LINUX Reference (from web)

# Files

Every UNIX program has three file streams opened for it when it starts up under normal conditions. One is for input, other is for output and another one is opened for printing error messages.

## Stdin

The input stream is referred to as the "standard input" stream (stdin). Under normal circumstances, File descriptor 0 is assigned to stdin.

## Stdout

The output stream is referred to as the "standard output" stream (stdout). Under normal circumstances, File descriptor 1 is assigned to stdout.

## Stderr

The error stream is referred to as the "standard error" stream (stderr). Under normal circumstances, File descriptor 2 is assigned to stderr.

## Stream redirection

### Output Redirection

The output from a command normally intended for standard output can be easily diverted to a file instead. This capability is known as output redirection.

If with the notation > , file is appended to any command that normally writes its output to standard output, the output of that command will be written to file instead of your terminal.

**cat test.c > test1.c**

The output of cat is redirected to another file in the above case.

### Input Redirection

Just as the output of a command can be redirected to a file, so can the input of a command be redirected from a file. As the **greater-than character** > is used for output redirection, the **less-than character** < is used to redirect the input of a command.

**./bin < inp**

The content of the file inp is redirected to the binary as it's input.

### Redirection of output of a process to another

The | operator takes output from one program, or process, and sends it to another.

**cat test.txt | grep "CTF"**

Here we are redirecting the output of the cat command to the grep command to chain the effect of multiple commands used at once.

### Redirection of file descriptors to pseudo-files

When needed , we can silence the output of certain commands by redirecting stdout and stderr to the pseudo-files like **/dev/null**.

**docker-compose up --build > /dev/null 2>&1**

Here in this command , all of it's stdout and stderr will be redirected to the **/dev/null**.

### Here Document

A here document is used to redirect input into an interactive shell script or program.

The general form of the here document is :

**command << delimiter**

**document**

**delimiter**

Here the shell interprets the << operator as an instruction to read input until it finds a line containing the specified delimiter. The delimiter specified tells the shell that the **here document** has completed.

You can also use here document to print multiple lines in a shell script like this

**cat << EOF**

**CTFs are fun but well**

**they're not easy sometimes**

**though u get a lot to learn :)**

**EOF**

Linux commands

Commonly used commands in linux. Open up terminal using ctrl+alt+t and do the exercises as you learn each command. Read [Files](https://wiki.bi0s.in/basics/files/) before starting this since you will need it to do the exercises.

Pipe doubt

Pipe is used to redirect the output of one command/program to input of another command/program.  
The character | is used for this purpose. You can see the below example.

cat file | grep gives the contents of file to grep.

ls

Lists the contents of the current directory or the directory that we mentioned.

Usage

* **ls** - lists the contents of the current directory.
* **ls dir** - lists the contents of directory dir.
* **ls -a dir** lists the contents including the **hidden files/folders**.
  + Hidden files/folders - files that start with a .
* **ls -l file** - gives information about owner, creation date and time, permissions for file.

For more information,

**$ man ls**

Exercise

* List the contents of Desktop directory.

mkdir

Used to create a new directory.

* **mkdir dir** - will make a new directory dir.
* **mkdir -p dir1/dir2** - creates the required parent directories if it doesn't already exist.

dir1 is created. dir2 is created in dir1.

For more information,

**$ man mkdir**

cd

Used to change from one directory to another.

* **cd dir** - changes the directory to dir.
* cd or **cd ~**- moves to home directory.
  + ~ - /home/{user}/
* **cd ..** - moves to parent directory of current directory.

For more information,

**$ man cd**

touch

Used to create a empty file.

* **touch file** - creates a new empty file with filename file.

For more information,

**$ man touch**

cat

Display the contents of a file or concatenate files.

* cat file - displays contents of file.
* **cat file1 file2 > file3** (file3 = file1 + file2),, **cat file1 file2 >> file3**  (file3 = file3 + file1 + file2)

if file3 doesn’t exist, it creates a new file file3  with contents of both file1 and file2, if exists, just copy contents in file1 and file2, and paste it in file3.

* **cat file1 >> file2** - adds contents of file1 to end of file2
* **cat file1 > file2** – removes all the contents int file2, and places all contents of file1 into file2.

For more information,**$ man cat**

cp

Used to make copies of files or directories.

* **cp file1 file2** - copies contents of file1 to a new file file2.

If file2 already exists, then it similar to **cat file1 > file2**, it will erase initial elements in file2.

* **cp -r dir1 dir2** - copies the directory dir1 to a new directory dir2 with the contents of dir1.

For more information, **$ man cp**

mv

Move a file or directory into another file or directory or rename files or directories.

* **mv file1 file2** – if file2 doesn’t exist, changes the name of file from file1 to file2. If file2 exists, then it deletes file2, and then follow above procedure.
* **mv dir1 dir2** – if dir2 doesn’t exist, changes the name of directory from dir1 to dir2. If dir2 exists in computer, it moves dir1 from current directory to dir2.
* **mv file1 file2 dir1** - moves both file1 and file2 to the directory dir1 given that the directory exists.

For more information, **$ man mv**

rm & rmdir

Removes a file or a directory.

* **rm file1** - removes the file file1. **rm file1 file2** - removes the file file1 and file 2.
* **rmdir dir1**- removes the directory dir1 only if the directory is empty.
* **rm -rf dir1** - removes the directory dir1 even if it has contents.

For more information, **$ man rm**, **$ man rmdir**

echo

Gives the argument to standard output.

* Cat Qecho argument - writes argument to standard output.

**$ echo "Hello World!"**

**Hello World!**

For more information, **$ man echo**

Exercise

* Recreate myfirst and test. Put the following data into test using echo.

Name Height Age Place

Arun 172 18 Delhi

Amy 160 22 NYC

Benny 175 35 LA

Xavier 165 15 Egypt

Mani 170 23 Florida

Answer: echo ”what u want to add” > test.txt

grep

Searches file for specified string or expression.

* **grep "hello" file** - gives the lines that contain hello.
* **grep -i "hello" file** - search for the string case insensitively.
* **grep -c "hello" file** - prints the number of lines which has the string.
* **grep -v "hello" file** - displays the line that are not matched.
* **grep "^hello" file** - matches the lines that start with the string.
* **grep "hello$" file** - matched the lines that end with the string.

For more information, **$ man grep**

Exercise

* Find lines that start with A from test file. (Use |)

tr

Used for deleting or translating characters.

* **tr -d 'w' file** - removes all w from the file. doubt
* **tr -s file** - replaces multiple spaces in a string with single space.
* **tr -cd 'w' file** - complement of what other option does. So here removes everything except w.

For more information, **$ man tr**

cut

The cut command is a command for cutting out the sections from each line of files.

* **cut -b 1,2,3 file.txt** - prints the first 3 bytes of each line in the file.
* **cut -b 2-5 file.txt** - prints the 2nd to 5th bytes of each line.
* **cut -d " " -f 1 file.txt** - d spefifies the delimiter and f specifies the field. In this example, 1st word from each line is printed.

For more information, **$ man cut**

Exercise

* Print only the age column from test file. Use the output of the above exercise. It will be something like cat test | tr <options> | cut <options>.

sort

Sort command sorts the contents of a text file, line by line.

* Lines starting with a number will appear before lines starting with a letter.
* Lines starting with a letter that appears earlier in the alphabet will appear before lines starting with a letter that appears later in the alphabet.
* Lines starting with a lowercase letter will appear before lines starting with the same letter in uppercase.
* sort file.txt - sorts the contents of the file.
* sort -o out.txt in.txt - puts the output to out.txt.
* sort -r file.txt - sorts in reverse order.
* sort -n file.txt - sorts the files in numeric order.
* sort -u file.txt - removes duplicate elements.

For more information,

**$ man sort**

Exercise

* Print the sorted age from test. Use the output of above command.
* Print the sorted name from test.

man

Displays a manual page for the command.

* **man command** - displays a manual page about the command.

For more information,

**$ man man**

pwd

* **pwd** - prints the absolute path of the current directory.

For more information, **$ man pwd**

clear

* **clear** - clears the teminal screen and put the cursor back at the top of the window.

For more information, **$ man clear**

find & locate

Used for locating a file/directory.

* **locate** - is fast way to locate files and uses a recorded snapshot of the system and might miss recent additions.
* **find** - can be also used to find files but might take some time depending on the size of file system.

For more information, **$ man locate**, **$ man find**

Ssh doubt

Used for logging into a remote machine.

* ssh sample.ssh.com - connects to a remote computer called sample.ssh.com.
* ssh [[email protected]](https://wiki.bi0s.in/cdn-cgi/l/email-protection) - can be used to log in as a different user.
* ssh [[email protected]](https://wiki.bi0s.in/cdn-cgi/l/email-protection) -p port - can be used to specify the port to connect to.

For more information, **$ man ssh**

Scp doubt

**scp** copies files between hosts on a network.

* **-r** - can be used for copying a directory recursively.

Remote to local machine:

* scp [[email protected]](https://wiki.bi0s.in/cdn-cgi/l/email-protection):file\_on\_remote.txt folder\_on\_local

Local to remote machine:

* scp /path/file\_on\_local [[email protected]](https://wiki.bi0s.in/cdn-cgi/l/email-protection):folder\_on\_server

For more information,

**$ man scp**

**Note:** For ssh and scp, you might have to provide ssh key using -i key.

Env doubt

Prints the defined environment variables.

Xxd doubt

Prints out the hex dump of the given file.

Usage

**$ xxd file.txt**

**00000000: 4865 6c6c 6f20 576f 726c 6421 0a Hello World!.**

For more information, **$ man xxd**

strings

Print the sequences of printable characters which are aleast 4 characters long in file.

Usage, **$ strings file.txt**

For more information, **$ man strings**

Conditions and loops

We can write conditional statements and loops in bash.

if

* **if [ exp1 condition exp2 ]; then statements; fi.**

statement gets executed if condition satisfies for exp1 and exp2.

condition can be any of the following:  
-**eq** equal to.  
-**ne** not equal to.  
-**gt** greater than.  
-**lt** less than.  
-**a** and.  
-**o** or.

These can be used for other conditional statements also.

if else

* **if [ exp1 condition exp2 ]; then statements; else statements; fi**

if else if

* **if [ exp1 condition exp2 ]; then statements; elif [ exp1 condition exp2 ] then statements; else statements; fi**

Exercise

* Try out if, if else and if else if by printing some string according to the condition.

for

To run a command multiple times, we can use bash loops.  
For running n times using for loops.

* for i in {1..n}; do command; done

The following is an alternative for the above loop.

* for((i=1;i<=n;i+=1)); do command; done

Exercise

* Print the multiplication table of 9 using for loop in both the ways mentioned above.

while

Bash also has while loops.

* **while condition; do command; done**

Examples:

* **while true; do echo "hello"; done**
* **while [ $i -le 5 ]; do echo $i; done doubt**

Exercise

* Do the same exercise as **for** but now using **while**.