FILE SYSTEM

expr : COMPUTATION AND STRING HANDLING

Performs two main functions:

1. Performs arithmetic operations on integers.
2. Manipulates strings.

- expr can handle only integers, division yields only integral part.

expr

COMMAND	DESCRIPTION
expr 8 + 4	Addition of two numbers
expr 8 – 4	Subtraction of two numbers
expr 8 / 4	Division of two numbers
expr 8 % 4	Modulus of a number
expr 8 * 4	Multiplication of two numbers

expr: STRING HANDLING

- For manipulating strings, **expr** uses two expression separated by a colon.

- String to be worked upon is placed on the left of the : and regular expression is placed on its right.

expr: STRING HANDLING

- expr can perform three important string functions:
  - Determine the length of the string
  - Extract a substring.
  - Locate the position of a character in a string.

expr: STRING HANDLING

COMMAND	DESCRIPTION
expr “abcdefgh” : ‘.*’	‘.*’ signifies to expr that it has to print number of characters matching the pattern. It returns length of the string=8
s=2003 expr “$s” : ‘..\(..\)’	Extracts last two characters
s=abcdefghi ; expr “$s” : ‘[^d]*d’	Locating the position of a character. Location of d is 4
expr linux : lin **Ans: 3**	returns the number of characters matched
$ expr linux : linx **Ans: 0**	the condition is string 2 entirely should match in string 1.
expr linux : '.*‘ **Ans: 5**	to match any number of characters

chmod

- chmod stands for change mode.
- Used to set the permission of one or more files for all three categories of users (user, group and others).
- It can be run by the user and the superuser.
- chmod command can be used in two ways:
  - In relative manner by specifying the changes to the current permissions.
  - In absolute manner by specifying the final permissions.

RELATIVE PERMISSIONS

- In this, chmod only changes the permissions specified in the command line and leaves the other permission unchanged.

- Syntax:

  chmod category operation permission filename(s)

RELATIVE PERMISSIONS

- Chmod takes its argument an expression that contains three components:
  - Category (user(u), group(g), others(o), all(a))
  - Operation (assign(+), remove(-))
  - Type of permission (read(r), write(w) and execute(x))

REFERENCES

Reference	Class	Description
u	owner	file's owner
g	group	users who are members of the file's group
o	others	users who are neither the file's owner nor members of the file's group
a	all	All three of the above, same as ugo

OPERATORS

Operator	Description
+	Adds the specified modes to the specified classes
-	Removes the specified modes from the specified classes
=	The modes specified are to be made the exact modes for the specified classes

MODES

```
r       Permission to read the file.
w       Permission to write (or delete) the file.
x       Permission to execute the file, or, in
        the case of a directory, search it.
```

RULES

- The very first column represents the type of the file i.e. is it a normal file or a directory where d represents a directory and – represents a normal file.

- The first set three letters after the file type tell what the Owner of the file, have permissions to do.

RULES

- The next three letters after the user’s permission are the group’s permissions.

- The last three letters in the permissions column tell us what the “others” may do. The general practice is to protect the files from external access so that others can’t write any files or directories. They may read(r) or execute(x) it.

EXAMPLE1

- Let’s change the assgn1_client.c permission so that the owner cannot write(w) in the file but can only read it.

BEFORE: -rw-rw-r-- mik mik a.c

COMMAND: chmod u=r a.c

AFTER: -r--rw-r-- mik mik a.c

RELATIVE PERMISSION

- chmod accepts multiple filenames in the command line.
- chmod accepts multiple expressions delimited by commas.
- In chmod, more than one permission can also be set.

RELATIVE PERMISSIONS

COMMAND	DESCRIPTION
chmod u+x abc.txt	Assigns execute permission to user but other permission remain unchange
chmod a+x abc.txt	a stands for all and implies ugo. All categories has execute permission.
chmod +x abc.txt	By default, a is implied.
chmod u+x file1 file2 file3	Assign the same set of permissions to a group of files
chmod go-r abc.txt ; ls –l abc.txt	Remove the read permission from both group and others.
chmod a-x, go+r abc.txt ; ls –l abc.txt	Removing execute permission from all and assigning read permission to group and other categories.
chmod o+wx abc.txt ; ls –l abc.txt	Assigning more than one permission to others category.

ABSOLUTE PERMISSION

- In this, chmod use three digit string as the expression.
- Octal numbers have the values 0-7.
- For each category, three permissions and three category are there so set of three bits can represent one octal digit.
- Permission of each category is represented by one octal digit.

ABSOLUTE PERMISSION

Binary	*Octal*	*Permissions*	*Significance*
000	0	---	No permissions
001	1	--x	Executable only
010	2	-w-	Writable only
011	3	-wx	Writable and executable
100	4	r--	Readable only
101	5	r-x	Readable and executable
110	6	rw-	Readable and writable
111	7	rwx	Readable, writable and executable

ABSOLUTE PERMISSION

COMMAND	**MEANING**
chmod 761 xstart	Assign all permissions to owner, read and write to group and only execute permission to others.

chmod

- File’s permissions can only be changed by the owner of the file.

cat

- cat (short for “concatenate“) command is one of the most frequently used command in Linux.
- Cat command allows us to:
  - create single or multiple files,
  - view contain of file,
  - concatenate files and
  - redirect output in terminal or files.
- General Syntax

  cat [OPTION] [FILE]...

cat

COMMAND	DESCRIPTION
cat filename1 > filename2	To copy the contents of one file to another file. Please make sure destination file is empty otherwise it will overwrite the contents of destination file.
cat filename1 >> filename2	Appends the contents to the destination file.
cat filename	To view contents of a file
cat file1 file2 file3 | sort > file4	Sorting Contents of Multiple Files in a Single File
cat file1 file2 > file3	Redirecting Multiple Files Contain in a Single File

cat

COMMAND	DESCRIPTION
cat filename 1 filename 2	To display multiple files
cat -n filename	To display contents of file with line numbers
cat > filename	To create file . When run, the command requires you to enter the information on the terminal. Once you're done, just press CTRL+d. To verify, use ls command or cat [name of new file]

cat

COMMAND	DESCRIPTION
cat -E filename	To make cat highlight line-ends. It can be done by displaying *$* at the end of each line.
cat -s filename	To suppress repeated empty lines
cat < filename	Redirecting Standard Input with Redirection Operator (use file name as a input for a command and output will be shown in a terminal.)

cp : COPYING A FILE

- Copies a file or group of files.
- Creates an exact image of the file on the disk with a different name.
- Syntax:

cp [OPTION] SOURCEFILE DESTFILE

- Syntax requires atleast two filenames.
- Destination can be ordinary file or directory.
- cp overwrites without warning the destination file if it exists.

cp

COMMAND	DESCRIPTION
cp file1 file2	To copy source file into destination file
cp –i file1 file2	To make cp prompt before overwriting
cp file1 file2 dir1	To copy multiple files into directories
cp –r dir1 dir2	To copy one dir to another dir
cp chap* dir1	Copies all files beginning with chap to dir1
cp file1 dir/dir1	File1 is copied to dir1 under dir

mv : RENAMING FILES

- It has two main functions:
  - Rename a file or directory
  - Moves a group of files to different directory.

- SYNTAX:
  mv [OPTION] SOURCE DESTINATION

- mv requires the user to have write permission for the directories the file will move between. This is because mv changes the file's location by editing the file list of each directory.

mv

COMMAND	DESCRIPTION
mv file1 file2	To rename a file
mv dir1 dir2	To rename a Directory
mv -i original new	To prompt for a confirmation before overwriting
mv file1 file2 dir1	Move the files to directory

rm : DELETING FILES

- **rm** stands for **remove.**

- rm command is used to delete or remove one or more files or directory.

- Be careful while running rm command because once you delete the files then you can not recover the contents of files and directory.

- Syntax:

  rm file1 file2

rm

COMMAND	DESCRIPTION
rm file1	Remove or delete a file.
rm file1 file2 file3	Delete multiple files at once.
rm -i file1	prompt before deleting a file
rm -d dir1	Delete a empty directory
rm –rf*	Removes forcefully in the current directory and below.
rm -ri dir1	Delete the files and sub-directories interactively

rm

COMMAND	DESCRIPTION
rm -f *.txt	Delete all the files of the current directory that ends with ‘.txt’
rm -f *.???	Delete all files of present working directory which has 3 characters in extension.
rm *	Deletes all the files
rm –f	Removes forcefully

clear

- By using this command, screen clears and cursor is positioned at the top-left corner of the screen.

umask

- Is for default file and directory permissions.
- It is shell built-in command though it also exists as an external command.
- UNIX has following default permissions for files and directories:
  - rw-rw-rw- (octal 666) for regular files
  - rwxrwxrwx (octal 777) for directories.

umask

- We don’t see these permissions when file or directory is created.
- This default is transformed by subtracting the user mask from it to remove one or more permissions.
- Umask 022 is octal number which has to be subtracted from the system default to obtain actual default.
- It becomes 644 for ordinary files and 755 for directories.

history

- It is also built-in command.
- By default, the command displays all events in the list.
- Each command is shown prefixed with the event number.
- Every command you run gets added to the list.
- Last command shows the history command itself.
- List can be big enough so we can restrict the size.
  - Example: history 7 shows only last seven commands.

grep

- It scans its input for a pattern and displays lines containing the pattern, the line numbers or filenames where pattern occurs.
- Syntax:

  grep options pattern filename(s)

GREP EXAMPLES

SYNTAX	EXAMPLE	MEANING
grep option f1.txt	**grep am f1.txt** Example: yamha lamborgini	Returns result for matching string “options”
grep –i option f1.txt	**grep –i am f1.txt**	Returns a result for case insensitive strings
grep –n option f1.txt	**grep –n am f1.txt**	Returns the matching strings along with the line number
grep –v option f1.txt	**grep –v am f1.txt**	Returns the result of lines not matching the search string.
grep –c option f1.txt	**grep –c am f1.txt**	Returns the number of lines in which the result matches search string.