

First steps on the command line in Linux/Unix



bwHPC support team Hohenheim

[illegible]

- When you log into bwUniCluster, you are on the command line
- PC: Open a console.
 - Ubuntu: open app “terminal”, short cut ALT+STRG+T
- You usually start in you home directory, e.g. where all non-system related data/code is hosted

Command line navigation

- **pwd** shows the path to the current directory
- **ls** shows the content of the current directory
- **ll** shows the content of the directory including file sizes,...
- **mkdir <name>** makes a new directory (subfolder of the folder you are in)
- **cd <folder>** changes the directory to <folder>
just **cd** returns to home directory (or **cd \$HOME**), **cd ..** jumps one directory up (from subfolder to folder)

- **cp** <file to copy> <name of copy> copies a file
- **mv** <file to move> <move target> moves/renames a file
- **touch** <filename> creates an empty file
- **rm** <filename> removes/deletes file (!PERMANENTLY!)

You can either specify a file within your current directory or a path to the file

Example:

```
$ ls                #check subfolders
$ mkdir test1       #make new subdirectory
$ touch test1/dummy1 #create file
$ touch $HOME/dum2  #create 2nd file
```

Where did we create the two files? How to delete them?



- **cp -r <folder to copy> <name of copy>** copies a folder
- **mv <folder to move> <move target>** moves/renames a file
- **rm -r <filename>** removes/deletes folder (!PERMANENTLY!)

You can copy/move/delete/... all files or those with a specific pattern via **wildcard** *

Example:

```
$ ls                #check files
$ touch test1.txt   #create file
$ touch test2.txt   #create file
$ ls *.txt          #create 2nd file
$ rm test*.txt      # Remove both
```

Could we delete the two files with an alternative use of the wildcard?

- You can autocomplete commands by the tab key (e.g. when entering paths to files)
- Scroll through your previous command line entries via   arrow keys
- Most command line commands have a manual via **man <command name>**
- **echo <argument>** writes the argument to the command line. Could be a text string or e.g. a bash variable
\$ echo "TEST"
\$ echo \$HOME #preset bash variable
- If you are stuck within a command and cannot continue: CTRL+C
(if that does not work, try CTRL+D, ESC or q).
Try it out: \$ echo "TEST"
- Copy & paste with the mouse (not via shortcut)

- For using the bwUniCluster, you will usually write scripts of bash commands to execute.
- Simple easy-to-use text editor: **nano <filename>**. If you open a non-existing file it will create this file (as soon as you save within nano). Usually available on any Linux.
- **nano** has not many functions/options. A much more performative editor is **vim** (which we will not need today)
 - **vim <filename>** opens the file
 - There is a built-in 25-30 min tutorial of vim, which is started by command **vimtutor** (resp. **vimtutor -g <language>**)

Both are installed on bwUniCluster. On your own system, just roll with your favourite editor!

- CTRL + letter activates all functions (STRG on German keyboard)
- CTRL + g: Help manual
- CTRL + o: save current file
- CTRL + x: exit nano
- Copy & paste as in command line via mouse, it copies to the location of the cursor
- Cursor move by arrow keys

All common functions (via CTRL+ letter) are shown at the bottom of the nano GUI

Adding to text files from the command line

- <command> > <text file> writes the output of the command to a text file (created if not yet existing, overwriting content if existing)
- <command> >> <text file> appends the output of the command to a text file (created if not yet existing, appending content if existing)

Example

```
$ cd test1
```

```
$ pwd > test3.txt
```

```
$ echo $HOME >> test3.txt
```

- **cat <text file name>** prints (=shows in command line) the text file

Example: `$ cat test3.txt`

You will need your 2FA device again
(and you need to be in the university network)!

- Linux/MacOS: Use scp
 - From bwUniCluster to your computer
scp <ho_name>@bwunicluster.scc.kit.edu:<remote file> <local file>
 - From your computer to bwUniCluster
scp <local file> <ho_name>@bwunicluster.scc.kit.edu:<remote file>
- Windows: via MobaXterm

After entering the command, you will be prompted for an (T)OTP and your service password

non-Hohenheim users: User name has different prefix

Works for any remote system with Unix (access control may differ)!

1. Find out what **ls -t -r -l** does. (-<letter> runs **ls** with a specific option)
2. Put a text file into the new folder test1/ that contains some text. Rename it.
- ★ 3. Write a list of the files in your home directory and put it into folder test1
4. Remove the directory test1/

- Text file that lists bash commands that should be run (executed line by line)
- First line: **#!/bin/sh** (so that commands are executed “as in command line”), then commands

- Example

```
test run.sh
#!/bin/sh
ls
pwd
```

- Execute command with
\$ sh <script name>
- Alternatively, make script executable (via **chmod +x**) and execute it (via **./**)
\$ chmod +x <script name>
\$./<script name>

Exercise

- Create a file testjob1.sh
- Should include 2 commands
 - Show THIS IS A TEST on the command line
 - Run the command **hostname** (shows the node you're on)
- Run it!

- Variables: You can run a bash script with variables via **sh <scriptname> <var1> <var2>** and use the variables within the script as **\$1** and **\$2**

Example: Script

```
test_run2.sh  
#!/bin/sh  
echo $1  
echo $2
```

Run it & output

```
[ho_ffreund@uc2n995 ~]$ sh test_run2.sh "Hello" "world"  
Hello  
world
```

- Bash scripting also allows the usual programming constructs: **for** loops, **if else** conditions, ... see our handout!

- Extract lines from a text file that includes a specific word via **grep**

```
$ cat <filename> | grep "testword"
```

Searches for string testword in the filename

Example: `$ cat test_run2.sh | grep "echo"`

- Note the **pipe operator** above

```
$ <Command 1> | <command 2>
```

Applies command 2 on the output of command 1

- **cut** cuts out specific parts in files, e.g. columns ("fields") in tab-delimited files

```
$ cut -f 1 -d " " test_table.txt
```

 extract column 1 in " " -delimited file

```
$ cut -f 1-2 -d " " test_table.txt
```

 extract column 1-2 in " " -delim. file

See **man <command>** for much more information

- **history** shows your full history of commands on this machine/node

Exercise: List all command line entries in the command history that featured `cd`!

Some convenience knowledge (after the first steps/for a bit experienced users)

- Under linux systems (e.g. Ubuntu): You can mount a folder from bwUniCluster on your local computer via **sshfs** - see details
 - In German: <https://wiki.ubuntuusers.de/FUSE/sshfs/>
 - In English: <https://embraceubuntu.com/2005/10/28/how-to-mount-a-remote-ssh-file-system-using-sshfs/>
- You can copy code/software to the bwUniCluster and compile programs. However, you have no sudo rights
- Access control: Control who can access or modify your files (usually only yourself and super users). On bwUniCluster, we recommend using access control lists:
https://wiki.bwhpc.de/e/Workspace#Sharing_Workspace_Data_within_your_Workgroup

There are a myriad of resources to learn more about bash and bash scripting

- Further bwHPC courses, see <https://training.bwhpc.de/>
- HPC resources https://hpc-wiki.info/hpc/Introduction_to_Linux_in_HPC
- Web resources, e.g. <https://www.redhat.com/sysadmin/learn-bash-scripting>