

the Shell

Johan Montelius

KTH

2021

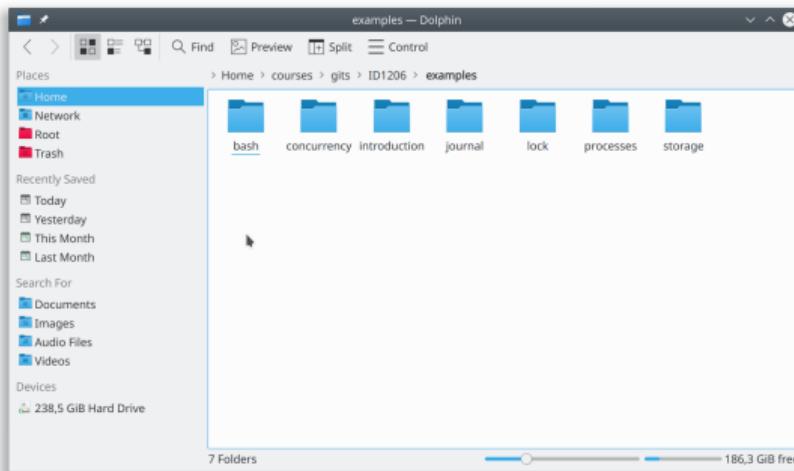
- the shell
- files and directories
- some tools: grep, wc, sed ...
- write a thesis: gcc, latex, gnuplot, make
- environment variables

the shell

```
Terminal File Edit View Search Terminal Help

johanmon:Orange:~$cd
johanmon:Orange:~$cd courses/ID2206/lectures/linux/
johanmon:Orange:~$ls
adm3a-keyboard.jpg      handout.snm          Makefile      slides.tex
Apple_IIe_keyboard-s.jpg handout.tex          mark.jpg      slides.toc
foo.txt                  handout.toc          mint.png      slides.vrb
handout.aux              handout.vrb          slides.aux    unity.png
handout.log              history-of-unix.png slides.log    windows10.png
handout.nav              hjkl.jpg            slides.nav    xubuntu.png
handout-nup.pdf          kubuntu.jpg          slides.out    slides.pdf
handout.out              lisp-machine-keyboard-2-left.jpg slides.snm
handout.pdf              lubuntu.png          slides.out    slides.snm
johanmon:Orange:~$make
make: Nothing to be done for 'all'.
johanmon:Orange:~$xpdf -fullscreen slides.pdf&
```

the file system



A screenshot of a terminal window titled "examples : bash — Konsole". The terminal shows the following command-line session:

```
johanmon@orange:~/courses/gits/ID1206/examples$ ls
johanmon@orange:~/courses/gits/ID1206/examples$ ls
johanmon@orange:~/courses/gits/ID1206/examples$ ls
bash concurrency introduction journal lock processes storage
johanmon@orange:~/courses/gits/ID1206/examples$ ls -l
total 28
drwxrwxr-x 8 johanmon johanmon 4096 nov 26 2018 bash
drwxrwxr-x 2 johanmon johanmon 4096 mar 4 17:27 concurrency
drwxrwxr-x 2 johanmon johanmon 4096 nov 15 2018 introduction
drwxrwxr-x 3 johanmon johanmon 4096 dec 10 2018 journal
drwxrwxr-x 2 johanmon johanmon 4096 nov 29 2018 lock
drwxrwxr-x 2 johanmon johanmon 4096 nov 13 2018 processes
drwxrwxr-x 3 johanmon johanmon 4096 dec 6 2018 storage
johanmon@orange:~/courses/gits/ID1206/examples$
```

the directory

Commands that you should to know:

Commands that you should to know:

- `ls` - list files and directories
- `mkdir` - make a directory
- `rmdir` - remove a directory
- `cd` - change directory
- `pwd` - path of working directory
- `touch` - touch a file
- `rm` - remove a file
- `mv` - move a file
- `cp` - copy a file
- `ln` - create a link (soft/hard) to a file
- `stat` - information about a file

The shell will *expand* any input, depending on files in the directory, before issuing command.

The shell will *expand* any input, depending on files in the directory, before issuing command.

- ~ precede by space - expands to home directory.

The shell will *expand* any input, depending on files in the directory, before issuing command.

- ~ precede by space - expands to home directory.
- * as in *.c - expands to a sequence of characters to matches files in the directory

The shell will *expand* any input, depending on files in the directory, before issuing command.

- ~ precede by space - expands to home directory.
- * as in *.c - expands to a sequence of characters to matches files in the directory
- ? as in f???.txt - expands to any single character

The shell will *expand* any input, depending on files in the directory, before issuing command.

- ~ precede by space - expands to home directory.
- * as in *.c - expands to a sequence of characters to matches files in the directory
- ? as in f???.txt - expands to any single character
- [06] as in ID120[06].pdf - expands one of the specified characters

The shell will *expand* any input, depending on files in the directory, before issuing command.

- ~ precede by space - expands to home directory.
- * as in *.c - expands to a sequence of characters to matches files in the directory
- ? as in f???.txt - expands to any single character
- [06] as in ID120[06].pdf - expands one of the specified characters
- \$ as in \$HOME - expands to the *variable* value (more on this later)

Expansion can be controlled by enclosing arguments in single quotes ' ', double quotes " " (variables will be expanded) or precede character by backslash \.

Some more or less simple ways to explore the content of a text file:

Some more or less simple ways to explore the content of a text file:

- `cat` - concatenate files
- `less` - less is of course more
- `head` - the beginning of a file
- `tail` - the end of a file
- `grep` - search a file for pattern
- `diff` - difference of two files

Some more or less simple ways to explore the content of a text file:

- cat - concatenate files
- less - less is of course more
- head - the beginning of a file
- tail - the end of a file
- grep - search a file for pattern
- diff - difference of two files
- sort - sort rows
- wc - word count
- uniq - remove duplicates
- tr - transpose char-by-char
- sed - stream editor
- awk - more powerful than sed

pipes and redirect

The shell can set a file as the standard input of a command or redirect the standard output and/or standard error.

pipes and redirect

The shell can set a file as the standard input of a command or redirect the standard output and/or standard error.

- < as in `wc < foo.txt` will set standard input.
- > as in `ls > out.txt` will set standard output.
- 2> as in `grep foo bar.txt 2> err.txt` will set standard error.

The power of the UNIX shell is the concept of *pipes*.

```
grep typedef foo.c | sort | uniq | less
```

pipes and redirect

The shell can set a file as the standard input of a command or redirect the standard output and/or standard error.

- < as in `wc < foo.txt` will set standard input.
- > as in `ls > out.txt` will set standard output.
- 2> as in `grep foo bar.txt 2> err.txt` will set standard error.

The power of the UNIX shell is the concept of *pipes*.

```
grep typedef foo.c | sort | uniq | less
```

Standard output of one command becomes standard input of the next command

Den bok jag nu sätter mig ner att skriva måste verka meningslös på många - om jag alls vågar tänka mig, att "många" får läsa den - eftersom jag alldeles självmant, utan någons order, börjar ett sådant arbete och åndå inte själv är riktigt på det klara med vad avsikten är.

Den bok jag nu sätter mig ner att skriva måste verka meningslös på många - om jag alls vågar tänka mig, att "många" får läsa den - eftersom jag alldeles självmant, utan någons order, börjar ett sådant arbete och åndå inte själv är riktigt på det klara med vad avsikten är.

2019 jag

1818 och

1505 att

1429 det

1045 i

979 en

:

from text to frequency list

Turn a raw text into and ordered frequency list.

- Remove special characters (.,?!;:-()") from text using sed or tr.

from text to frequency list

Turn a raw text into and ordered frequency list.

- Remove special characters (.,?!;-()") from text using sed or tr.
- Replace space by linefeed to turn the text into a list of words.

from text to frequency list

Turn a raw text into and ordered frequency list.

- Remove special characters (.,?!;:-()") from text using sed or tr.
- Replace space by linefeed to turn the text into a list of words.
- Sort the list using sort.

from text to frequency list

Turn a raw text into and ordered frequency list.

- Remove special characters (.,?!;-()") from text using sed or tr.
- Replace space by linefeed to turn the text into a list of words.
- Sort the list using sort.
- Remove duplicates but add frequency using uniq.

from text to frequency list

Turn a raw text into and ordered frequency list.

- Remove special characters (.,?!;-()") from text using sed or tr.
- Replace space by linefeed to turn the text into a list of words.
- Sort the list using sort.
- Remove duplicates but add frequency using uniq.
- Sort the result using sort.

Everything is of course connected using pipes.

from text to frequency list

Turn a raw text into and ordered frequency list.

- Remove special characters (.,?!;-()") from text using sed or tr.
- Replace space by linefeed to turn the text into a list of words.
- Sort the list using sort.
- Remove duplicates but add frequency using uniq.
- Sort the result using sort.

Everything is of course connected using pipes.

Experiment yourself, the devil is in the details.

Run the benchmark and save the result in a text file.

Run the benchmark and save the result in a text file.

Use gnuplot to produce a graph.

Run the benchmark and save the result in a text file.

Use `gnuplot` to produce a graph.

Write the thesis, including the graph, using `LATEX`.

Run the benchmark and save the result in a text file.

Use `gnuplot` to produce a graph.

Write the thesis, including the graph, using `LATEX`.

Set up a `Makefile` to automate the process.

gnuplot

gnuplot

gnuplot

- generate graphs from data in text file (tab separated)

gnuplot

- generate graphs from data in text file (tab separated)
- interactive or from script

gnuplot

- generate graphs from data in text file (tab separated)
- interactive or from script
- not a program for statistics (for statistics use R)

gnuplot

- generate graphs from data in text file (tab separated)
- interactive or from script
- not a program for statistics (for statistics use R)

pdflatex

pdflatex

- will let you focus on content

pdflatex

- will let you focus on content
- easy to include content from other files

pdflatex

- will let you focus on content
- easy to include content from other files
- generates pdf

make

make

make

- the work horse in any UNIX project

make

- the work horse in any UNIX project
- script will set up the dependencies between files

make

- the work horse in any UNIX project
- script will set up the dependencies between files
- will run programs as needed to produce final output i.e “make”

make

- the work horse in any UNIX project
- script will set up the dependencies between files
- will run programs as needed to produce final output i.e “make”
- used for programming as well as documentation

The shell maintains a set of variables that can be accessed from the shell, but not immediately from child processes.

shell variables

The shell maintains a set of variables that can be accessed from the shell, but not immediately from child processes.

- `set` - control the shell environment
- `<variable>=<value>` - defines a variable value
- `$<variable>` - access variable from shell
- `HOME` - home directory
- `PWD` - current directory
- `PATH` - paths searched when looking for executables
- `USER` - user name

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

- `export <variable>` - make variable accessible from child process

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

- `export <variable>` - make variable accessible from child process
- `printenv` - list all environment variables

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

- `export <variable>` - make variable accessible from child process
- `printenv` - list all environment variables
- `env` - run command in specified environment

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

- `export <variable>` - make variable accessible from child process
- `printenv` - list all environment variables
- `env` - run command in specified environment

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

- `export <variable>` - make variable accessible from child process
- `printenv` - list all environment variables
- `env` - run command in specified environment

Functions from standard library.

- `getenv()` - get the value of variable

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

- `export <variable>` - make variable accessible from child process
- `printenv` - list all environment variables
- `env` - run command in specified environment

Functions from standard library.

- `getenv()` - get the value of variable
- `putenv()` - set the value of variable

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

- `export <variable>` - make variable accessible from child process
- `printenv` - list all environment variables
- `env` - run command in specified environment

Functions from standard library.

- `getenv()` - get the value of variable
- `putenv()` - set the value of variable
- `execle()` - execute command in new environment

The *environment* is a set variables that can be accessed by programs using the standard library function call `getenv()`.

The *shell* will set up a set of *exported* variables that will be visible as environment variables when a child process is created.

- `export <variable>` - make variable accessible from child process
- `printenv` - list all environment variables
- `env` - run command in specified environment

Functions from standard library.

- `getenv()` - get the value of variable
- `putenv()` - set the value of variable
- `execle()` - execute command in new environment
- : - there are more

package - a set of source files and scripts

package - a set of source files and scripts

configure - check that everything is available, build Makefile

package - a set of source files and scripts

configure - check that everything is available, build Makefile

make - make, compile, environment variables define the target

package - a set of source files and scripts

configure - check that everything is available, build Makefile

make - make, compile, environment variables define the target

execute - execute, environment variables describe the session

package - a set of source files and scripts

configure - check that everything is available, build Makefile

make - make, compile, environment variables define the target

execute - execute, environment variables describe the session

- the shell - your interface to any UNIX system

- the shell - your interface to any UNIX system
- files and directories - learn to navigate the tree

- the shell - your interface to any UNIX system
- files and directories - learn to navigate the tree
- shell and environment variables

- the shell - your interface to any UNIX system
- files and directories - learn to navigate the tree
- shell and environment variables
- work with text file, connect sequences with pipes

- the shell - your interface to any UNIX system
- files and directories - learn to navigate the tree
- shell and environment variables
- work with text file, connect sequences with pipes

Do learn gnuplot, latex and make before starting your thesis.

- the shell - your interface to any UNIX system
- files and directories - learn to navigate the tree
- shell and environment variables
- work with text file, connect sequences with pipes

Do learn gnuplot, latex and make before starting your thesis.