1) Introduction to unix command line and perl

Matt Webster
IMBIM, BMC
matthew.webster@imbim.uu.se

Perl course details

- course book – Learning Perl (6 ed.)
- lectures cover chapters
- morning lectures + afternoon exercises
- UU user name and password C
- goal: exercises relevant to real tasks

- helpers: Andreas Wallberg, Erik Axelsson, Alvaro Martinez, Martin Dahlö

- me: matthew.webster@imbim.uu.se

Basic perl programming for biological sciences

Course textbook:
Learning Perl - 6th Edition (Randal L Schwartz, Brian D Foy, Tom Phoenix)

Preliminary Schedule

Date	Time	Room	Topics	Chapters in *Learning Perl (6th ed.)*
9 Oct (Tue)	09.00-12.00	A9:001	1) Introduction to unix command line and perl	1
10 Oct (Wed)	09.00-12.00	A7:114	2) Scalar data 3) Lists / arrays	2, 3
10 Oct (Wed)	13.15-17.00	A6:003 (datorsal)		
11 Oct (Thu)	09.00-12.00	A7:114	4) Subroutines 5) Input/output	4, 5
11 Oct (Thu)	13.15-17.00	A6:003 (datorsal)		
12 Oct (Fri)	09.00-12.00	A1:104b	6) Hashes 7) Regular expressions	6,7,8,9
12 Oct (Fri)	13.15-17.00	A6:003 (datorsal)		
15 Oct (Mon)	09.15-12.00	C8:317	8) Control structures 9) Strings & sorting	10,14
15 Oct (Mon)	13.15-17.00	A6:003 (datorsal)		
16 Oct (Tue)	13.15-17.00	A6:003 (datorsal)	10) Advanced topics: modules, file tests, directories	11,12,13 (+15,16,17)
17 Oct (Wed)	08.15-10.00	A6:003 (datorsal)	11) Bioinformatics exercises	-

2012-10-03
matthew.webster@imbim.uu.se

Why perl?

- Perl is an Open Source project
- Perl is a cross-platform programming language.

- Perl is a very popular programming language, especially for bioinformatics
- Perl allows a rapid development cycle
- Perl is strong in text manipulation
- Perl can easily handle files and directories
- Perl can easily run other programs

Why unix?

- Powerful command line tools
- doesn't waste computer resources on graphics
- designed for lots of small programs
  - can link programs together
- Easy to combine with scripts (e.g. perl), automation, queue systems
- Necessary for using servers like UPPMAX via ssh
- Particularly useful for manipulating large files
- Linux/Mac OS X terminal
- more precise than point and click
- Necessary for supercomputing

Bioinformatics ingredients

- unix command line
- perl scripts
- R (statistics)
- many other command line applications

- perl scripts are the glue:
  - manipulating large text files
  - changing file formats
  - summarizing data from huge files

unix basics

- Each word you type in the command line runs a program. So it is easy to add your own commands – just add, or write, another program.

- The output of the program is returned to the **terminal** unless you say otherwise. So all your interaction is through one text window. This makes it easy to log in **remotely.**

Best practice in unix

- don't use spaces in filenames, better to use _
- tab completion
- Unix is CaSe SeNsiTivE
- The extension (eg .txt, .fasta) can be any number of letters and is optional. It's for your own convenience so you know what kind of file is what.
- You can only have one file in the same directory with the same name. **If you make another one, the old file will be deleted. This is VERY easy to do.**

Navigating in directories

- `ls`
  - list contents of current working directory

- commands have options following –
  - e.g. `ls -a`
  - lists hidden files also

- `mkdir`
  - make a directory
  - `mkdir test_folder`

Navigating in directories

- `cd`
  - change directory
  - `cd test_folder`

- The current directory (.)
  - `cd .` stay where you are (does nothing)

- The parent directory (..)
  - `cd ..` go up one directory

Navigating in directories

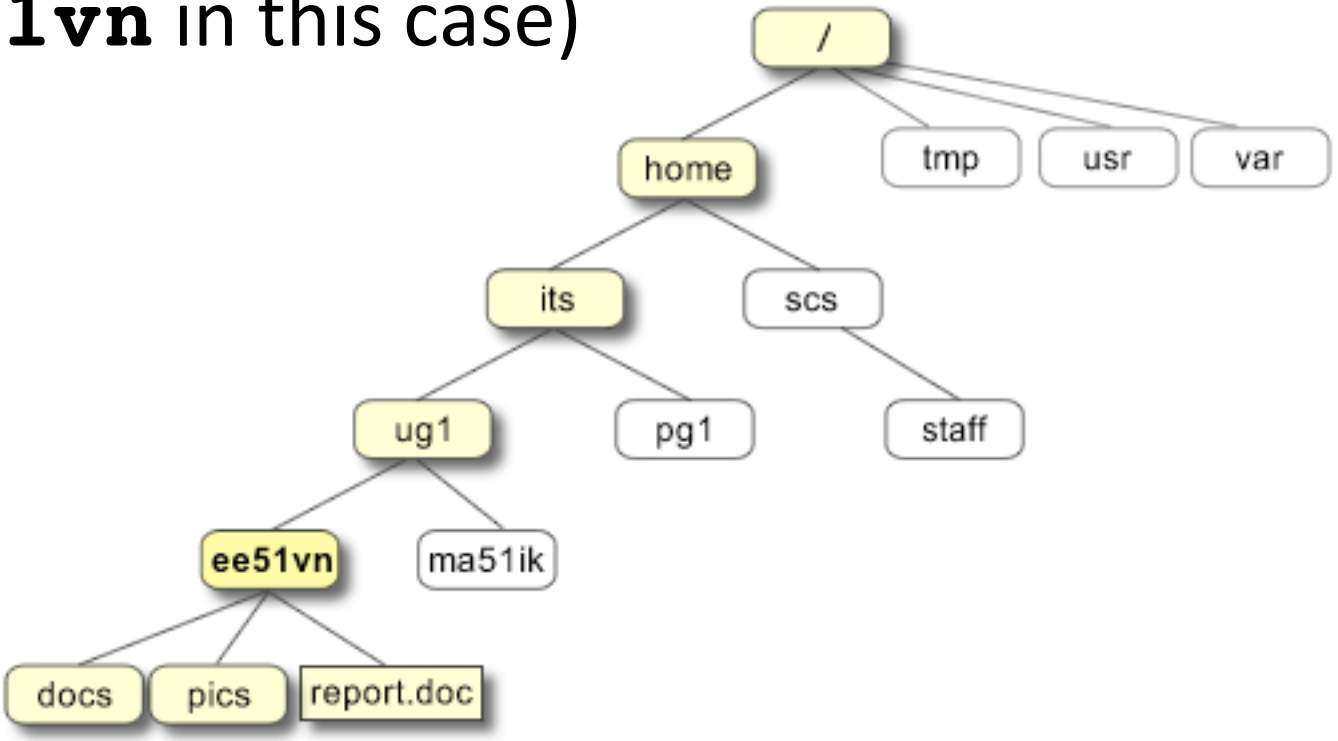
- `pwd`
  - print working directory
  - will give something like:
  - **`/home/its/ug1/ee51vn`**

- The top-level root directory is called " / "

Navigating in directories

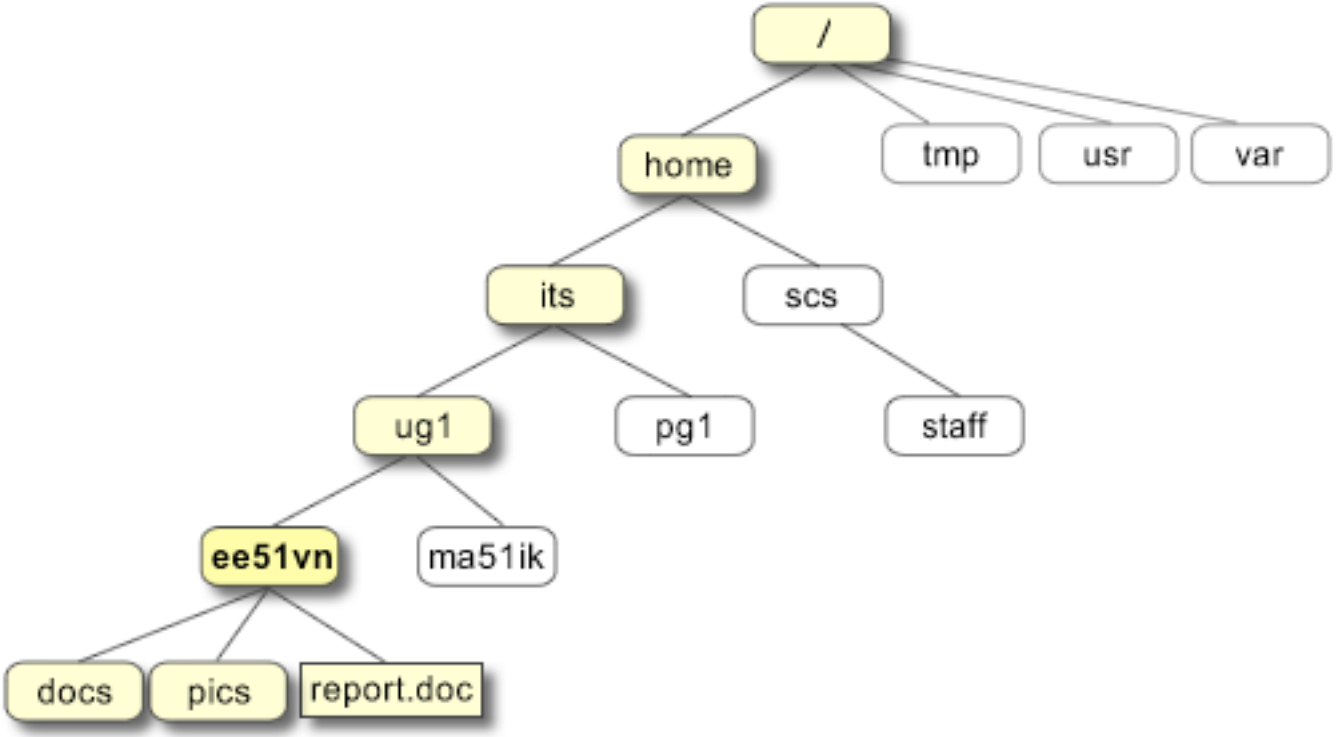
- `cd`, `ls` and `pwd` can be used to explore the file system
- `cd` (by itself) returns to home-directory

(**`ee51vn`** in this case)



Navigating in directories

- how to list **`report.doc`** from **`ma51ik`**?



path names

- how to list **`report.doc`** from **`ma51ik`**?

```
1)   cd ../ee51vn
     ls report.doc
2)   ls ../ee51vn/report.doc
3)   ls /home/its/ug1/ee51vn/report.doc
4)   ls ~/report.doc
```

" ~ " refers to your home directory

reminder: always use tab complete!

file manipulation

- `cp file1 file2`
  - copy a file
  - if file2 is " . " then it will copy to current directory
  - `cp /vol/examples/tutorial/science.txt .`

- `mv file1 file2`
  - move a file to a different directory
    - `mv example1.txt example_files/`
  - rename a file
    - `mv example1.txt example2.txt`

file manipulation

- `rm` *`file`*
  - remove (delete) a file
  - `rm example_files/example1.txt`

- `rmdir` *`directory`*
  - remove a directory
    - `rmdir example_files/`

displaying files

- `cat` *`file`*
  - print whole file on screen (keeps scrolling)
- `head` *`file`*
  - print first 10 lines
- `tail` *`file`*
  - print last 10 lines
- `less` *`file`*
  - interactive viewer (space bar for next page, q to quit)
- `pico` *`file`*
  - simple text editor

searching files

- `wc file`
  - word count
  - prints number of lines, words, characters
- `less file` followed by `/keyword`
  - similar to using "find" in word or web browser
- `grep 'keyword' file`
  - search a file for *'keyword'*

getting help

- `man` *`command`*
  - prints manual page for command
  - e.g. `man less` will print all different options within `less`

redirecting output

- to send output to a file rather than the screen, use " > "
- *e.g.* `cat example1.txt > example2.txt`
  - will print contents of `example1.txt` into a new file `example2.txt`
  - (same result as copying the file)

- to append to a file, use " >> "
- e.g. `cat example1.txt >> example2.txt`
  - if `example2.txt` already exists, it will now have double the contents of `example1.txt`

redirecting output

- to concatenate two files, use

```
cat example1.txt example2.txt > example3.txt
```

pipes

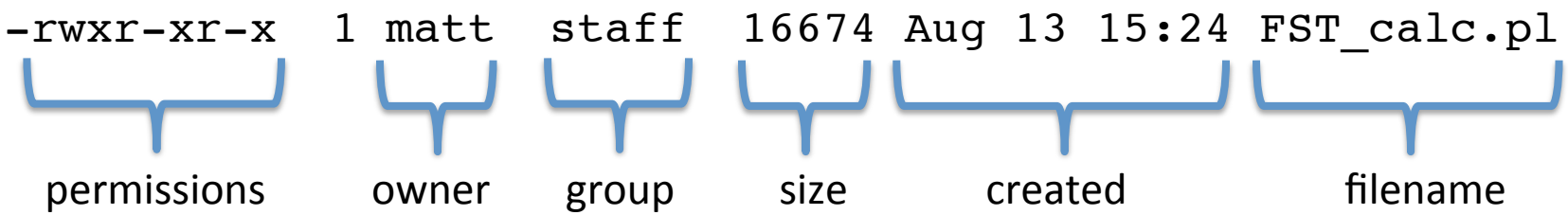
- to send output to another command rather than the screen, use " | "
- *e.g.* `cat example1.txt | wc`
  - will send contents of `example1.txt` to `wc`
  - (performs word count on file)

wildcards: * and ?

- `ls *.txt`
  - will list all files ending in `.txt`
- `ls ?ouse`
  - will list files like mouse and house

file permissions

- `ls -l`
  - long listing of files
  - includes additional information



for this file:

owner permissions	`rwx`
group permissions	`r-x`
all permissions	`r-x`

`r` = read
`w` = write
`x` = execute

changing file permissions

Symbol	Meaning
u	user
g	group
o	other
a	all
r	read
w	write (and delete)
x	execute (and access directory)
+	add permission
-	take away permission

changing file permissions

- `chmod +x file`
  - allow file to be executed by owner
- `chmod a+rw file`
  - add read and write permissions to all
- `chmod go-rwx file`
  - remove all permissions to group and others

processes

- `ps`
  - list processes you are running
- `top`
  - view all processes in real time

processes

- `^C (control-C)`
  - kill a process running in the foreground
- `^Z`
  - suspend a process running in the foreground
- `bg`
  - start running the suspended process in the background
- use " & " to start a process in the background
  - e.g. `cat bigfile1 bigfile2 > bigfile3 &`

to kill a background process

- use `ps` to find the process number (PID)
  - `ps`
  - `kill` *`PID`*
- if it doesn't work
  - `kill -9` *`PID`*

other useful commands

- `df`
  - how much space on the server?
- `du`
  - how many kb in each subdirectory
- `gzip` *`file`*
  - compress a file into a `.gz` file
- `gunzip` *`file`*
  - expand a `.gz` file
- `diff file1 file2`
  - compare *file1* and *file2*

operations with a server

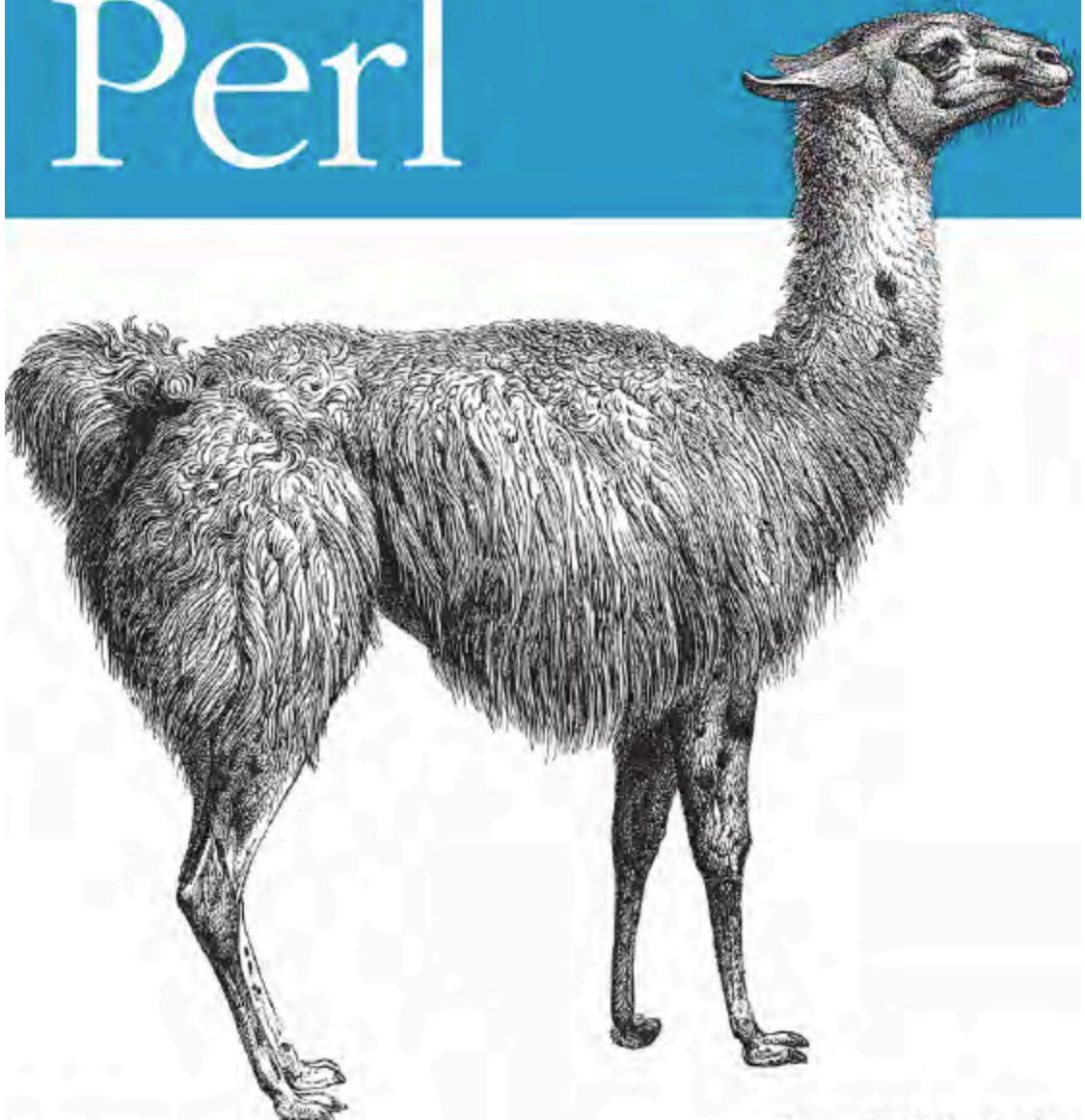
- `ssh username@kalkyl.uppmax.uu.se`
  - log in to kalkyl

- `scp username@kalkyl.uppmax.uu.se:glob/example.txt .`
  - copy `glob/example` from kalkyl to where you are currently logged in

Logging out

```
exit
```

Learning Perl



perl basics

- you don't COMPILE perl scripts
- perl scripts are compiled when you run them
- a perl script is just a text file

how to write a perl script

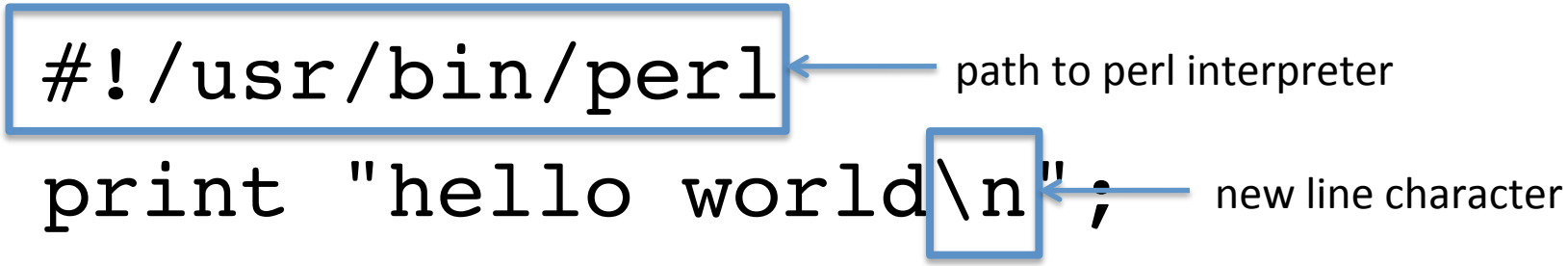
- use any text editor
- anything that can save a plain text file
  - TextEdit
  - pico, nano in terminal
- more useful ones have syntax highlighting
  - gedit
  - emacs
  - TextWrangler
  - many other options

some code:

```
#!/usr/bin/perl -w

use strict;

#lupa file generator

my $phen_file = "6set.pheno";
my $mark_file = "success_mark.txt";
my $geno_file = "Final_dat_LUPA.ped";
my $all_pref= "HD24_";

my %lupa_id;
my %outprefix;
my $lupa_prefix="cHD24_";
my %phenout;
my %pref_id;
my %affstat;
my %sex;

open IN,$phen_file or die;
chomp (my $line = <IN>);
$line=~s/[\t]+$//g;
my $firstline= $line."\tPATIENT\tSEX\tAFFSTAT";
my $i=1;
while (<IN>) {
    chomp (my $line = $_);
    $line=~s/[\t]+$//g;
```

every line ends with a semicolon

your first perl script



save as "example.pl"

two ways to run example.pl

1) `perl example.pl`

2) `chmod +x example.pl`
   `./example.pl`

`chmod +x` makes it executable and is only needed once

```
#!/usr/bin/perl -w
use strict;

#lupa file generator

my $phen_file = "6set.pheno";
my $mark_file = "success_mark.txt";
my $geno_file = "Final_dat_LUPA.ped";
my $all_pref= "HD24_";

my %lupa_id;
my %outprefix;
my $lupa_prefix="cHD24_";
my %phenout;
my %pref_id;
my %affstat;
my %sex;

open IN,$phen_file or die;
chomp (my $line = <IN>);
$line=~s/[\t]+$//g;
my $firstline= $line."\tPATIENT\tSEX\tAFFSTAT";
my $i=1;
while (<IN>) {
    chomp (my $line = $_);
    $line=~s/[\t]+$//g;
    my @data = split ("\t",$line);
    my $id=$lupa_prefix.$i;
    my $previd=$data[7];
    $previd=~s/ //g;
    $previd=~s/\-/_/g;
    $previd=~s/\//_/g;
```

interpreter (topic 1)

string (topic 2)

hash (topic 6)

private variable (topic 4)

file handle (topic 5)

array (topic 3)

regular expression (topic 7)

Key topics in the course (1)

topic	title	chapters in *Learning Perl (6th ed.)*	key operators / functions
1	introduction to unix command line and perl	1	`print`

2	scalar data	2	`if`
			`chomp`
			`while`
			`<STDIN>`

3	lists / arrays	3	`push, pop`
			`shift, unshift`
			`splice`
			`foreach`
			`reverse`
			`sort`

4	subroutines	4	`sub`
			`return`
			`my`
			`use strict`

5	input/output	5	`printf`
			`open, close`
			`die`

6	hashes	6	`keys, values`
			`exists`
			`delete`

Key topics in the course (2)

topic	title	chapters in *Learning Perl (6th ed.)*	key operators / functions

7	regular expressions	7,8,9	`m//`
			`s///`
			`split`
			`join`

8	control structures	10	`unless`
			`else`
			`elsif`
			`for`
			`last`
			`next`

9	strings + sorting	14	`index`
			`substr`
			`sprintf`

10	advanced topics: modules, file tests, directories	11,12,13 (+15,16,17)	`use`
			`-e`
			`chdir`
			`glob`
			`unlink`
			`system`

11	bioinfomatics exercises		

summary (1): unix

`cd`

change directory

`ls`

list files

`mkdir`

make directory

`pwd`

print working directory

`cp` `mv`

copy and move files

`cat`

print contents of file

`head` `tail`

print start or end of file

`less`

interactively view file

summary (2): unix

`wc`

word count

`grep pattern`

find lines matching *pattern*

`man command`

print manual page for command

`>` `>>`

redirect output to new file or append to file

`|`

send output to command

`*` `?`

wildcards

`chmod`

change permissions

`ps` `top`

view processes, or interactively view processes

summary (3): unix

```
bg
```
run stopped process in background
```
command &
```
run *command* in background
```
df
```
how much space on the drive?
```
du directory
```
how much space used by *directory*
```
gzip   gunzip
```
compress or uncompress file with gzip
```
ssh user@server.uu.se
```
login to server
```
exit
```
logout

summary: first steps in perl

```
#!/usr/bin/perl
# the first line: the path to the perl interpreter
print
# prints to screen
;
# at the end of each line
```