Lindsay Graham

Linux Administration

Week 2 Lab – Grep

Grep Tutorial

Grep is a powerful pattern matching tool that can be very useful for advanced searching. It can match both plain text and regular expressions (regex). This overview will demonstrate several use cases for the grep command, providing examples of the command syntax and its corresponding output. It will also provide a short explanation of how the command works and why. For these examples, the file used was named GrepLab. For clarity, the file will be referred to as 'filename' throughout this tutorial.

Example 1. Print all lines containing the string Lane

- **grep -i “Lane” filename**

Figure 1.1 Output of all lines that contain the word “Lane” in the file

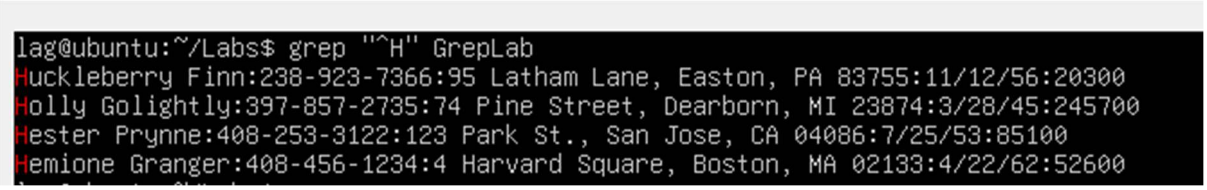
The syntax for this command is as follows: grep -i "text to match" filename. The -i option when used with grep will ignore case sensitivity. The prompt showed the word “Lane” with a capital “L” but did not specify that capitalization was important. Below is an example of the same command with a lower case “lane” and the -i option to illustrate how case sensitivity is ignored. The results are the same as above.

Figure 1.2 Output of all lines that contain the word “Lane” in the file

Example 2. Print all lines where the person’s first name starts with H

- **grep “^H” filename**

Figure 2.1 Output of all lines where the person’s first name starts with “H”

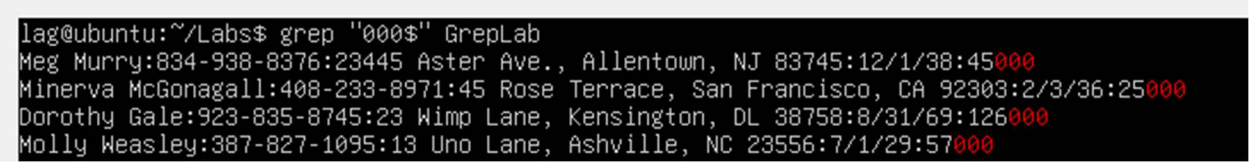


The syntax for this command is as follows: grep “^text to match” filename. The ^ symbol before the text specifies that grep should match at the beginning of a line. So here, the command says look for “H” at the beginning of the line. While not always possible, it is easier to use grep if you have an idea of how the file contents are laid out. Had first names not been the start of each line, a different command may be needed to find the first names that start with “H.”

Example 3. Print all lines ending in three zeros (000)

- **grep “000$” filename**

Figure 3.1 Output of all lines ending in three zeros “000”



The syntax for this command is as follows: grep “text to match$” filename. The $ symbol after the text (i.e., 000$) specifies that grep should match the end of a line. So here, the command says look for “000” at the end of the line.

Example 4. Print all lines that don’t contain 408

- **grep -v “408” filename**

Figure 4.1 Output of all lines that don’t contain “408”

```
Huckleberry Finn:238-923-7366:95 Latham Lane, Easton, PA 83755:11/12/56:20300
Betty Boop:245-836-8357:635 Cutesy Lane, Hollywood, CA 91464:6/23/23:14500
Dorian Gray:385-375-8395:3567 Populus Place, Caldwell, NJ 23875:6/18/68:23400
Holly Golightly:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700
Ebenezer Scrooge:548-834-2348:583 Laurel Ave., Kingsville, TX 83745:10/1/35:58900

Westley Pirate:284-758-2857:23 Edgecliff Place, Lincoln , NB 92743:7/25/53:85100
Westley Pirate:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11/3/35:58200
Westley Pirate:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11/3/35:58200
Lizzie Bennett:674-843-1385:20 Parak Lane, Duluth, MN 23850:4/12/23:780900
Nancy Drew:674-843-1385:20 Parak Lane, Duluth, MN 23850:4/12/23:780900
Jo March:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/65:35200
Victor Frankenstein:835-365-1284:454 Easy Street, Decatur, IL 75732:2/28/53:123500
Ephram Hardy:293-259-5395:235 CarltonLane, Joliet, IL 73858:8/12/20:56700
Meg Murry:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Lucy Pevensie:385-573-8326:832 Ponce Drive, Gzary, IN 83756:12/1/46:268500

Mary Poppins:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Sir Lancelot:837-835-8257:474 Camelot Boulevard, Bath, WY 28356:5/13/69:24500

Zippy Pinhead:834-823-8319:2356 Bizarro Ave., Farmount, IL 84357:1/1/67:89500
Dorothy Gale:923-835-8745:23 Wimp Lane, Kensington, DL 38758:8/31/69:126000
Popeye Sailor:156-454-3322:945 Bluto Street, NotHere, USA 29358:3/19/35:22350
Luna Lovegood:385-898-8357:38 Fife Way, Abilene, TX 39673:1/5/58:95600

Molly Weasley:387-827-1095:13 Uno Lane, Ashville, NC 23556:7/1/29:57000
Arya Stark:438-910-7449:8235 Maple Street, Wilmington, VM 29085:9/23/63:68900
lag@ubuntu:~/Labs$
```

The syntax for this command is as follows: grep -v “text to match” filename. The -v option specifies that grep should return all lines that don’t match the pattern, which in this case is “408.” Since no lines in the results contain “408”, there's nothing to highlight in red.

Example 5. Print all lines where birthdays are in the year 1935

- **grep “[0-9]*/*[0-9]*/35” filename**

Figure 5.1 Output of all lines where birthdays are in the year 1935

```
lag@ubuntu:~/Labs$ grep "[0-9]*/*[0-9]*/35" GrepLab
Ebenezer Scrooge:548-834-2348:583 Laurel Ave., Kingsville, TX 83745:10/1/35:58900
Westley Pirate:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11/3/35:58200
Westley Pirate:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11/3/35:58200
Popeye Sailor:156-454-3322:945 Bluto Street, NotHere, USA 29358:3/19/35:22350
lag@ubuntu:~/Labs$
```

\b

The command syntax, with regex in quotes between grep and filename, is as follows:

- grep
- [0-9]* : Zero or more digits (0-9)
- /: Matches the literal forward slash that is in the birthday format in the file.
- [0-9]* : Zero or more digits (0-9) again

- /: Another literal forward slash
- 35: matches the number 35
- filename

Grep matches the pattern above to any identical patterns in the file. If the birthdates were formatted differently (i.e., yyyy-mm-dd), then a different regex would be needed.

Example 6. Print all lines where the phone number is in an area code that starts with an 8

- **grep "8[0-9]\{2\}\-[0-9]\{3\}\-[0-9]\{4\}" filename**

Figure 6.1 Output of all lines where the phone number is in an area code that starts with an 8

```
lag@ubuntu:~/Labs$ grep "8[0-9]\{2\}\-[0-9]\{3\}\-[0-9]\{4\}" GrepLab
Victor Frankenstein:835-365-1284:454 Easy Street, Decatur, IL 75732:2/28/53:123500
Meg Murry:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Mary Poppins:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Sir Lancelot:837-835-8257:474 Camelot Boulevard, Bath, WY 28356:5/13/69:24500
Zippy Pinhead:834-823-8319:2356 Bizarro Ave., Farmount, IL 84357:1/1/67:89500
lag@ubuntu:~/Labs$ _
```

The command syntax, with regex in quotes between grep and filename, is as follows:

- grep
- 8: Matches a pattern with literal 8 as the first digit
- [0-9]\{2\}\ : Matches two or more digits 0-9
- -: Matches the literal dash that is the phone number format in the file
- [0-9]\{3\}*\ : Matches three or more digits 0-9
- -: Another literal dash
- [0-9]\{4\}*\ : Matches four or more digits 0-9
- filename

Grep matches that regex pattern to patterns in the file. The numbers in square brackets (i.e., [0-9]) specify a digit from 0-9 ,and the numbers in curly brackets (i.e., {2\}) specify how many of the preceding digit to match. The backslashes escape the special characters like the curly brackets. Like with birthdates, if the phone numbers were formatted differently (i.e., (888)888-8888), a different regex would be needed.

Example 7. Print all lines containing an uppercase letter, followed by four lower case letters, a space and one upper case letter

- **grep -E '[A-Z]+([a-z]{4}) [A-Z]' filename**

Figure 7.1 Output of all lines containing an uppercase letter, followed by four lower case letters, a space and one upper case letter

```
lag@ubuntu:~/Labs$ grep -E '[A-Z]+([a-z]{4}) [A-Z]' GrepLab
Betty Boop:245-836-8357:635 Cutesy Lane, Hollywood, CA 91464:6/23/23:14500
Holly Golightly:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700
Lizzie Bennett:674-843-1385:20 Parak Lane, Duluth, MN 23850:4/12/23:780900
Nancy Drew:674-843-1385:20 Parak Lane, Duluth, MN 23850:4/12/23:780900
Jo March:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/65:35200
Meg Murry:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Lucy Pevensie:385-573-8326:832 Ponce Drive, Gzary, IN 83756:12/1/46:268500
Zippy Pinhead:834-823-8319:2356 Bizarro Ave., Farmount, IL 84357:1/1/67:89500
Popeye Sailor:156-454-3322:945 Bluto Street, NotHere, USA 29358:3/19/35:22350
Daenerys Targaryen:408-724-0140:1222 Oxbow Court, Sunnyvale, CA 94087:5/19/66:34200
Molly Weasley:387-827-1095:13 Uno Lane, Ashville, NC 23556:7/1/29:57000
Arya Stark:438-910-7449:8235 Maple Street, Wilmington, VM 29085:9/23/63:68900
lag@ubuntu:~/Labs$ _
```

The command syntax, with regex in single quotes between grep and filename, is as follows:

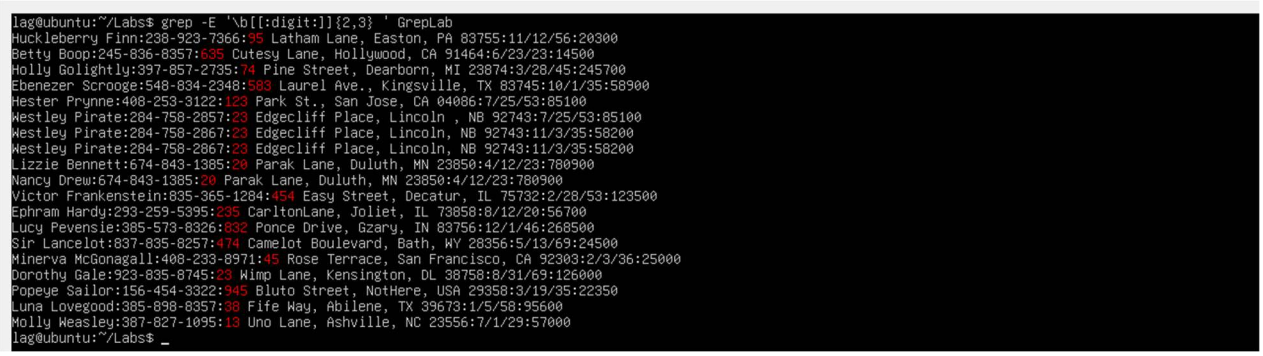
- grep
- -E: Extended regex option
- [A-Z] : Matches a capital letter A-Z
- +: Matches one or more
- ([a-z]{4}): Matches four lower case letters a-z
- Literal space
- [A-Z] : Matches a capital letter A-Z
- filename

The -E option represents extended regex. Extended regex are considered more powerful and flexible than basic regex. With basic regex, certain characters (?, +, {, |,(,)) need to be escaped to use them literally. With extended regex, they are literal by default. When using extended regex, you should use single quotes, or it could lead to unexpected results.

Example 8. Print all lines where the address begins with a two or three digit number

- **grep -E '\b[[:digit:]]{2,3}' filename**

Figure 8.1 Output of all lines where the address begins with a two or three digit number



The command syntax, with regex in single quotes between grep and filename, is as follows:

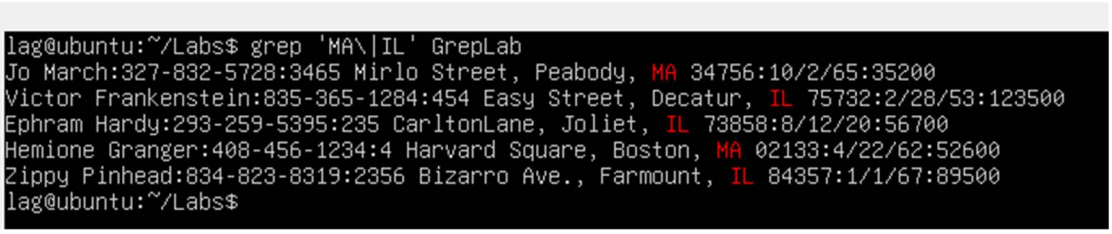
- grep
- \b: Word boundary association that matches the beginning of a word
- [[:digit:]]{2,3} : Matches two to three digits 0-9
- Literal space
- filename

The [[:digit:]] character class matches any digit 0-9. The \b is considered a word boundary and can mark the position where a word starts or ends. This command searches for a word starting with two to three digits.

Example 9. Print all lines where the person lives in Mass or Illinois

- **grep 'MA\|IL' filename**

Figure 9.1 Output of all lines where the person lives in Mass or Illinois

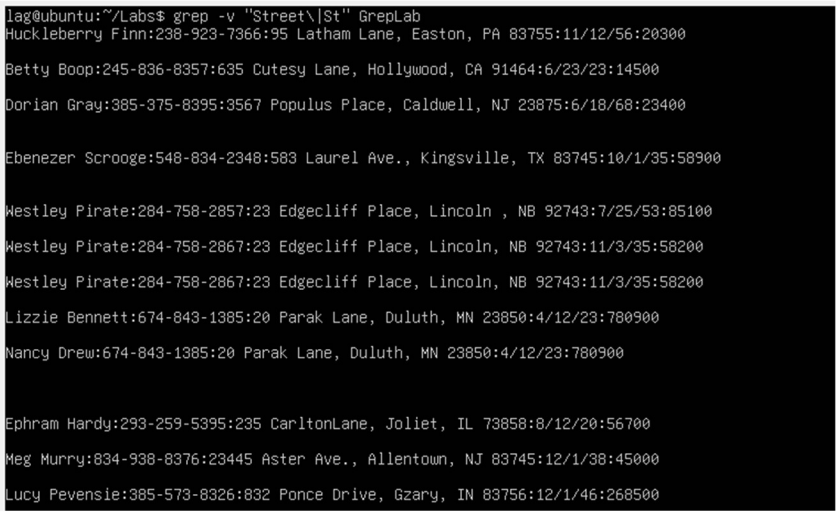


The syntax for this command is as follows: grep 'text to match \| text to match' filename. The pipe symbol represents 'OR' and can separate possible matches. The backslash is needed to escape the special pipe character. This command is essentially saying, find 'MA' OR 'IL.' The pipe symbol can work with both literal strings and regular expressions.

Example 10. Print all lines containing the addresses that aren't on a street (i.e., Street, St.)

- **grep -v "Street\|St" filename**

Figure 10.1 Output of lines containing the addresses that aren't on a street



The syntax for this command is as follows: grep -v “text to not match\|text to not match” filename. As mentioned earlier, the -v option returns results that do not match the specified pattern, and the pipe can serve as an OR to search multiple patterns. This command is essentially saying, return lines that do not contain ‘Street’ OR ‘street.’

References

https://www.aholdengouveia.name/LinuxAdmin/Grep.html

https://quickref.me/grep.html

https://caspar.bgsu.edu/~courses/Stats/Labs/Handouts/grepsearch.htm

https://storm.cis.fordham.edu/~zhang/cs3130/assignment/lab2_sol.html

https://linuxize.com/post/regular-expressions-in-grep/

https://pressbooks.senecapolytechnic.ca/uli101/chapter/extended-regular-expressions.