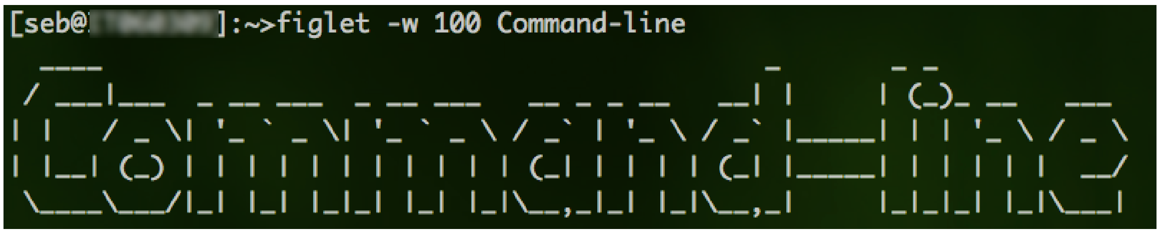
203341 Linux/Unix command-line tutorial

Sebastian Schmeier

2015-07-17

Contents

Unix/Linux command-line tutorial



```
[seb@[redacted]]:~>figlet -w 100 Command-line
  ____                                          _       _ _
 / ___|___  _ __ ___  _ __ ___   __ _ _ __   __| |     | (_)_ __   ___
| |   / _ \| '_ ` _ \| '_ ` _ \ / _` | '_ \ / _` |_____| | | '_ \ / _ \
| |__| (_) | | | | | | | | | | | (_| | | | | (_| |_____| | | | | |  __/
 \____\___/|_| |_| |_|_| |_| |_|\__,_|_| |_|\__,_|     |_|_|_| |_|\___|
```

This tutorial is based on a Linux/Unix *command-line*. Using the *command-line* requires a Linux/Unix operating system. The easiest way to try out a Linux system without actually installing it on your computer is a LiveCD. A LiveCD is a DVD that you prepare (e.g. burn a Linux distribution on it) and insert in your computer. You would restart you computer and can run Linux from the DVD without any installation requirements. This is helpful for trying out a distribution of Linux not for actual work.

Another route would be to use a virtual machine. Software to create a virtual machine is free, e.g. VirtualBox.

Common flavors of Linux ready for download are e.g. Ubuntu or if you are thinking of going the bioinformatics route, BioLinux, which includes many pre-installed bioinformatics tools (this is also the distribution we will be using).

0. Learning outcomes

1. Be able to operate comfortably the Linux command-line.
2. Be able to navigate the unix directory structure on the command-line.
3. Be able to start command-line programs and getting help/information about programs.
4. Be able to explain the concept of a unix pipe.

1.1 Introduction

This is a collection of commands and programs I put together for working under Linux/Unix shells. It is not comprehensive. It includes very basic stuff. Tutorial style. This is bash syntax but most of it will work on other shells (tcsh, sh) as well.

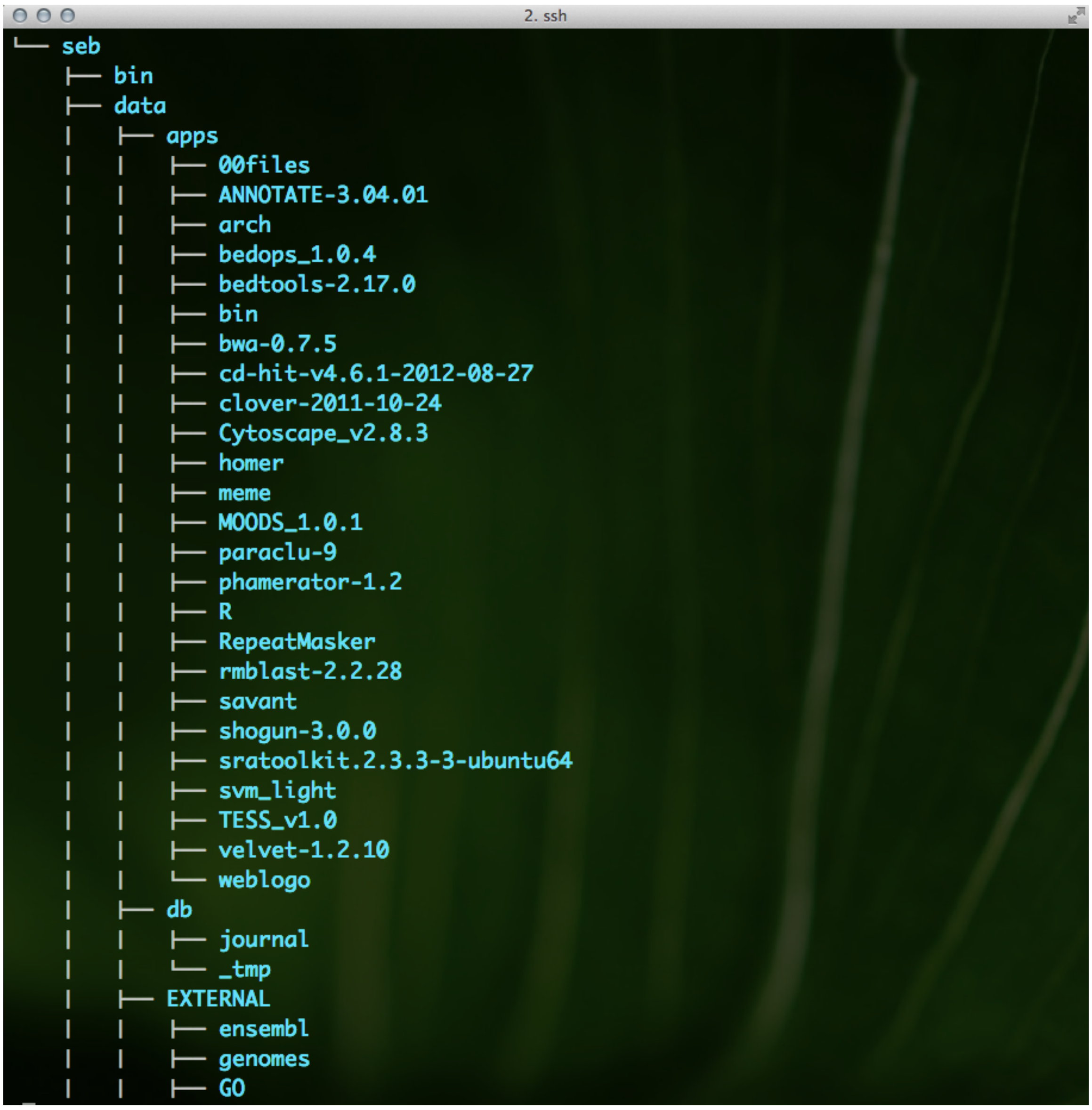
What is a shell? Here I shamelessly quote Wikipedia:

> In computing, a shell is a user interface for access to an operating system's services. In general, operating system shells use either a command-line interface (**CLI**) or graphical user interface (GUI), depending on a computer's role and particular operation...
>
> **CLI** shells allow some operations to be performed faster in some situations, especially when a proper GUI has not been or cannot be created. However, they require the user to memorize all commands and their calling syntax, and also to learn the shell-specific scripting language, for example bash script.

Hint! If you see a ***grey*** box and a line starting with the " **$**" command-line prompt, this means this is command-line code and you can paste it (without the **$**) into the command-line and hit "Enter" to run it. If you see a "#" at the start of a line, this denotes a comment.

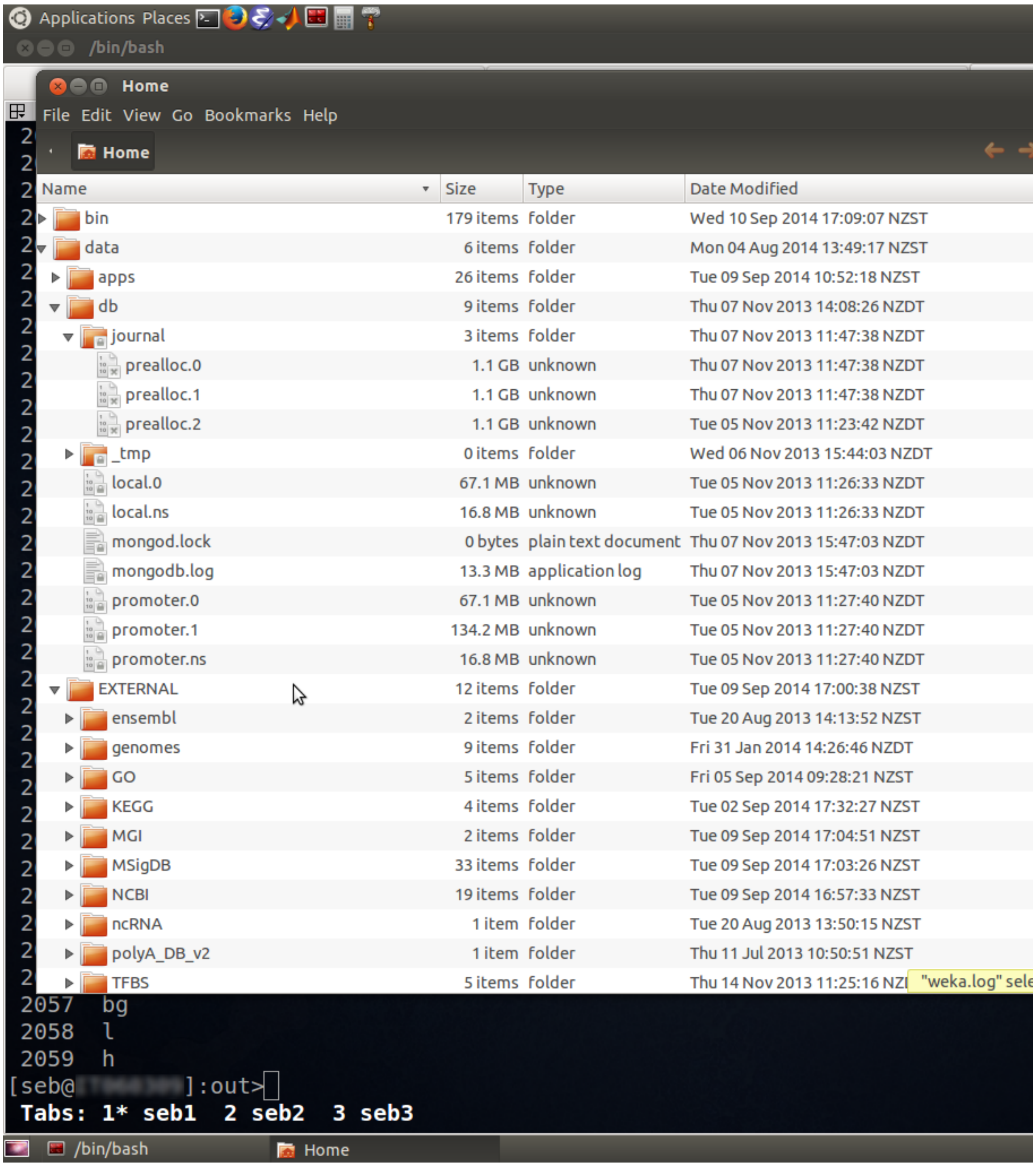
1.2 Upfront some words regarding the linux file-system

The directory structure in a Linux system is not much different from any other system you worked with, e.g. Windows, MacOSX. It is essentially a tree structure.



```
└── seb
    ├── bin
    ├── data
    |   ├── apps
    |   |   ├── 00files
    |   |   ├── ANNOTATE-3.04.01
    |   |   ├── arch
    |   |   ├── bedops_1.0.4
    |   |   ├── bedtools-2.17.0
    |   |   ├── bin
    |   |   ├── bwa-0.7.5
    |   |   ├── cd-hit-v4.6.1-2012-08-27
    |   |   ├── clover-2011-10-24
    |   |   ├── Cytoscape_v2.8.3
    |   |   ├── homer
    |   |   ├── meme
    |   |   ├── MOODS_1.0.1
    |   |   ├── paraclu-9
    |   |   ├── phamerator-1.2
    |   |   ├── R
    |   |   ├── RepeatMasker
    |   |   ├── rmblast-2.2.28
    |   |   ├── savant
    |   |   ├── shogun-3.0.0
    |   |   ├── sratoolkit.2.3.3-3-ubuntu64
    |   |   ├── svm_light
    |   |   ├── TESS_v1.0
    |   |   ├── velvet-1.2.10
    |   |   └── weblogo
    |   ├── db
    |   |   ├── journal
    |   |   └── _tmp
    |   ├── EXTERNAL
    |   |   ├── ensembl
    |   |   ├── genomes
    |   |   ├── GO
```

To navigate the file-system you can use a file-manager e.g. Nautilus.



Name	Size	Type	Date Modified
bin	179 items	folder	Wed 10 Sep 2014 17:09:07 NZST
data	6 items	folder	Mon 04 Aug 2014 13:49:17 NZST
apps	26 items	folder	Tue 09 Sep 2014 10:52:18 NZST
db	9 items	folder	Thu 07 Nov 2013 14:08:26 NZDT
journal	3 items	folder	Thu 07 Nov 2013 11:47:38 NZDT
prealloc.0	1.1 GB	unknown	Thu 07 Nov 2013 11:47:38 NZDT
prealloc.1	1.1 GB	unknown	Thu 07 Nov 2013 11:47:38 NZDT
prealloc.2	1.1 GB	unknown	Tue 05 Nov 2013 11:23:42 NZDT
_tmp	0 items	folder	Wed 06 Nov 2013 15:44:03 NZDT
local.0	67.1 MB	unknown	Tue 05 Nov 2013 11:26:33 NZDT
local.ns	16.8 MB	unknown	Tue 05 Nov 2013 11:26:33 NZDT
mongod.lock	0 bytes	plain text document	Thu 07 Nov 2013 15:47:03 NZDT
mongodb.log	13.3 MB	application log	Thu 07 Nov 2013 15:47:03 NZDT
promoter.0	67.1 MB	unknown	Tue 05 Nov 2013 11:27:40 NZDT
promoter.1	134.2 MB	unknown	Tue 05 Nov 2013 11:27:40 NZDT
promoter.ns	16.8 MB	unknown	Tue 05 Nov 2013 11:27:40 NZDT
EXTERNAL	12 items	folder	Tue 09 Sep 2014 17:00:38 NZST
ensembl	2 items	folder	Tue 20 Aug 2013 14:13:52 NZST
genomes	9 items	folder	Fri 31 Jan 2014 14:26:46 NZDT
GO	5 items	folder	Fri 05 Sep 2014 09:28:21 NZST
KEGG	4 items	folder	Tue 02 Sep 2014 17:32:27 NZST
MGI	2 items	folder	Tue 09 Sep 2014 17:04:51 NZST
MSigDB	33 items	folder	Tue 09 Sep 2014 17:03:26 NZST
NCBI	19 items	folder	Tue 09 Sep 2014 16:57:33 NZST
ncRNA	1 item	folder	Tue 20 Aug 2013 13:50:15 NZST
polyA_DB_v2	1 item	folder	Thu 11 Jul 2013 10:50:51 NZST
TFBS	5 items	folder	Thu 14 Nov 2013 11:25:16 NZI

```
 2057  bg
 2058  l
 2059  h
[seb@[redacted]]:out>
 Tabs: 1* seb1  2 seb2  3 seb3
```

However, on the command-line we navigate via commands and not via mouse clicks. Why is this necessary to use the command-line in the first place? Strictly speaking it is not, if you do not want to make use of programs on the command-line. However, the power of the Linux system becomes only obvious once we learn to make use of the command-line, thus navigating the directory structure via commands is one of the **most important skills** for you to know.

1.3 Open a terminal

Open a terminal window and you are are ready to go. On your linux desktop find: **Application** –> **Accessories** –> **Terminal** (for Gnome environent) or type "Terminal" in the search box.

1.4 Getting help about command-line programs (`man`)

This is likely a task you will perform quite often, so it is good that you know how to do it.

Hint! The program `man` is your most important friend.

With `man` getting help is as easy as:

```
$ man pwd
```

```
WD(1)                     BSD General Commands Manual                     PWD(1)

NAME
     pwd -- return working directory name

SYNOPSIS
     pwd [-L | -P]

DESCRIPTION
     The pwd utility writes the absolute pathname of the current working
     directory to the standard output.

     Some shells may provide a builtin pwd command which is similar or identical
     to this utility.  Consult the builtin(1) manual page.
.
.
.
```

Hint! You can navigate the view down with "j" and up with "k". You exit the view with "q".

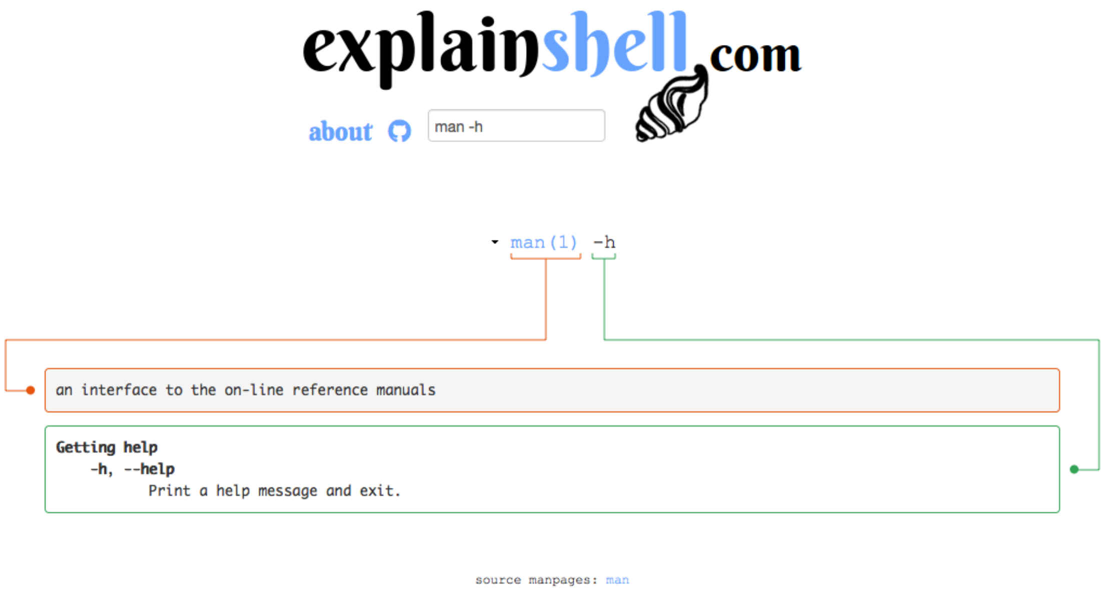
Lets look at the manual pages of `man` itself:

```
$ man man
```

```
man(1)                                                                        man(1)

NAME
       man - format and display the on-line manual pages

SYNOPSIS
       man  [-acdfFhkKtwW] [--path] [-m system] [-p string] [-C config_file]
       [-M pathlist] [-P pager] [-B browser] [-H htmlpager] [-S section_list]
       [section] name ...

DESCRIPTION
```

```
       man formats and displays the on-line manual pages.  If  you  specify
       section,  man only  looks in that section of the manual.  name is
       normally the name of the manual page, which is typically the name of
       a command, function,  or  file.
.
.
.
```

Another very helpful resource is the explainshell.com webpage, that lets you write down a *command-line* to see the help text that matches each argument.



```
man(1) -h

an interface to the on-line reference manuals

Getting help
    -h, --help
           Print a help message and exit.

source manpages: man
```

1.5 Navigating the directory tree on the command-line

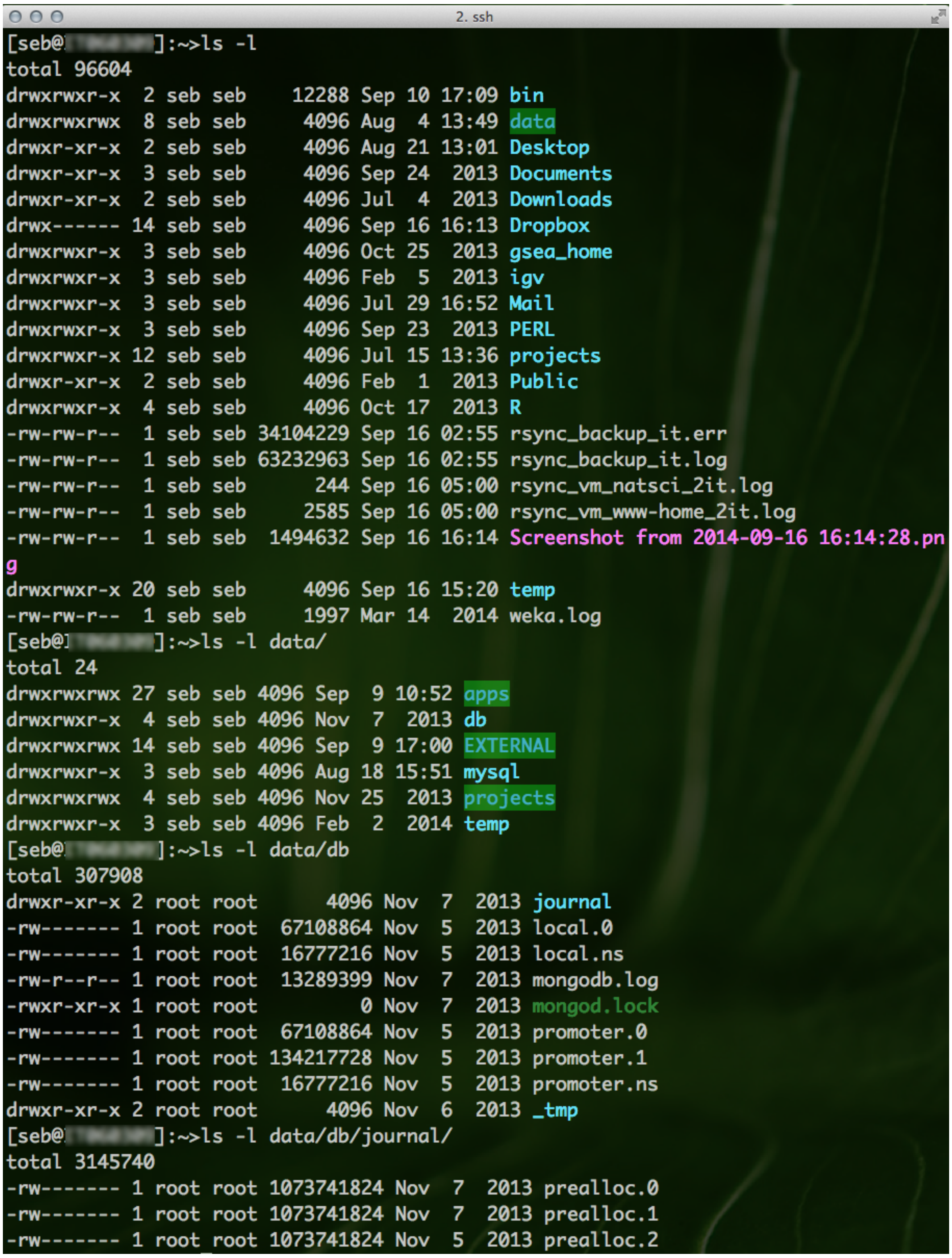
This is possibly one of the most important skills you need to learn. You need to understand where you are in the file-system, how to get to a certain directory that contains the files/programs you want to work with.

Identify the current directory path / Where am I? (`pwd`)

```
# What directory am I in?
# Find out using the "pwd" command (print name of current/working directory)
$ pwd
# you should see something like /home/seb
```

Investigate directories / list directory content (`ls`)

```
# list the current directory elements implicitly
$ ls

# the same in a nicer format
$ ls -l

# List a particular directory (e.g. temp/) explicitly
$ ls -l temp/
```



2. ssh

```
[seb@[redacted]]:~>ls -l
total 96604
drwxrwxr-x  2 seb seb    12288 Sep 10 17:09 bin
drwxrwxrwx  8 seb seb     4096 Aug  4 13:49 data
drwxr-xr-x  2 seb seb     4096 Aug 21 13:01 Desktop
drwxr-xr-x  3 seb seb     4096 Sep 24  2013 Documents
drwxr-xr-x  2 seb seb     4096 Jul  4  2013 Downloads
drwx------ 14 seb seb     4096 Sep 16 16:13 Dropbox
drwxrwxr-x  3 seb seb     4096 Oct 25  2013 gsea_home
drwxrwxr-x  3 seb seb     4096 Feb  5  2013 igv
drwxrwxr-x  3 seb seb     4096 Jul 29 16:52 Mail
drwxrwxr-x  3 seb seb     4096 Sep 23  2013 PERL
drwxrwxr-x 12 seb seb     4096 Jul 15 13:36 projects
drwxr-xr-x  2 seb seb     4096 Feb  1  2013 Public
drwxrwxr-x  4 seb seb     4096 Oct 17  2013 R
-rw-rw-r--  1 seb seb 34104229 Sep 16 02:55 rsync_backup_it.err
-rw-rw-r--  1 seb seb 63232963 Sep 16 02:55 rsync_backup_it.log
-rw-rw-r--  1 seb seb      244 Sep 16 05:00 rsync_vm_natsci_2it.log
-rw-rw-r--  1 seb seb     2585 Sep 16 05:00 rsync_vm_www-home_2it.log
-rw-rw-r--  1 seb seb  1494632 Sep 16 16:14 Screenshot from 2014-09-16 16:14:28.pn
g
drwxrwxr-x 20 seb seb     4096 Sep 16 15:20 temp
-rw-rw-r--  1 seb seb     1997 Mar 14  2014 weka.log
[seb@[redacted]]:~>ls -l data/
total 24
drwxrwxrwx 27 seb seb 4096 Sep  9 10:52 apps
drwxrwxr-x  4 seb seb 4096 Nov  7  2013 db
drwxrwxrwx 14 seb seb 4096 Sep  9 17:00 EXTERNAL
drwxrwxr-x  3 seb seb 4096 Aug 18 15:51 mysql
drwxrwxrwx  4 seb seb 4096 Nov 25  2013 projects
drwxrwxr-x  3 seb seb 4096 Feb  2  2014 temp
[seb@[redacted]]:~>ls -l data/db
total 307908
drwxr-xr-x 2 root root      4096 Nov  7  2013 journal
-rw------- 1 root root  67108864 Nov  5  2013 local.0
-rw------- 1 root root  16777216 Nov  5  2013 local.ns
-rw-r--r-- 1 root root  13289399 Nov  7  2013 mongodb.log
-rwxr-xr-x 1 root root         0 Nov  7  2013 mongod.lock
-rw------- 1 root root  67108864 Nov  5  2013 promoter.0
-rw------- 1 root root 134217728 Nov  5  2013 promoter.1
-rw------- 1 root root  16777216 Nov  5  2013 promoter.ns
drwxr-xr-x 2 root root      4096 Nov  6  2013 _tmp
[seb@[redacted]]:~>ls -l data/db/journal/
total 3145740
-rw------- 1 root root 1073741824 Nov  7  2013 prealloc.0
-rw------- 1 root root 1073741824 Nov  7  2013 prealloc.1
-rw------- 1 root root 1073741824 Nov  5  2013 prealloc.2
```

Moving around in the file system / changing directories (`cd`)

```
# Where am I?
$ pwd
/home/seb
# change into directory "Desktop" explicitly with command "cd" (change directory)
$ cd /home/seb/Desktop
# Where am I?
$ pwd
/home/seb/Desktop
# you moved to the Desktop directory

# Go to your home directory from any position in the directory tree
$ cd
# Where am I?
$ pwd
/home/seb

# A shortcut for the home directory is ~/
# This command will change to /home/user/Desktop from any position in
# the directory tree.
$ cd ~/Desktop
$ pwd
/home/seb/Desktop

# Go one directory up in the directory tree with the ".." operator
$ cd ..
# Where am I?
$ pwd
/home/seb

# Go two directories up in one go
$ cd ../..
# Where am I?
$ pwd
/
# Now you are at the file-system root

# Back to home directory
$ cd
```

1.6 File/Directory-handling

Create an empty directory (`mkdir`)

```
# Lets create a directory with the program "mkdir"
$ mkdir temp
```

Create a new empty text-file (`touch`)

```
# first change into the temp directory
$ cd temp
$ ls
total 0
# now create empty file
$ touch file1.txt
# list directory content
$ ls
file1.txt
```

Copy files/directories (`cp`)

```
# create empty file again
$ touch file1.txt
$ ls -l
total 4
-rw-rw-r-- 1 seb seb 0 Jul 17 17:45 file1.txt
$ cp file1.txt file2.txt
$ ls
file1.txt  file2.txt
$ ls -l
total 4
4 -rw-rw-r-- 1 seb seb 0 Jul 17 17:45 file1.txt
0 -rw-rw-r-- 1 seb seb 0 Jul 17 17:46 file2.txt

# back to home directory
$ cd
# copy temp to temp2
# -r stands for recursive
$ cp -r temp1 temp2
```

Move a file/directory (`mv`)

```
$ pwd
/home/seb/temp
$ ls
file1.txt  file2.txt
# move files
$ mv file1.txt file3.txt
$ ls
file2.txt  file3.txt
# move directories
$ mv dir1 dir2
# will not work because we miss "dir1"
```

Delete a file/directory (`rm`)

```
$ cd temp
# delete a file explicitly
$ rm file1.txt
# delete all files starting with "file"
$ rm file*
```

Warning! Avoid using `rm *`, as this will erase all files in the directory.

```
# Delete a whole directory.
# Back to the home directory
$ cd
# Where am I?
$ pwd
/home/seb
# -r stands for recursive
$ rm -r temp2/
```

Warning! Deleting files with the `rm` command does really delete them. They are not moved to a trash can, they are gone forever, thus take care of what you delete.

1.7 Investigate files

Note! Download two sample-files here and here.

Put them in the "temp" directory you created or somewhere else where you find them easily on the command-line.

Look into files (`less`)

```
$ less file1.txt
```

Hint! Move a line down with "j", up with "k", and you can get out of the view with "q".

Print the head/tail of files (`head` and `tail`)

```
# first 2 lines
$ head -2 file1.txt
AA,1,2,3,4
CC,9,10,11,12
# last 3 lines
$ tail -3 file1.txt
CC,9,10,11,12
BB,5,6,7,8
AAA,13,14,15,16
```

Note! Here we see for the first time another important concept of programs on the command-line. many of them print the results to what is called "standard-out", which in our case currently is the terminal window.

Concatenate content of files (`cat`)

```
$ cat file1.txt file2.txt
AA,1,2,3,4
CC,9,10,11,12
BB,5,6,7,8
AAA,13,14,15,16
ZZZ,9,10,11,12
XXX,1,2,3,4
YYY,5,6,7,8
BB,5,6,7,8
# all files starting with "file":
$ cat file*
AA,1,2,3,4
CC,9,10,11,12
BB,5,6,7,8
AAA,13,14,15,16
ZZZ,9,10,11,12
XXX,1,2,3,4
YYY,5,6,7,8
BB,5,6,7,8
# print content from one file to stdout:
$ cat file1.txt
AA,1,2,3,4
CC,9,10,11,12
BB,5,6,7,8
AAA,13,14,15,16
```

Note! `cat` also prints output by default to standard-out, currently the terminal window.

Count number of rows of a file (`wc`)

```
$ wc -l file1.txt
4 file1.txt
# -l stands for lines, by default wc

$ man wc
WC(1)                                  User Commands                                  WC(1)

NAME
           wc - print newline, word, and byte counts for each file
.
.
.
```

1.8 Other operations on files

Sorting files (`sort`)

```
$ cat file1.txt
AA,1,2,3,4
CC,9,10,11,12
BB,5,6,7,8
AAA,13,14,15,16

# sort on complete line
$ sort file1.txt
AA,1,2,3,4
AAA,13,14,15,16
BB,5,6,7,8
CC,9,10,11,12

# sort a comma-seperated file on third field
$ sort -t ',' -k3,3 file1.txt
CC,9,10,11,12
AAA,13,14,15,16
AA,1,2,3,4
BB,5,6,7,8

# sort a comma-seperated file on third field according to numbers
$ sort -t ',' -k2,2n file1.txt
AA,1,2,3,4
BB,5,6,7,8
CC,9,10,11,12
AAA,13,14,15,16
```

Extract columns of a file (`cut`)

```
# cut -d'seperator' -fCOLUMN,COLUMN,...  file.txt, e.g.
# cut out second column
$ cut -d ',' -f 2 file1.txt
1
9
5
13

# cut out column 1,3,4,5
$ cut -d ',' -f 1,3-5 file1.txt
AA,2,3,4
CC,10,11,12
BB,6,7,8
AAA,14,15,16
```

Search lines with certain pattern (grep and egrep)

```
# print only lines of a file that contain a pattern:
$ grep 'AAA' file1.txt

# print only lines that do _not_ contain the pattern:
$ grep -v 'AAA' file1.txt

# the same using regular expressions
$ egrep 'A+.+14' file1.txt
# Lets investigate what is happening here:
# 'A+.+14'
# Look for at least one A ("A+")
# followed by random characters (".") ath least one or more ("+")
# followed by a 14
```

1.8 Compression magic

Compress/Extract a file (gzip)

```
$ gzip file1.txt
# will produce a file called file1.txt.gz in gzip format, and delete file1.txt

# Extract a gzipped-file
$ gzip -d file1.txt.gz
```

Look into compressed files on-the-fly (zless and zcat)

We do not need to decompress a file to use look at its content (most of my text files are stored in gzip format):

```
$ zless file1.txt.gz
$ zcat file1.txt.gz
$ zcat file1.txt.gz
```

Compress/decompress using zip (zip)

```
# Compress into file.zip archive
$ zip file.zip file1.txt

# Extract a zipped-file/archive
$ unzip file1.zip
```

1.9 Redirecting standard-out / pipes

Redirecting output from programs to other programs

```
$ cat file2.txt
ZZZ,9,10,11,12
XXX,1,2,3,4
YYY,5,6,7,8
BB,5,6,7,8

# Cut out second column of file
$  cut -d ',' -f2 file2.txt

# This can be rewritten using the output of cat as input to cut using "|" operator
$ cat file2.txt | cut -d ',' -f2
9
1
5
5
```

Note! In the first command we are using `cut` explicitly with a file, whereas in the last example we used the output from one program `cat` as input for `cut` concatenated with the | pipe operator.

As most unix programs except input from standard in and most programs can write to standard out we essentially can concatenate many programs one after each other to perform many operations in one go.

In this example we aim at counting the unique lines of the second column of "file2.txt". This will be done using the program `uniq`, which need sorted input from the program `sort`.

```
# make lines uniq using uniq, and sort

$ cat file2.txt | cut -d ',' -f2
9
1
5
5
# get unique lines
$ cat file2.txt | cut -d ',' -f2 | sort | uniq
1
5
9
# Now also count
$ cat file2.txt | cut -d ',' -f2 | sort | uniq | wc -l
3
```

Redirecting output into a file

This can be done with the ">" operator.

```
# Find all lines in file that contain a "5"
$ cat file2.txt | grep '5' > extractedLines.txt
$ cat extractedLines.txt
YYY,5,6,7,8
BB,5,6,7,8

# We can also append to an existing file with ">>"
# Find all lines that contain a "1" folowed by a "3"
$ cat file1.txt | egrep '1.*3' >> extractedLines.txt
$ cat extractedLines.txt
YYY,5,6,7,8
BB,5,6,7,8
AA,1,2,3,4
AAA,13,14,15,16
```

2.1 Processes

2.2 File privileges

File: index.md** - **Sebastian Schmeier** - **Last update: 2015-07-10