deluser : deleting a user
$usermod : modify user account $ usermod -G wheel username
$groupadd : add a group (Ex - $groupadd <groupname>)
(cat /etc/group file contains group and user information )
$id <username> : gives details about user
$adduser <un> <gn> : adding user to an existing group
$usermod -G groupname username
```

Yum package manager:

- A package is a compressed archive of all the files for a software to run .
- Mutliple linux distributions contain various versions and types of kernels . All packages may not be compatible with all linux flavors. Installation of said packages may fail due to dependencies
- Package manager is a software responsible for installing, upgrading , configuring and removing packages from linux
- DPKG,APT,RPM and YUM are some of the most common package managers
```

- RPM stands for redhat package manager . Amazon linux also makes use of the same . File extension is ".rpm"
- basic modes of operation
    - Install
    - Uninstall
    - Query
    - Upgrade
    - Verify
- To install package we can use

```
#download below package for testing using command
wget https://download-ib01.fedoraproject.org/pub/epel/7/aarch64/Packages/s/stress-1.0.4-16.el7.aarch64.rpm

#if we want to see all the available rpm packages
rpm -qa

# To install a certain package , you can use below command. where -i stands for install and v is verbose which gives detail information about the operation
rpm -iv /var/lib/rpm/stress-1.0.4-16.el7.aarch64.rpm
# To uninstall the package we'll use -e
rpm -e /var/lib/rpm/stress-1.0.4-16.el7.aarch64.rpm
# in order to upgrade the package use -U
rpm -Uv /var/lib/rpm/stress-1.0.4-16.el7.aarch64.rpm
# -q stands for query . You can use the same to verify if the package exists and if it is installed
rpm -q /var/lib/rpm/stress-1.0.4-16.el7.aarch64.rpm
# -V is used to verify the source of the package to ensure it is from a trusted source
```

- From testing above commands , we can conclude that RPM doesnt reolve package dependencies
- FOr this we can use higher level package manager called Yum
- Yellow Dog Updater Modified . Works on RPM based systems
- Yum repos have .repo as extension . It still uses rpm in the background
- Yum depends on repositories . Which contains the information of all required rpm files . You can see them in `/etc/yum.repos.d`

```
# to see all repos added to your system
yum repolist

#if we want to know which packages must be installed in order for a command to work
yum provides wget
# to remove a package
```

```
yum remove httpd

# to update a package
yum update httpd
# to update all packages
yum update
# in order to see the packages installed recently
yum history list
```

- Other than the primary commands listed above , below are few other reference commands
- list all available packages `yum list available`
- to list all installed packages `yum list installed`
- list installed and available packages `yum list all`
- list installed and available kernel packages `yum list kernel`
- get info related to a package `yum info httpd`
- get the dependencies of a particular package `yum deplist httpd`
- search whether a particular package is available `yum search httpd`
- delete packages saved in cache `yum clean packages`
- clean out all packages and meta data from cache `yum clean all`

Filters in aws cli - JMESPath

- In AWS cli we can always chose the type of output one can get after a command ex. text ,json
- JSON is a default and preffered way of output for these commands
- A JSON object is an unordered set of name/value pairs.
    - An object begins with `{left brace` and ends with `}right brace`.
    - Each name is followed by `:colon` and the name/value pairs are separated by `,comma`.
- An array is an ordered collection of values.
    - An array begins with `[left bracket` and ends with `]right bracket`. Values are separated by `,comma`.
- A value can be a `string in double quotes`, or a `number`, or `true or false` or `null`, or an `object` or an `array`. These structures can be nested.

```
{
  "key": "value",
  "keys": "must always be enclosed in double quotes",
  "numbers": 0,
  "strings": "Hello, world",
  "hasbools": true,
  "nothingness": null,
  "objects": {
    "comment": "Most of your structure will come from objects.",
    "array": [0, 1, 2, 3, "Arrays can have anything in them.", 5],
    "anotherobject": {
```

```
      "comment": "These things can be nested, very useful."
    }
  }
}
```

- You will always feel the need to pick out the relevant details from a json output primarily when dealing with scripts
- `--query` parameter helps us pick out exactly what we need from the output . We have seen this breifly in kms cli

```
# Lets run below command to get a feel of generic json based outputs of aws cli
aws ec2 describe-volumes --region us-east-1
# As you can observe , there are bunch of details which are displayed , all of
which might not be relevant for the time being
aws ec2 describe-volumes --query 'Volumes[0]'
# above command will only fetch the first value from  the array of volumes .
# In order to filter out relevant details like  volume id and instance id, we can
add relevant values. here we are using dictionary notation i.e. {} which can get
the nested information like instance id
aws ec2 describe-volumes --query 'Volumes[*].
{ID:VolumeId,InstanceId:Attachments[0].InstanceId,AZ:AvailabilityZone,Size:Size}'
# To filter the output by the value of a specific field, use ? expression
aws ec2 describe-volumes --query 'Volumes[?AvailabilityZone==`ap-south-1b`]'
# To list all volumes that were created after a specified date
aws ec2 describe-volumes --query 'Volumes[?CreateTime>=`2020-02-07`].
{Id:VolumeId,CreateTime:CreateTime,AZ:AvailabilityZone}
```