Chapter 2d. filters

Commands that are created to be used with a **pipe** are often called **filters**. These **filters** are very small programs that do one specific thing very efficiently. They can be used as **building blocks**.

This chapter will introduce you to the most common **filters**. The combination of simple commands and filters in a long **pipe** allows you to design elegant solutions.

2d.1. cat

When between two **pipes**, the **cat** command does nothing (except putting **stdin** on **stdout**).

```
[clim@sop pipes]$ tac count.txt | cat | cat | cat | cat | cat
five

four
three
two
one
```

2d.2. tee

Writing long **pipes** in Unix is fun, but sometimes you may want intermediate results. This is were **tee** comes in handy. The **tee** filter puts **stdin** on **stdout** and also into a file. So **tee** is almost the same as **cat**, except that it has two identical outputs.

```
[clim@sop pipes]$ tac count.txt | tee temp.txt | tac
one

two
three
four
five

[clim@sop pipes]$ cat temp.txt
five

four
three
two
```

2d.3. grep

The **grep** filter is famous among Unix users. The most common use of **grep** is to filter lines of text containing (or not containing) a certain string.

```
[clim@sop pipes]$ cat tennis.txt
Amelie Mauresmo, Fra

Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA

[clim@sop pipes]$ cat tennis.txt | grep Williams
Serena Williams, usa
```

You can write this without the cat.

```
[clim@sop pipes]$ grep Williams tennis.txt
Serena Williams, usa

Venus Williams, USA
```

One of the most useful options of grep is **grep -i** which filters in a case insensitive way.

```
[clim@sop pipes]$ grep Bel tennis.txt
Justine Henin, Bel

[clim@sop pipes]$ grep -i Bel tennis.txt
```

```
Kim Clijsters, BEL
Justine Henin, Bel
[clim@sop pipes]$
```

Another very useful option is **grep -v** which outputs lines not matching the string.

```
[clim@sop pipes]$ grep -v Fra tennis.txt
Kim Clijsters, BEL

Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[clim@sop pipes]$
```

And of course, both options can be combined to filter all lines not containing a case insensitive string.

```
[clim@sop pipes]$ grep -vi usa tennis.txt
Amelie Mauresmo, Fra

Kim Clijsters, BEL
Justine Henin, Bel
[clim@sop pipes]$
```

With **grep -A1** one line **after** the result is also displayed.

```
clim@debian5:~/pipes$ grep -A1 Henin tennis.txt
Justine Henin, Bel

Serena Williams, usa
```

With **grep -B1** one line **before** the result is also displayed.

```
clim@debian5:~/pipes$ grep -B1 Henin tennis.txt
Kim Clijsters, BEL

Justine Henin, Bel
```

With **grep -C1** (context) one line **before** and one **after** are also displayed. All three options (A,B, and C) can display any number of lines (using e.g. A2, B4 or C20).

```
clim@debian5:~/pipes$ grep -C1 Henin tennis.txt
Kim Clijsters, BEL

Justine Henin, Bel
Serena Williams, usa
```

2d.4. cut

The **cut** filter can select columns from files, depending on a delimiter or a count of bytes. The screenshot below uses **cut** to filter for the username and userid in the **/etc/passwd** file. It uses the colon as a delimiter, and selects fields 1 and 3.

```
[[clim@sop pipes]$ cut -d: -f1,3 /etc/passwd | tail -4
Figo:510

Pfaff:511
Harry:516
Hermione:517
```

When using a space as the delimiter for **cut**, you have to quote the space.

```
[clim@sop pipes]$ cut -d" " -f1 tennis.txt
Amelie

Kim
Justine
Serena
Venus
```

This example uses **cut** to display the second to the seventh character of **/etc/passwd**.

```
[clim@sop pipes]$ cut -c2-7 /etc/passwd | tail -4
igo:x:

faff:x
arry:x
ermion
```

2d.5. tr

You can translate characters with **tr**. The screenshot shows the translation of all occurrences of e to E.

```
[clim@sop pipes]$ cat tennis.txt | tr 'e' 'E'
AmEliE MaurEsmo, Fra

Kim ClijstErs, BEL
JustinE HEnin, BEl
SErEna Williams, usa
```

Here we set all letters to uppercase by defining two ranges.

```
[clim@sop pipes]$ cat tennis.txt | tr 'a-z' 'A-Z'
AMELIE MAURESMO, FRA

KIM CLIJSTERS, BEL
JUSTINE HENIN, BEL
SERENA WILLIAMS, USA
VENUS WILLIAMS, USA
```

Here we translate all newlines to spaces.

```
[clim@sop pipes]$ cat count.txt
one
```

```
three
four
five

[clim@sop pipes]$ cat count.txt | tr '\n' ' '
```

The **tr -s** filter can also be used to squeeze multiple occurrences of a character to one.

```
[clim@sop pipes]$ cat spaces.txt
one    two        three

     four   five  six

[clim@sop pipes]$ cat spaces.txt | tr -s ' '
one two three
```

You can also use **tr** to 'encrypt' texts with **rot13**.

```
[clim@sop pipes]$ cat count.txt | tr 'a-z' 'nopqrstuvwxyzabcdefghijklm'
bar

gjb
guerr
sbhe
svir

[clim@sop pipes]$ cat count.txt | tr 'a-z' 'n-za-m'
bar

gjb
guerr
sbhe
```

This last example uses **tr -d** to delete characters.

```
clim@debian5:~/pipes$ cat tennis.txt | tr -d e
Amli Maursmo, Fra

Kim Clijstrs, BEL
Justin Hnin, Bl
Srna Williams, usa
```

2d.6. wc

Counting words, lines and characters is easy with **wc**.

```
[clim@sop pipes]$ wc tennis.txt

  5  15 100 tennis.txt

[clim@sop pipes]$ wc -l tennis.txt

5 tennis.txt

[clim@sop pipes]$ wc -w tennis.txt
```

2d.7. sort

The **sort** filter will default to an alphabetical sort.

```
clim@debian5:~/pipes$ cat music.txt
Queen

Brel

Led Zeppelin
Abba

clim@debian5:~/pipes$ sort music.txt
Abba
```

But the **sort** filter has many options to tweak its usage. This example shows sorting different columns (column 1 or column 2).

```
[clim@sop pipes]$ sort -k1 country.txt
Belgium, Brussels, 10

France, Paris, 60

Germany, Berlin, 100

Iran, Teheran, 70

Italy, Rome, 50

[clim@sop pipes]$ sort -k2 country.txt
Germany, Berlin, 100
```

The screenshot below shows the difference between an alphabetical sort and a numerical sort (both on the third column).

```
[clim@sop pipes]$ sort -k3 country.txt
Belgium, Brussels, 10

Germany, Berlin, 100

Italy, Rome, 50

France, Paris, 60

Iran, Teheran, 70

[clim@sop pipes]$ sort -n -k3 country.txt
Belgium, Brussels, 10
```

2d.8. uniq

With **uniq** you can remove duplicates from a **sorted list**.

```
clim@debian5:~/pipes$ cat music.txt
Queen

Brel
Queen
Abba

clim@debian5:~/pipes$ sort music.txt
Abba

Brel
Queen
Queen

clim@debian5:~/pipes$ sort music.txt |uniq
```

uniq can also count occurrences with the **-c** option.

```
clim@debian5:~/pipes$ sort music.txt |uniq -c

      1 Abba

      1 Brel
```

2d.9. comm

Comparing streams (or files) can be done with the **comm**. By default **comm** will output three columns. In this example, Abba, Cure and Queen are in both lists, Bowie and Sweet are only in the first file, Turner is only in the second.

```
clim@debian5:~/pipes$ cat > list1.txt
Abba

Bowie
Cure
Queen
Sweet

clim@debian5:~/pipes$ cat > list2.txt
Abba

Cure
Queen
Turner

clim@debian5:~/pipes$ comm list1.txt list2.txt

                Abba

Bowie
```

The output of **comm** can be easier to read when outputting only a single column. The digits point out which output columns should not be displayed.

```


clim@debian5:~/pipes$ comm -12 list1.txt list2.txt
Abba

Cure
Queen

clim@debian5:~/pipes$ comm -13 list1.txt list2.txt
Turner
```

2d.10. od

European humans like to work with ascii characters, but computers store files in bytes.The example below creates a simple file, and then uses **od** to show the contents of the file in hexadecimal bytes

```
clim@laika:~/test$ cat > text.txt
abcdefg

1234567

clim@laika:~/test$ od -t x1 text.txt
```

The same file can also be displayed in octal bytes.

```
clim@laika:~/test$ od -b text.txt

0000000 141 142 143 144 145 146 147 012 061 062 063 064 065 066 067 012
```

And here is the file in ascii (or backslashed) characters.

```
clim@laika:~/test$ od -c text.txt
0000000   a   b   c   d   e   f   g  \n   1   2   3   4   5   6   7  \n

0000020
```

2d.11. sed

The **s**tream **ed**itor **sed** can perform editing functions in the stream, using **regular expressions**.

```
clim@debian5:~/pipes$ echo level5 | sed 's/5/42/'
level42

clim@debian5:~/pipes$ echo level5 | sed 's/level/jump/'
jump5
```

Add **g** for global replacements (all occurrences of the string per line).

```
clim@debian5:~/pipes$ echo level5 level7 | sed 's/level/jump/'
jump5 level7

clim@debian5:~/pipes$ echo level5 level7 | sed 's/level/jump/g'
jump5 jump7
```

With **d** you can remove lines from a stream containing a character.

```
clim@debian5:~/test42$ cat tennis.txt
Venus Williams, USA

Martina Hingis, SUI
Justine Henin, BE
Serena williams, USA
Kim Clijsters, BE
Yanina Wickmayer, BE

clim@debian5:~/test42$ cat tennis.txt | sed '/BE/d'
Venus Williams, USA
```

2d.12. pipe examples

2d.12.1. who | wc

How many users are logged on to this system ?

```
[clim@sop pipes]$ who

root     tty1         Jul 25 10:50

clim     pts/0        Jul 25 09:29 (laika)

Harry    pts/1        Jul 25 12:26 (barry)
```

2d.12.2. who | cut | sort

Display a sorted list of logged on users.

```
[clim@sop pipes]$ who | cut -d' ' -f1 | sort
Harry

clim
clim
```

Display a sorted list of logged on users, but every user only once .

```
[clim@sop pipes]$ who | cut -d' ' -f1 | sort | uniq
Harry

clim
```

2d.12.3. grep | cut

Display a list of all bash **user accounts** on this computer. Users accounts are explained in detail later.

```
clim@debian5:~$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
clim:x:1000:1000:clim,,,:/home/clim:/bin/bash
serena:x:1001:1001::/home/serena:/bin/bash
clim@debian5:~$ grep bash /etc/passwd | cut -d: -f1
root

clim
```

2d.13. practice: filters

1. Put a sorted list of all bash users in bashusers.txt.
2. Put a sorted list of all logged on users in onlineusers.txt.
3. Make a list of all filenames in **/etc** that contain the string **conf** in their filename.
4. Make a sorted list of all files in **/etc** that contain the case insensitive string **conf** in their filename.
5. Look at the output of **/sbin/ifconfig**. Write a line that displays only ip address and the subnet mask.
6. Write a line that removes all non-letters from a stream.
7. Write a line that receives a text file, and outputs all words on a separate line.
8. Write a spell checker on the command line. (There may be a dictionary in **/usr/share/dict/** .)