REGIONALES HOCHSCHUL-
RECHENZENTRUM KAISERSLAUTERN

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

RHRK-Seminar

High Performance Computing with the Cluster „Elwetritsch“

Course instructor: Dr. Josef Schüle, RHRK

Overview

- Cluster Elwetritsch
- Accessing Elwetritsch
- Desktop
- Linux Basics
  - shell
  - environment
  - filesystem
  - files
  - scripting

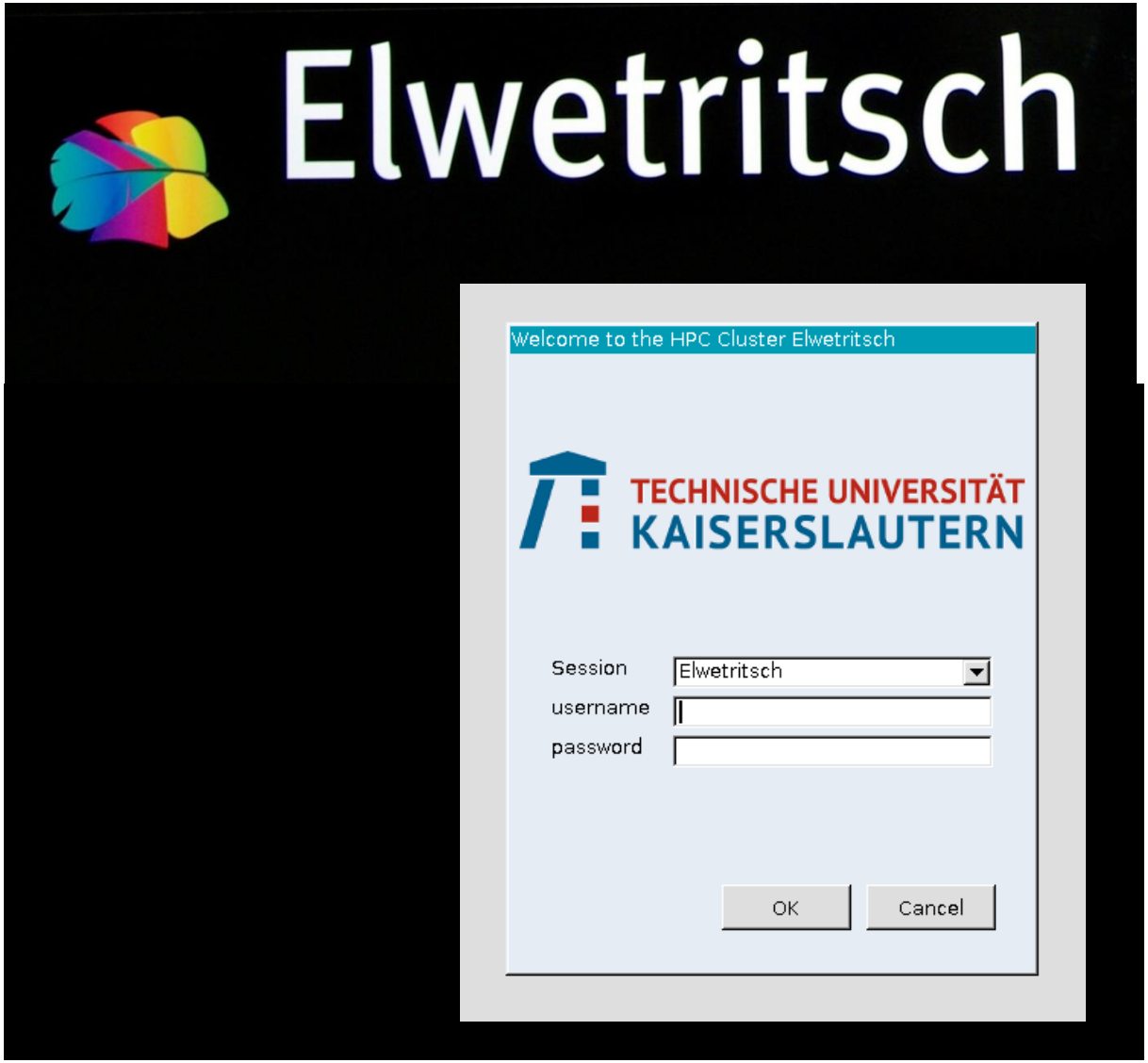
Login to cluster

Premises

- RHRK account
- supervisor

Protocols and Clients

- SSH
- RDP
- NX



Cluster „Elwetritsch“

Elwetritsch basics

- 4 login nodes
- many nodes for computing
- some visualization nodes
- special nodes

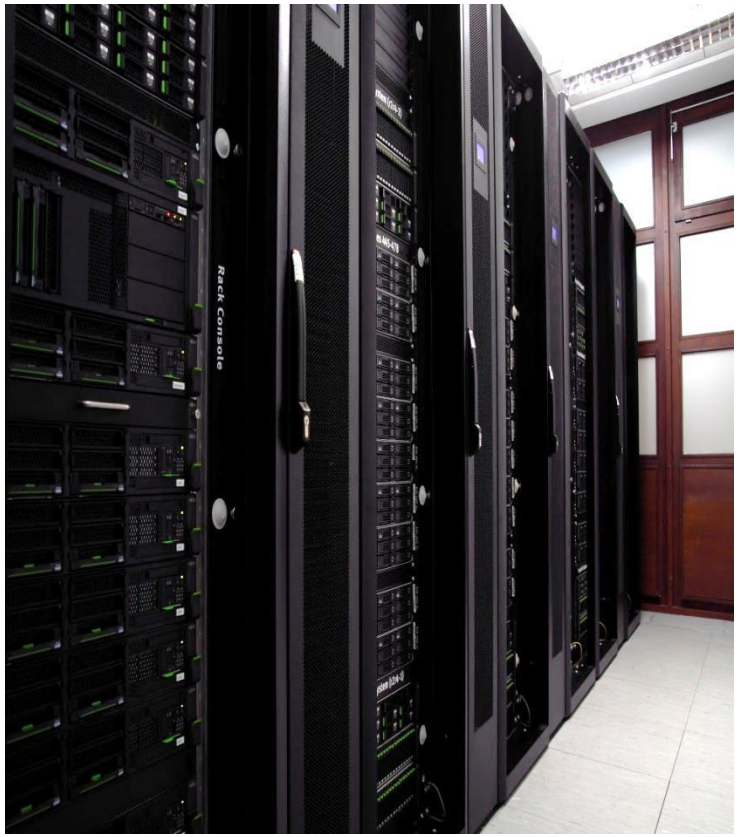
Access

- access limited
- contingents for projects
- Access only via

  elwe1-4.rhrk.uni-kl.de

  (elwe1-2.rarp-kl.de - external usage)
- Ubiquitous accessible

  no VPN required



Premises

RHRK Account

- Account for employees
  URL: https://www.rhrk.uni-kl.de/en/services/accounts
- Extended student‘s account
  Supervisor may grant access via https://serviceportal.rhrk.uni-kl.de/
- Testing/setting account‘s privileges
  URL: https://elwe.rhrk.uni-kl.de/cockpit/

Hints for students

- Extended student‘s account required
- Limited in time (Hiwi or Bachelor/Master work)
- Arising costs (not usage of Elwetritsch itself, but usage of commercial software licenses) forwarded to AG

Connecting Basics

ssh usage: Please read ssh.shtml carefully.

Linux and OS X:

- SSH intrinsic available
- VPN required
- Command: `ssh -X elwe4.rhrk.uni-kl.de` allows opening of windows (X-forwarding)
- MACs require extension for an open source X11-Client or Apple X11

Windows:

- install Tunnelier client- ssh client and winscp client

Recommended

  - fast and effective file transfer
  - VPN required

RDP = Remote Desktop Protocol

No VPN

- **Windows:**
  - Start -> Remote -> remote desktop
  - Configuration:
    - Computer: elwe1-4.rhrk.uni-kl.de
    - Display -> configuration: full page (Ruler to the right)
    - **Display -> Color: True Color (24 Bit)**

- **Linux:**
  - use a client like rdesktop or xfreerdp
- **OS X:**
  - Client required, like CoRD
  - Retina-Displays are not suited

NX

- **NX Server is installed on elwe-nx**
  - Download Client v5 at www.nomachine.com
  - Turn on Auto-Update to assure newest version
- **Konfiguration:**
  - No UDP
  - Protocol: SSH
  - Host: elwe-nx.rhrk.uni-kl.de
  - Port: 22
  - virtual desktop -> fully qualified desktop
  - user defined session -> like ssh login with X11 enabled

▪ others

x2go

The x2go client provides simple and easy to install virtual desktop to elwetritsch. It provides a desktop on the head nodes with same functionality as NX. The client can be downloaded from http://wiki.x2go.org.

NICE-DV

This is mainly intended for users of Ansys and Abaqus which otherwise have problems to visualize their workbenches.
You may download a client and/or connect via a browser to https://elwe.rhrk.uni-kl.de/enginframe/vdi/
Please close all windows after your work. Otherwise licenses are blocked and the licenses are expensive and thus limited.

Common Login Problems

- **ssh login works, but not RDP**
  - If your password contains the $-sign -> has to be changed
  - ➢ https://passwort.uni-kl.de/
- **No login with RDP possible**
  - you perhaps worked with python and used a "conda install"
  - ➢ please read conda specific information
  - ➢ send a mail to hotline@rhrk.uni-kl.de - "conda install" corrupts your .bashrc-file and has to be fixed
- **Quota exceeded**
  - login via RDP or NX is not possible any more
  - ➢ login via SSH and remove data.
  - ➢ please have a look at our FAQ - "How to delete files"

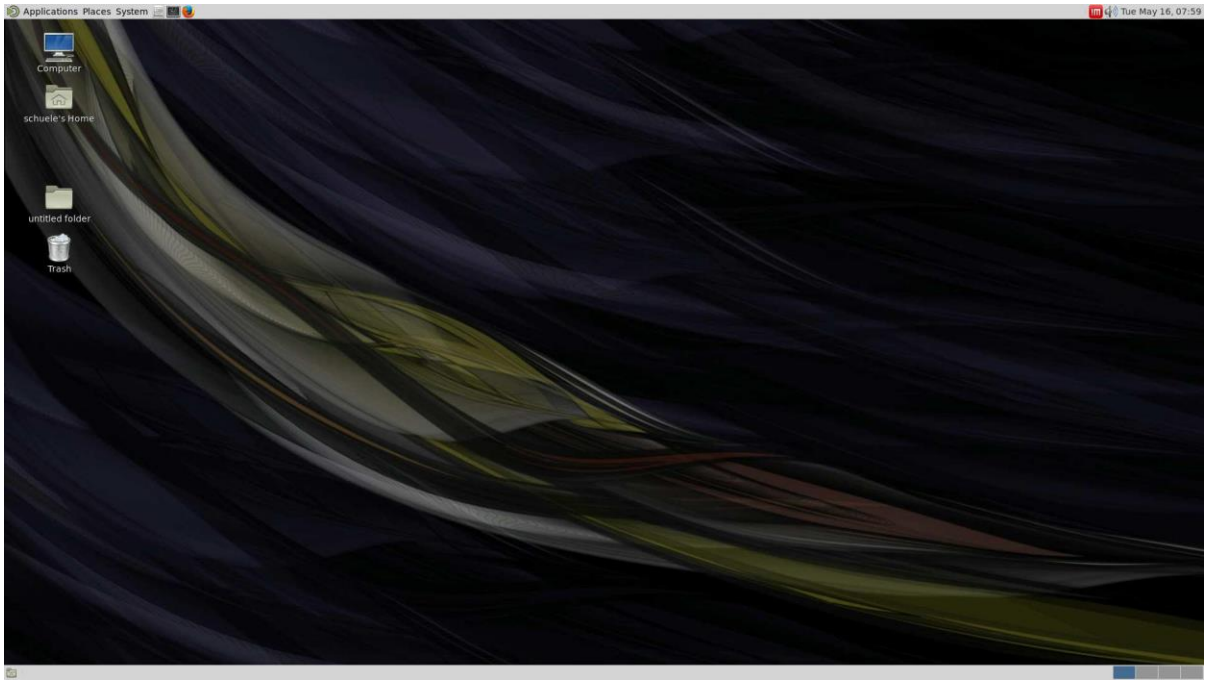
Common Login Problems

- **Desktop has vanished after re-login**
  - RDP connects to previous sessions only, if all parameters are the same especially server, resolution and color depth.
  - Otherwise a new session is opened and the old one remains open
  - ➢ Control it with

    ```
    ps -fu $USER | grep [X]vnc
    ```

    If so, kill the old session by killing the ID given with above command sequence

Desktop



GNOME-Desktop

- **Applications**
- **Places**
- **System**

Other Desktops

- **KDE**
- **XFCE**

GNOME Desktop

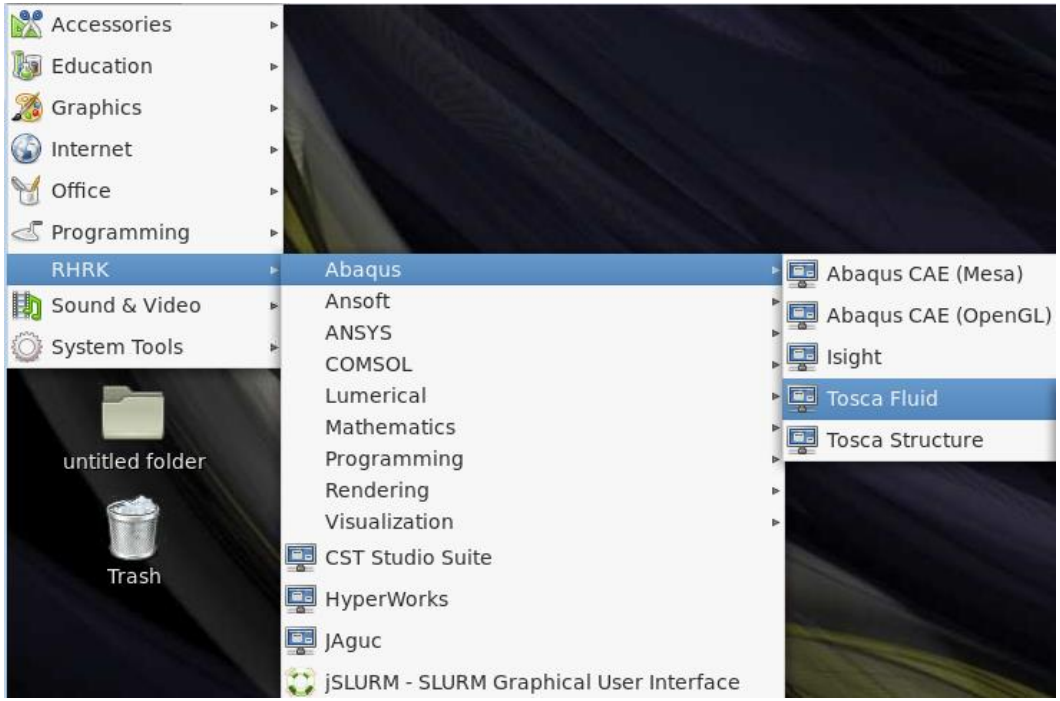
- **Classical graphical desktop**
  - file manager
  - menus
  - common look & feel
- **set of accessories**

file managers

- **don‘t delete files into Trash**
  - reduce trash to prevent quota from exceeding
  - see "How to delete many files" in our FAQ
- **/scratch/<userid>**
  - HOME is not intended for large amounts of data - but /scratch filesystem
- **Using Caja: Add /scratch/<userid> to bookmark**
  - not for directories with many (> 1000) files.
  - instead open a command window and type "ls" or "ls -lt"
  - use "rm" to delete files
  - change behavior into list - not icons

Menus

- **Menu RHRK**
  - configured applications
  - more possible after request



Hints for GNOME caja

-> Accessories -> files opens caja

- **use right mouse to**
  - open folder in new window
  - please use Delete instead of Move to Trash

  (if really needed - we keep backup of the files for some days, see **FAQ** for a restore)
  - key „Entf“ moves data to trash
  - open with
- **clean trash on a periodically basis**
- **open a bookmark for frequently used folders**
  - especially meaningful for /scratch/<userid>
- **much much better is a console window and some unix commands**
  - ls - list files
  - rm - remove files
  - mv - move files
  - cd - change directory

Drag & Drop files

Use mv instead - if really needed:

- **Any interrupt will produce chaos**
- **Drag & Drop - files and folder**
  - change in filesystem -> copy (plus added to cursor)
  - in same filesystem -> move (arrow added to cursor)
  - copy in same filesystem
    - keep „Strg“ pressed
  - move in different filesystems
    - keep „Shift“ pressed
  - set a link
    - keep „Shift“ and „Strg“ pressed
- **Don‘t use it to transport large amount of data or folder with many files (see FAQ)**

Different Desktops

- **RDP-Login allows to switch the desktops**
  - GNOME: `# rhrk-enable-desktop-gnome` (default)
  - KDE: `# rhrk-enable-desktop-kde`
  - XFCE4: `# rhrk-enable-desktop-xfce4`
  - Fluxbox: `# rhrk-enable-desktop-fluxbox`
  - IceWM: `# rhrk-enable-desktop-icewm`

- **NX - the session may be changed specifying a starting script**
  - GNOME: selectable
  - KDE: selectable
  - XFCE4: /usr/bin/startxfce4
  - Fluxbox: /usr/bin/startfluxbox
  - IceWM: /usr/bin/icewm-session

Linux Basics

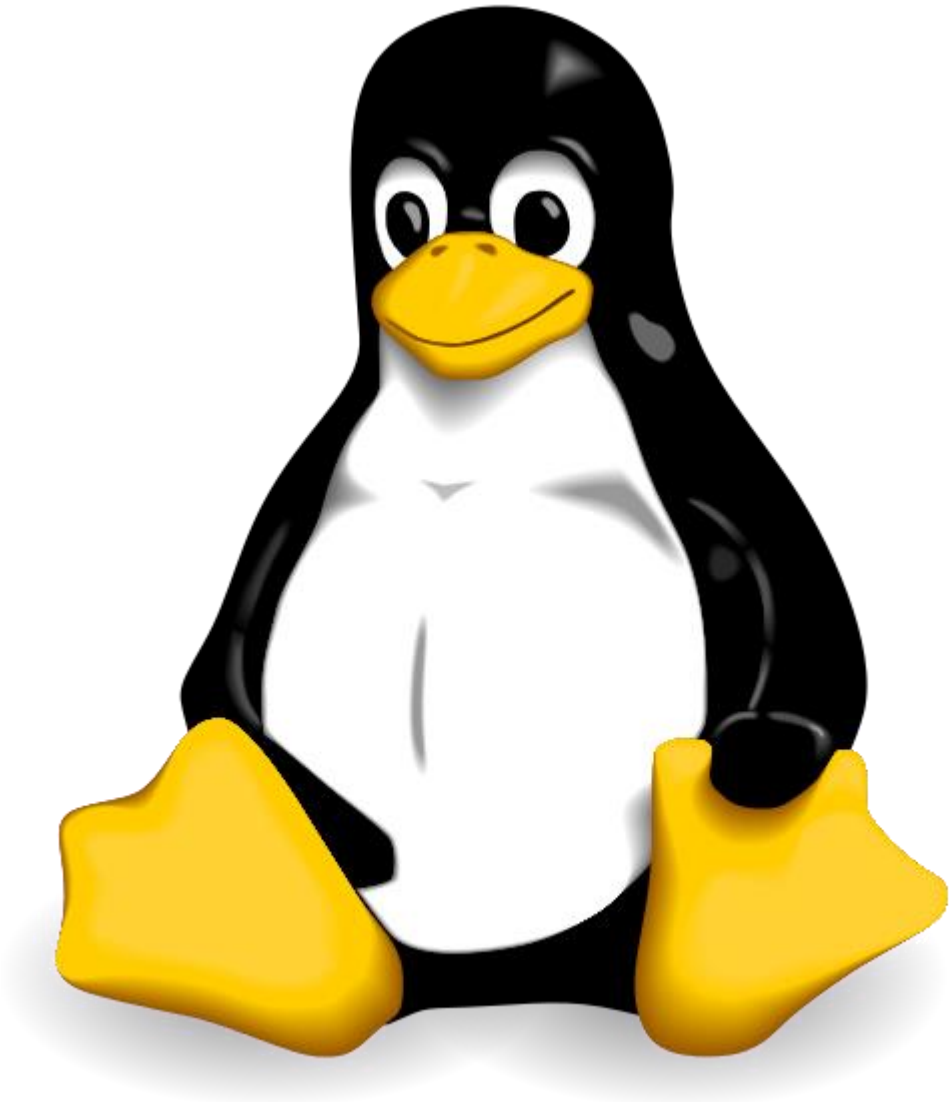
Terminal / Shell

- **bash**
- **command syntax**
- **Man pages**

file system

- **files**
- **folders**
- **rights**

further hints



Terminal / Shell

- **Most effective is working within a shell**
- **Most commands are only available in a shell**
- **starting a shell**
  - login with SSH
  - start a terminal/console on the desktop
- **bash is the default shell**
  - scripts and examples in the cluster documentation refer to this shell
  - configuration in $HOME/.bashrc
  - other available shells: ksh, csh, tcsh
- **Which shell do I use?**
  - the shell is directly connected with the account
  - More: https://elwe.rhrk.uni-kl.de/cockpit/

bash - background process

A background process may be started for any command. The current shell is ready to proceed with other commands

- **Start**
  - add "&" to the command (e.g. xterm &)
  - type „CTRL-Z“ while a command runs, followed by „bg“ (=background)
- **Properties**
  - stdin and stdout remain at calling process
  - ending the calling process will end all it‘s background processes
  - command jobs provides a list of running background processes. Processes obtain an internal number 1-
  - `kill %1` will end the first back ground process
  - `fg %1` will move first back ground process into foreground again
- **Example:**
  - `find $HOME -name "*.c" -print > found 2>/dev/null &`
  - `jobs`
  - `kill %1`

bash - stdin, stdout, stderr

- associated files are stdin (input), stdout (output), stderr (error protocol)
- input /output redirection
  - < input will be read from file
  - > output will be written into file (overwrite)
  - >> output will be appended to file
  - 2> error will be written into file (overwrite)
  - 2>> error will be appended to file
  - 2>&1 stderr is redirected to stdout
- examples:
  - `cat /etc/hosts > test`
  - `echo "# Test" >> test`
  - `tail test`

bash - command sequences

- **command sequences:**
  - commands following each other may be written into one line
  - a ; (semi colon) separates two commands
- **command groups:**
  - example: redirect the output of two commands into one file
  - `ps; who > test ; cat test` (who)
  - `ps > test ; who > test ; cat test` (who)
  - `ps > test; who >> test; cat test` (both)
  - `(ps ; who) > test; cat test` (both)
- **Pipes**
  - commands are chained. Output of first is input to second
  - example: `ls | wc -w` (number of files in folder)
    - intermediate result may be saved to file
    - Solution: `ls | tee test | wc -w ; cat test`

man pages for help

- **Almost everything provides a so called manual page**
  - example: `man ls`
  - man pages are ordered (`man 1 kill` and `man 2 kill`)
  - man pages for C-functions (man malloc)
- **`apropos` scans all man pages for given word (in case you do not know the exact wording of a command)**

  (example: `apropos terminal`)
- **further information:**
  - `man -k fopen`
  - `whatis fopen`
  - `info fopen`

bash - environment

- **Variables in shell may be used like variables in programs**
- **Usage:**
  - `VARIABLE=value` (no blank) assigns a value
  - `export VARIABLE` makes `VARIABLE` visible to child processes
  - access via `$VARIABLE` or `${VARIABLE}` (preferable in case of blanks in value)
  - `echo ${VARIABLE}` returns value
  - which variables are used: `set` or `env`
  - remove with either `unset VARIABLE` or `export VARIABLE=`
  - extension like in: `export PATH=/bin:${PATH}`
- **certain variables:**
  - Exit-Code: **$?**, Process-ID: **$$**, **$!** (last background process)
  - scripting: **$0** (name of script), **$1…$9** (arguments passed), **$#** (number of arguments), **$***, **$@** (all arguments)

bash - environment examples

for example find out, if your colleague is logged in:

ps -ef | grep -v $USER | grep colleague; echo $?

- $USER is set in your environment
- $? is the return code

kill -9 $$

- ends your current shell ($$ process id)

echo "echo \$3" > my_echo; . my_echo 1 2 3 4

- will write a file my_echo with line

  `echo $3`

  (\ is needed to print $-sign)
- then my_echo is executed with 4 arguments
- the 3th is printed

bash - wildcards

- **naming convention:**
  - A word containing *, ? or […]
  - shell extends it before execution
- **meaning:**
  - * anything, even nothing
  - ? one single character
  - […] a set of characters
  - - defines a range in […] like a-z (all small letters)
  - ! negation, if used as first sign after [like [!a-z] all but small letters
  - \ next special character (like . or *) should not be read as is
- **example:**
  - `ls /etc/h*`
  - `ls /etc/h*t?`
  - `ls /etc/h[oi]*`

filesystem

- **file types known to Linux**
  - Normal files (text and binary)
  - folders
  - device files
  - pipes
- **all files are hierarchically located below /**
- **no letters for drives like in windows**
  - different partitions are mounted at a well defined place below /:
    `df -lh`
  - this is even true for network drives (like NFS): `df -h`
  - `/net` and `/media` may be filled after access by automounter
  - "." and ".." are special folder names indicating this folder (.) and the folder above in the hierarchy (..)

folder hierarchy

- current folder path in file system hierarchy
  - `pwd`
- change into new folder
  - `cd /some/other/directory`
- short time change into another folder
  - `pushd /some/other/directory`
  - …
  - `popd`
- moving into your HOME
  - `cd ~` or `cd $HOME`
- important folders on Elwetritsch:
  - /home, /scratch, /software, /rhrk/home: `cd $(rhrk-home)`

file and folder access rights


```
$ ls -l
total 5          (1)
-rw-------   1   user   users  386    Aug 4  9:12     brief
-rw-r-----   1   user   users   78    Aug 9 17:01     text.1
drwxr--r--   1   user   users  113    Sep 4 11:24     upros
   (2)       (3)      (4)      (5)      (6)            (7)
```

- **(1) number of blocks occupied by files listed**
- **(2) type and access rights:**
  - „d“ = Directory, „b“ = special file (block oriented), „c“ = special file(character oriented)
  - „l“ = symbolic link, „p“ = Pipe, „-“ = normal file
  - followed by 9 „Protection-Bits“,3 for user rights, 3 for group, 3 for all others. r=readable, w=writable, x=executable (=readable in case of a folder)
- **(3) number of hard links**
- **(4) name of owner and group (s. command id)**
- **(5) size in bytes**
- **(6) date and time of last modification**
- **(7) name of file/folder**

access rights details

- owner/user ("u"), group ("g") and others ("o") have three flavors of rights
  - reading ("r"), writing ("w"), and executing ("x"), in case of a folder reading in it
- octal notation of rights:

r	w	X	Oktal	meaning
0	0	0	0	no rights
0	0	1	1	just execution
0	1	0	2	just writing
0	1	1	3	writing and executing
1	0	0	4	just reading
1	0	1	5	reading and executing
1	1	0	6	reading and writing
1	1	1	7	full access rights

access rights folders

- **right to read**
  - allows to list the contents (not necessarily to read the file therein. This is guided by their bits).
- **right to write**
  - generate new files
  - rename files
  - remove files
- **right to execute**
  - enter the folder (not reading it‘s content)
  - accessing (known) files therein if they allow it

modifying access rights

Command: `chmod <who><allowance><what> file`

- **<who>**
  - u (user=owner)
  - g (group)
  - o (others)
  - a (all)
- **<allowance>**
  - + may (added)
  - - may not (removed)
  - = may exactly like specified
- **<what>**
  - r (read)
  - w (write)
  - x (execute)

modifying access rights

Command: `chmod <octal code> file/folder`

- **rights are overwritten to specified octal code**
  - `chmod 640 file` (user read/write, group read)
  - `chmod 755 folder` (user all, group/others read/execute)
- **access rights generating a file**
  - `umask` is responsible
    - octal code "000" negates rights (negation to `chmod` !)
    - notation like `chmod`
  - `umask 077` will lead to rwx------ (077 = 000111111)
  - `umask u+rwx,g-rwx,o-rwx`
  - `umask -S`

working with text files

- looking at files
  - full file: `cat`
  - paging and searching ("/search string"): `less` or `more`
  - end of a file: `tail`
  - head of a file: `head`
  - full file in reverse order: `tac`
- examples:
  - `cat /etc/hosts`
  - `less /etc/hosts`
  - `head -10 /etc/hosts`
  - `tail -10 /etc/hosts`
  - watching a file while a process appends to it:

    `tail -f <file>`

working with files

- **copy:**
  - `cp "old" "new"`
- **rename/ move:**
  - `mv "old" "new"`
- **delete:**
  - `rm "file"`
- **link:**
  - symbolic link: `ln -s /etc/hosts myhosts`
    - allows to link to files in different file systems
    - link indicated by -> in ls -l
    - origin deleted, link remains but is invalid
  - true (hard) link: `ln „existing file“ „new file“`
    - only within same file systems
    - both are identical (no copies). Is one removed, the other one does still exist with it‘s content.
    - if one is changed - both are changed (identical)

working in folders

- **listing the content:**
  - `ls, ls -l, ls -al, ls -alrt`
- **generating a folder:**
  - `mkdir "dir", mkdir -p "dir"`
- **copying a small folder:**
  - `cp -rp "old dir" "new dir"`
- **copying a large folder (different file system):**
  - tar cf - "old dir" | (cd "new place"; tar xf -)
- **rename / move:**
  - `mv "old dir" "new dir"`
- **remove:**
  - `rm "dir", rm -r "dir", rm -rf "dir"` (Caution you won't be asked anymore!)
- **copying/moving files from one folder to another:**
  - `cp dat1 dat2 dat3 … "dir"`
  - `mv dat1 dat2 dat3 … "dir"`

collection of commands

command	meaning	action
cd	Change Directory	change into folder
ls	LiSt	lists contents of a folder
cp	CoPy	copies files/folders
mv	MoVe	renames or moves into different folder
rm	ReMove	removes files/folders
ln	LiNk	generates a link
mkdir	MaKe DIRectory	generates a folder
rmdir	ReMove DIRectory	removes a folder
umask	User MASK	sets umask for generating files/folders
chmod	CHange MODe	modifies access rights
chown	CHange OWNer	modifies owner (only root)
chgrp	CHange GRouP	modifies group (if user is in both or root)

hints

- **file system:**
  - Linux is case sensitive
  - avoid "Umlaute" or blanks (need a "\ " to write)
  - hidden files begin with ".", "ls -a" makes them visible
  - use variables instead of fix names (in case we change the mount points)
    - $HOME for /home/$USER
    - $SCRATCH for /scratch/$USER
- **bash, system and process management**
  - your own processes: `ps -fu $USER`
  - `top` lists all processes and their current load
  - `watch` allows periodic execution of a command

bash - scripting

- **executable files beginning with**

  `#!/bin/bash`

  containing a command sequence you otherwise would have typed in the terminal
  - `executable: carry an x in ls -l`
  - `chmod +x script` makes script executable

- **executed with `./script`**
  - contents of environment var. PATH is searched.
  - current directory is "."
  - PATH=$PATH:.; script

  extends the search path and executes script in current folder - but should not be used (in case you mistype in /tmp)

- **# indicate a comment**

- **scripting in command line possible as well, like:**
  - `for name in hello you ; do echo $name ; done  (difference to "hello you")`
  - `for i in $(seq 1 4); do echo $i; done`
  - `for (( i=0;i<4;i++ )) ; do echo $i ; done`
  - ```
    select v in hello you; do
      case $v in "hello") echo $v;; "you") echo "me too";; *) echo "no"; break;;
      esac
    done
    ```
  - `if [ "$?" -eq "0" ]; then echo "ok"; else echo "wrong"; fi`
  - `i=0; while [ $i -lt 4 ]; do echo $i; ((i++)); done`
  - `i=0 until [ $i -gt 4 ]; do echo $i; ((i++)); done`
    ```

bash - calling scripts

Syntax: script [options] [arguments]

- **options**
  - control behavior of a script/command
  - often a single letter (`-o`)
  - in most cases single letters may be combined (`-avh`)
  - each script/command may have its own set of options
    (s. manual page or `-h/-?` or `--help`)
  - some options need additional arguments
    (`sort -o sorted_file`)
  - -- alone indicates end of options
- **arguments**
  - further information
  - in most cases files or folders where something should be done

bash - calling scripts

- commands, options and arguments are " " separated
- some commands are special
  - long options starting with "--" instead of just "-"
  - no "-" in front of an option (like `tar tv file.tar`)
  - further example: `find . -name core -print`
- command may be used to start other commands
  - that is, an argument may be an argument with options

writing scripts

Use an editor

- joe
- vi
- emacs
- pluma
- …. many more

```
#!/bin/bash
#first line to specify the interpret
OPTIONS=$(getopt --options x:h --long help -- "$@")
eval set -- "$OPTIONS"
while true; do
        case $1 in
                -x) echo "explain $2";shift;shift;;
                -h|--help) echo "I will help you"; exit 2;;
                --) shift; break;;
        esac
done
echo "Arguments are $@"
```

`file script.sh`

(good praxis to end a script with .sh, but could be .nothing_whatever_you_like)

`chmod u+x script.sh` # change permission, so that file may be executed

./script.sh -x why? -- we all

working with files

- **search for a pattern:**
  - `grep pattern file (options: -i ignores case -v all but)`
- **change contents:**
  - `sed -e 's/a/e/g' input > output (substitutes a by e)`
- **select words:**
  - `cat input | awk '{print $1}' (prints 1 word in each line)`
- **copying according to file pattern:**

```
for f in $(ls */*out); do
        d=${f/\/*/ }  # select directory
        fn=${f/*\// } # select filename in directory
        fn=${fn%%.out} # remove suffix
        fn=${fn##slurm} # remove prefix
        fn=$d$fn     # combine directory and number
        mv $f new/$fn        # move finally
done
```

files before:	afterwards:
E1/slurm-30.out	new/E1-30
E1/slurm-35.out	new/E1-35
E2/slurm-70.out	new/E2-70

questions - comments

The HPC Team consists of several people. You address them all sending an email at

hotline@rhrk.uni-kl.de

- describe questions in detail
- give hints to files and directories (thereby granting access for us)
- add a screenshot

- High Performance Computing on Elwetritsch
- Part I

Vielen Dank
Thank You