

## 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  $\neq$  '/')

/usr/bin/ls	example for an absolute file path
/home/foo/	example for an absolute directory path
bin/a.out	example for a relative file path
.	path to the <i>working directory</i>
..	path to the parent directory

### File system hierarchy

/	Primary hierarchy and root directory
/bin	Command binaries
/dev	Device files
/etc	System-wide configuration files
/sbin	System binaries
/tmp	Temporary files
/usr	Secondary hierarchy for user data
/usr/bin	Non-essential command binaries
/usr/include	Standard include files
/usr/lib	System wide libraries
/usr/local	Tertiary hierarchy for local data
/usr/sbin	Non-essential system binaries
/var	Variable files

## 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

Unity/GNOME	Ctrl + Alt + T
Mac OS	Cmd + [ ] → "terminal" → [ ]
Bash on Windows	Win + R → "bash" → [ ]

## Shell

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

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

### Prompt

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

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

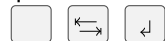
### Line editing

Ctrl + A	Go to the beginning of the line
Ctrl + E	Go to the end of the line
Ctrl + U	Clean up to the beginning of the line
Ctrl + K	Clean up to the end of the line
Ctrl + C	Cancel the current command line

### Special characters

The following characters can't be used directly:

| & ; < > ( ) \$ ` \" \* ? [ # ~ = %



\ preserves the literal value of the following character  
' ' preserves the literal values of enquoted characters  
" " preserves the literal values of enquoted characters  
except the characters ` \$ \

### Expressions

~	<i>home directory</i> of the current user
*	matches any character sequence
?	matches a single character
\${var}	value of the environment variable <i>var</i>

## Shell utilities

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

## Input output redirection

<i>cmd1</i>   <i>cmd2</i>	starts <i>cmd1</i> and <i>cmd2</i> and redirects the output of <i>cmd1</i> to the input of <i>cmd2</i>
<i>cmd</i> > <i>file</i>	starts <i>cmd</i> and redirects its output to <i>file</i> , content of <i>file</i> is completely overwritten
<i>cmd</i> >> <i>file</i>	starts <i>cmd</i> and redirects its output to <i>file</i> , the output is append after content of <i>file</i>
<i>cmd</i> < <i>file</i>	starts <i>cmd</i> and redirects <i>file</i> to its input

## Job control

*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

<i>cmd</i> &	starts <i>cmd</i> as background job (id is printed)
fg % <i>job</i>	puts the job <i>job</i> in foreground
bg % <i>job</i>	continues suspended job <i>job</i> in background
ps	prints the process ids of all active jobs
kill <i>pid</i>	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
Ctrl + C	aborts the active job (most of the times)