### Directories and files

# Concepts

- Files are organised in a directory tree/hierarchy
- Everything is a file (e.g. keyboard, printers, ...)
- Each process has access to the files *stdin* (input), *stdout* (buffered output), *stderr* (unbuffered output)
- Each process operates in a working directory
- Each user has a home directory

#### **Paths**

Path = Identifier for the location of file/directory

- Paths consists of a parent directory list + file/directory
- Files and directories are separated by a '/
- Directory paths may contain a trailing '/'

Absolute path = Full location (first character = '/')

Relative path = Relative location (first character ≠ '/')

path to the directory itself

path to the directory itself
path to the parent directory

/usr/bin/ls example for an absolute file path
example for an absolute directory path
./a.out example for a relative file path

# File system hierarchy

/ Root directory

/bin Essential command executables

/dev Device files

/etc System-wide configuration files /sbin Essential administrative executables

/tmp Temporary files

/usr System resources for users /usr/bin Command executables

/usr/local Site-local data

/usr/sbin Administrative executables

/var Variable files

man hier (or man file-hierarchy on recent Linux
distributions) to get a more detailed overview

# Terminal (emulator)

Text terminal = Computer interface for text entry/display Terminal emulator = Application that emulates a text terminal in a graphical environment

Examples for terminal emulators: xterm, urxvt, guake

### Opening a terminal

### Shell

*Unix shell* = User interface that accepts commands to operate a computer

man intro to get an introduction into basic shell usage

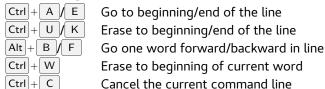
Examples for shell programs: sh, bash, zsh, fish, ksh

## **Prompt**

Prompt = Text sequence that precedes each line that
prompts the user to enter a command

[foo@bar /var/www]\$ example prompt in bash ⇒ user foo is operating in the working directory /var/www at the computer with the host name bar

# Line editing



#### Metacharacters

The following characters have special meaning and sometimes they can't be used directly as arguments/words:

Their special meaning can be disabled:

preserves the literal value of the following character
preserves the literal values of enquoted characters
like ' but characters ` \$ \ retain their meaning

#### Expansions

•	
~	home directory of the current user
*	matches any character sequence
?	matches a single character
[]	matches a character enclosed by the braces
\${ <i>var</i> }	value of the environment variable <i>var</i>
\$(cmd)	output of <i>cmd</i>
\$(( <i>expr</i> ))	result of the mathematical expression expr

#### Shell utilities

apropos text	searches the manual pages for <i>text</i>
cat file	prints the contents of <i>file</i>
<b>cd</b> dir	changes the working directory to directory
<pre>chmod prm file</pre>	changes permissions of file to pri
<pre>cp src dst</pre>	copies the file/directory src to dst
echo text	prints text
<b>file</b> file	determines the file type of <i>file</i>
<b>find</b> dir expr	finds files in <i>dir</i> that match <i>expr</i>
<pre>grep expr file</pre>	searches for pattern <i>expr</i> in <i>file</i>
<b>ls</b> dir	list the entries in the directory dir
man cmd	displays the manual for <i>cmd</i>
<b>mkdir</b> dir	creates the directory dir
mv src dst	moves/renames <i>src</i> to <i>dst</i>
pwd	prints the current working directory
rm file	removes the file <i>file</i>
sort	sorts lines of text from input
touch file	creates the empty file file

### Input output redirection

put output i	can ección
cmd1   cmd2	runs <i>cmd1</i> and <i>cmd2</i> and redirects the output of <i>cmd1</i> to the input of <i>cmd2</i>
<pre>cmd &gt; file</pre>	runs <i>cmd</i> and redirects output to <i>file</i> , content of <i>file</i> is overwritten
<pre>cmd &gt;&gt; file</pre>	runs <i>cmd</i> and redirects output to <i>file</i> ,
<pre>cmd &lt; file</pre>	the output is appended to file runs cmd and redirects file to its input

#### Job control

Ctrl + C

Job = Shell command and its associated process(es)

- Each job has a job id and corresponding process ids
- Jobs can run in the foreground or in the background
- The execution of a job can be temporarily suspended

cmd &	starts <i>cmd</i> as background job (id is printed)
<b>fg</b> %job	puts the job $job$ in foreground
<b>bg</b> %job	continues suspended job <i>job</i> in background
ps	prints the process ids of all active jobs
<b>kill</b> pid	terminates a process with the process id <i>pid</i>
Ctrl + S	suspends active job
Ctrl + $Q$	continues active job
Ctrl + Z	puts active job to background and suspends it

sends an EOF character

aborts the active job (most of the times)