Introduction to the Linux Commandline for NGS analyses

Release 0.5

Grainne Kerr, Holger Dinkel, Matt Betts

September 17, 2013

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION TO THE LINUX COMMANDLINE

1.1 Why Use the Commandline

- It's **fast**. Productivity is a word that gets tossed around a lot by so-called power users, but the command line can really streamline your computer use, assuming you learn to use it right.
- It's **easier to get help**. The command line may not be the easiest thing to use, but it makes life a whole lot easier for people trying to help you and for yourself when looking for help, especially over the internet. Many times it's as simple as the helper posting a few commands and some instructions and the recipient copying and pasting those commands. Anyone who has spent hours listening to someone from tech support say something like, "OK, now click this, then this, then select this menu command" knows how frustrating the GUI alternative can be.
- It's nearly **universal**. There are hundreds of Linux distros out there, each with a slightly different graphical environment. Thankfully, the various distros do have one common element: the command line. There are distro-specific commands, but the bulk of commands will work on any Linux system.
- It's **powerful**. The companies behind those other operating systems try their best to stop a user from accidentally screwing up their computer. Doing this involves hiding a lot of the components and tools that could harm a computer away from novices. Linux is more of an open book, which is due in part to its prominent use of the command line.

1.2 General Remarks Regarding Using UNIX/Linux Systems

- **Test before run**. Anything written here has to be taken with a grain of salt. On another system – be it a different Linux distribution or another UNIXoid operating system – you might find the same command but without the support of some of the options tought here. It is even possible, that the same option has a different meaning on another system. With this in mind always make sure to test your commands (specially the "dangerous" ones which remove or modify files) when switching from one system to the other.
- **The Linux/UNIX environment**. The behaviour of many commands is influenced or controlled by the so-called "environment". This environment is the sum of all your environment variables. Some of these environment variables will be shown towards the end of this course.
- **UPPERCASE, lowercase**. Don't forget that everything is case-sensitive.
- **The Filesystem**. Linux filesystems start on top at the root directory (sic!) "/" which hierarchically broadens towards the ground. The separator between directories or directories and files in Linux is the slash ("/").

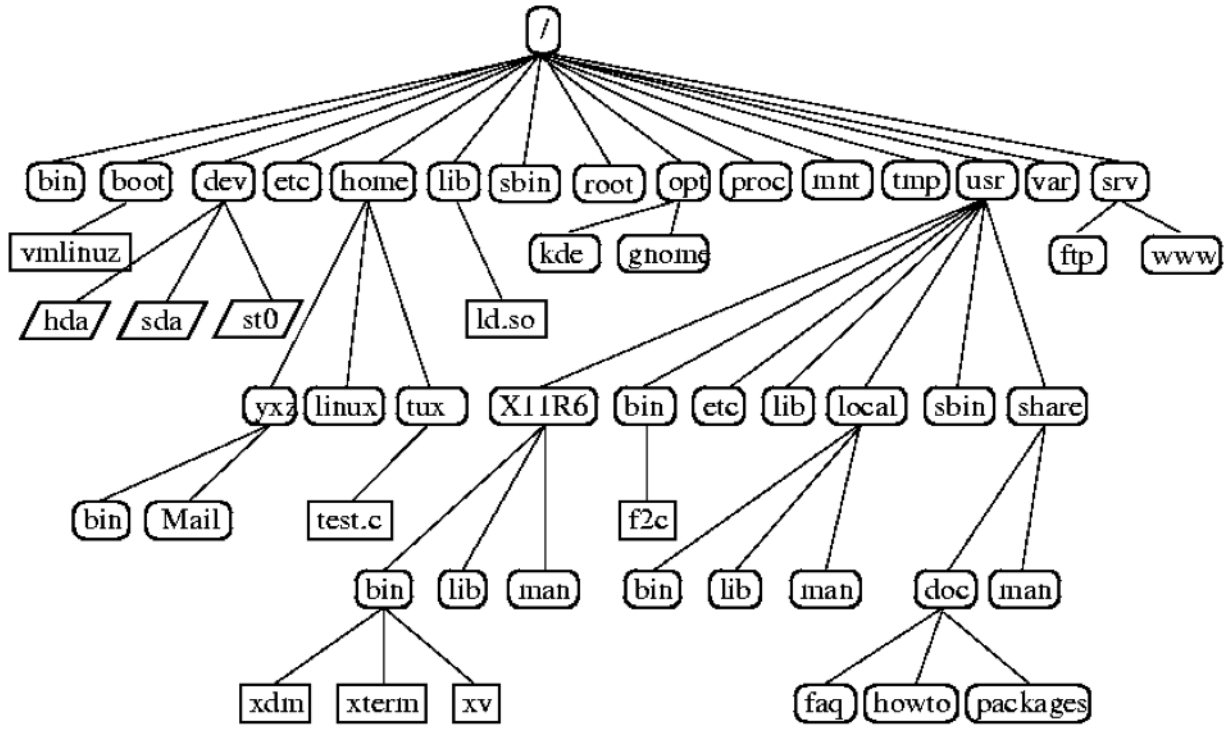
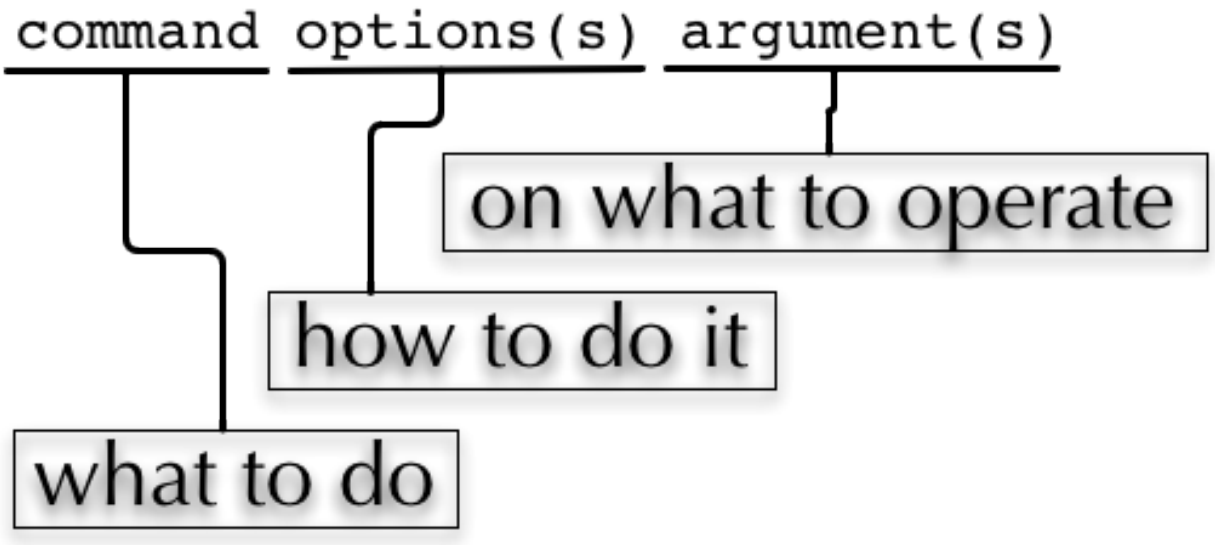


Figure 1.1: Depending on the Linux distribution you might or might not find all of above directories. Most important directories for you are `/bin` and `/usr/bin` (sometimes also `/usr/local/bin`) which contain the user software, `/home` which usually contains the users' homedirectories and `/tmp` which can be used to store temporary data (beware: Its content is regularly removed!).

Note: The terms "directory" and "folder" are used interchangeably in this document.

1.3 General Structure of Linux Commands

Linux commands have the following general structure:



commandline options (sometimes called comandline switches) commonly have one of the two following forms: The short form `-character` or the long form `--string`. E.g.

```
> man -h
> man --help
```

Short options are usually – though not always – concatenable:

```
> ls -l -A -h
> ls -lAh
```

Some options require an additional argument, which is added with a blank to the short form and with an equal sign to the long form:

```
> ls -I "*.pdf"
> ls --ignore="*.pdf"
```

Since Linux incorporates commands from different sources, options can be available in one or both forms and you'll also encounter options with no dash at all and all kinds of mixtures:

```
> tar cf file.tar -C .. file/
> ps auxgww
```

1.4 A Journey Through the Commands

Please note that all examples and usage instructions below are just a glimpse of what you can do and reflect our opinion on what's important and what's not. Most of these commands support many more options and different usages. Consult the manpages to find them. Typographical conventions: Commands and examples are written in Courier. User Input is written in Courier bold and placeholders are generally written in italic.

1.4.1 Getting Help

`-h/--help` option, no parameters

Many commands support a "help" option, either through `-h` or through `--help`. Other commands will show a help page or at least a short usage overview if you provide wrong commandline options

man - show the manual page of a command

Usage: `man command or file`

```
> man man
man(1)

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

SYNOPSIS
 man [-acdfFhkKtwW] [--path] [-m system] [-p string] [-C config_file]
[...]
```

For the navigation within a man-page see the chapter regarding less below.

Note: The behaviour of man is dependent of the $PAGER environment variable

---

apropos – list manpages containing a keyword in their description

Usage: `apropos keyword`

```
> apropos who
[...]
> who                  (1)  - show who is logged on
> who                  (1)  - display who is on the system
> whoami               (1)  - print effective userid
```

Use apropos to find candidates for specific tasks

/usr/share/doc

The `/usr/share/doc` directory in some Linux distributions contains additional documentation of installed software packages

1.4.2 Who am I, where am I

whoami – Print your username

Usage: `whoami`

```
> whoami
fthommen
```

hostname – Print the name of the computer

Usage: `hostname`

```
> hostname
pc-teach01
```

pwd – Print the current working directory

Usage: `pwd`

```
> pwd
/home/fthommen
```

date – Print current date and time

Usage: `date`

```
> date
Tue Sep 25 19:57:50 CEST 2012
```

Note: The command time does something completely different than date and is not used to show the current time.

1.4.3 Moving Around

cd – Change the working directory

Usage: `cd [new_directory]`

```
# pwd
/home/fthommen
# cd /usr/bin
# pwd
/usr/bin
```

Special directories:

- "`.`": The current working directory
- "`..`": The parent directory of the current working directory
- "~": Your homedirectory

Note: Using cd without a directory is equivalent to "`cd ~`" and changes into the users's homedirectory

Note: Please note the difference between absolute pathes (starting with "/") and relative pathes (starting with a directory name)

```
$ pwd
/usr
$ cd /bin
$ pwd
/bin

> pwd
/usr
> cd bin
> pwd
/usr/bin
```

1.4.4 See What's Around

ls - List directory contents

Usage: `ls [options] [file(s) or directory/ies]`

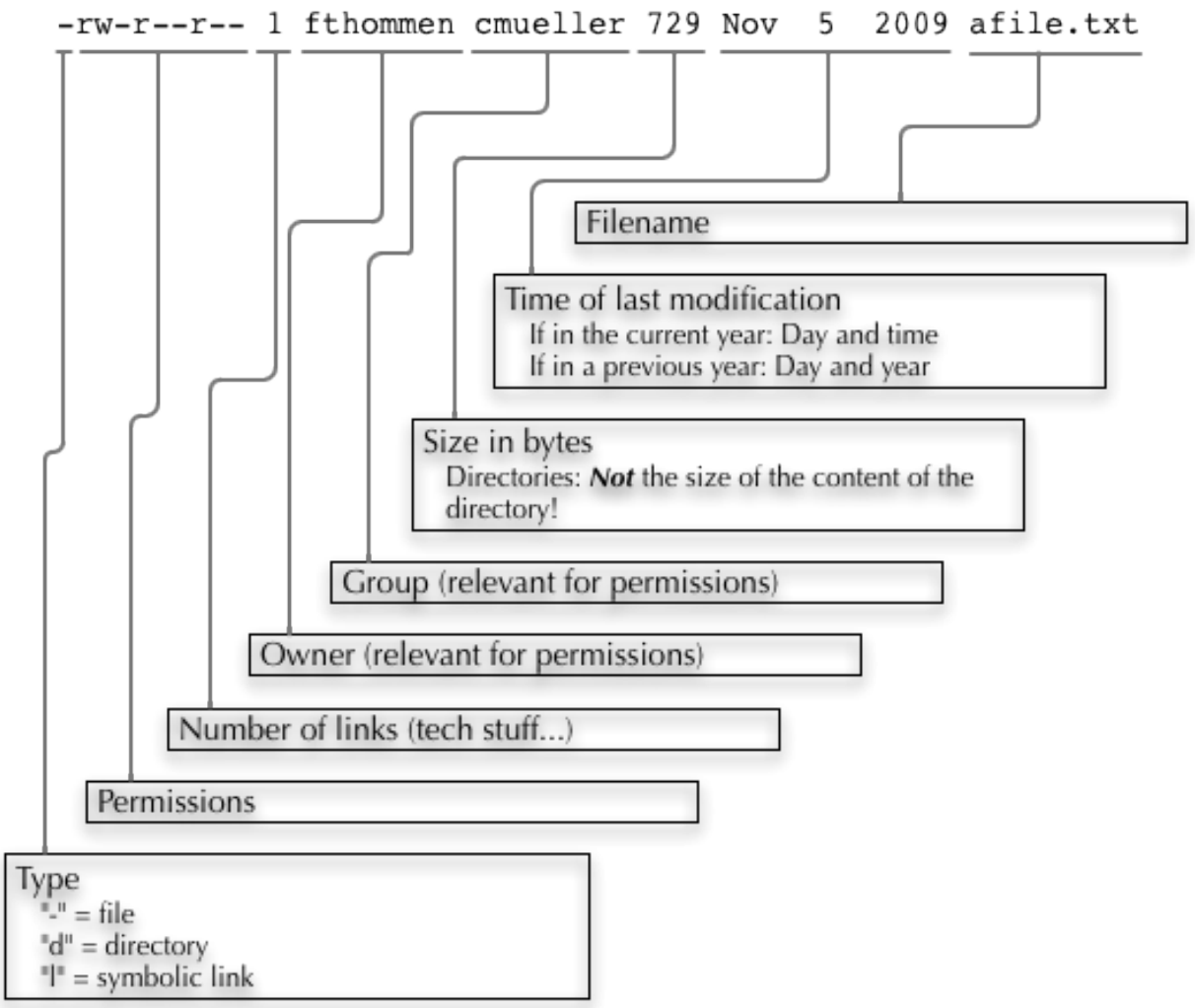
```
> ls
/home/fthommen
> ls -l aa.pdf
-rw-r--r-- 1 fthommen cmueller 0 Sep 24 10:59 aa.pdf
```

Useful options:

-l	Long listing with permissions, user, group and last modification date
-1	Print listing in one column only
-a	Show all files (hidden, "." and "..")
-A	Show almost all files (hidden, but not "." and "..")
-F	Show filetypes (nothing = regular file, "/" = directory, "*" = executable file, "@" = symbolic link)
-d	Show directory information instead of directory content
-t	Sort listing by modification time (most recent on top)

Digression: Shell globs

Files and folders can't only be referred to with their full name, but also with so-called "Shell Globs", which are a kind of simple pattern to address groups of files and folders. Instead of explicit names you can use the following placeholders:


-rw-r--r-- 1 fthommen cmueller 729 Nov 5 2009 afile.txt
Filename
Time of last modification
If in the current year: Day and time
If in a previous year: Day and year
Size in bytes
Directories: Not the size of the content of the directory!
Group (relevant for permissions)
Owner (relevant for permissions)
Number of links (tech stuff...)
Permissions
Type
"-" = file
"d" = directory
"l" = symbolic link

- `?`: Any single character
- `*`: Any number of any character (including no character at all)
- `[...]`: One of the characters included in the brackets. Use "-" to define ranges of characters

Examples:

- `*.pdf`: All files having the extension ".pdf"
- `?.jpg`: Jpeg file consisting of only one character
- `[0-9]*.txt`: All files starting with a number and having the extension ".txt"
- `*.???`: All files having a three-character extension

Note: The special directory "~" mentioned above is a shell glob, too.

1.4.5 Organize Files and Folders

touch – Create a file or change last modification date of an existing file

Usage: `touch file(s) or directory/ies`

```
> ls afile
ls: afile: No such file or directory
> touch afile
> ls afile
afile

> ls -l aa.pdf
-rw-r--r-- 1 fthommen cmueller 0 Sep 24 10:59 aa.pdf
> touch aa.pdf
> ls -l aa.pdf
-rw-r--r-- 1 fthommen cmueller 0 Sep 25 22:01 aa.pdf
```

rm – Remove files and directories

Usage: `rm [options] file(s)`

```
rm -r [options] directory/ies
> ls afile
afile
> rm afile
> ls afile
ls: afile: No such file or directory
```

Useful options:

-i	Ask for confirmation of each removal
-r	Remove recursively
-f	Force the removal (no questions, no errors if a file doesn't exist)

Note: rm without the –i option will usually not ask you if you really want to remove the file or directory

---

mv – Move and rename files and folders

Usage: `mv [options] sourcefile destinationfile`

```
mv [options] sourcefile(s) destinationdirectory
> ls *.txt
a.txt
> mv a.txt b.txt
> ls *.txt
b.txt
```

Useful options:

-i	Ask for confirmation of each removal

Note: You cannot overwrite an existing directory by another one with mv

---

mkdir – Create a new directory

Usage: `mkdir [options] directory`

```
> ls adir/
ls: adir/: No such file or directory
> mkdir adir
> ls adir
```

Useful options:

-p	Create parent directories (when creating nested directories)

```
> mkdir adir/bdir
mkdir: cannot create directory 'adir/bdir': No such file or directory
> mkdir -p adir/bdir
```

rmdir – Remove an empty directory

Usage: `rmdir directory`

```
> rmdir adir/
```

Note: If the directory is not empty, rmdir will complain and not remove it

---

cp – Copy files and folders

Usage: `cp [options] sourcefile destinationfile` .. note:: cp [options] sourcefile(s) destinationdirectory

```
> cp P12931.fasta backup_of_P12931.fasta
```

Useful options:

-r	Copy recursively
-i	Interactive operation, ask before overwriting an existing file

-p	Preserve owner, permissions and timestamp

View Files

cat – Print files on terminal (concatenate)

Usage: `cat [options] file(s)`

```
> cat  P12931.fasta backup_of_P12931.fasta
[...]
```

less – View and navigate files

Usage: `less [options] file(s)`

```
> less  P12931.fasta backup_of_P12931.fasta
[...]
```

Note: This is the default "pager" for manpages under Linux unless you redefine your $PAGER environment variable

---

Navigation within less:

Key(s):	Effect:
up, down, right, left:	use cursor keys
top of document:	g
bottom of document:	G
search:	"/" + search-term
find next match:	n
find previous match:	N
quit:	q

Extracting Informations from Files

grep – Find lines matching a pattern in textfiles

Grep is a command-line utility for searching plain-text data sets for lines matching a regular expression.

Usage: `grep [options] pattern file(s)`

```
> grep -i ensembl P04637.txt
DR   Ensembl; ENST00000269305; ENSP00000269305; ENSG00000141510.
DR   Ensembl; ENST00000359597; ENSP00000352610; ENSG00000141510.
DR   Ensembl; ENST00000419024; ENSP00000402130; ENSG00000141510.
DR   Ensembl; ENST00000420246; ENSP00000391127; ENSG00000141510.
DR   Ensembl; ENST00000445888; ENSP00000391478; ENSG00000141510.
DR   Ensembl; ENST00000455263; ENSP00000398846; ENSG00000141510.
```

Useful options:

-v	Print lines that do not match
-i	Search case-insensitive
-l	List files with matching lines, not the lines itself

-L	List files without matches
-c	Print count of matching lines for each file

head – Print first lines of a textfile

Head is a program on Unix and Unix-like systems used to display the beginning of a text file or piped data.

Usage: `head [options] file(s)`

```
> head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
```

Useful options:

-n num	Print num lines (default is 10)

tail – Print last lines of a textfile

Usage: `tail [options] file(s)`

```
> tail -n 3 /etc/passwd
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
gdm:x:42:42::/var/gdm:/sbin/nologin
sabayon:x:86:86:Sabayon user:/home/sabayon:/sbin/nologin
```

Useful options:

-n num	Print num lines (default is 10)
-f	"Follow" a file (print new lines as they are written to the file)

Useful Filetools

file – determine the filetype

Usage: `file [options] file(s)`

```
> file /bin/date
/bin/date: ELF 32-bit LSB executable
> file /bin
/bin: directory
> file SRC_HUMAN.fasta
SRC_HUMAN.fasta: ASCII text
```

Note: The command file uses certain tests and some magic to determine the type of a file

---

which – find a (executable) command

Usage: `which [options] command(s)`

```
> which date
/bin/date
> which eclipse
/usr/bin/eclipse
>
```

find – search/find files in any given directory

Usage: `find [starting path(es)] [search filter]`

```
> find /etc
/etc
/etc/printcap
/etc/protocols
/etc/xinetd.d
/etc/xinetd.d/ktalk
[...]
>
```

`find` is a powerful command with lots of possible search filters. Refer to the manpage for a complete list.

Examples:

- Find by name:

```
> find . -name SRC_HUMAN.fasta
./SRC_HUMAN.fasta
```

- Find by size: (List those entries in the directory /usr/bin that are bigger than 500kBytes)

```
> find /usr/bin -size +500k
/usr/bin/oparchive
/usr/bin/kiconedit
/usr/bin/opjitconv
[...]
```

- Find by type (d=directory, f=file, l=link)

```
> find . -type d
.
./adir
```

Useful Terminal Tools

clear – Clear the "screen"

Usage: `clear`

```
> clear
```

In case the output of the terminal/screen gets cluttered, you can use `clear` to clear the screen...

If this doesn't work, you can use `reset` to perform a re-initialization of the terminal:

reset – Reset your terminal

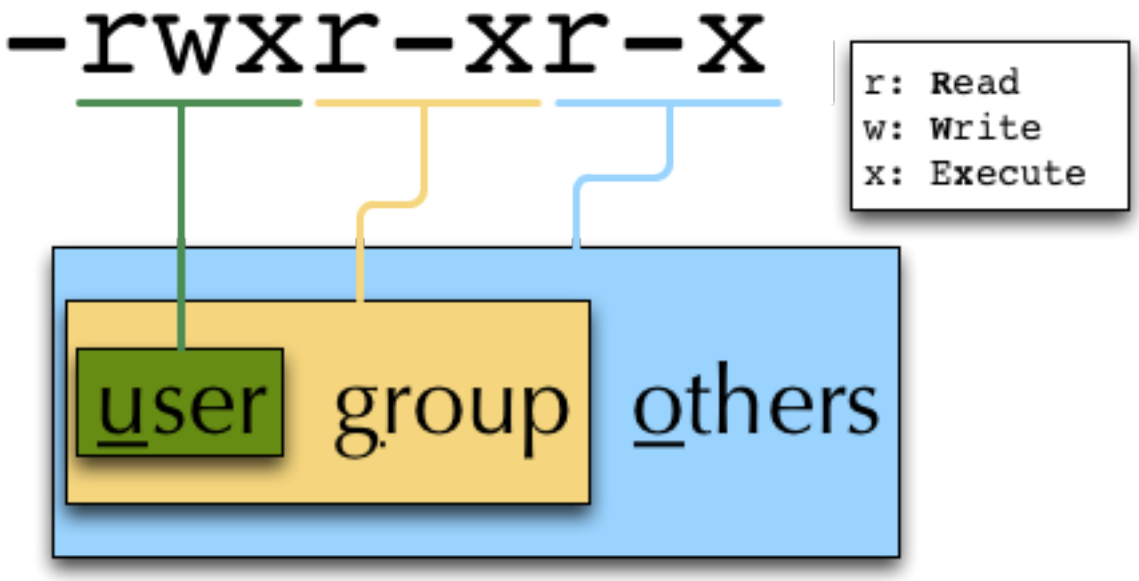
Usage: `reset [options]`

```
> reset
```

Permissions

using ls -l to view entries of current directory:

```
> ls -l
drwxr-xr-x 2 dinkel gibson 4096 Sep 17 10:46 adir
lrwxrwxrwx 1 dinkel gibson   15 Sep 17 10:45 H1.fasta -> H2.fasta
-rw-r--r-- 1 dinkel gibson  643 Sep 17 10:45 H2.fasta
```



Changing Permissions

Permissions are set using the `chmod` (change mode) command. **Usage**: `chmod [options] mode(s) files(s)`

```
> ls -l adir
drwxr-xr-x 2 dinkel gibson 4096 Sep 17 10:46 adir
> chmod u-w,o=w adir
> ls -l adir
dr-xr-x-w- 2 dinkel gibson 4096 Sep 17 10:46 adir
```

The mode is composed of

Who		What		Which permission	
u:	user/owner	+:	add this permission	r:	read
g:	group	-:	remove this permission	w:	write
o:	other	=:	set exactly this permission	x:	execute
a:	all | xx		xx	xx	xx

Add executable permission to the group:

```
> chmod g+x file
```

Revoke this permission:

```
> chmod g-x file
```

Allow all to read a directory:

```
> chmod a+rx adir/
```

Remote access

To execute commands at a remote machine/server, you need to log in to this machine. This is done using the `ssh` command (secure shell). In its simplest form, it takes just the machinename as parameter (assuming the username on the local machine and remote machine are identical):

```
> ssh remote_server
```

Note: Once logged in, use `hostname, whoami,` etc. to determine on which machine you are currently working!

To use a different username, you can use either:

```
> ssh username@remote_server
```

or

```
> ssh -l username remote_server
```

When connecting to a machine for the first time, it might display a warning:

```
> ssh sub-master
The authenticity of host 'sub-master (10.11.4.84)' can't be established.
RSA key fingerprint is 47:a4:0f:7b:c2:0f:ef:91:8e:65:fc:3c:f7:0c:53:8d.
Are you sure you want to continue connecting (yes/no)?
```

Type *yes* here. If this message appears a second time, you should contact your IT specialist...

To disconnect from the remote machine, type:

```
> exit
```

Copying files to and from remote computers can be done using `scp` (secure copy). The order of parameters is the same as in `cp`: first the name of the source, then the name of the destination. Either one can be the remote part.

```
> scp localfile server:/remotefile

> scp server:/remotefile localfile
```

An alternative username can be provided just as in ssh:

```
> scp username@server:/remotefile localfile
```

IO and Redirections

Redirect

Redirect the output of one program into e.g. a file: (Caution: you can easily overwrite files by this!) Inserting the current date into a new file:

```
> date > file_containing_date
```

Filtering lines containing the term "src" from FASTA files and inserting them into the file lines_with_src.txt:

```
> cd /exercises/
> grep -i "src" *.fasta > lines_with_src.txt
```

Append

Append something to a file (rather than overwriting it):

```
> date >> file_containing_date
```

Pipe

Use the | pipe symbol (`|`) to feed the output of one program into the next program. Here: use `ls` to show the directory contents and then use `grep` to only show those that contain fasta in their name:

```
> cd /exercises
> ls | grep fasta
EPSINS.fasta
FYN_HUMAN.fasta
P12931.fasta
SRC_HUMAN.fasta
```

Environment Variables

Environment variables are a set of dynamic named values that can affect the way running processes will behave on a computer.

$HOME

Contains the location of the user's home directory. Although the current user's home directory can also be found out through the C functions `getpwuid` and `getuid`, $HOME is often used for convenience in various shell scripts (and other contexts).

Note: Do not change this variable unless you have a good reason and you know what you are doing!

$PATH

$PATH contains a colon-separated list of directories that the shell searches for commands that do not contain a slash in their name (commands with slashes are interpreted as file names to execute, and the shell attempts to execute the files directly).

$PAGER

The $PAGER variable contains the path to the program used to list the contents of files through (such as `less` or `more`).

$PWD

The $PWD variable points to the current directory. Equivalent to the output of the command `pwd` when called without arguments.

Displaying environment variables:

Use `echo` to display individual variables *set'* or `env` to view all at once:

```
> echo $HOME
/localhome/teach01
> set
...
> env
...
```

Setting an environment variable:

Use `export` followed by the variable name and the value of the variable (separated by the equal sign) to set an environment variable:

```
> export PAGER=/usr/bin/less
```

Note: An environment variable is only valid for your current session. Once you logout of your current session, it is lost or reset.

CHAPTER

TWO

EXERCISES

2.1 Misc. file tools

1. Which tool can be used to determine the type of a file?
2. Use it on the following files/directories and compare the results:
   (a) `/usr/bin/tail`
   (b) ~
   (c) `/exercises/SRC_HUMAN.fasta`

2.2 Searching

1. Which tool can be used to search for files or directories?
2. Use it to find all directories in the `/exercises` directory
3. Search for the file date in the `/bin` directory
4. List those entries in the directory `/bin` that are bigger than 400kBytes

2.3 Misc. terminal

1. Which two tools can be used to redraw/empty the screen?

2.4 Permissions

1. Create a directory called testpermissions
2. Change your working directory to testpermissions
3. Create a directory called adir.
4. Use the command `which date` to find out where the date program is located.
5. Copy this date program into the directory adir.
6. Check the permissions of the copied program date
7. Change the permissions on date to remove the executable permissions.

8. Check the permissions of the program date
9. Try running it as `./date` or `adir/date` (depending on your current working directory)
10. Change the permissions back so that the file is executable.
11. Try running it as `./date` or `adir/date` (depending on your current working directory)
12. Copy a textfile from a previos exercise into adir, then change the permissions, so you are not allowed to write to it.
13. Then change the permissions so you can't read/cat it either.
14. Change your working directory to testpermissions, and then try changing the permissions on adir.
15. What are the minimum permissions (on the directory) necessary for you to be able to execute `adir/date`?

2.5 Remote access

1. Login to machine "sub-master.embl.de" (using your own username)
2. Use exit to quit the remote shell (Beware to not exit your local shell)
3. Use clear to empty the screen after logout from the remote server
4. Use the following commands locally as well as on the remote machine to get a feeling for the different machines:

1. `hostname`
2. `whoami`
3. `cat /etc/hostname`
4. `ls -la ~/`

1. Copy the file `/etc/motd` from machine sub-master.embl.de into your local home directory
2. Determine the filetype and the permissions of the file that you just copied
3. Login to your neighbor's machine (ask him for the hostname) using the username `teach01` (password will be given by teacher)

2.6 IO and Redirections

1. Use date in conjuction with the redirection to insert the current date into the (new) file current_date (in your homedirectory).
2. Inspect the file to make sure it contains (only a single line with) the date.
3. Use `date` again to append the current date into the same file.
4. Again, check that this file now contains two lines with dates.
5. Use `grep` to filter out lines containing the term "TITLE" from all PDB files in the exercises directory and use redirection to insert them into a new file pdb_titles.txt.
6. (OPTIONAL) Upon inspection of the file pdb_titles.txt, you see that it also contains the names of the files in which the term was found. Use either the grep manpage or `grep --help` to find out how you can suppress this behaviour. Redo the previous exercise such that the output file pdb_titles.txt only contains lines starting with TITLE.

CHAPTER

THREE

FILE FORMATS

File Types to research and present:

- Fasta/fastq (sanger and illumina)/fai
- GFF/GTF
- Sam/Bam/Bai
- Vcf/bcf/Pileup/interval/ROD
- Bed/bigBed
- Wig/bigWig

Questions you could answer in your presentation (5 – 7 min)

- **What does each line in the file represent?**
- **What does each column represent?**
- **What character separates the columns? Can you write this character?**
- **Is there a file header?**
- **What online repositories are available to download this data and how can I download it?**
- What type of user/database/online tools uses this file format/who came up with it?
- Show an example of the file.
- What is the difference between the files you've been given?
- Can you convert between one file format and another?
- Any other information interesting information?

Note: Questions in bold should be asked about any file format you have been given.

Unsure about jargon? Ask for help!

3.1 Useful links about file formats

3.1.1 VCF:

http://www.1000genomes.org/wiki/analysis/variant-call-format/vcf-variant-call-format-version-42

3.1.2 GFF:

- http://www.ensembl.org/info/website/upload/gff.html
- http://www.sanger.ac.uk/resources/software/gff/spec.html

3.1.3 BED/BIGBED/WIG/BIGWIG:

- http://genome.ucsc.edu/FAQ/FAQformat.html#format1
- http://www.ensembl.org/info/website/upload/bed.html
- http://www.ensembl.org/info/website/upload/wig.html

3.1.4 SAM/BAM/BAI:

- http://genome.sph.umich.edu/wiki/SAM
- http://samtools.sourceforge.net/SAMv1.pdf
- http://blog.nextgenetics.net/?e=18 = bitwise flags

3.1.5 FASTA/FASTQ:

- http://en.wikipedia.org/wiki/FASTQ_format#Illumina_sequence_identifiers
- http://bioinf.comav.upv.es/courses/sequence_analysis/sequence_file_formats.html
- http://blog.nextgenetics.net/?e=33 = phred quality

3.1.6 All files!

http://www.broadinstitute.org/igv/FileFormats http://genome.ucsc.edu/FAQ/FAQformat

3.1.7 NEXT STEPS

Look at the databases mentioned in the presentations and go through them in a live demo. Typical will be Ensembl, UCSC, NCBI, GATK, 1000 genomes

SAM very important – re-enforce QUAL score, CIGAR string, FLAG Quality scores – make reference to illumina vs sanger (Sanger pretty standard now)

CHAPTER

FOUR

APPENDIX

4.1 Literature

4.1.1 Algorithms

- Bowtie: Langmead B, Trapnell C, Pop M, Salzberg SL. "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome", Genome Biology <http://genomebiology.com/2009/10/3/R25>.
- Tophat: Trapnell C, Pachter L, Salzberg SL, TopHat: discovering splice junctions with RNA-Seq, Bioinformatics, 25(9):1105-1111 <http://bioinformatics.oxfordjournals.org/cgi/content/abstract/btp120>.
- Cufflinks: Trapnell C, Williams BA, Pertea G, Mortazavi AM, Kwan G, van Baren MJ, Salzberg SL, Wold B, Pachter L.Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation <http://dx.doi.org/10.1038/nbt.1621> Nature Biotechnology
- BWA: Li et al " Fast and accurate short read alignment with Burrows-Wheeler transform" http://bioinformatics.oxfordjournals.org/content/25/14/1754.abstract

4.1.2 File Manipulation tools

- Samtools - http://samtools.sourceforge.net/
- Picard - http://picard.sourceforge.net/index.shtml
- FastQC - http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
- HTSeq - http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

4.1.3 Data Standards

- http://encodeproject.org/ENCODE/dataStandards.html
- Sam format - http://samtools.sourceforge.net/SAM1.pdf
- http://www.ebi.ac.uk/arrayexpress/
- http://www.ebi.ac.uk/ena/

4.1.4 Normalization

- Oshlack A. et al "From RNA-seq reads to differential expression results." Genome Biology
- Robinson and Oshlack "A scaling normalization method for differential expression analysis of RNA-seq data." Genome Biology
- http://www.biomedcentral.com/1471-2105/12/480/abstract - GC content normalization
- http://www.bepress.com/ucbbiostat/paper291/ - GC bias

4.1.5 Differential Expression

- Tarazona et al. Differential expression in RNA-seq: A matter of depth, 2011, Genome research 21 (12) p. 2213-23 <http://genome.cshlp.org/cgi/content/abstract/gr.124321.111v1>
- Anders and Huber, "Differential expression analysis for sequence count data", 2010, Genome Biology, 11 (10) p. R106 <http://genomebiology.com/2010/11/10/R106>
- Robinson et al, "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data", 2010, Bioinformatics (Oxford, England) 26 (1) p. 139-40 < http://bioinformatics.oxfordjournals.org/content/26/1/139.long>

INDEX