Unix/Linux Tutorial for Beginners
Session VI

Mihaela Martis

NBIS & Faculty of Medicine and Health Sciences
Division Cell Biology, IKE

NBIS NATIONAL BIOINFORMATICS INFRASTRUCTURE SWEDEN

elixir

LINKÖPING UNIVERSITY

How to find files?

- find – finds recursively files matching a certain search pattern in a directory hierarchy
- basic syntax: find *<path> <search-term>*
- the search can be limited to a few directories deep → -maxdepth
  - -maxdepth 1 – search only within the current directory
  - -maxdepth 2 – search within the current directory and its subdirectories

Example

```
$ find ~/data/ -name "*fastq*"
~/data/fastq
~/data/fastq/zeaMays/SRR2889759_1.fastq.gz
~/data/fastq/zeaMays/SRR2889754_2.fastq.gz
~/data/fastq/zeaMays/SRR2889759_2.fastq.gz
~/data/fastq/zeaMays/SRR2889754_1.fastq.gz
~/data/fastq/SRR494009_1.fastq.gz
~/data/fastq/SRR494009_2.fastq.gz
~/data/fastq/DRR001013.fastq
```

```
$ find ~/data/ -maxdepth 1 -name "*fastq*"
~/data/fastq
```

```
$ find ~/data/ -maxdepth 2 -name "*fastq*"
~/data/fastq
~/data/fastq/SRR494009_1.fastq.gz
~/data/fastq/SRR494009_2.fastq.gz
~/data/fastq/DRR001013.fastq
```

find expressions and operators

- -name <pattern> – match a filename to <pattern>
- -empty – matches empty files or directories
- -type <d|f|l> – matches only directories | files | links
- '!'expression – negation
- expression -and expression – logical 'and'

Example negation

```
$ find ~/data/ -type f '!' -name '*fastq*'
~/data/fasta/barley_CDS.fa
~/data/fasta/brachy_CDS.fa
~/data/fasta/wheat_PEP.fa
~/data/fasta/ZMpep.bz2
~/data/fasta/subset/myCDS.fa
~/data/fasta/subset/brachy_CDS_subset.fa
~/data/fasta/subset/barley_subset_CDS.fa
~/data/fastq/myFirstFastQCrun.fq
~/data/gff/Pygoscelis_adeliae.gff
~/data/others/animals.txt
~/data/others/employees.txt
~/data/others/salmon.txt
~/data/others/ATH_GO_GOSLIM.txt
~/data/others/Annas_shoppingsList.txt
~/data/others/new_employees.txt
~/data/others/ids.txt
~/data/others/Peters_shoppingsList.txt
~/data/poetry/theSickRoseWilliamBlake.txt
~/data/poetry/rosesRobertBurns.txt
~/data/poetry/aWhiteRoseJohnBoyleOreily.txt
~/data/vcf/mySample_somaticMutations.vcf
```

Example multiple conditions

```
$ find ~/data/ -type f -name '*fastq*' -or -name '*fa'
~/data/fasta/barley_CDS.fa
~/data/fasta/brachy_CDS.fa
~/data/fasta/wheat_PEP.fa
~/data/fasta/subset/myCDS.fa
~/data/fasta/subset/brachy_CDS_subset.fa
~/data/fasta/subset/barley_subset_CDS.fa
~/data/fastq/zeaMays/SRR2889759_1.fastq.gz
~/data/fastq/zeaMays/SRR2889754_2.fastq.gz
~/data/fastq/zeaMays/SRR2889759_2.fastq.gz
~/data/fastq/zeaMays/SRR2889754_1.fastq.gz
~/data/fastq/SRR494009_1.fastq.gz
~/data/fastq/SRR494009_2.fastq.gz
~/data/fastq/DRR001013.fastq
```

Search by type

- find all directories in the current working directory

```
$ find . -type d
./
./data
./thesis
./tools
./tools/old
```

Search by type

- find all directories in the current working directory

```
$ find . -type d
./
./data
./thesis
./tools
./tools/old
```

- find all files in the current directory

```
$ find . -type f
./haiku.txt
./tools/stats
./tools/old/oldtool
./thesis/empty-draft.md
```

Search by type

- find all directories in the current working directory

```
$ find . -type d
./
./data
./thesis
./tools
./tools/old
```

- find all files in the current directory

```
$ find . -type f
./haiku.txt
./tools/stats
./tools/old/oldtool
./thesis/empty-draft.md
```

Search by type

- find all directories in the current working directory

```
$ find . -type d
./
./data
./thesis
./tools
./tools/old
```

- find all files in the current directory

```
$ find . -type f
./haiku.txt
./tools/stats
./tools/old/oldtool
./thesis/empty-draft.md
```

Search by type

- find all directories in the current working directory

```
$ find . -type d
./
./data
./thesis
./tools
./tools/old
```

- find all files in the current directory

```
$ find . -type f
./haiku.txt
./tools/stats
./tools/old/oldtool
./thesis/empty-draft.md
```

- it goes automatically into subdirectories to find everything that matches the given pattern

Search by name

- find all '.txt' files in the current directory

```
$ find . -name '*.txt'
./animals.txt
./salmon.txt
./test.txt
./groceries.txt

$ find /home/duck/exercises/ -name '*.pdf'
/home/duck/exercises/bootcamp.pdf
/home/duck/exercises/Linux_worshop.pdf
/home/duck/exercises/Lecture.pdf
```

Ignore the case or invert match

- to ignore the case use -iname instead of -name

```
$ find . -iname "*.TxT"
./animals.txt
./salmon.txt
./test.txt
./groceries.txt
```

Ignore the case or invert match

- to ignore the case use -iname instead of -name

```
$ find . -iname "*.TxT"
./animals.txt
./salmon.txt
./test.txt
./groceries.txt
```

- search for files that do not match a given name or pattern

```
$ find . -not -name "*.txt"
./thesis.pdf
./screen.png
$ find . ! -name "*.txt"
./thesis.pdf
./screen.png
```

Combine find with other commands

- tools can be combined by using pipes ('|') or using '$()'
- combine wc and find to count the lines in all files

```
$ wc -l $(find . -name '*.txt')
  8 ./animals.txt
  6 ./salmon.txt
  0 ./test.txt
 11 ./groceries.txt
 25 total
```

Combine find with other commands

- tools can be combined by using pipes ('|') or using '$()'
- combine wc and find to count the lines in all files

```
$ wc -l $(find . -name '*.txt')
  8 ./animals.txt
  6 ./salmon.txt
  0 ./test.txt
 11 ./groceries.txt
 25 total
```

- combine grep and find to find all lines starting with 'b'

```
$ grep "^b" $(find . -name '*.txt')
./groceries.txt:butter
./groceries.txt:buttermilk
./groceries.txt:banana
```

File compression

- reduces the space necessary to store data
- increases data transfer speed
- 2 types:
  - *lossless compression* $\rightarrow$ allows the original data to be perfectly reconstructed from the compressed data.
    Many different formats: gzip, bzip2, ...
  - *lossy compression* $\rightarrow$ allows the original data to be only approximately reconstructed from the compressed data.
    Popular for compressing media formats: JPEG, MP3, MPEG-4, ...

gzip compression

- 'classic' method of compressing data using the DEFLATE algorithm
- advantages: fast, resource efficient in term of memory usage during compression and decompression
- disadvantage: it compresses data less thoroughly than other
- use it if you need a lot of quick compressions and decompressions
- file ending .gz
- compression:

```
$ gzip sourcefile
$ gzip -r directory
$ ls
sourcefile.gz directory.gz
```

- decompression: use the -d flag

```
$ gzip -d sourcefile.gz
```

bzip2 compression

- Burrows-Wheeler algorithm
- greater compression at the cost of longer compression time
- memory requirements are greater than *gzip*
- file ending .bz2
- compression:

```
$ bzip2 sourcefile
$ ls
sourcefile.bz2
```

- decompression: use the -d flag

```
$ bzip2 -d sourcefile.bz2
```

tar

- collects many files into one archive file, uncompressed

```
$ tar cvf archive_name.tar dirname/
```

c – create a new archive

v – verbosely list files which are processed

f – following is the archive file name

tar

- collects many files into one archive file, uncompressed

```
$ tar cvf archive_name.tar dirname/
```

c – create a new archive

v – verbosely list files which are processed

f – following is the archive file name

- is often paired with *gzip*, *bzip2* to compress archives of files

```
$ tar czvf archive_name.tar.gz dirname/

$ tar cvfj archive_name.tar.bz2 dirname/
```

z – filter the archive through gzip

j – filter the archive through bzip2

Extracting an archive using tar

- extract an uncompressed tar archive → x

```
$ tar xvf archive_name.tar
```

Extracting an archive using tar

- extract an uncompressed tar archive → x

```
$ tar xvf archive_name.tar
```

- extract a tar archive with gzip or bzip2 compression

```
$ tar xzvf archive_name.tar.gz

$ tar xjvf archive_name.tar.bz2
```

Listing an archive using tar

- view the tar archive file content without extracting the content

```
$ tar tvf archive_name.tar

$ tar tzvf archive_name.tar.gz

$ tar tjvf archive_name.tar.bz2
```

Adding a file to an existing archive

- r – add additional files to existing, uncompressed tar archive; overwrites if the files already exists

```
$ tar rvf archive_name.tar new_file.txt

$ tar rvf archive_name.tar newDir/
```

- u – add a file to an existing archive, but only if the file is not already there or if the file is newer than that in archive

```
$ tar uvf archive_name.tar new_file.txt
```

Ownership & Permissions

- multi-user environment
- problem: keep people from editing or executing other people files
- solution: assign different owner and permissions to files and directories

Owners

- users, groups, and others
- user → name of the person who owns the file
- group → group of users that co-own the file (useful in projects)
- others → everyone else

Permissions

- four permissions that a file/directory can have:
  - read (r) – file can be opened and read; contents of directories can be listed
  - write (w) – file can be modified; contents of directories can be renamed or deleted
  - execute (x) – file can be executed as a program or shell script; directories can be entered using 'cd'
  - no permission (-)

Interpreting the permissions

```
$ ls -lh
-rw-rw-r-- 1 mihaelmr mihaelmr 31M 12. Okt 22:18 barley_CDS.fa
-rw-rw-r-- 1 mihaelmr mihaelmr 41M 11. Okt 13:59 brachy_CDS.fa
-rw-rw-r-- 1 mihaelmr mihaelmr 27M 11. Okt 18:26 wheat_PEP.fa
```

- -rw-rw-r– – describes the files permission

  'It is a regular file. The user has read & write permission, but not execute. The group is allowed to read and write, but not execute. Everyone else (other) has read permission but not write & execute.'

```
$ ls -lh
drwxrwsr-x 2 mihaelmr mihaelmr 4,0K 12. Okt 22:18 fasta
drwxrwsr-x 2 mihaelmr mihaelmr 4,0K 11. Okt 18:19 others
drwxrwsr-x 2 mihaelmr mihaelmr 4,0K 11. Okt 18:44 poetry
```

- drwxrwxr-x – describes the directory permission

  'It is a directory in which all user groups have execute permissions, to enter the directory by default. Others are not allowed to write in the directory.'

Interpreting the permissions II

d	rwx	r-x	rwx
-	rw-	rw-	r–

→ user (u), group (g), others (o)
→ **'d'** – directory or '-' if file
→ **'r'** – read
→ **'w'** – write
→ **'x'** – execute
→ '-' – no permission

Changing file permissions

- chmod – changes the permissions of files or directories
- only the owner can set file permissions
  1. for which group are the permissions: user (u), group (g), other (o), all (a)
  2. define if the permissions should be added or removed
     - + – add permissions
     - - – remove previous permissions
     - = – add new permissions
  3. specify the kind of permissions: **r, w, x, -**

chmod - Examples

- make a file executable by the owner (u) and members of the group (g)

```
$ ls -l animal.pl
-rw-rw-r-- 1 duck duck 136 8nov 16.28 animal.pl

$ chmod ug+x animal.pl
$ ls -l animal.pl
-rwxrwxr-- 1 duck duck 136 8nov 16.28 anima.pl
```

- deny all ('a'), including the owner, the right to write to a file
  $\rightarrow$ this prevents accidental deletion

```
$ ls -l
-rw-rw-r-- 1 duck duck 512 11. Okt 18:44 myFile.txt

$ chmod a-w myFile.txt
$ ls -l
-r--r--r-- 1 duck duck 512 11. Okt 18:44 myFile.txt
```

Changing file permissions using digits

read = 4, write = 2
execute = 1, no permission = 0

```
$ chmod 555 myFile.txt

$ ls -l myFile.txt
-r-xr-xr-x 1 duck duck 512 11. Okt 18:44 myFile.txt
```

Summary

- use find to locate files and directories
- use chmod to change the file permissions
- compress large data to save disk space and speed up the file transfer
- use gzip, bzip2, or tar in combination with one of the 2 compression methods