Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

Introduction to GNU/Linux and the Shell

07/10/2019 | J. Albert-von der Gönna

Leibniz Supercomputing Centre

Bavarian Academy of Sciences and Humanities

IT Service Backbone for the Advancement of Science and Research

250
employees
approx.

57
years of
IT support

Computer Centre
for all Munich Universities

Regional Computer Centre
for all Bavarian Universities

National Supercomputing Centre
(GCS)

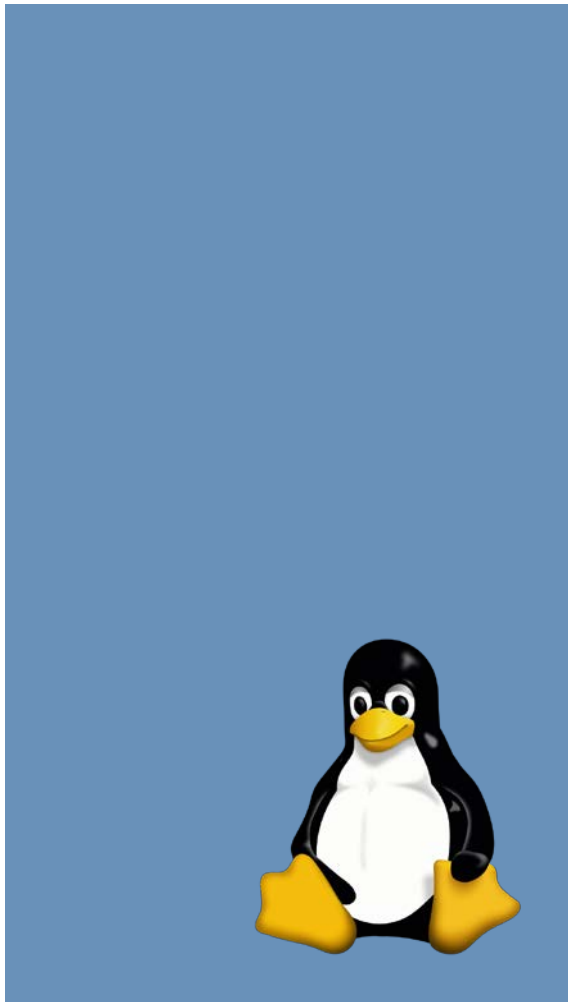
European Supercomputing Centre
(PRACE)

Course Information

- The aim of this course is to provide an introduction to GNU/Linux and the Unix Shell
- You will probably benefit the most, if you’re not yet familiar with GNU/Linux and the Unix Shell, but if you plan to work on the HPC and/or Compute Cloud infrastructure provided by LRZ
  -> by the end of this workshop, you should have the basic skills to successfully interact with GNU/Linux-based systems
- Consider the following – especially during hands-on sessions:
  -> you may want to partner up with the person sitting next to you
  -> it may be beneficial to sit back and watch the slides/demos
  -> the slides will be made available after the workshop
  -> generally: please ask, if you have any questions

What is GNU/Linux

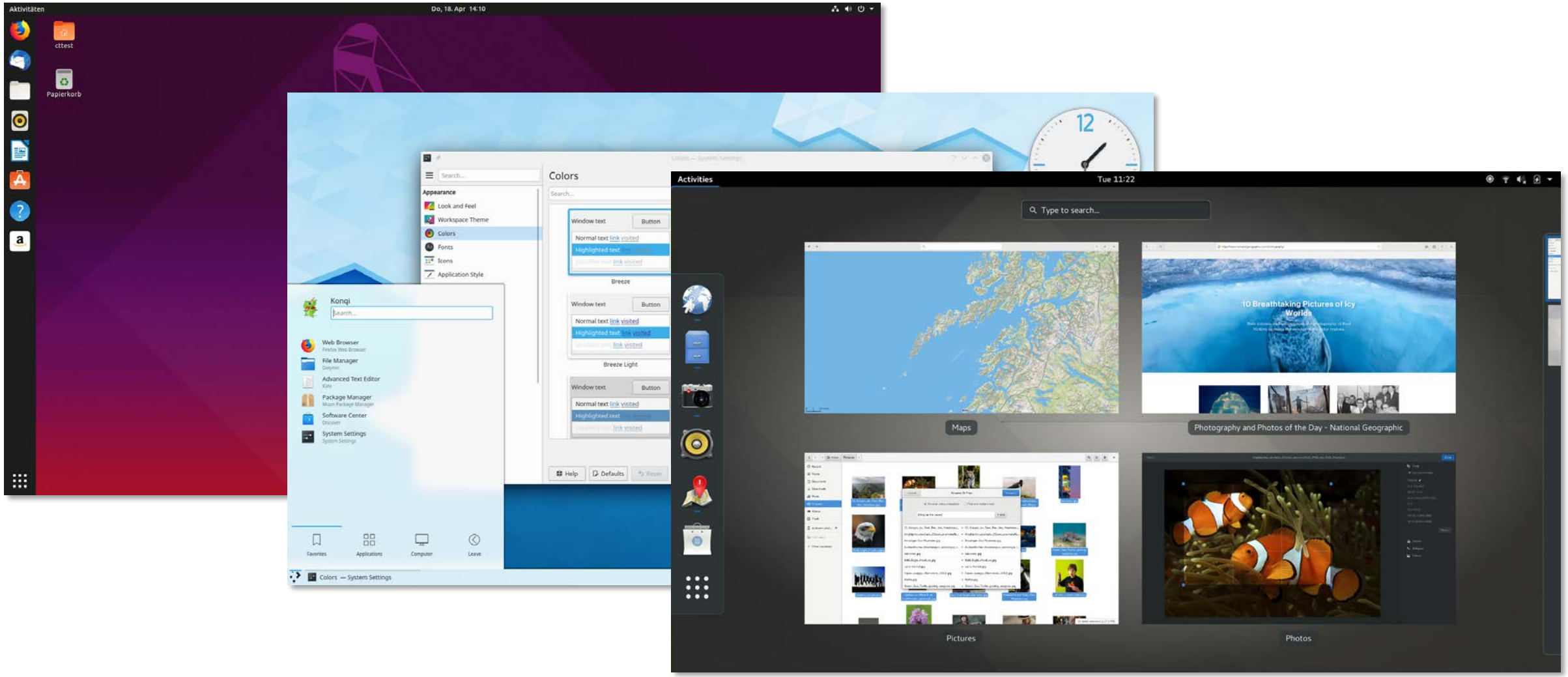
- Free and open-source operating system
- Alternative to
  Microsoft Windows, Apple macOS, Google Android …
- Generally consists of the Linux kernel, libraries and tools,
  a desktop environment and various applications
  (e.g. web browser, office suite, …)
- Different distributions:
  Arch Linux, Debian, Fedora/RHEL, openSUSE/SLES, Ubuntu, …



GNU General Public License (GPL)

- Use:
  “Run the program as you wish, for any purpose.”
- Study:
  “Study how the program works,
  and change it so it does your computing as you wish.”
- Share:
  “Redistribute copies so you can help your neighbor.”
- Improve:
  “Distribute copies of your modified versions to others.”

Desktop


Aktivitäten
Do, 18. Apr 14:10
cttest
Papierkorb
Colors
Appearance
Look and Feel
Workspace Theme
Colors
Fonts
Icons
Application Style
Breeze
Breeze Light
Help
Defaults
Konqi
Web Browser
File Manager
Advanced Text Editor
Package Manager
Software Center
System Settings
Favorites
Applications
Computer
Leave
Colors — System Settings
Activities
Tue 11:22
Type to search...
Maps
10 Breathtaking Pictures of Icy Worlds
Photography and Photos of the Day - National Geographic
Pictures
Photos

Linux and High Performance Computing (HPC)

ZDNet

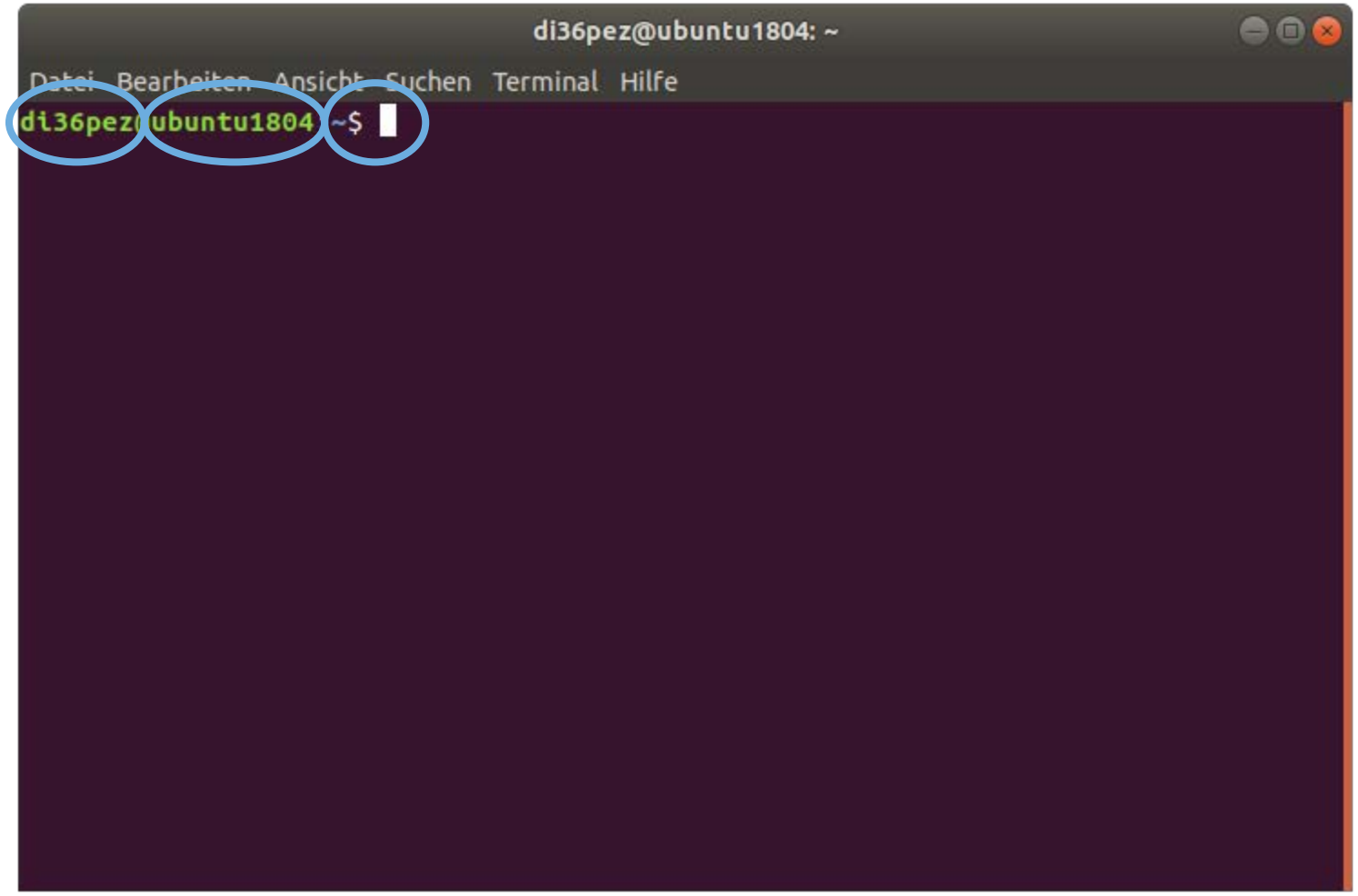
CENTRAL EUROPE MIDDLE EAST SCANDINAVIA AFRICA UK ITALY

Linux totally dominates supercomputers

It finally happened. Today, all 500 of the world's top 500 supercomputers are running Linux.

By Steven J. Vaughan-Nichols for Linux and Open Source | November 14, 2017 -- 20:04 GMT (20:04 GMT) | Topic: Innovation

Server


di36pez@ubuntu1804: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
di36pez@ubuntu1804 ~$

Let’s get started!

Start a Unix-like shell environment…

- GNU/Linux: launch your favorite terminal application
- macOS: launch Terminal
- Windows:
  - Windows 10: Install the Windows Subsystem for Linux https://docs.microsoft.com/en-us/windows/wsl/install-win10
  - Alternatively (on older Windows versions):
    - Install Git BASH (as part of Git for Windows) https://gitforwindows.org/
    - or MobaXterm https://mobaxterm.mobatek.net/
  - Or visit https://bellard.org/jslinux/ in your browser…

https://bellard.org/jslinux/

Browser-based Linux Emulation

JSLinux

Run Linux or other Operating Systems in your browser!

The following emulated systems are available:

CPU	OS (Distribution)	User Interface	VFsync access	Startup Link	VM Config	Comment
x86	Linux 4.12.0 (Buildroot)	Console	Yes	click here	url	
x86	Linux 4.12.0 (Buildroot)	X Window	Yes	click here	url	
x86	Windows 2000	Graphical	No	click here	url	Disclaimer.
x86	FreeDOS	VGA Text	No	click here	url	
riscv32	Linux 4.12.0 (Buildroot)	Console	Yes	click here	url	
riscv32	Linux 4.12.0 (Buildroot)	X Window	Yes	click here	url	
riscv64	Linux 4.12.0 (Buildroot)	Console	Yes	click here	url	gcc not available yet.
riscv64	Linux 4.12.0 (Fedora 25)	Console	No	click here	url	Longer boot time due to larger libraries.

The World is Your Oyster

```
Loading...

Welcome to JS/Linux (x86)

Use 'vflogin username' to connect to your account.
You can create a new account at https://vfsync.org/signup .
Use 'export_file filename' to export a file to your computer.
Imported files are written to the home directory.

[root@localhost ~]#
```

What is “your” username?

What is the hostname of the machine?


```
Loading...

Welcome to JS/Linux (x86)

Use 'vflogin username' to connect to your account.
You can create a new account at https://vfsync.org/signup .
Use 'export_file filename' to export a file to your computer.
Imported files are written to the home directory.

[root@localhost ~]#
```


Let’s explore the file system…

File System Hierarchy Standard (FHS)

- On a Unix-like system (pretty much) everything is a file
- All files and directories appear (somewhere) under the root directory “/” (even if stored on different – possibly – remote devices)
- Use `$ pwd` to get the name of the current working directory
- Use `$ ls` to list all files and directories in the current directory
- Use `$ ls /` to list all files and directories in the root directory
- Use `$ ls /any/other/dir` to list all files and directories in the specified directory

Exploring the File System

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# ls
dos       hello.c
[root@localhost ~]# ls /
bin       etc       lib       linuxrc   mnt       proc      run       sys       usr
dev       home      lib32     media     opt       root      sbin      tmp       var
[root@localhost ~]#
```

/bin: command binaries (e.g. ls)
/etc: configuration files
/home: (regular) users' home directories
/lib: libraries (for binaries in /bin et al.)
/media: mount points for removable media
/mnt: mounted filesystems
/root: home directory for the root user
/sbin: system binaries
/usr: secondary hierarchy for read-only user data
/var: variable, i.e. changing files

- Take a look at the contents of /usr…
- Are any (regular) user home directories on your system?

Detailed Listing of All Files

```
[root@localhost ~]# ls /usr
bin                                libexec
i486-buildroot-linux-uclibc        local
include                            sbin
lib                                share
lib32                              var
lib64
[root@localhost ~]# ls /home
```

Use the `l` and `a` options with `ls` (i.e. `ls -la`) to get a detailed listing of all files in your current (home) directory (we will cover most of this information later).

Can you spot a difference to the previous listing (using just `ls`)?

```
[root@localhost ~]# ls -la
total 20
drwx------    3 root     root          135 Oct 24 19:34 .
drwxrwxrwx   19 root     root          457 Feb 27 13:44 ..
-rw-------    1 root     root            0 Jul  8  2017 .Xauthority
-rwxr-xr-x    1 root     root           28 Jun 24  2017 .xsession
drwxr-xr-x    3 root     root          163 Aug 20  2011 dos
-rw-r--r--    1 root     root          242 Jul 15  2017 hello.c
[root@localhost ~]#
```

General Command Syntax

This kind of input can serve as general example to distinguish different components:

```
$ ls -la /home
```

`ls` is the **command**
with the **options** (also **switches** or **flags**) `-la` and
the **argument** `/home`

Options generally start with either a single dash `-` (as above) or two dashes `--`.

Getting Help

- Two common ways to find out how a command works and which options it accepts…

- Pass the `--help` option to the command:
  `$ ls --help`
- Read a command's manual (man pages), using the `man` command:
  `$ man ls`
  (use the arrow keys to move up and down, press `q` to quit the man page)

- What effect does the `-h` option have on the `ls` command?
- Can you spot other interesting options?

Directories

- Create a new directory in your current (home) directory called „my_dir“:
  `$ mkdir my_dir`
- Change your current working directory to this folder:
  `$ cd my_dir`

Navigating Directories

```
[root@localhost ~]# mkdir my_dir
[root@localhost ~]# cd my_dir/
[root@localhost my_dir]# ls
[root@localhost my_dir]#
```

Notice the changing prompt…

What does the “~” symbol represent?

You could use `cd ..` to move (back) to the parent folder
(a dot . represents the current, two dots .. the parent folder)

Absolute vs. relative paths:
specifying a location with a leading slash `/` indicates a start at the root of the file system (absolute), omitting it leads to an interpretation relative to the current directory

Tip: use the tab key for auto-completion

File Manipulation

- Make sure you‘re located in the my_dir directory created earlier

- Create a new (text) file by “touching” it:
  `$ touch my_file`

- Can you spot the newly created file it in a file listing?
- What‘s the content of this new file? How can you tell?

File Manipulation

```
[root@localhost my_dir]# touch my_file
[root@localhost my_dir]# ls
my_file
[root@localhost my_dir]# ls -la
total 8
drwxr-xr-x    2 root     root           61 Feb 27 14:27 .
drwx------    4 root     root          158 Oct 24 19:34 ..
-rw-r--r--    1 root     root            0 Feb 27 14:27 my_file
[root@localhost my_dir]#
```

Can you spot the size of this (empty) file?

On most systems, you can use editors like e.g. nano, vim or emacs to edit text files directly in the console.

Use nano to modify the existing file (write something to it):

```
$ nano my_file
```

Note the shortcuts along the bottom of the *nano* screen; “^” represents the Control (CTRL) key

Use nano to create another file and write a couple of lines:

```
$ nano another_file.txt
```

Note the file extension: what’s the purpose and effect?

File Manipulation and Redirection

- There is a tool called cat. What does it do?
- “Concatenate FILE(s) to standard output.
  With no FILE, or when FILE is -, read standard input.”
- Use cat to display the contents of my_file

```
$ cat my_file
```

- The shell allows for input/output redirection using > (and <)
- Use cat to write something to nice_file.txt and display it afterwards

```
$ cat > nice_file.txt
write something nice here
and add another line
CTRL+C
$ cat nice_file.txt
```

File Manipulation and Redirection

- Files can be appended using >>

```
$ echo “yet another line of text” >> nice_file.txt
$ cat nice_file.txt
```

- Using << allows for the creation of here documents (input stream literals), the general format is:

```
command << delimiter (commonly EOF)
input stream
delimiter
```

- Try the following:

```
$ tr a-z A-Z << EOF
> all lower case
> o rly?
> EOF
```

Pipes

- Commands can be chained using | (the pipe). It will instruct the shell to use the output of one command directly as input for another command. Pipes can be used consecutively.

```
$ echo “some fancy words” | wc –l
$ echo “some fancy words” | tr “ ” “\n” | wc -l
```

File Manipulation

- Create a copy of “my_file” called “my_file1”:
  `$ cp my_file my_file1`

- Rename/move the copy “my_file1” to “new_file”:
  `$ mv my_file1 new_file`

- Delete the original file “my_file” :
  `$ rm my_file`
  Note: there is no trash bin or undo.

- Take a look at the file listing. What is the expected output? Does it match?

File Manipulation

```
[root@localhost my_dir]# ls
my_file
[root@localhost my_dir]# cp my_file my_file1
[root@localhost my_dir]# ls
my_file   my_file1
[root@localhost my_dir]# mv my_file1 new_file
[root@localhost my_dir]# ls
my_file   new_file
[root@localhost my_dir]# rm my_file
[root@localhost my_dir]# ls
new_file
[root@localhost my_dir]#
```

File Manipulation

- Create two more copies of “new_file”, “01.bak” and “02.bak”

```
$ cp new_file 01.bak
$ cp new_file 02.bak
```

- Move to you home directory.

```
$ cd ..

$ cd
$ cd /path/to/home/dir
```

- Copy “new_file” to your home directory.

```
$ cp my_dir/new_file .
```

- Make a (full) copy of “my_dir” called “another_dir”.

```
$ cp –r my_dir another_dir
```

Wildcards

- Wildcards can be used for character matching (and can be combined)…
  - Zero or more characters -> `*`
    ```
    $ ls -la mydir/n*
    ```
  - Exactly one character -> `?`
    ```
    $ ls -la mydir/0?.bak
    ```

- Count the combined number of words in all files with a file extension

```
$ cat */*.??? | wc -w
```

Searching: grep

- Use `grep` to select lines from text files that match simple patterns, e.g.

```
$ grep something my_dir/nice_file.txt
$ grep “another line” ./*
```

- Consider options like

  -w: Select only those lines containing matches that form whole words
  -n: Prefix (file name and) line number to each match
  -i: Make search case-insensitive
  -v: Invert search, i.e. output non-matching lines

  and many more…

Searching: find

- The `find` command searches for files, e.g.

```
$ find .
$ find . -type d
$ find . -name “*.txt”
$ find . -type f -name “a*” -exec ls -la '{}' \;
```

Shell Scripting

- Use shell scripts to save and re-use commands
- Create a new file myscript.sh containing the line

```
echo “This script is simple.”
```

- Once saved, you can run it explicitly

```
$ bash myscript.sh
```

Shell Scripting

- Modify the script to allow for argument use. Add the line:

```
echo “This $1 is $2.”
```

- Provide the needed arguments when calling the script

```
$ bash myscript.sh “scripting” “getting somewhere”
```

Shell Scripting

- Add a shebang interpreter directive as the first line for direct execution:

```
#!/bin/bash
echo “This script is simple.”
```

- Afterwards, call the script directly

```
$ ./myscript.sh
```

- What is going on?

Ownership and Permissions

```
[root@localhost ~]# ls -la
total 20
drwx------    3 root     root          135 Oct 24 19:34 .
drwxrwxrwx   19 root     root          457 Feb 27 13:44 ..
-rw-------    1 root     root            0 Jul  8  2017 .Xauthority
-rwxr-xr-x    1 root     root           28 Jun 24  2017 .xsession
drwxr-xr-x    3 root     root          163 Aug 20  2011 dos
-rw-r--r--    1 root     root          242 Jul 15  2017 hello.c
[root@localhost ~]#
```

- Every file/directory is owned by a specific user
  (usually the original creator, but this can be changed)
- Every user is member of a (primary) group (and potentially additional ones)
- Notice the two “root” columns:
  the first one is the owner of the respective file/directory
  the second one is the group assigned to the file/directory

Ownership and Permissions

```
[root@localhost ~]# ls -la
total 20
drwx------    3 root     root          135 Oct 24 19:34 .
drwxrwxrwx   19 root     root          457 Feb 27 13:44 ..
-rw-------    1 root     root            0 Jul  8  2017 .Xauthority
-rwxr-xr-x    1 root     root           28 Jun 24  2017 .xsession
drwxr-xr-x    3 root     root          163 Aug 20  2011 dos
-rw-r--r--    1 root     root          242 Jul 15  2017 hello.c
[root@localhost ~]#
```

- Permissions (access rights) for files and directories are managed in three different classes: user, group and others
- Three specific permissions apply to each class:
  - read (a file or the names of files in a directory)
  - write (modify a file or the entries of a directory)
  - execute (a file or access file contents of a directory)

Shell Scripting

- Use chmod to change file permissions/mode bits

```
$ chmod +x myscript.sh
```

- Afterwards, call the script directly

```
$ ./myscript.sh
```

File Manipulation

- Finally, let‘s clean up: completely delete “another_dir”.

```
$ rm –r another_dir
```

Again: there is no trash bin or undo.

- What about `rmdir`?

Additional material

Visit e.g. https://linuxjourney.com/ for interactive tutorials

One more thing: Environment Variables

- Named values that can influence how programs are run
  (e.g. by providing context information)
- Use the command `env` to print these variables in the current environment
- To print a specific environment variable, use the `echo $VARNAME` command
  e.g. `$ echo $HOME`
- To set (or change) a specific environment variable,
  use the `export VARNAME=<value>` command
- On our HPC clusters, we use a “module system” to adjust these environment variables, this e.g. allows for providing/running different versions of the same application.