SWE-204 OPERATING SYSTEMS SSUET/QR/113

**LAB # 2**

**File Creation and Listing Commands**

**File Naming Rules**

When you create a new file, there are some rules governing the naming of the file. These rules are related to the length of the file name and the characters allowed in naming a file.

Generally a file name in LINUX can be as long as 256 characters. The rules that apply to the file names are as follows:

1. A file name can be a combination of letters, numbers and special characters.
2. All letters, both upper (A-Z) and lower case (a-z) can be used.
3. Numbers from 0 – 9 can be used.
4. Special characters like plus (+), minus (-), underscore (\_), or dot (.) can be used.
5. LINUX is case sensitive and uppercase and lowercase letters are treated separately. So myfile, Myfile, MyFile, and myFilE are different names.

**Basic Operations on Files**

The most common operations on files are creating new files, deleting unnecessary files, and displaying the contents of a file.

**Creating a File with *cat* command**

A file is a named area on the disk(s) where you can store information. The **cat** command is a basic command used for creating new files containing text.

**Example:**

**$ Cat > newfile** this is first line. <Enter>

this is the second line. <Enter> this is third and last line. <Enter>

<**CTRL+d**> or <**CTRL+Z**>

Linux Operating System 6

SWE-204 OPERATING SYSTEMS SSUET/QR/113

**Note:** Press the*Control*and*d*or z keys simultaneously to end the text entry process andsave the file.

**Displaying the contents of a file**

To see the contents of a file use the **cat** command again likes:

**$ cat newfile**

This is first line.

This is the second line. This is third and last line.

**Redirecting Standard Output**

The Redirection of stdout is controlled by **“>”** the greater-than symbol. The command takes input from the keyboard but sends its output to a file on disk. Note that the error messages still go to the terminal screen.

**$ cat newfile**

This is first line.

This is the second line. This is third and last line.

To redirect the output of the cat command we use the following step.

**$ cat newfile > newfile2**

Now the **cat** command displayed nothing, as the output of the command is redirected to a file. If we check the contents of file **newfile2**, it will contain the same text as **newfile.** This is another way of copying files.

**Joining Two or More Files**



Two or more files can be joined into a single file by the use of the ***cat*** command and redirecting output to a file. Let supose there are three files in your home directory, like ***file1 file2 and file3***. you will join these files in one file like;

Linux Operating System 7

SWE-204 OPERATING SYSTEMS SSUET/QR/113

***1. $ cat file1 file2 file3 > file4***



The following command creates a new file, containing all files in the directory.

***2. $ cat \* > file5***

**Appending to a File**

In the case of output redirection with the “>” symbol, the file to which we redirect the output of a command is overwritten. It means that the previous contents of a file are destroyed. We can use the double redirection symbol “>>” to keep the previous contents of a file. In such a case the output of a command is appended to the file.

***$ cat file1 >> file2***

This command means that ***file2*** still contains the old contents of ***file2***. In addition to this, the contents of ***file1*** are added to the end of ***file2***. If ***file2*** does not exist, it is created. This is very useful feature and is used in many situations.

**Redirecting Standard Error**

There is no special symbol for redirecting Standard Error **(stderr).** The same “>” symbol is used but with the number 2 attached in front of it. We use **“2>”** for **stderr** redirection to tell the shell that we want to redirect the error messages instead of stdout (file descriptor value is 0). Consider the following commands;

***$ ls xyz***

xyz not found

***$ ls xyz > abc*** xyz not found

We tried to redirect the output, but still the message is displayed on our screen. The redirection had no effect because we are trying to redirect stdout while the command is generating stderr messages. Now try it;

***$ ls xyz 2> abc***

Now there is nothing displayed because the error message has been stored in a file with name **abc**. You can use the **cat** command to verify it.

***Standard File Descriptors***

|  |  |  |
| --- | --- | --- |
|  | ***File Descriptor Number*** | ***Description*** |
|  |  |  |
|  | *0* | *Standard Input* |
|  | *1* | *Standard Output* |
|  | *2* | *Standard Error* |

Linux Operating System 8

SWE-204 OPERATING SYSTEMS SSUET/QR/113

**Example:**



***$ ls –i***

137717 history

137656 psg

The **inode** number contains: the location of files and directories on the disk; access permissions; owner and group ids; file link count; time of last modification; time of last access; device identification number for special files; and a variety of other information. **inode** numbers are used extensively by the system as you change directories and performvarious work.

 ***ls –p, ls -F***

Since you may have subdirectories within the directory you are listing, you may want to use the ***-p*** option to ***ls,*** which puts a “/” in after directory names.



***ls -R***

Since the subdirectories you are listing probably them, you may want to recursively list these. The example will perform this recursive listing.

have files and subdirectories beneath **-R** option to **ls** shown in the following

Linux Operating System 9

SWE-204 OPERATING SYSTEMS SSUET/QR/113

**Summary**

ls – List the contents of a directory



**Options**

**-a** List all entries.

**-b** Print nongraphic characters.

**-c** Use time file was last modified for producing order in which

files are listed.

**-d** List only the directory name, not its contents.

**-f** Assume each argument is a directory.

**-g** Only the group is printed and not the owner.

**-i** Print the inode number in the first column of the report.

**-m** List the contents across the screen separated by commas.

**-o** List the information in long form (-l) except that group is

omitted.

**-p** Put a slash (/) at the end of directory names.

**-s** Show the size in blocks instead of bytes.

**-t** List in order of time saved with most recent first.

**-A** Same as–a except current and parent directories aren’t listed.

**-C** Multicolumn output produced.

**-L** List file or directory to which link points.

**-R** Recursively list subdirectories.

There are also some shorthand command names for issuing **ls** with options. For instance, **ll** is equivalent to **ls -l** and **lsr** is equivalent to **ls -R**.

Linux Operating System 10

SWE-204 OPERATING SYSTEMS SSUET/QR/113

**File Name Expansion and wild cards**

An overview of file name expansion is useful to insure you’re comfortable with this topic.

1. (\*) To list all files in a directory that end in **“.c”**, you could do the following:

**$ ls \*.c**

**conf.c SAM.c newfile.c**

2. (?) To find all of the files in a directory named “conf” with an extension of one character, you could do the following:

**$ ls conf.?**

**conf.c conf.o conf.1**

3. [list] To list all of the files in a directory named “conf” with only t he extension “c” or “o”, you could do the following:

**$ ls conf.{co} conf.c conf.o**

4. str{str1,str2,str3,….} To list files that start with “ux”, and have the extension “300” or “700”, you could do the following:

**$ ls ux\*.{700,300}**

**uxbootlf.700 uxinstfs.300 unistker.300 unistker.700**

5. (~) To list the files in your home directory you could use **~** :

|  |  |  |  |
| --- | --- | --- | --- |
| **$ ls -a ~** |  |  |  |
| **.** | **.cshrc.org** | **.login** | **.shrc.org** |
| **..** | **.exrc** | **.abcd** | **.defg.c** |
| **.chrcs** | **.history** | **.profile** |  |



If you want to see the contents of the directory **/abc** without opening it do the following command:

**$ ls /abc**

images pictures games …..

Linux Operating System 11

SWE-204 OPERATING SYSTEMS SSUET/QR/113

 *ls -l*

To list all information about the contents of directory you would use the **–l** option to **ls** as shown in the following example.

-rwxr-x--x 1 denie users 99 dec 6 09:18 story

Issuing this command has produced a lot of information relating to file called **story.** Now understand the first set of characters (-rwxr-x--x). This set of characters is made up of 4 distinct fields, as describe here:

|  |  |
| --- | --- |
| 1st Character | File Type |
| ------------------ | ------------------------------------------------------- |
| **-** | An ordinary file such as a text file. |
| **b** | A block special file which is a device file. |
| **c** | A character special file which is device file. |
| **d** | A directory. |
| **l** | A link. |
| **n** | A network special file. |
| **p** | A named pipe special file, also called a FIFO. |
| **S** | A socket |

**“Who” and “whoami”**

If you see an unattended terminal to which a user is logged in? You can use the ***whoami*** command to identify who is logged in. When you use this command, you areactually asking the system to tell “you” about “yourslef.”

***$ whoami*** username

The **who** command, tells you the login names of all users logged into the system, along with their time and the terminal line they are using .

***$ who***

*operator* *pts/ta Jan 23 14:56*

**Some Magical Differences**

The following command creates a file, containing all list of files and directories.

First list your current directory with ***ls*** command. Then enter this;

***1. Ls > filename***

**Note:** The (>) means put the output from the command into the following file, rather thandisplay it on the terminal screen.(‘>’ is the redirection symbol).



Consider the ***who*** command. We redirected its output to a file with the name ***whofile.***

1. ***who > whofile***

**Sorting**

Let us suppose we have a file with name *unsorted* with thw following lines in it;

***$ cat unsorted***

Linux Operating System 13

CE-303 OPERATING SYSTEMS SSUET/QR/113

This is number 1

This is number 3

This is number 5

This is number 4

This is number 2

We can use the ***sort*** command to arrange(sort) these lines. When we use the sort command with input redirection to this file, this result appears.

***$ Sort < unsorted***

**Searching**

We can find a particular word with ***grep*** command like;

**$ grep abc <*filename*>**

For multiple words use it like;

**$ grep “sir syed” <*filename*>**

**Note:** For case-insensitive use ***–i*** with grep command.

**Copying and moving files**

**Copying Files**

The files are copied with ***cp*** command. The source and destination file names are supplied to the ***cp*** command as arguments. The first argument is the source file name, and second argument is the destination file name. For example;



**$ cp file1 file2**

This command copies ***file1*** from the current directory to ***file2*** in the current directory.

**Moving and Renaming Files**

The ***mv*** command is used for renaming files and moving files from one place to another in the directory structure.

 For Rename a File;

**$ mv myfile newfile**

Linux Operating System 14

CE-303 OPERATING SYSTEMS SSUET/QR/113

 For Moving a File;

**$ mv file1 ./lab**

**Note:** You must be careful with the ***mv*** command, as it will overwrite any existing file ifthe destination file name matches any source file. And it will do it without any warning.

**Miscellaneous File Handling Commands**

**The Head and Tail**

Sometimes you need to view only the first or last lines of a text file. By default, the ***head*** command lists the first ten lines of a text file, and the ***tail*** command lists the last ten lines of a file.

**$ head -2 <*filename*>**

**$ tail -5 <*filename*>**

**Counting Characters, Words, and Lines in a Text File**

Many times you want to know how many characters, words, or lines there are in a file. We use ***wc***(*word count* ) command for this purpose.

**$ wc <*filename*>**

It displays the number of lines, words, and characters, respectively.

Use ***wc*** ***–*** ***c*** for characters, ***wc*** ***–l*** for lines, and ***wc*** ***–*** ***w*** for words.

**Command Aliases**

An alias is a different name for a Linux command. Aliases can also be used to remember commands by assigning them more meaningful names. They can be set using ***alias*** command.

***$ alias dir=ls***

**Disk Usage by Directory**

The ***du*** command shows a breakdown of disk usage by directory. For example, ***cd*** to your home directory and type "***du -k***" (The *-k* option displays output in Kilobytes)

Linux Operating System 15

CE-303 OPERATING SYSTEMS SSUET/QR/113

**$ cd ~ $ du –k**

|  |  |
| --- | --- |
| 1 | ./News |
| 3 | ./mail |
| 713 | ./public\_html/photo |

1363 .

The output lists the number of Kilobytes used by each subdirectory. In this example, the user is using a total of 1363 Kilobytes or about 1.5 Megabytes of disk space.

The ***-s*** option causes ***du*** to work in summary mode. In summary mode, ***du*** only outputs the total amount of disk space used by a directory structure. For example

**$ du -k -s**

1363.

**Lab task**

1. **date** (for Current Date).
2. **ps**
3. **fgrep**
4. **finger**
5. **cal** (For Calender).
6. **free**
7. Create a file name ***<yearfile>*** which displays the contents of current year.
8. Display calendar and date Simultaniously.
9. df ./
10. df –k ./
11. du –s \*
12. zcmp
13. zdiff
14. Create multiple files and perform all commands.
15. Remove non empty directory.

**Home Assignment**

1. Write in detail about “***w”*** command and also the difference between ***who*** and ***w*** command.
2. Write the difference of the following: >, <, >>.

**Note:** Write in your hand writing.

**Lab Task:**

Linux Operating System 12