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 '/')

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 /home/foo/

Example for a relative file path ./a.out

File system hierarchy

Root directory /

/bin Essential command executables

/dev Device files

System-wide configuration files /etc

Manually added software /opt

/sbin Essential administrative executables

Temporary files /tmp

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

/usr/local Site-local data

/usr/sbin Administrative executables

/var Variable files

man hier (or man file-hierarchy on recent Linux Expansions 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, quake

Opening a terminal

 $\left| \mathsf{Ctrl} \right| + \left| \mathsf{Alt} \right| + \left| \mathsf{Alt} \right|$ Unity/GNOME ightarrow "terminal" ightarrow [ightarrowMac OS Bash on Windows

Shell

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

man intro to get an introduction into basic shell usage

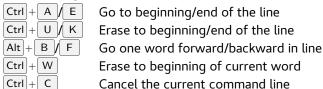
Examples for shell programs: sh, bash, zsh, fish, ksh

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

~	Home directory of the current user
*	Matches any character sequence
?	Matches a single character
[]	Matches any one of the enclosed characters
\${ <i>var</i> }	Value of the environment variable <i>var</i>
\$(<i>cmd</i>)	Output of <i>cmd</i>
\$((<i>expr</i>))	Result of the mathematical expression expr

Shell utilities

apropos text	Searches the manual pages for text
cat file	Prints the contents of file
cd dir	Changes the working directory to dir
<pre>chmod prm file</pre>	Changes permissions of file to prm
cp src dst	Copies the file/directory src to dst
echo text	Prints text
file file	Determines the file type of <i>file</i>
find dir expr	Finds files in dir that match expr
grep expr file	Searches for pattern <i>expr</i> in <i>file</i>
ls dir	List the entries in the directory dir
man cmd	Displays the manual for <i>cmd</i>
mkdir 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

cmd1 cmd2	Runs $\mathit{cmd1}$ and $\mathit{cmd2}$ and redirects the
<pre>cmd > file</pre>	output of <i>cmd1</i> to the input of <i>cmd2</i> Runs <i>cmd</i> and redirects output to <i>file</i> , content of <i>file</i> is overwritten
<pre>cmd >> file</pre>	Like >> but appends output to file
<pre>cmd < file cmd <<< text</pre>	Runs <i>cmd</i> and redirects <i>file</i> to its input Runs <i>cmd</i> with input <i>text</i>

Job control

Ctrl + D

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)
fg %jid	Puts job with id jid in foreground
bg %jid	Continues job with id jid in background
jobs	Prints ids of jobs in current shell
kill pid	Terminates process with the PID <i>pid</i>
ps	Prints PIDs of processes in current terminal
Ctrl + S / Q	Suspends/resumes active job
Ctrl + Z	Puts job to background and suspends it
Ctrl + C	Aborts job (most of the times)

Sends an EOF character