



Linux

Linux – Terminal commands for searching and regular expressions

Dan Richter

08 Sep 2020

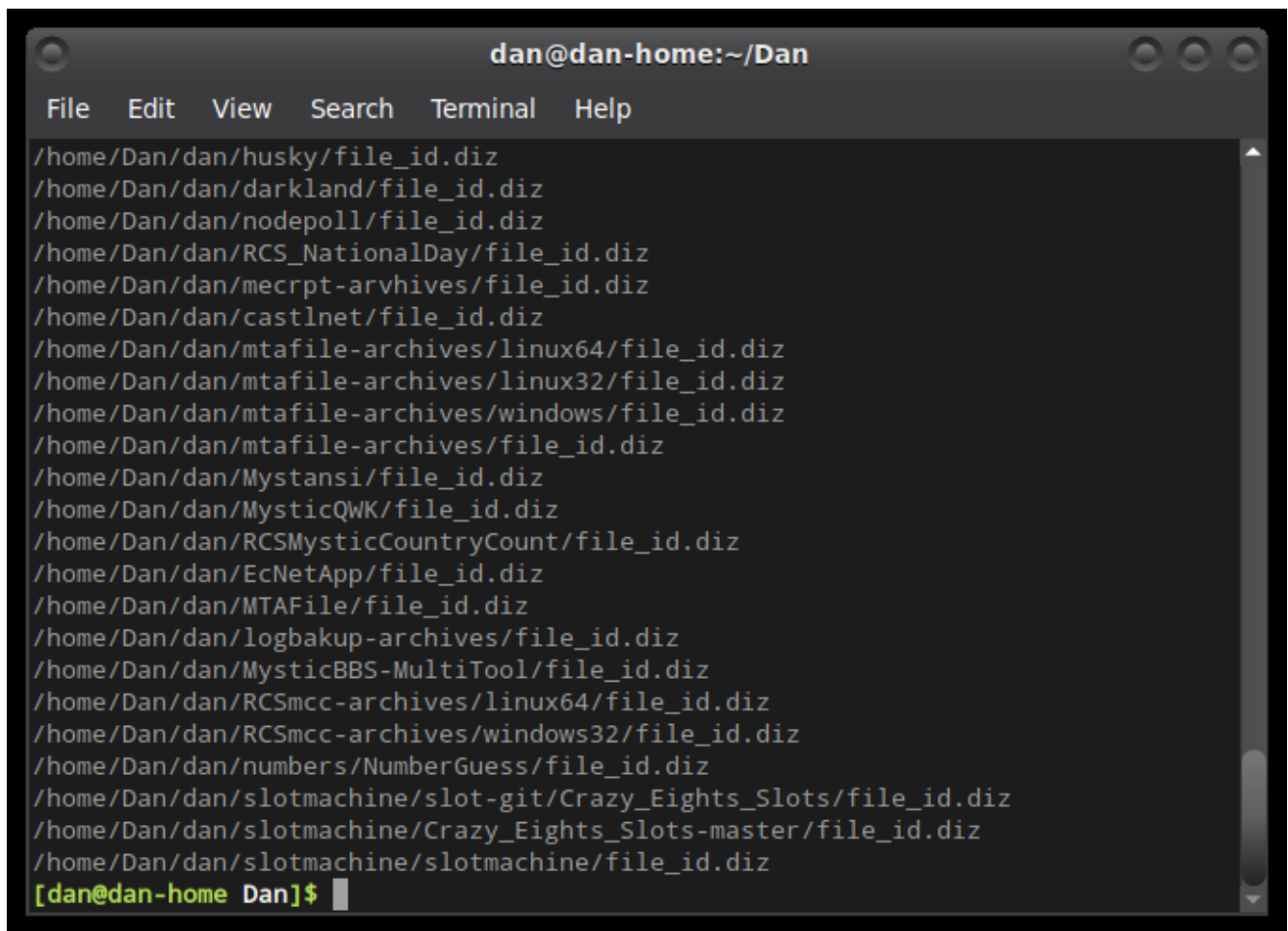
You should have a pretty good basic understanding on how things work in the Linux terminal. As I've stated before, there are many more commands and switches available that are beyond the scope of these writings. Perhaps in future writings we may get more into the advanced commands.

Let's take a look at some commands that will help you find files, directories or text.

find

The find command is a very useful command. It has options for finding files by name, date, size, permissions, ownership, etc.

So, you want to find all the files in your home directory named 'file_id.diz'. With this command, you can do that by typing in 'find /home -name file_id.diz'.

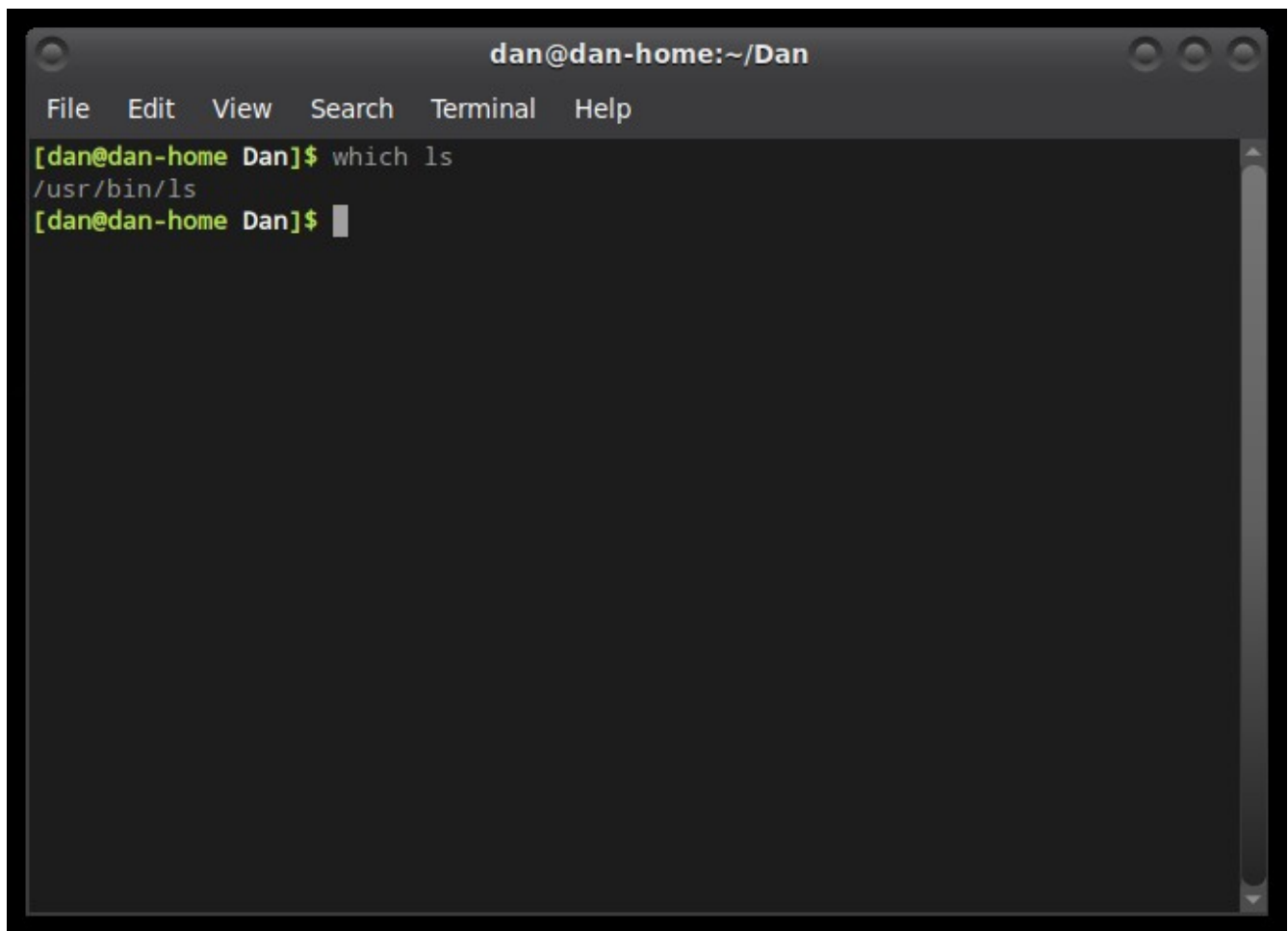
A terminal window titled 'dan@dan-home:~/Dan' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal displays a list of file paths, each ending in 'file_id.diz'. The paths are: /home/Dan/dan/husky/, /home/Dan/dan/darkland/, /home/Dan/dan/nodepoll/, /home/Dan/dan/RCS_NationalDay/, /home/Dan/dan/mecrpt-arvhives/, /home/Dan/dan/castlnet/, /home/Dan/dan/mtafile-archives/linux64/, /home/Dan/dan/mtafile-archives/linux32/, /home/Dan/dan/mtafile-archives/windows/, /home/Dan/dan/mtafile-archives/, /home/Dan/dan/Mystansi/, /home/Dan/dan/MysticQWK/, /home/Dan/dan/RCSMysticCountryCount/, /home/Dan/dan/EcNetApp/, /home/Dan/dan/MTAFile/, /home/Dan/dan/logbakup-archives/, /home/Dan/dan/MysticBBS-MultiTool/, /home/Dan/dan/RCSmcc-archives/linux64/, /home/Dan/dan/RCSmcc-archives/windows32/, /home/Dan/dan/numbers/NumberGuess/, /home/Dan/dan/slotmachine/slot-git/Crazy_Eights_Slots/, /home/Dan/dan/slotmachine/Crazy_Eights_Slots-master/, and /home/Dan/dan/slotmachine/slotmachine/. The prompt is '[dan@dan-home Dan]\$' with a cursor.

```
dan@dan-home:~/Dan
File Edit View Search Terminal Help
/home/Dan/dan/husky/file_id.diz
/home/Dan/dan/darkland/file_id.diz
/home/Dan/dan/nodepoll/file_id.diz
/home/Dan/dan/RCS_NationalDay/file_id.diz
/home/Dan/dan/mecrpt-arvhives/file_id.diz
/home/Dan/dan/castlnet/file_id.diz
/home/Dan/dan/mtafile-archives/linux64/file_id.diz
/home/Dan/dan/mtafile-archives/linux32/file_id.diz
/home/Dan/dan/mtafile-archives/windows/file_id.diz
/home/Dan/dan/mtafile-archives/file_id.diz
/home/Dan/dan/Mystansi/file_id.diz
/home/Dan/dan/MysticQWK/file_id.diz
/home/Dan/dan/RCSMysticCountryCount/file_id.diz
/home/Dan/dan/EcNetApp/file_id.diz
/home/Dan/dan/MTAFile/file_id.diz
/home/Dan/dan/logbakup-archives/file_id.diz
/home/Dan/dan/MysticBBS-MultiTool/file_id.diz
/home/Dan/dan/RCSmcc-archives/linux64/file_id.diz
/home/Dan/dan/RCSmcc-archives/windows32/file_id.diz
/home/Dan/dan/numbers/NumberGuess/file_id.diz
/home/Dan/dan/slotmachine/slot-git/Crazy_Eights_Slots/file_id.diz
/home/Dan/dan/slotmachine/Crazy_Eights_Slots-master/file_id.diz
/home/Dan/dan/slotmachine/slotmachine/file_id.diz
[dan@dan-home Dan]$
```

Evidently, I have quite a few variations of that file in my home directory. :)

which

This command, while similar it find, will only look for executable files. Let's say I want to find the executable file called 'ls', as I use it all the time. I could type in 'which ls' and it will tell me where it is.

A terminal window titled 'dan@dan-home:~/Dan' with a menu bar containing 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command '[dan@dan-home Dan]\$ which ls' and its output '/usr/bin/ls'. A second prompt '[dan@dan-home Dan]\$' is visible with a cursor.

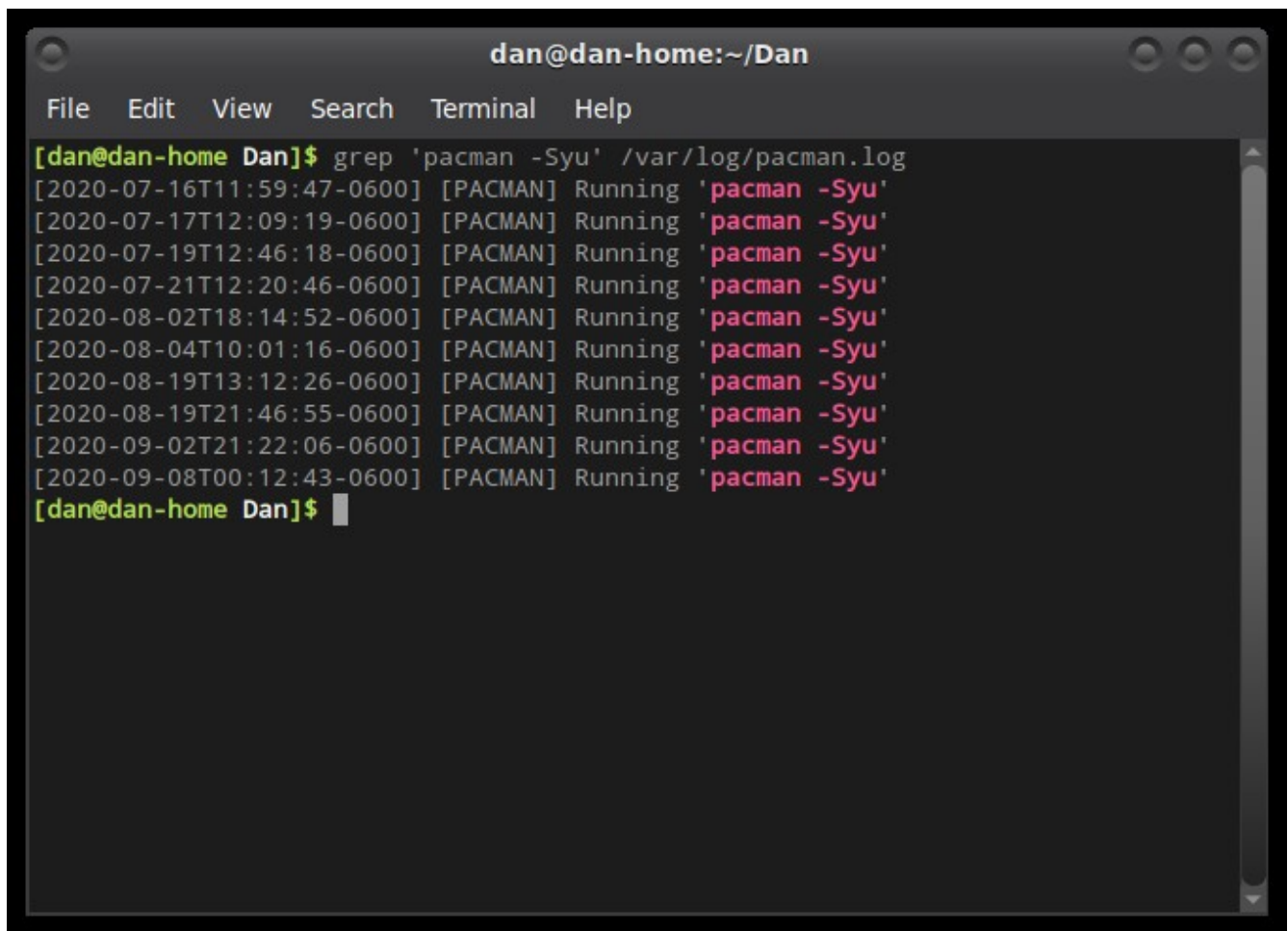
```
dan@dan-home:~/Dan
File Edit View Search Terminal Help
[dan@dan-home Dan]$ which ls
/usr/bin/ls
[dan@dan-home Dan]$
```

grep

This has to be one of the most used commands on my computer. I think this gets used almost as much as 'ls'.

Grep has a couple different ways it can be used. Let's say your looking through a large text file, such as a log file, trying to find an IP address. Instead of scrolling through thousands of lines in hopes that you don't miss the IP address you're looking for, you could use grep.

Let's take a look and see how many times recently I've run the 'pacman -Syu' command. There is a 'pacman.log' file located in the /var/log directory that I will be searching with 'grep 'pacman -Syu' /var/log/pacman.log'

A terminal window titled 'dan@dan-home:~/Dan' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows a command to search for 'pacman -Syu' in the file /var/log/pacman.log. The output consists of ten lines, each showing a timestamp, the word '[PACMAN]', the word 'Running', and the command 'pacman -Syu' in red text. The prompt is '[dan@dan-home Dan]\$' with a cursor.

```
dan@dan-home:~/Dan
File Edit View Search Terminal Help
[dan@dan-home Dan]$ grep 'pacman -Syu' /var/log/pacman.log
[2020-07-16T11:59:47-0600] [PACMAN] Running 'pacman -Syu'
[2020-07-17T12:09:19-0600] [PACMAN] Running 'pacman -Syu'
[2020-07-19T12:46:18-0600] [PACMAN] Running 'pacman -Syu'
[2020-07-21T12:20:46-0600] [PACMAN] Running 'pacman -Syu'
[2020-08-02T18:14:52-0600] [PACMAN] Running 'pacman -Syu'
[2020-08-04T10:01:16-0600] [PACMAN] Running 'pacman -Syu'
[2020-08-19T13:12:26-0600] [PACMAN] Running 'pacman -Syu'
[2020-08-19T21:46:55-0600] [PACMAN] Running 'pacman -Syu'
[2020-09-02T21:22:06-0600] [PACMAN] Running 'pacman -Syu'
[2020-09-08T00:12:43-0600] [PACMAN] Running 'pacman -Syu'
[dan@dan-home Dan]$
```

Probably not as often as I should... So, 'grep' will display any line in the over 2000 line log file, that contains the search parameter of 'pacman -Syu'.

Another way that grep can be used, is to find text within a directory of files. For example, if I search through all of my tagline files for the keyword of 'Wisconsin', I get a pretty good list. I'm using the command of 'grep Wisconsin *'.

```
dan@dan-home:~/Dan/temp3
File Edit View Search Terminal Help
Taglines Galore j.tags.txt:Joel Robinson a Wisconsin farmer?
Taglines Galore j.tags.txt:Joel Robinson a Wisconsin farmer? E...I...E...I DON'T
THINK SO!
Taglines Galore m.tags.txt:Member of FidoNet Tagline Thief Wisconsin Guild chapt
er. :-}
Taglines Galore m.tags.txt:My Canada includes Wisconsin from November until Apri
l
Taglines Galore n.tags.txt:Northwestern Wisconsin
Taglines Galore quotes.tags.txt:"In Wisconsin, that barn would say DIESEL/CHEESE
/GIFTS." -- Servo
Taglines Galore quotes.tags.txt:"Joel Robinson a Wisconsin farmer? E...I...E...I
DON'T THINK SO!"
Taglines Galore quotes.tags.txt:"Something tells me we're not in Northern Wiscon
sin anymore" - Yakko
Taglines Galore s.tags.txt:Sheboygan Wisconsin, U.S.A., Earth, Sometimes
Taglines Galore s.tags.txt:Something tells me we're not in Northern Wisconsin an
ymore - Yakko
Taglines Galore s.tags.txt:State of Wisconsin Motto. Eat cheese or die!
Taglines Galore t.tags.txt:The Story of Wisconsin Sausage - Tom on title 'Bloodl
ust'
Taglines Galore t.tags.txt:Transplanted from Wisconsin.
Taglines Galore v.tags.txt:Visit Wisconsin!...You'll LOVE our "dairy air"!
Taglines Galore v.tags.txt:Visit beautiful Wisconsin Dells.
[dan@dan-home temp3]$
```

This command will let you know which files it found the keyword in, and what the lines contained. Very useful when trying to find specific information, but don't know which file it is in.

sed

The sed command is also very useful if you get into writing any bash file scripts. I've written some that will do web scraping, and sed is my go-to utility.

If you have a text file that you would like to change or remove specific words, sed is a great tool. In the example below, there is a text file that contains the text 'This is a text file'. Instead of opening an editor to change one of the words, I could use sed (Stream Editor), using the following command:

'sed -i 's/a/the/' text.txt'. What this is doing, is I told sed to substitute (s) the word 'a' with the word 'the'. The -i switch tells sed to make the changes to the file.

```
dan@dan-home:~/Dan/temp9
File Edit View Search Terminal Help
[dan@dan-home temp9]$ cat text.txt
This is a text file
[dan@dan-home temp9]$ sed -i 's/a/the/' text.txt
[dan@dan-home temp9]$ cat text.txt
This is the text file
[dan@dan-home temp9]$
```

Again, this is the very basics of what this tool can accomplish, but as you can see, this can be very useful in either doing a quick change, or working with a large document.

Just to give you an example of how this tool can be used, utilizing the '|' pipe between commands, this is from one of my web scraping scripts. Remember, the output of a command can be piped to another command.

```
cat $FILE | tail -n +121 | sed -n -e '/Products You May Like/, $!p' | sed -e 's/\([^\]]*\)/\]/g' | sed 's/by  
admin 0 Comments//g' | sed '/Share on/d' | sed "s/^\^/g" | sed "s/^\^/g" | sed '/Photo by/d' | sed  
'/Next\:/d' | sed '/Images[)]/d' | sed 's/Read the//g' | sed 's/original article here\./g' | sed 's/^\^ //g' | sed  
'/^$/N;\n$/D' > post$count.txt
```

While this line is working on an HTML file, it uses sed to find specific matches, and either remove them, or replace them with blank space. The final output is a “cleaned” text file. (if everything works according to plan)