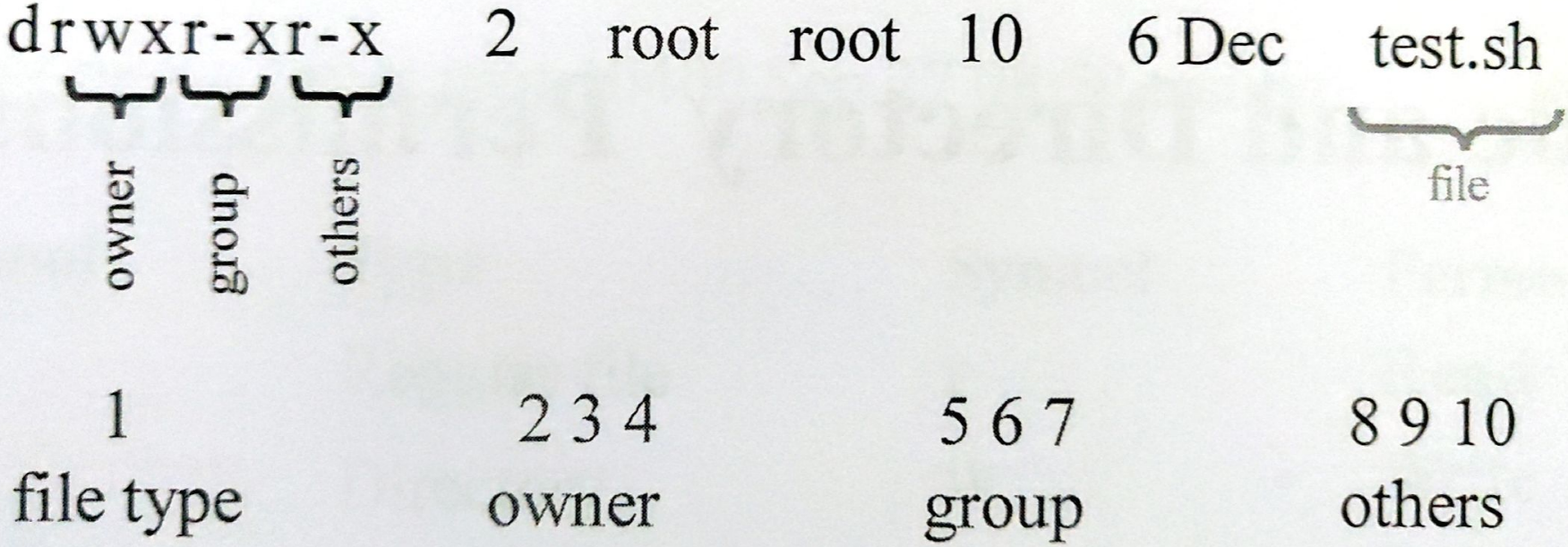
File and Directory Permissions

Permissions:

format from 'ls -l' command



Permissions:

```
$ ls -l
-rw-rw-r-- 1 root root 10400 Sep 27 08:52 test.txt
```

Symbol	Type	Symbol	Permission
-	Regular file	r	Read
d	Directory	w	Write
l	Symbolic link	x	Execute

Permissions - Files vs Directories:

Permission	File	Directory
Read (r)	Allows a file to be read.	Allows file names in the directory to be read.
Write (w)	Allows a file to modified.	Allows entries to be modified within the directory.
Execute (x)	Allows execution of a file.	Allows access to contents and metadata for entries.

Directory Permissions Revisited:

- Permissions on a directory can effect the files in the directory.
- If the file permissions look correct, start checking directorypermissions.
- Work your way up to the root.

- read permission – 4, write – 2, execute 1

Eg. rwxrw-r-- =764
673 = rw-rwx-wx

- **Chmod** (**ch**ange **mode**) is used to change the permissions on a file.

(owner) (group) (others)

chmod [number][number][number] file1

Number = (read)4 + (write)2 + (execute)1

- Example: Chmod 754 file1

for owner: *read*, *write* and *execute* permissions (4+2+1)

for group: *read* and *execute* permissions (4+0+1)

for others: only *read* permission (4+0+0)

Permission Categories:

Symbol	Category
u	User
g	Group
o	Other
a	All

Changing Permissions:

Item	Meaning
chmod	Changemode command
ugoa	User category user, group, other, all
+-=	Add, subtract, or set permissions
rwx	Read, Write, Execute

Ex: chmod u+rwx test.sh

Octal	String	Description
0	---	No permissions
1	--x	Execute only
2	-w-	Write only
3	-wx	Write and execute (2+1)
4	r--	Read only
5	r-x	Read and execute (4+1)
6	rw-	Read and write (4+2)
7	rwx	Read, write, and execute (4+2+1)

Commonly Used Permissions:

Symbolic	Octal
-rwx------	700
-rwxr-xr-x	755
-rw-rw-r--	664
-rw-rw----	660
-rw-r--r--	644

Groups:

- Every user is in at least one group.
- Users can belong to many groups.
- Groups are used to organize users.
- The groups command displays a user's groups.
- You can also use id -Gn.

Working with Groups:

- New files belong to the primary group.
- The chgrp command changes the group.
  - *$ groups*
  - *$ id –Gn*
  - *$ groups adminuser*
  - *$ chgrp vagrant test.sh*

File Creation Mask:

- File creation mask determines default permissions.
- If no mask were used permissions would be:
  - 777 for directories
  - 666 for files

The umask Command:

umask [-S] [mode]

- Sets the file creation mask to mode, if given.
- Use -S to for symbolic notation.

	Directory	File
Base Permission	777	666
Subtract Umask	-022	-022
Creations Permission	755	644

	Directory	File
Base Permission	777	666
Subtract Umask	-002	-002
Creations Permission	775	664

Octal Subtraction Is an Estimation:

	Directory	File
Base Permission	777	666
Subtract Umask	-007	-007
Creations Permission	770	660 *

Common umask modes:

- 022
- 002
- 077
- 007

Special Modes:

- umask 0022 is the same as umask 022
- chmod 0644 is the same as chmod 644

Summary:

- Symbolic permissions
- Numeric / octal permissions
- File versus directory permissions
- Changing permissions
- Working with groups
- File creation mask

Finding Files and Directories

Rise 'n' Shine Technologies

The find Command:

$ find [path…] [expression]

-- Recursively finds files in path that match expression.

-- If no arguments are supplied it find all files in the current directory.

find Options:

-name pattern	Find files and directories that match pattern.
-iname pattern	Like -name, but ignores case.
-ls	Performs an ls on each of the found items.
-type f/d	Performs file or directory

find Options:

-mtime days # Finds files that are days old.
-size num # Finds file that are of size num.
-newer file # Finds files that are newer than file.

-exec command {} \;
Run command against all the files that are found.

Examples:

-- Find all Empty Files

To file all empty files under certain path.

 # find /tmp -type f -empty

-- Find all Empty Directories

To file all empty directories under certain path.

 # find /tmp -type d –empty

-- Find and remove Multiple File

To find and remove multiple files such as **.mp3** or **.txt**, then use.

 # find . -type f -name ".txt" -exec rm -f {} \;*

Examples:

-- Find Last 50 Days Modified Files

To find all the files which are modified **50** days back.

 # find / -mtime 50

-- Find Last 50-100 Days Modified Files

To find all the files which are modified more than **50** days back and less than **100** days.

 # find / -mtime +50 –mtime -100

-- Find Size between 50MB – 100MB

To find all the files which are greater than **50MB** and less than **100MB**.

 # find / -size +50M -size -100M

Editing Files with Vi

Rise 'n' Shine Technologies

The Vi Editor:

- Has advanced and powerful features
- Not intuitive
- Harder to learn than nano
- Requires a time investment

The Vi Editor:

$ vi [file]	Edit file.
$ vim [file]	Same as vi, but more features.
$ view [file]	Starts vim in read-only mode.

Vi Modes:

Mode	Key
Command	Esc
Insert	i I a A

i	Insert at the cursor position.
I	Insert at the beginning of the line.
a	Append after the cursor position.
A	Append at the end of the line.

Vi Line Mode:

:w	Writes (saves) the file.
:w!	Forces the file to be saved.
:q	Quit.
:q!	Quit without saving changes.
:wq!	Write and quit.
:x	Same as :wq.

Vi Line Mode:

:n	Positions the cursor at line n.
:$	Positions the cursor on the last line.
:set nu	Turn on line numbering.
:set nonu	Turn off line numbering.

Vi - Deleting Text:

x	• Delete a character.
dw	• Delete a word.
dd	• Delete a line.
D	• Delete from the current position.

Vi - Copying and Pasting:

yy	Yank (copy) the current line.
dd	cut the current line
nyy	will copy n number of lines
ndd	will cut n of lines
p	Paste the most recent cut or yanked text.

Vi –Searching:

/<pattern>	Start a forward search.
?<pattern>	Start a reverse search.

Multiple Files:

vim -o file1 file2 --> will open 2 files vertically

vim -O file1 file2 --> will open 2 files horizontally

ctrl + w --> to switch the files

Wildcards

Rise 'n' Shine Technologies

What You Will Learn:

- What wildcards are.
- When and where they can be used.
- The different types of wildcards.
- How to use wildcards with various commands.

Wildcards:

- A character or string used for pattern matching.
- Globbing expands the wildcard pattern into a list of files and/or directories.
- Wildcards can be used with most commands.
  - ls
  - rm
  - cp

Wildcards:

- *- matches zero or more characters.
  - *.txt
  - a*
  - a*.txt
- ?- matches exactly one character.
  - ?.txt
  - a?
  - a?.txt

More Wildcards - Character Classes:

- [] - A character class.
  - Matches any of the characters included between the brackets. Matches exactly one character.
  - [aeiou]
  - ca[nt]*
    - can
    - cat
    - candy
    - catch

More Wildcards - Character Classes:

- [!] - Matches any of the characters NOT included between the brackets.
- Matches exactly one character.
  - [!aeiou]*
    - baseball
    - cricket

More Wildcards – Ranges:

- Use two characters separated by a hyphen to create a range in a character class.
- [a-g]*
  - Matches all files that start with a, b, c, d, e, f, or g.
- [3-6]*
  - Matches all files that start with 3, 4, 5 or 6.

Named Character Classes:

- [[:alpha:]]
- [[:alnum:]]
- [[:digit:]]
- [[:lower:]]
- [[:space:]]
- [[:upper:]]

Matching Wildcard patterns:

- \ - escape character.
- Use if you want to match a wildcard character.
    - Match all files that end with a question mark:
        - *\?
            - done?

Summary:

- *
- ?
- []
- [0-3]
- [[:digit:]]

I/O Redirection

Rise 'n' Shine Technologies

Summary:

- *
- ?
- []
- [0-3]
- [[:digit:]]

Input/Output Types:

I/O Name	Abbreviation	File Descriptor
Standard Input	stdin	0
Standard Output	stdout	1
Standard Error	stderr	2

Redirection:

>	Redirects standard output to a file. Overwrites (truncating) existing contents.
>>	Redirects standard output to a file. Appends to any existing contents.
<	Redirects input from a file to a command.

Redirection:

& # Used with redirection to signal that a file descriptor is being used.

2>&1 # Combine stderr and standard output.

2>file # Redirect standard error to a file.

The Null Device:

>/dev/null Redirect output to nowhere.

$ ls files.txt not-here 1> out 2>out.err

$ ls files.txt not-here > out.both 2>&1

$ ls files.txt not-here 2>/dev/null

$ ls files.txt not-here > /dev/null 2>&1

Summary:

- Standard input
- Standard output
- Standard error
- Redirection
- Null device

Comparing Files

Rise 'n' Shine Technologies

Comparing the Contents of Files:

$ diff file1 file2	Compare two files.
$ sdiff file1 file2	Side-by-side comparison.
$ vimdiff file1 file2	Highlight differences in vim.

diff Output:

$ diff file1 file2

3c3

...

LineNumFile1-Action-LineNumFile2

Action = (A)dd (C)hange (D)elete

diff Output:

```
$ diff file1 file2
3c3
< this is a line in a file.
---
>    This is a Line in a File!
```

```
<    Line from file1
>    Line from file2
```

sdiff Output:

```
$ sdiff      file1      file2
line in      file1      |          line      in      file2
                        >          more      in      file2
```

|	Differing lines
<	Line from file1
>	Line from file2

Vimdiff:

Ctrl-w	w	Go to next window
:q		Quit (close current window)
:qa		Quit all (close both files)
:qa!		Force quit all

Searching in Files

Using Pipes

Rise 'n' Shine Technologies

The grepCommand:

$ grep # Display lines matching a pattern.

$ grep pattern file

grep Options:

-i	Perform a search, ignoring case.
-c	Count the number of occurrences in a file.
-n	Precede output with line numbers.
-v	Invert Match. Print lines that don't match.
-An	print n number lines after occurence
-Bn	print n number of lines before occurence

The file Command:

$ file file_name # Display the file type.

$ file sales.data sales.data: ASCII text

$ file *

bin: directory

test.tar: POSIX tar archive

Pipes:

| Pipe symbol

command-output | command-input

grep pattern file

cat file | grep pattern

The cut Command:

$ cut [file]

-- Cut out selected portions of file.
-- If file is omitted, use standard input.

cut Options:

-d delimiter

Use delimiter as the field separator.

-f N

Display the Nth field.

Searching and Pipe Example:

- Find all users named "adminuser" in /etc/passwd.
- Print accountname andreal name.
- Print in alphabetical order by accountname.
- Print in atabular format.

- grep –i vagrant /etc/passwd | cut –d: -f1,5 | tr “:” “ ” | column –t

Piping Output to a Pager:

- more
- less

Summary:

- grep
- file
- cut
- tr
- column
- more
- less
- Pipes

Customizing the Shell Prompt

Rise 'n' Shine Technologies

Customizing the Shell Prompt

- Use an environment variable to customize.
- Bash, ksh, and sh use $PS1.
- Csh, tcsh, and zsh use $prompt.
- Command to get the current bash -

```
$ echo $0

$ echo $SHELL
```

Customizing the Prompt with PS1

\d	Date in "Weekday Month Date" format "Tue May 26"
\h	Hostname up to the first period
\H	Hostname
\n	Newline
\t	Current time in 24-hour HH:MM:SS format
\T	Current time in 12-hour HH:MM:SS format

Customizing the Prompt with PS1

\@	Current time in 12-hour am/pm format
\A	Current time in 24-hour HH:MM format
\u	Username of the current user
\w	Current working directory
\W	Basename of the current working directory
\$	if the effective UID is 0, a #, otherwise a $

Persist PS1 Changes:

```
$ echo 'export PS1="[\u@\h \w]\$ "' >> ~/.bashrc
```

Using Aliases

Rise 'n' Shine Technologies

Aliases

- Shortcuts
- Use for long commands
- Use for commands you type often

Creating Aliases:

List aliases:

$ alias

createaliases:

$ alias name=value

Aliases:

- Fix Typos

  $ alias grpe='grep'

- Make Linux behave like another OS.

  $ alias cls='clear'

Removing Aliases:

unalias name # Removethe“name”alias.

unalias –a # Remove allaliases

Persisting Aliases:

- Add aliases to your personal initialization files.

  .bash_profile

Environment Variables

Rise 'n' Shine Technologies

Environment Variables:

- To store value in a variable
- Name/Value pairs
- Can be changed how an application behaves.
- An Example Environment Variable:

  HOME=/home/vagrant

Viewing Environment Variables

-- printenv # Print all or part of environment.

-- echo $ENV_VAR # Print the ENV_VAR variable.

Examples:

$ printenv HOME
/home/vagrant

$ printenv home

Common Environment Variables

Variable	Description
EDITOR	# Program to run to perform edits
HOME	# Home directory of the user.
LOGNAME	# The login name of the user.
MAIL	# The location of the user's local inbox.
OLDPWD	# The previous working directory.
PATH	# A list of directions to search for commands.

Common Environment Variables

Variable	Description
PS1	# The primary prompt string
PWD	# Present working directory.
USER	# The username of the current user.

Creating Environment Variables:

Syntax:

export var="value"

Example:

export EDITOR="vi"

export TZ="Asian/Calcutta"

Removing Environment Variables

unset - delete an environment variable.

Syntax:

$ unset VAR

Example:

$ unset TZ