HPC.NRW

INTRODUCTION TO LINUX

(in an HPC context)

Version 20.09 | HPC.NRW Competence Network

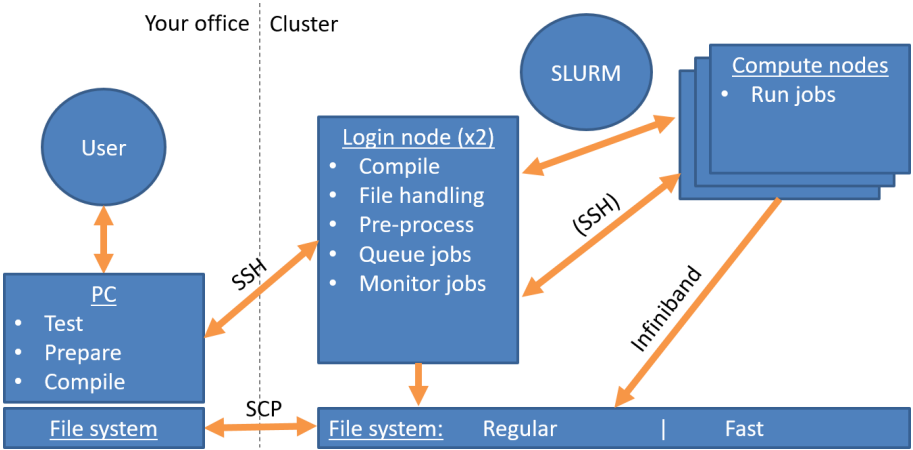
INTRODUCTION AND AGENDA

HPC.NRW Competence Network

INTRODUCTION TO LINUX

LOGICAL STRUCTURE OF A CLUSTER

HPC.NRW



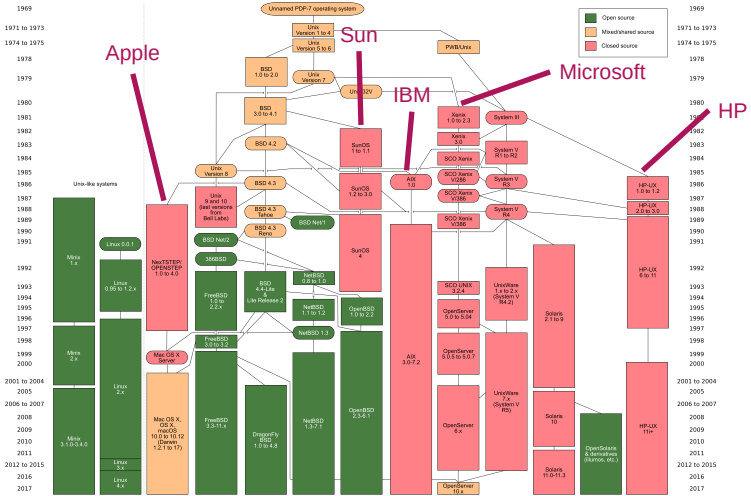
A BRIEF HISTORY OF UN*X

HPC.NRW Competence Network

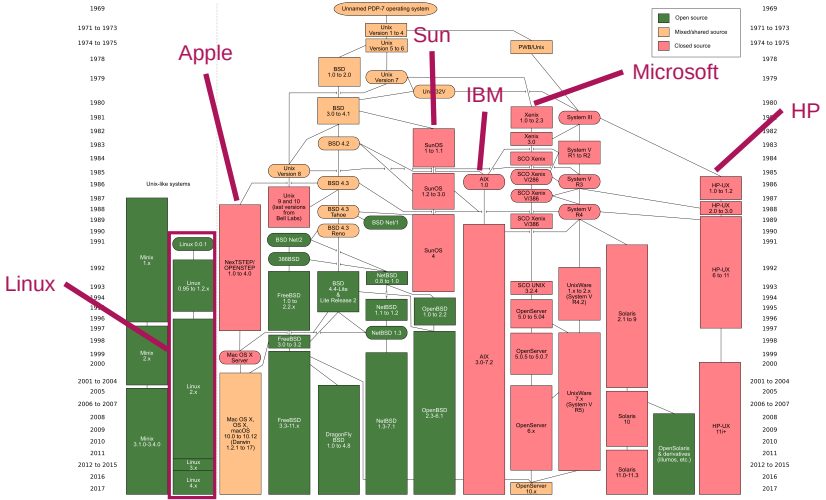
INTRODUCTION TO LINUX

- 1969: Unix (Bell Laboratories)
  - Written in C
  - Already a successor to Multics
  - Over time, many variations
- 1990: POSIX Standard
  - Interface that all Unix systems implement
  - Adopted by Unix-like systems (including Linux)
  - Already many tools that we still use

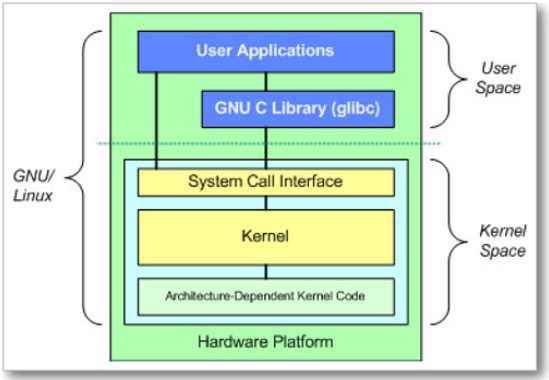
HISTORICAL BACKGROUND



HISTORICAL BACKGROUND



- Two separate initiatives:
  - GNU (GNU's Not Unix)
    - 1984: Richard Stallman and others
  - Linux
    - 1991: Linus Torvals
- Nowadays: GNU/Linux:
  - Linux kernel
  - GNU utilities
- Many distributions (**distros**)



SOME IMPORTANT DISTROS

HPC.NRW

- High reliability (e.g. servers): Red Hat Enterprise Linux
  - Developer “playground”: Fedora
  - Community variant: CentOS (HoRUS cluster)
- User-friendliness: Ubuntu
  - Community variant: Mint
  - “Parent”: Debian
- Workplace (especially Germany): Suse
- Specialized: e.g. Kali Linux (hacking tools)
  - Also runnable without installation

POPULARITY OF DISTROS

HPC.NRW

- Computers with Linux:
  - **500 out of the Top 500 supercomputers (2020)**
  - (Web) servers: 95 %
  - Mobile devices: 60 – 80 % of mobile devices (almost all Android)
  - Desktop PCs: 1 – 2 %

- Popular desktop distros (no good figures):
  - Linux Mint
  - Ubuntu

WHICH LINUX SHOULD I USE

HPC.NRW

- In principle: any Linux will do
  - So much for the theory …
- What do I want to use it for?
  - Server vs. desktop
  - Stability vs. flexibility
  - Easy to learn vs. lots of features
- What software comes with it/is available?
  - Legal status of packages
- What is support/maintenance like?
  - Including documentation

THE COMMAND LINE

HPC.NRW Competence Network

INTRODUCTION TO LINUX

– A line where you type commands

COMMAND LINE

HPC.NRW

- A line where you type commands
- Advantages
  - Simple (nearly always works)
  - Fast (if you know the commands)
  - Easy to program (in comparison to GUI)

COMMAND LINE

HPC.NRW

- A line where you type commands
- Advantages
  - Simple (nearly always works)
  - Fast (if you know the commands)
  - Easy to program (in comparison to GUI)
- Disadvantage
  - lots of memorizing (no visual exploration)

COMMAND LINE

HPC.NRW

- A line where you type commands
- Advantages
  - Simple (nearly always works)
  - Fast (if you know the commands)
  - Easy to program (in comparison to GUI)
- Disadvantage
  - lots of memorizing (no visual exploration)
- Other terms:
  - CLI (command line interface)
  - Console / Terminal
  - Shell

WHY THE NAME "SHELL"

HPC.NRW

- Isolates execution environments

- Multiple layers of shell possible

WHY THE NAME "SHELL"

HPC.NRW

- Isolates execution environments
- Multiple layers of shell possible
- Important for user: be aware where you are
  - Changes to environment not available in parent shell by default
  - Children processes get stopped if parent process ends

COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

HPC.NRW

COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

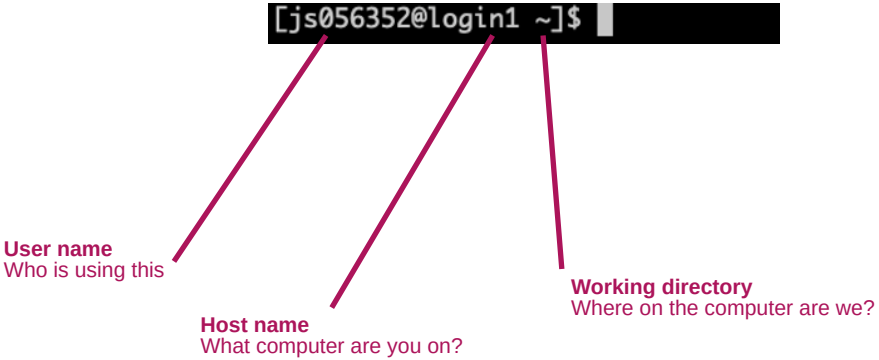
HPC.NRW

User name
Who is using this

Host name
What computer are you on?

COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

HPC.NRW



COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

HPC.NRW

Command prompt
- Indicates system is ready to receive input
- `$` : convention (also `>`)

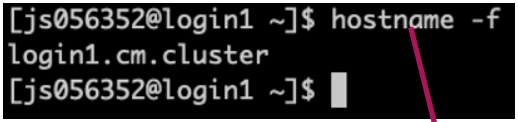
User name
Who is using this

Host name
What computer are you on?

Working directory
Where on the computer are we?

COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

HPC.NRW

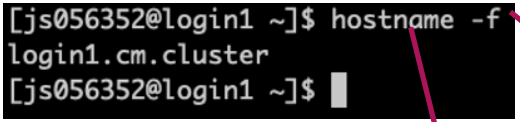


Command

- What to do
- Typed by user

COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

HPC.NRW

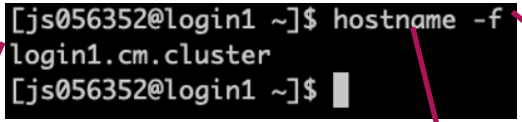


Options
- Command-specific
- Several conventions

Command
- What to do
- Typed by user

COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

HPC.NRW



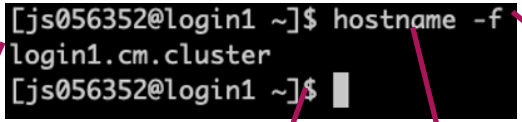
Options
- Command-specific
- Several conventions

Command
- What to do
- Typed by user

Output
- What the command returns
- No rules

COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

HPC.NRW



Options
- Command-specific
- Several conventions

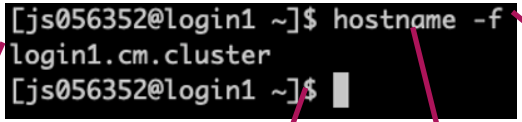
New command prompt
Execution completed

Command
- What to do
- Typed by user

Output
- What the command returns
- No rules

COMMON ELEMENTS OF THE CONSOLE PROMPT STATEMENT

HPC.NRW



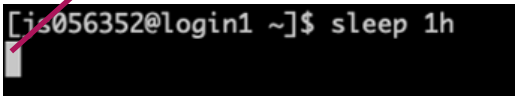
Options
- Command-specific
- Several conventions

New command prompt
Execution completed

Command
- What to do
- Typed by user

Output
- What the command returns
- No rules

Running



<Enter> run command

<Enter> run command

<Up-Arrow> navigate command history back in time

<Enter> run command

<Up-Arrow> navigate command history back in time

<Down-Arrow> navigate command history forward in time

<Enter> run command

<Up-Arrow> navigate command history back in time

<Down-Arrow> navigate command history forward in time

<Tab> auto-completion (a.k.a. “tab completion”)

- Completes to common prefix of multiple options
- More than one possibility: nothing shown
- Second Tab to list possible completions

HPC.NRW

<Enter> run command

<Up-Arrow> navigate command history back in time

<Down-Arrow> navigate command history forward in time

<Tab> auto-completion (a.k.a. “tab completion”)

- Completes to common prefix of multiple options
- More than one possibility: nothing shown
- Second Tab to list possible completions

<Ctrl-C> abort current command.

- Always case-sensitive
  - Popular source for errors

- Always case-sensitive
  - Popular source for errors

- Command line options:
  - Usually start with dash
  - Common convension: single dash for “short options”, double dash for “long options”
    - Example: `sbatch --time 0:30:00` is identical to `sbatch -t 0:30:00`
  - Note: specific commands may deviate from convention

HOW TO FIND CORRECT COMMAND/OPTIONS

HPC.NRW

- Internet (seriously)
  - Very extensive community
  - Stack Overflow/Stack Exchange

HOW TO FIND CORRECT COMMAND/OPTIONS

HPC.NRW

- Internet (seriously)
  - Very extensive community
  - Stack Overflow/Stack Exchange

- Man page:
  - `man <command name>`

HOW TO FIND CORRECT COMMAND/OPTIONS

HPC.NRW

- Internet (seriously)
  - Very extensive community
  - Stack Overflow/Stack Exchange
- Man page:
  - `man <command name>`
- Built-in help:
  - Often `-h` or `--help` option
  - Often identical to man page

LINUX DIRECTORY STRUCTURE

HPC.NRW Competence Network

INTRODUCTION TO LINUX

DIRECTORY STRUCTURE

HPC.NRW

- Directory tree structure different from Windows
  - No drive letters (`C:\`)
  - Top level (mostly) identical on every Linux system
  - “Mounting points”: location of hard drive in tree structure

DIRECTORY STRUCTURE

HPC.NRW

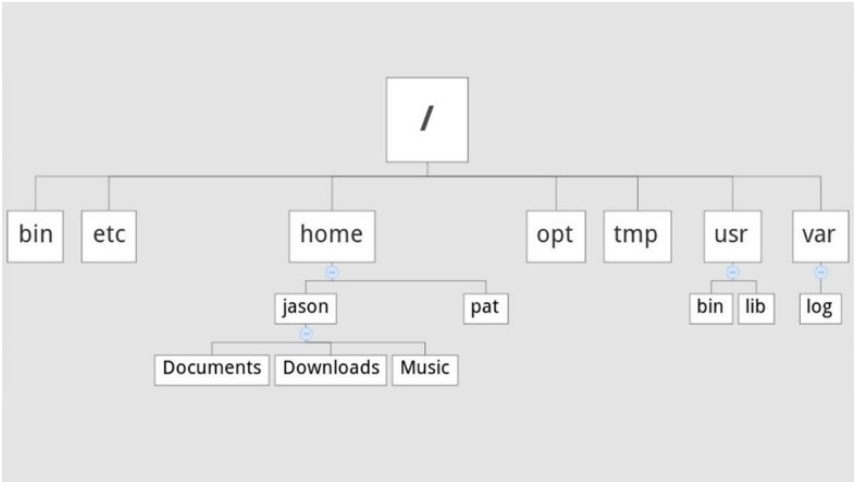
- Directory tree structure different from Windows
  - No drive letters (`C:\`)
  - Top level (mostly) identical on every Linux system
  - “Mounting points”: location of hard drive in tree structure
- “Path”: location inside file system
  - Example:
  - Absolute path (starts with `/`)
  - Relative path: relative to (current) **working directory**

DIRECTORY STRUCTURE

HPC.NRW

- Directory tree structure different from Windows
  - No drive letters (`C:\`)
  - Top level (mostly) identical on every Linux system
  - “Mounting points”: location of hard drive in tree structure
- “Path”: location inside file system
  - Example:
  - Absolute path (starts with `/`)
  - Relative path: relative to (current) **working directory**
- Print working directory: `pwd`

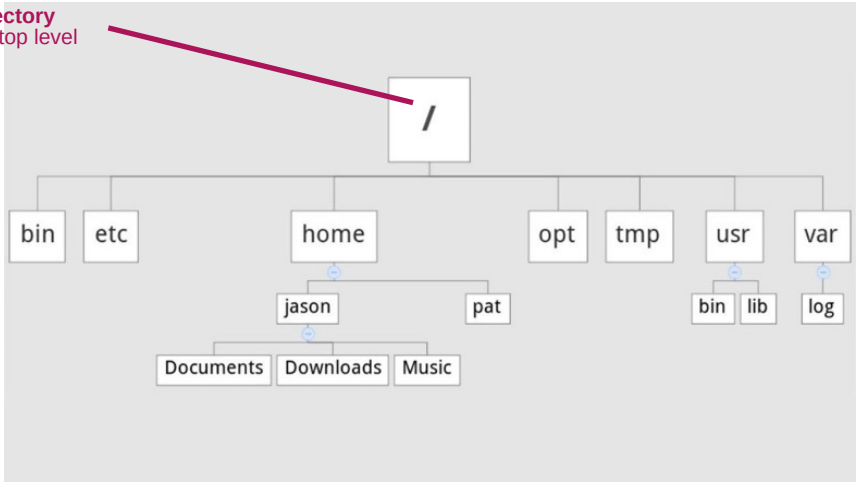
EXAMPLE DIRECTORY STRUCTURE



EXAMPLE DIRECTORY STRUCTURE

HPC.NRW

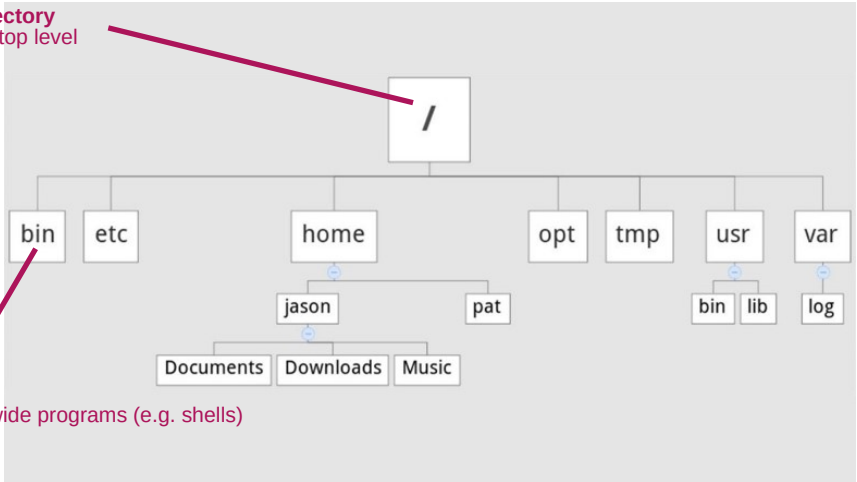
Root directory
Absolute top level



EXAMPLE DIRECTORY STRUCTURE

HPC.NRW

Root directory
Absolute top level

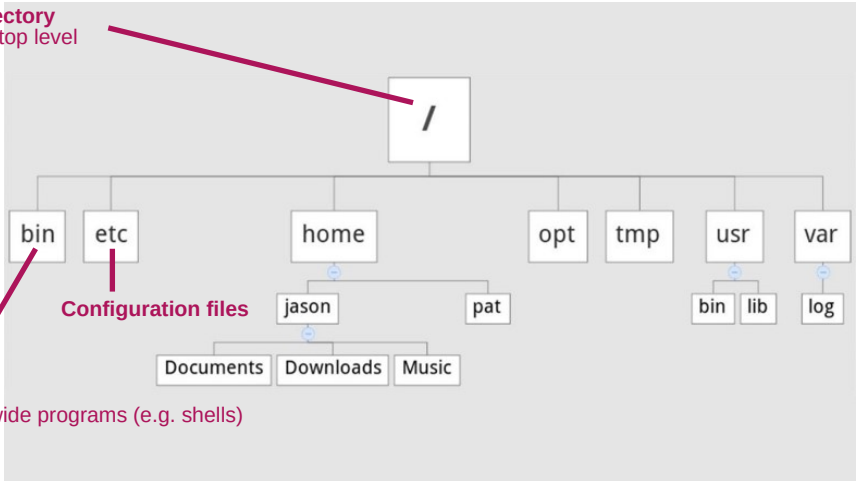


Binaries
System-wide programs (e.g. shells)

EXAMPLE DIRECTORY STRUCTURE

HPC.NRW

Root directory
Absolute top level



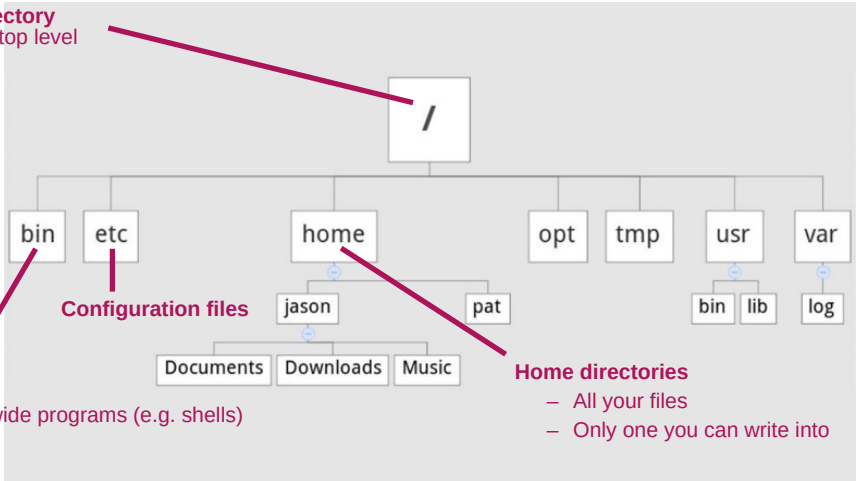
Configuration files

Binaries
System-wide programs (e.g. shells)

EXAMPLE DIRECTORY STRUCTURE

HPC.NRW

Root directory
Absolute top level



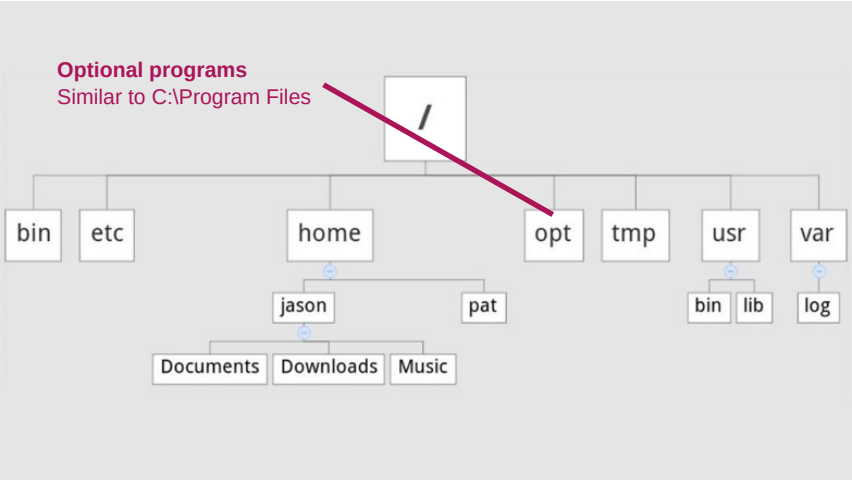
Configuration files

Binaries
System-wide programs (e.g. shells)

Home directories
- All your files
- Only one you can write into

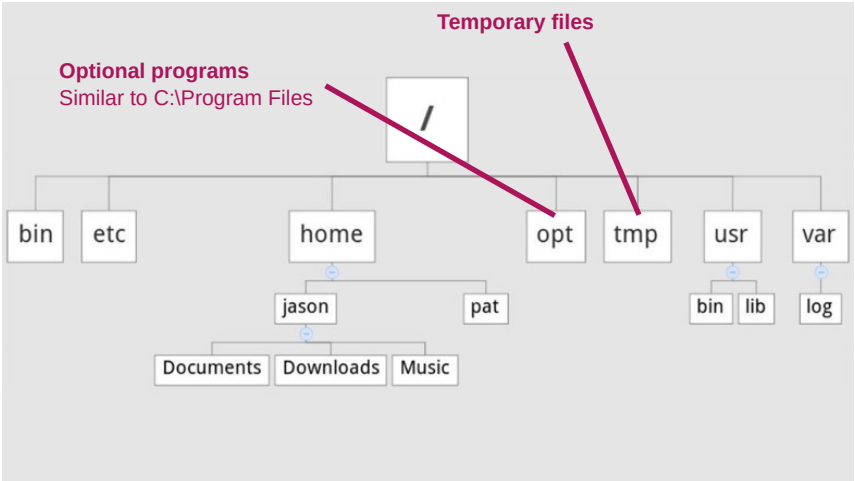
EXAMPLE DIRECTORY STRUCTURE

HPC.NRW

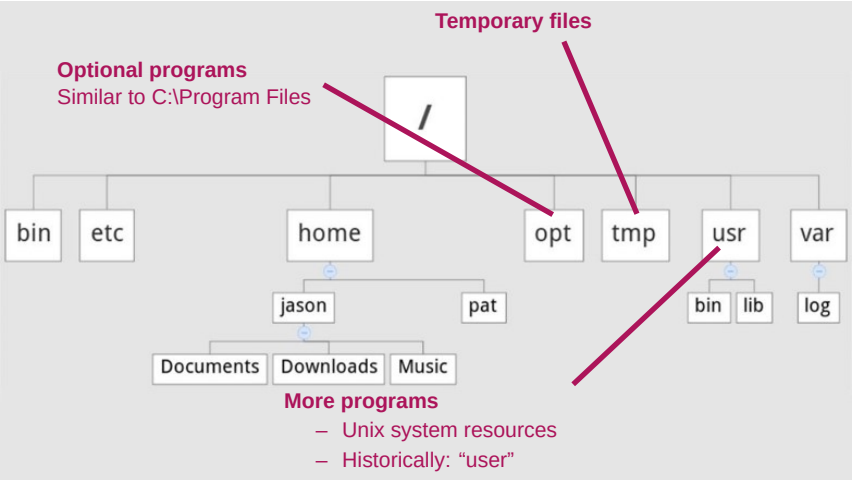


EXAMPLE DIRECTORY STRUCTURE

HPC.NRW

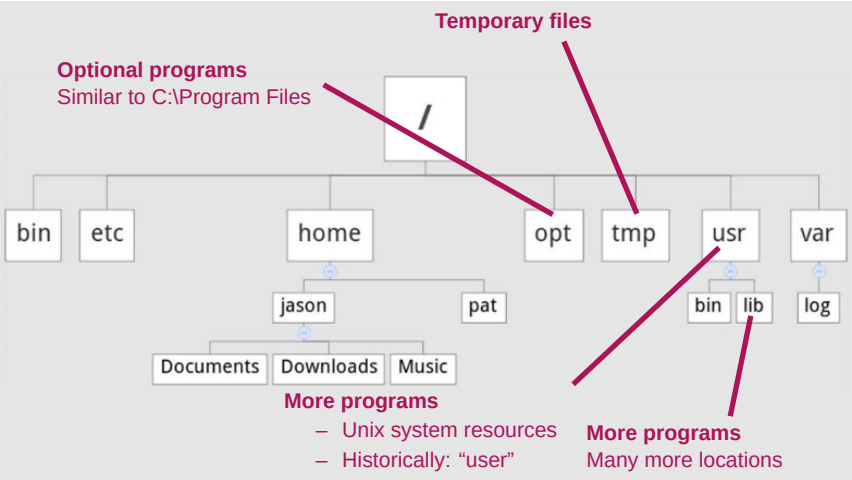


EXAMPLE DIRECTORY STRUCTURE



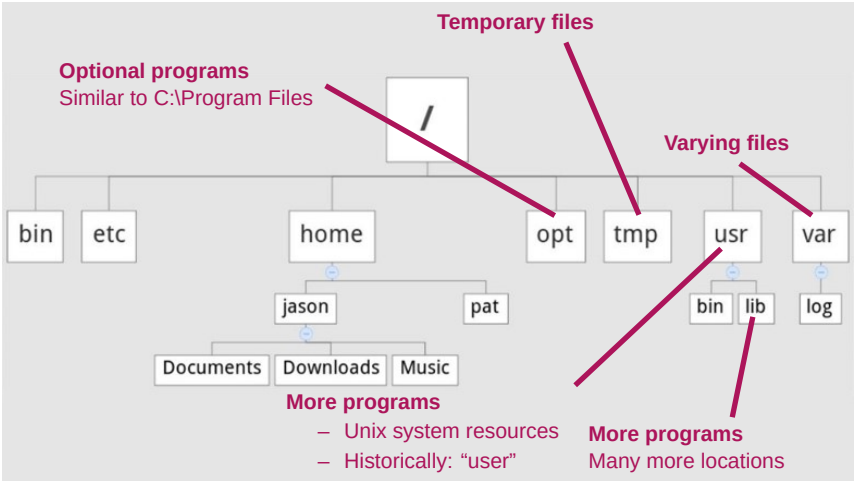
EXAMPLE DIRECTORY STRUCTURE

HPC.NRW



EXAMPLE DIRECTORY STRUCTURE

HPC.NRW



MORE ABOUT FILES AND DIRECTORIES

HPC.NRW

- Linux principle: everything is a file
  - `/dev`: Device files
  - `/proc`: System information files

MORE ABOUT FILES AND DIRECTORIES

HPC.NRW

- Linux principle: everything is a file
  - `/dev`: Device files
  - `/proc`: System information files
- (Almost) every command is a program or script somewhere

  `which <Commandname>` to see

MORE ABOUT FILES AND DIRECTORIES

HPC.NRW

- Linux principle: everything is a file
  - `/dev`: Device files
  - `/proc`: System information files
- (Almost) every command is a program or script somewhere
  `which <Commandname>` to see
- Special abbreviations for directories:
  - `.` (period): current directory
  - `..` (two periods): parent directory
  - `~` (tilde sign): your home directory

- `cd` Command (change directory)
  - Part of POSIX standard

- `cd` Command (change directory)
  - Part of POSIX standard
- Usage: `cd <Path>`
  - Can be relative or absolute
  - Must have at least execute permissions
    - Possible to execute but not read a file
  - May be special character, e.g. `cd ..` (parent directory)

- `cd` Command (change directory)
  - Part of POSIX standard
- Usage: `cd <Path>`
  - Can be relative or absolute
  - Must have at least execute permissions
    - Possible to execute but not read a file
  - May be special character, e.g. `cd ..` (parent directory)
- Common mistake: `cd..` (no space in between)
  - Often defined as an alias to mitigate typo

- `ls` Command
  - Short for “List”
  - List directory contents

- `ls` Command
  - Short for “List”
  - List directory contents
  - One of the most common commands in Linux (like `dir` in Windows)

- `ls` Command
  - Short for “List”
  - List directory contents
  - One of the most common commands in Linux (like `dir` in Windows)
  - `ls -l` is so common that it often has its own shortcut: `ll`
  - Can also show hidden files with `-a`
  - Can sort results, e.g. `-t` to sort by time modified

CONSOLE-SPECIFIC COMMANDS AND SHORTCUTS

HPC.NRW

<Middle Mouse> paste selected text
  - NOT Ctrl-C / Ctrl-V, see below

<Ctrl-C> stop current command

<Ctrl-Z> suspend current command

<Ctrl-D> send “End-of-File” to application
  - Will usually quit console when on an empty command line

- Quit console with `exit` (SSH connection: back to local console)
- Clear screen: `clear` command

FIRST, A WARNING

HPC.NRW

- Console has no “Undo” button
- Usually no “Are you sure you want to delete” dialog

FIRST, A WARNING

HPC.NRW

- Console has no “Undo” button
- Usually no “Are you sure you want to delete” dialog
- If root: can theoretically destroy entire system

FIRST, A WARNING

HPC.NRW

- Console has no “Undo” button
- Usually no “Are you sure you want to delete” dialog
- If root: can theoretically destroy entire system

- Never run a command which you don’t understand
  - “Lol, try `sudo rm -rf /` ” – many idiots on the internet

FIRST, A WARNING

HPC.NRW

- Console has no “Undo” button
- Usually no “Are you sure you want to delete” dialog
- If root: can theoretically destroy entire system

- Never run a command which you don’t understand
  - “Lol, try `sudo rm -rf /` ” – many idiots on the internet
- Make sure you are in the right directory
- Make sure you are not root unless necessary
- Check for spelling errors

FILES

HPC.NRW Competence Network

INTRODUCTION TO LINUX

- Linux: extensions do not matter
  - But: conventions to help humans
  - Some programs also look at extensions

- Linux: extensions do not matter
  - But: conventions to help humans
  - Some programs also look at extensions
- Most important: text file or not?
  - Configuration files
  - Scripts
  - System information files

- Linux: extensions do not matter
  - But: conventions to help humans
  - Some programs also look at extensions
- Most important: text file or not?
  - Configuration files
  - Scripts
  - System information files
- Binary file: generally not searchable
- Use `file <filename>` to identify file type

FILE MANIPULATION COMMANDS

HPC.NRW

- Simple commands to handle files
  - Most also work on directories

FILE MANIPULATION COMMANDS

HPC.NRW

- Simple commands to handle files
  - Most also work on directories

- You already know `ls`

FILE MANIPULATION COMMANDS

HPC.NRW

- Simple commands to handle files
  - Most also work on directories
- You already know `ls`
- Rename file/directory: `mv <oldname> <newname>` (move)

FILE MANIPULATION COMMANDS

HPC.NRW

- Simple commands to handle files
  - Most also work on directories
- You already know `ls`
- Rename file/directory: `mv <oldname> <newname>` (move)
- Copy file/directory: `cp <filename> <newname>` (copy)
  - Also needs `-r` for directories

– Create directory: `mkdir <dirname>`

- Create directory: `mkdir <dirname>`
- Create empty file: `touch filename`
  - Updates access time on an existing file

- Create directory: `mkdir <dirname>`
- Create empty file: `touch filename`
  - Updates access time on an existing file
- Remove file/directory: `rm <filename>`
  - Check access permissions!
  - To delete content of subdirectories: `rm -r` (recursive)
  - Common option: `-f` (force) → never prompt for confirmation

- Select more than one file/directory
- Option to specify patterns: wildcards
  - Also called globbing

WILD CARDS

HPC.NRW

- Select more than one file/directory
- Option to specify patterns: wildcards
  - Also called globbing
- Most important
  - `*` zero or more characters
  - `?` exactly one character
  - `[]` range of characters

SEARCHING FOR FILES

HPC.NRW

- Use `find` command
- Syntax: `find <targetdir> <options>`
  - Example: `find . -name "ex1.txt" -type f`

SEARCHING FOR FILES

HPC.NRW

- Use `find` command
- Syntax: `find <targetdir> <options>`
  - Example: `find . -name "ex1.txt" -type f`
- Allows very complex searches
  - Wildcards
  - Only files modified after X

SEARCHING FOR FILES

HPC.NRW

- Use `find` command
- Syntax: `find <targetdir> <options>`
  - Example: `find . -name "ex1.txt" -type f`
- Allows very complex searches
  - Wildcards
  - Only files modified after X
- Allows executing command for every found file: `-exec`

- Wildcards: common source of problems, especially in scripts
  - Expanded by shell before being given to program
  - Problem not limited to `find` command

- Wildcards: common source of problems, especially in scripts
  - Expanded by shell before being given to program
  - Problem not limited to `find` command
- Example: `find` command `-name` option

```
$ find . -type f -name *test*
```

  - The `find` command is handed multiple names, cannot handle this

- Wildcards: common source of problems, especially in scripts
  - Expanded by shell before being given to program
  - Problem not limited to `find` command
- Example: `find` command `-name` option

```
$ find . -type f -name *test*
```

  - The `find` command is handed multiple names, cannot handle this
- Fix: `$ find . -type f -name "*test*"`
  - Now string with wildcards is handed to `find` command

SEARCHING FILES

HPC.NRW Competence Network

INTRODUCTION TO LINUX

SEARCHING FILE CONTENTS

HPC.NRW

- Use `grep` command
- Syntax: `grep <options> <string> <filename>`
  - Example `grep -i -r "test" example*.txt`

SEARCHING FILE CONTENTS

HPC.NRW

- Use `grep` command
- Syntax: `grep <options> <string> <filename>`
  - Example `grep -i -r "test" example*.txt`
- Like `find`, very powerful due to options + wildcards

- Use `grep` command
- Syntax: `grep <options> <string> <filename>`
  - Example `grep -i -r "test" example*.txt`
- Like `find`, very powerful due to options + wildcards
- Common options:
  - `-r` Recursive (include subdirectories)
  - `-i` Ignore upper/lower case
  - `-I` Ignore binary files (capital i)

- Many different ways to display and edit text
  - Simplest: `cat` command
    - Outputs contents of a text file to console

- Many different ways to display and edit text
  - Simplest: `cat` command
    - Outputs contents of a text file to console
  - More advanced: `less` command
    - Allows going back and forth
    - Also used by man pages

- Many different ways to display and edit text
  - Simplest: `cat` command
    - Outputs contents of a text file to console
  - More advanced: `less` command
    - Allows going back and forth
    - Also used by man pages
  - Others:
    - `head`: display first lines
    - `tail`: display last lines

- Console has three main ways of communicating with process (so-called streams)
  - Standard input (`stdin`)
  - Standard output (`stdout`)
  - Standard error (`stderr`)

- Console has three main ways of communicating with process (so-called streams)
  - Standard input (`stdin`)
  - Standard output (`stdout`)
  - Standard error (`stderr`)

- `stdin`: what you type into console

- Console has three main ways of communicating with process (so-called streams)
  - Standard input (`stdin`)
  - Standard output (`stdout`)
  - Standard error (`stderr`)

- `stdin`: what you type into console

- `stdout` + `stderr`: what you see in console
  - Two separate streams so you can separate error messages from normal output

SIDE NOTE: WHY ARE THEY CALLED "STREAMS"

HPC.NRW

- What is a “stream” in computing terms?
  - Intermediate storage
  - Stuff is put into it, stuff gets taken out
  - Those two may happen at any time, overlapping

SIDE NOTE: WHY ARE THEY CALLED "STREAMS"

HPC.NRW

- What is a “stream” in computing terms?
  - Intermediate storage
  - Stuff is put into it, stuff gets taken out
  - Those two may happen at any time, overlapping

- Example: streaming video
  - Video gets partially downloaded, you can already view it
  - You can pause, jump etc.

SIDE NOTE: WHY ARE THEY CALLED "STREAMS"

HPC.NRW

- What is a “stream” in computing terms?
  - Intermediate storage
  - Stuff is put into it, stuff gets taken out
  - Those two may happen at any time, overlapping
- Example: streaming video
  - Video gets partially downloaded, you can already view it
  - You can pause, jump etc.
- In console: text gets written into stream and taken out
  - Input and output can be (re)directed to other sources/targets

– Input/output streams can be redirected

- Input/output streams can be redirected
- Redirect stdout

```
command > filename
```

- Input/output streams can be redirected
- Redirect stdout

```
command > filename
```

- Redirect stderr

```
command 2> filename
```

CONSOLE INPUT AND OUTPUT

HPC.NRW

- Input/output streams can be redirected
- Redirect stdout

  ```
  command > filename
  ```

- Redirect stderr

  ```
  command 2> filename
  ```

- Redirect stdin

  ```
  command < filename
  ```

CONSOLE INPUT AND OUTPUT

HPC.NRW

- Input/output streams can be redirected
- Redirect stdout

  ```
  command > filename
  ```
- Redirect stderr

  ```
  command 2> filename
  ```
- Redirect stdin

  ```
  command < filename
  ```
- Use output of one command as input to another: pipe symbol

  ```
  command1 | command2
  ```

- Stream redirection can do even more

  `command >> filename` will append to file without overwriting

  - Streams are numbered:

    0: `stdin`, **1**: `stdout`, **2**: `stderr`

  - Examples:

    ```
    command > out.log 2> err.log
    ```

    ```
    command 2>&1 > out_err.log
    ```

USE GREP ON SEARCH RESULTS

HPC.NRW

- Common situation:
  - Command with a lot of text output
  - You are looking for something inside output

USE GREP ON SEARCH RESULTS

HPC.NRW

- Common situation:
  - Command with a lot of text output
  - You are looking for something inside output
- Solution: pipe output into `grep`

```
$ ll | grep -i test
```

USE GREP ON SEARCH RESULTS

HPC.NRW

- Common situation:
  - Command with a lot of text output
  - You are looking for something inside output
- Solution: pipe output into `grep`

```
$ ll | grep -i test
```

- Note that there is no file specified in the `grep` call

USE GREP ON SEARCH RESULTS

HPC.NRW

- Common situation:
  - Command with a lot of text output
  - You are looking for something inside output
- Solution: pipe output into `grep`

```
$ ll | grep -i test
```

- Note that there is no file specified in the `grep` call
- See how pipes can be useful?

USERS AND GROUPS

- Linux is a *multi-user* system
  - Everyone should only be able to access own files
    - Others only see / change what you want them to
  - Some files / directories should only be accessible to admins

USERS AND GROUPS

HPC.NRW

- Linux is a *multi-user* system
  - Everyone should only be able to access own files
    - Others only see / change what you want them to
  - Some files / directories should only be accessible to admins

- Everyone is logged in as a specific user (account)
  - Every user has certain permissions

USERS AND GROUPS

HPC.NRW

- Linux is a *multi-user* system
  - Everyone should only be able to access own files
    - Others only see / change what you want them to
  - Some files / directories should only be accessible to admins

- Everyone is logged in as a specific user (account)
  - Every user has certain permissions

- Only admins can set permissions for others

Each file and directory has certain permissions

- Determines what you can do
  - *You can't break what you can't use!*

Each file and directory has certain permissions

- Determines what you can do
  - *You can't break what you can't use!*
- `root` user (superuser) can do everything

Each file and directory has certain permissions

- Determines what you can do
  - *You can't break what you can't use!*
- `root` user (superuser) can do everything
- Users may get temporary root permissions

  `sudo <Command>`

Each file and directory has certain permissions

- Determines what you can do
  - *You can't break what you can't use!*
- `root` user (superuser) can do everything
- Users may get temporary root permissions

  `sudo <Command>`
- Users belong to groups
  - Each user has a primary group

- Read:
  - Who can read contents of file/directory

- Read:
  - Who can read contents of file/directory
- Write:
  - Who can change contents of file/directory

- Read:
  - Who can read contents of file/directory
- Write:
  - Who can change contents of file/directory
- Execute:
  - File: who can execute file (like any program)
  - Directory: who can traverse directory
    - Can execute files inside but not see them

EXAMPLE OUTPUT OF LS -L

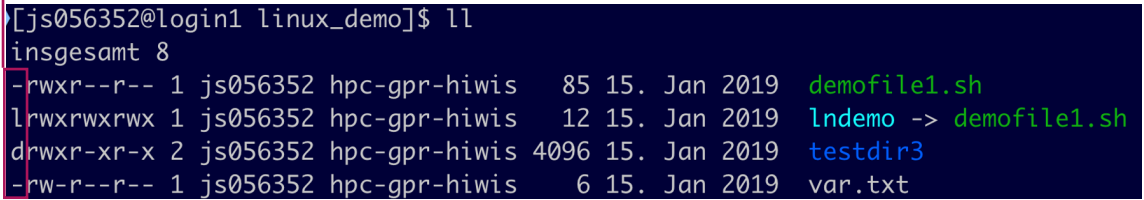
HPC.NRW

```
[js056352@login1 linux_demo]$ ll
insgesamt 8
-rwxr--r-- 1 js056352 hpc-gpr-hiwis   85 15. Jan 2019  demofile1.sh
lrwxrwxrwx 1 js056352 hpc-gpr-hiwis   12 15. Jan 2019  lndemo -> demofile1.sh
drwxr-xr-x 2 js056352 hpc-gpr-hiwis 4096 15. Jan 2019  testdir3
-rw-r--r-- 1 js056352 hpc-gpr-hiwis    6 15. Jan 2019  var.txt
```

EXAMPLE OUTPUT OF LS -L

HPC.NRW

Is it a file, link (`l`) or directory (`d`)?



EXAMPLE OUTPUT OF LS -L

HPC.NRW

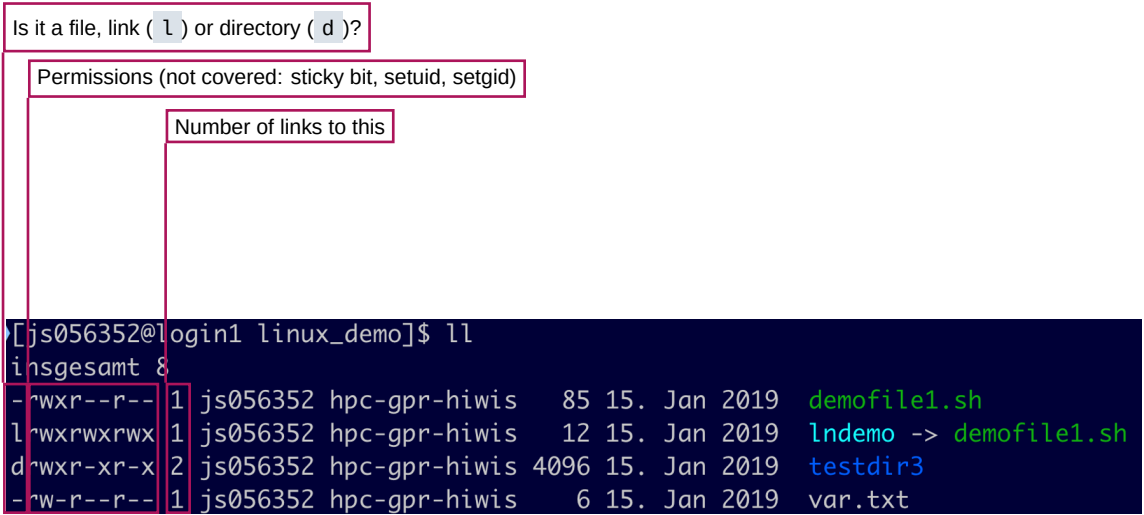
Is it a file, link (`l`) or directory (`d`)?

Permissions (not covered: sticky bit, setuid, setgid)

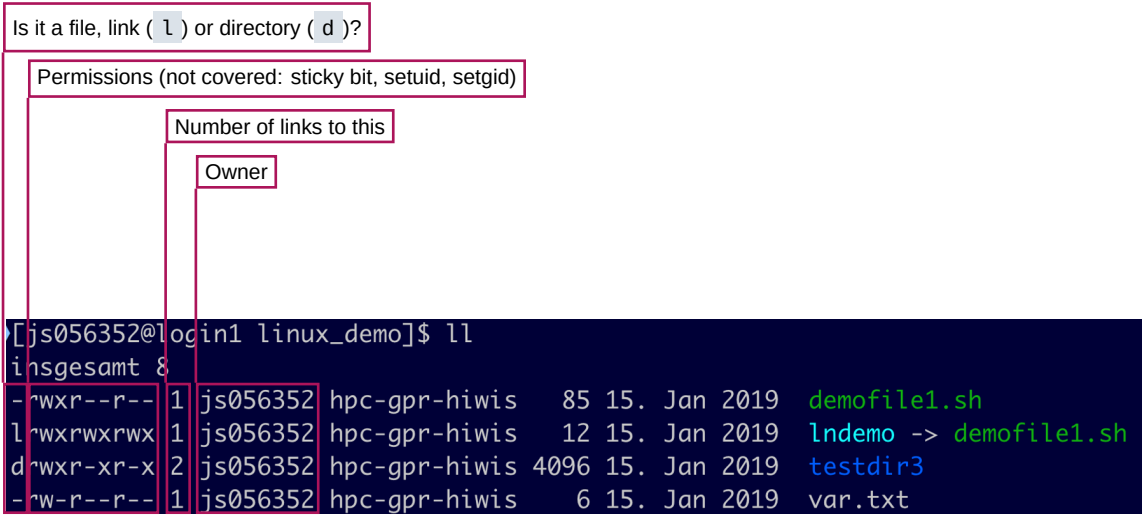
```
[js056352@login1 linux_demo]$ ll
insgesamt 8
-rwxr--r-- 1 js056352 hpc-gpr-hiwis   85 15. Jan 2019  demofile1.sh
lrwxrwxrwx 1 js056352 hpc-gpr-hiwis   12 15. Jan 2019  lndemo -> demofile1.sh
drwxr-xr-x 2 js056352 hpc-gpr-hiwis 4096 15. Jan 2019  testdir3
-rw-r--r-- 1 js056352 hpc-gpr-hiwis    6 15. Jan 2019  var.txt
```

EXAMPLE OUTPUT OF LS -L

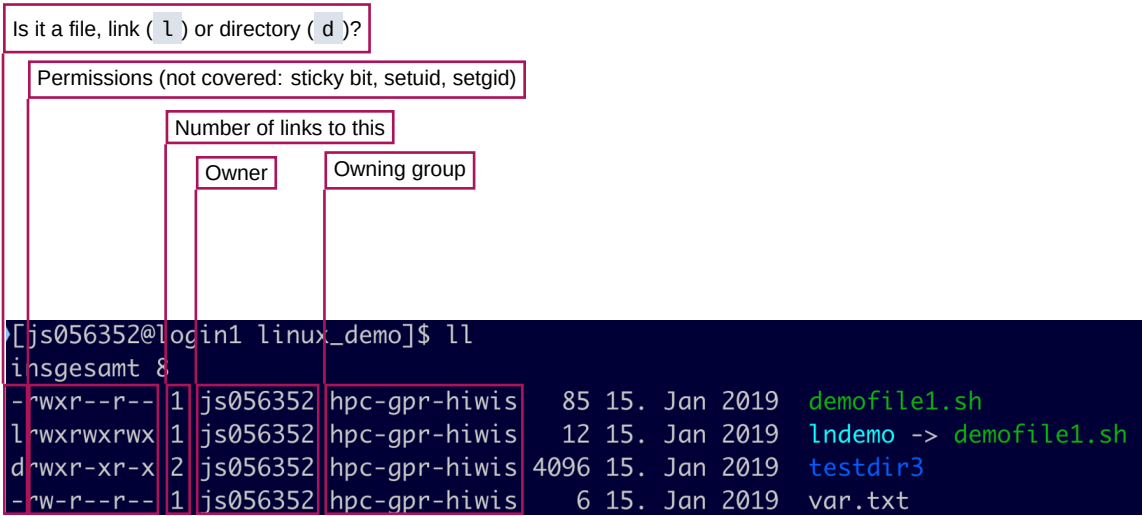
HPC.NRW



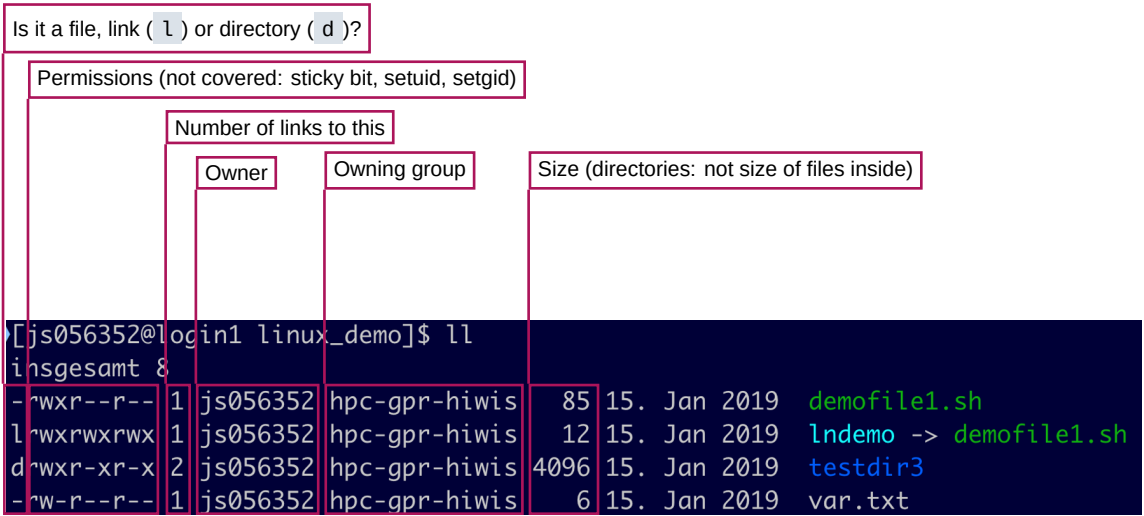
EXAMPLE OUTPUT OF LS -L



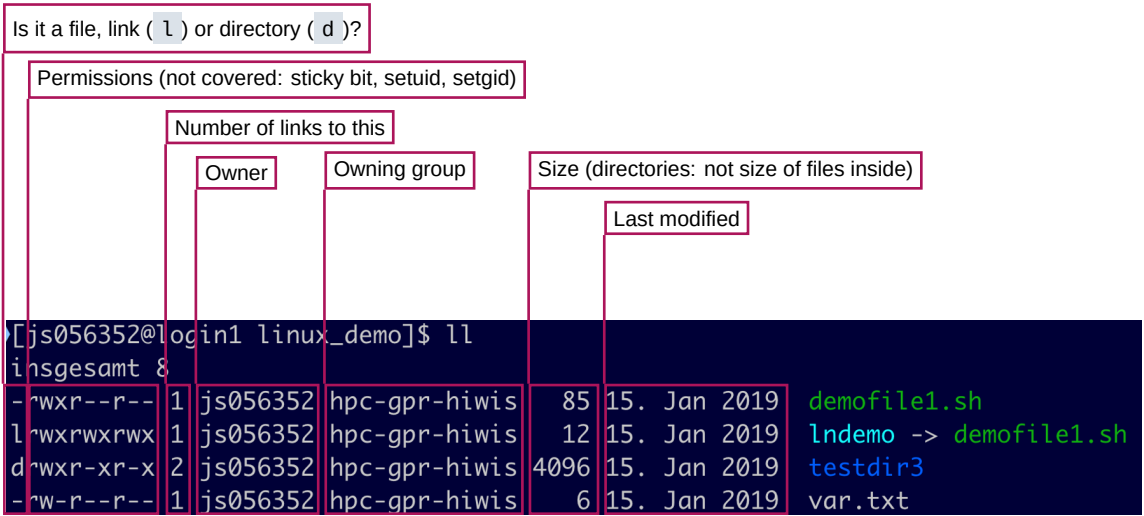
EXAMPLE OUTPUT OF LS -L



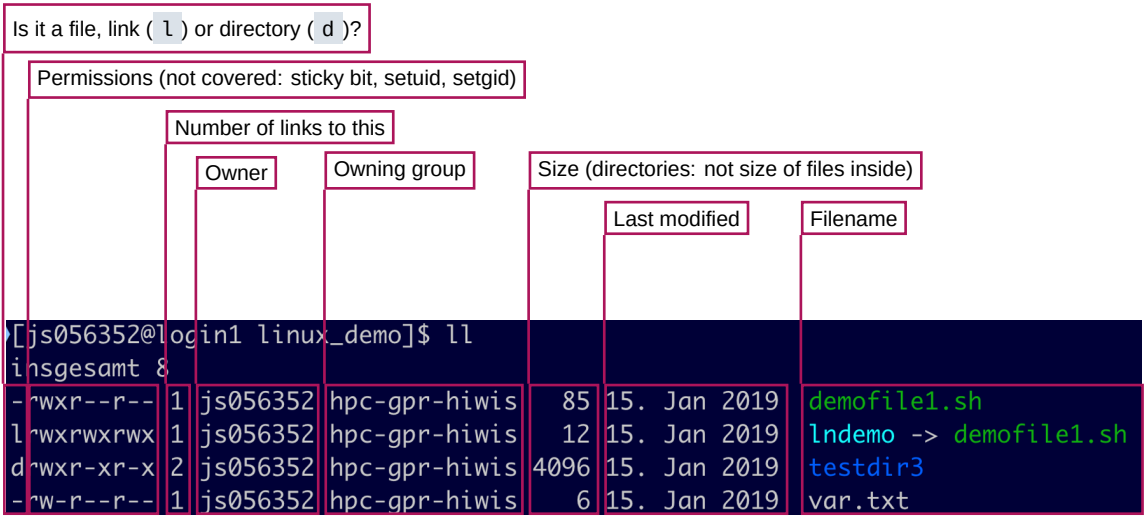
EXAMPLE OUTPUT OF LS -L



EXAMPLE OUTPUT OF LS -L



EXAMPLE OUTPUT OF LS -L



```
drwxr-xr-x 2 js056352 hpc-gpr-hiwis
-rw-r--r-- 1 js056352 hpc-gpr-hiwis
```

- (`-`) not set
- (`r`) read
- (`w`) write
- (`x`) execute

HPC.NRW

```
drwxr-xr-x 2 js056352 hpc-gpr-hiwis
-rw-r--r-- 1 js056352 hpc-gpr-hiwis
```

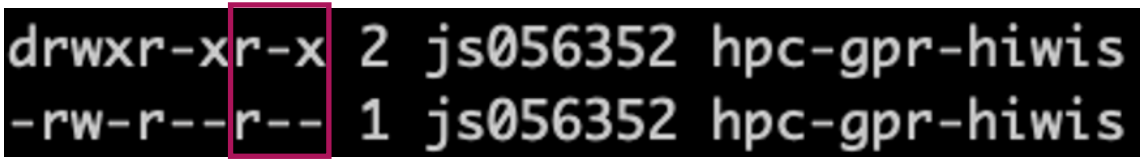
- (`-`) not set
- (`r`) read
- (`w`) write
- (`x`) execute

PERMISSIONS DISPLAY EXPLAINED

HPC.NRW

```
drwxr-xr-x 2 js056352 hpc-gpr-hiwis
-rw-r--r-- 1 js056352 hpc-gpr-hiwis
```

- (`-`) not set
- (`r`) read
- (`w`) write
- (`x`) execute



Other (o)

- (`-`) not set
- (`r`) read
- (`w`) write
- (`x`) execute

- Modify owner/group (needs `root`):
  - `chown <NewOwner> <filename>`
  - `chown <NewOwner>:<NewGroup> <filename>`

CHANGING PERMISSIONS

HPC.NRW

- Modify owner/group (needs `root`):
  - `chown <NewOwner> <filename>`
  - `chown <NewOwner>:<NewGroup> <filename>`
- Modify permissions:
  - `chmod u+x <Filename>`

    `u` = User, `g` = Group, `o` = Other, `a` = All

    `+` or `-`

    `r` = Read, `w` = Write, `x` = Execute

Demo 10

- Process: running instance of a program
  - System
  - User
  - User (manually launched)

PROCESSES

HPC.NRW

- Process: running instance of a program
  - System
  - User
  - User (manually launched)
- Like Windows
  - Equivalent to Task Manager: `top`
  - Short overview: `pstree`

- Process: running instance of a program
  - System
  - User
  - User (manually launched)
- Like Windows
  - Equivalent to Task Manager: `top`
  - Short overview: `pstree`
- Each process has an owner
  - Process can/can't do what owner can/can't do

- Process: running instance of a program
  - System
  - User
  - User (manually launched)
- Like Windows
  - Equivalent to Task Manager: `top`
  - Short overview: `pstree`
- Each process has an owner
  - Process can/can’t do what owner can/can’t do
- Each process has an ID number (PID)

– If you enter command, it runs in the shell

- If you enter command, it runs in the shell
- Enter `<command> &` to start it in background
  - Good if command launches window, console still usable

- If you enter command, it runs in the shell
- Enter `<command> &` to start it in background
  - Good if command launches window, console still usable

- Send foreground command to background by Ctrl-Z (pauses it) and typing `bg`

- If you enter command, it runs in the shell
- Enter `<command> &` to start it in background
  - Good if command launches window, console still usable

- Send foreground command to background by Ctrl-Z (pauses it) and typing `bg`
- Bring to foreground with `fg <Job-ID>`
  - Caution: job ID is different from process ID!
  - Can be displayed with `jobs`

OUTPUT AND NAVIGATION IN TOP

HPC.NRW

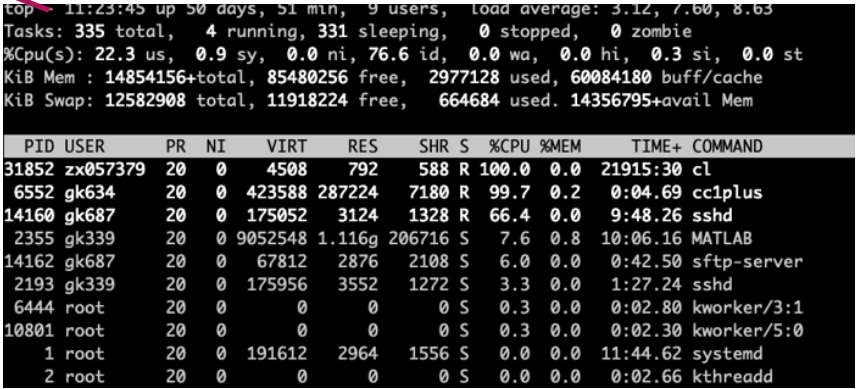
```
top - 11:23:45 up 50 days, 51 min,  9 users,  load average: 3.12, 7.60, 8.63
Tasks: 335 total,   4 running, 331 sleeping,   0 stopped,   0 zombie
%Cpu(s): 22.3 us,  0.9 sy,  0.0 ni, 76.6 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem : 14854156+total, 85480256 free,  2977128 used, 60084180 buff/cache
KiB Swap: 12582908 total, 11918224 free,   664684 used. 14356795+avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
31852 zx057379  20   0    4508    792    588 R 100.0  0.0  21915:30 cl
 6552 gk634     20   0  423588 287224   7180 R  99.7  0.2    0:04.69 cc1plus
14160 gk687     20   0  175052   3124   1328 R  66.4  0.0    9:48.26 sshd
 2355 gk339     20   0 9052548 1.116g 206716 S   7.6  0.8  10:06.16 MATLAB
14162 gk687     20   0   67812   2876   2108 S   6.0  0.0    0:42.50 sftp-server
 2193 gk339     20   0  175956   3552   1272 S   3.3  0.0    1:27.24 sshd
 6444 root      20   0       0      0      0 S   0.3  0.0    0:02.80 kworker/3:1
10801 root      20   0       0      0      0 S   0.3  0.0    0:02.30 kworker/5:0
    1 root      20   0  191612   2964   1556 S   0.0  0.0   11:44.62 systemd
    2 root      20   0       0      0      0 S   0.0  0.0    0:02.66 kthreadd
```

OUTPUT AND NAVIGATION IN TOP

HPC.NRW

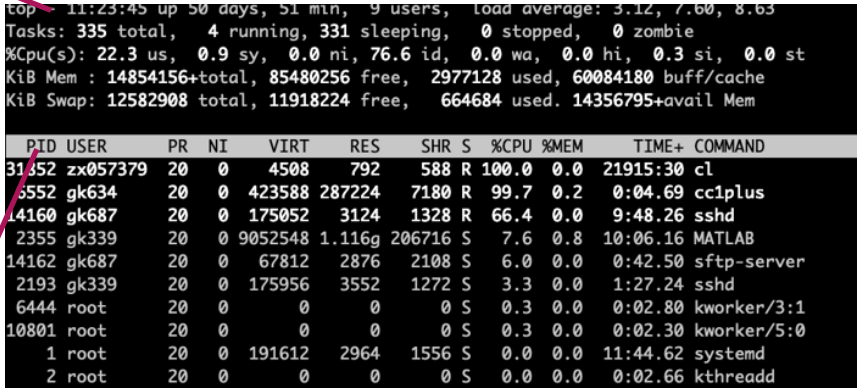
Total resource use



OUTPUT AND NAVIGATION IN TOP

HPC.NRW

Total resource use

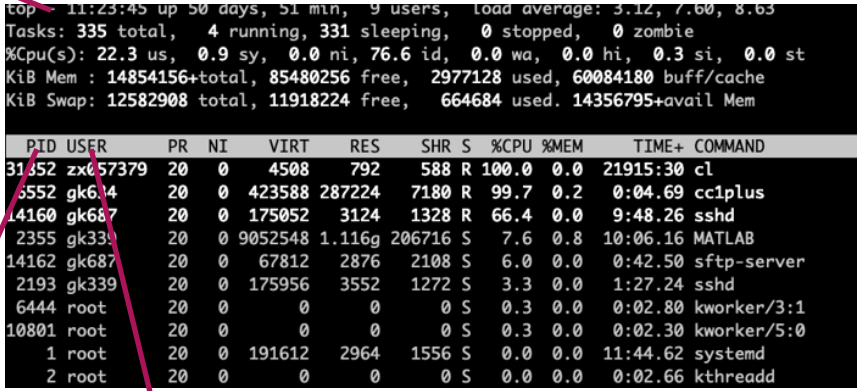


Process ID

OUTPUT AND NAVIGATION IN TOP

HPC.NRW

Total resource use



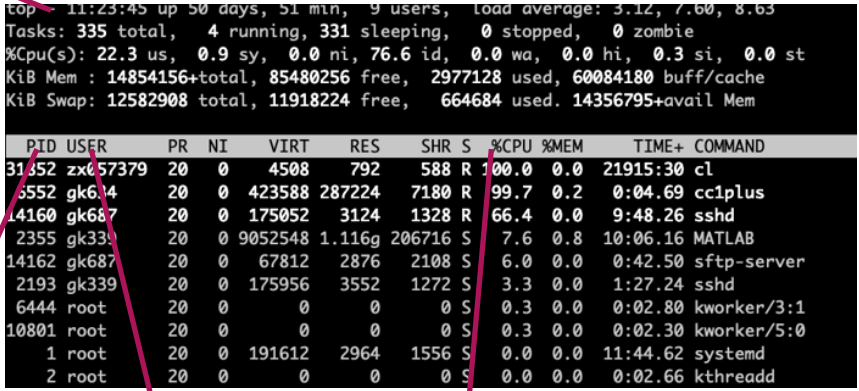
Process ID

Owner

OUTPUT AND NAVIGATION IN TOP

HPC.NRW

Total resource use



Process ID

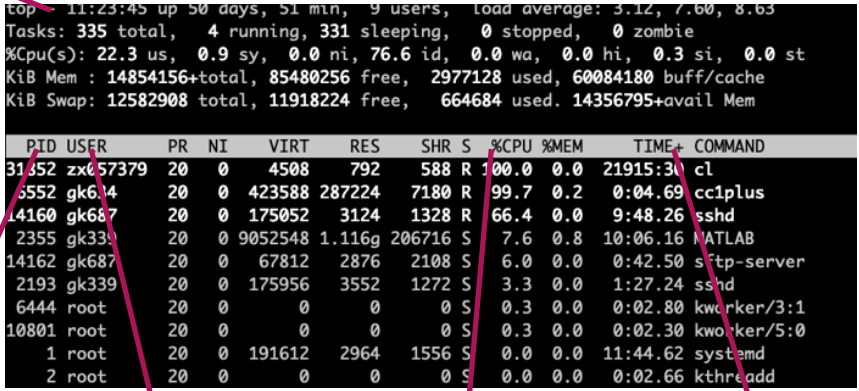
Owner

Resource use

OUTPUT AND NAVIGATION IN TOP

HPC.NRW

Total resource use



Process ID

Owner

Resource use

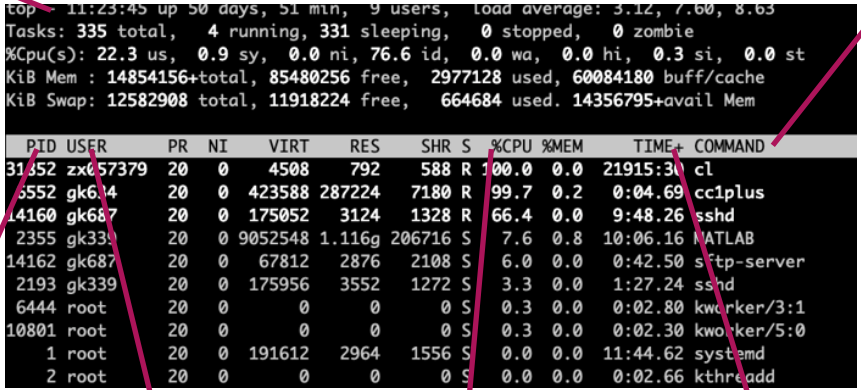
Runtime

OUTPUT AND NAVIGATION IN TOP

HPC.NRW

Total resource use

Command name



Process ID

Owner

Resource use

Runtime

OUTPUT AND NAVIGATION IN TOP

HPC.NRW

- Single-letter commands to navigate `top`
  - `u`: filter processes from a specific user
  - `k`: kill a specific process
  - `h`: show help
  - `f`: toggle displayed columns
  - `x`: highlight current sort columnt
  - `<>`: select column to sort for
  - `R`: Reverse sorting
  - `q`: quit top

THE VIM TEXT EDITOR

HPC.NRW Competence Network

INTRODUCTION TO LINUX

TEXT EDITOR VIM

HPC.NRW

- Default Linux text editor: `vi`
  - Usually: `vim` (vi improved), includes syntax highlighting

TEXT EDITOR VIM

HPC.NRW

- Default Linux text editor: `vi`
  - Usually: `vim` (vi improved), includes syntax highlighting
- Completely inside console

TEXT EDITOR VIM

HPC.NRW

- Default Linux text editor: `vi`
  - Usually: `vim` (vi improved), includes syntax highlighting
- Completely inside console
- Advantages:
  - Always available
  - Very fast *once you know commands*

HPC.NRW

- Default Linux text editor: `vi`
  - Usually: `vim` (vi improved), includes syntax highlighting
- Completely inside console
- Advantages:
  - Always available
  - Very fast *once you know commands*
- Disadvantages:
  - Interface unlike most text editors
  - Steep learning curve

Command Mode

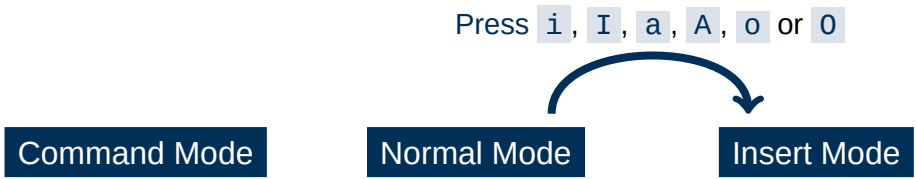
- Save/load
- Quit vim
- …

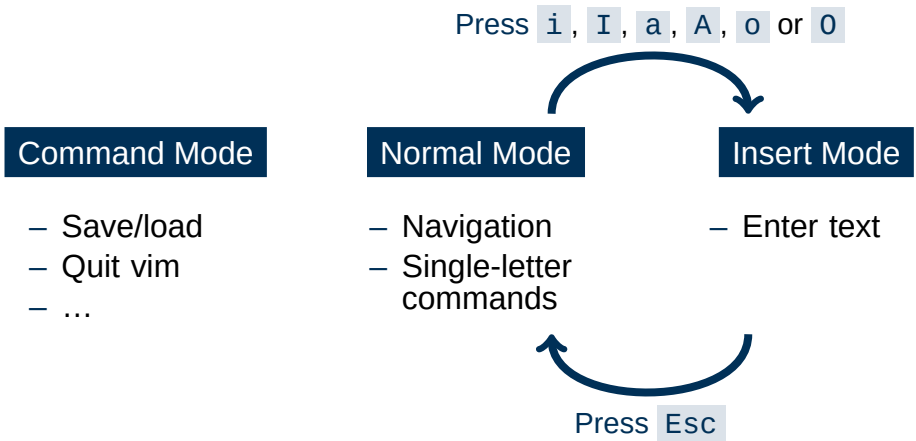
Normal Mode

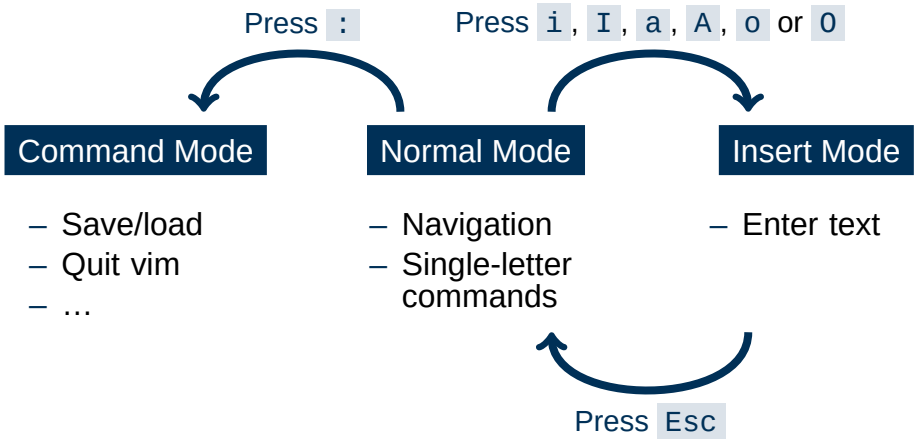
- Navigation
- Single-letter commands

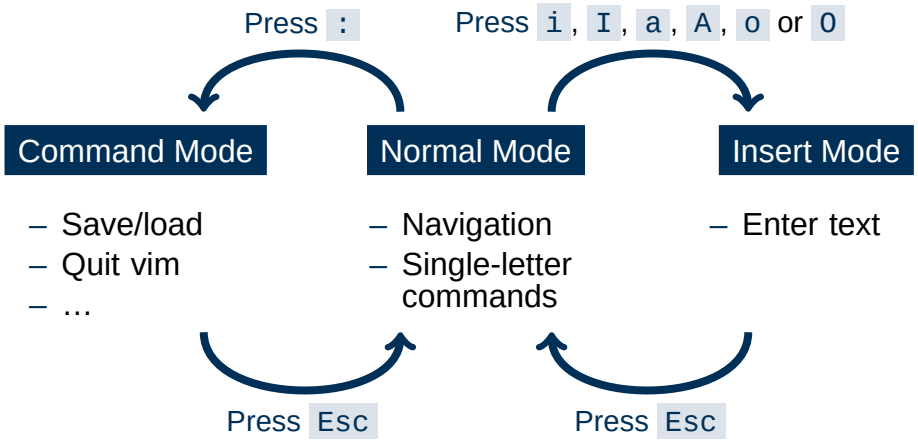
Insert Mode

- Enter text


Press i, I, a, A, o or O
Command Mode
Normal Mode
Insert Mode
– Save/load
– Quit vim
– …
– Navigation
– Single-letter commands
– Enter text


Press i, I, a, A, o or O
Command Mode
Normal Mode
Insert Mode
– Save/load
– Quit vim
– …
– Navigation
– Single-letter commands
– Enter text
Press Esc


Press :
Press i, I, a, A, o or O
Command Mode
Normal Mode
Insert Mode
– Save/load
– Quit vim
– …
– Navigation
– Single-letter commands
– Enter text
Press Esc


Press :
Press i, I, a, A, o or O
Command Mode
Normal Mode
Insert Mode
– Save/load
– Quit vim
– …
– Navigation
– Single-letter commands
– Enter text
Press Esc
Press Esc

COMMON VIM COMMANDS (MOVEMENT)

HPC.NRW

<arrow keys> move cursor in arrow direction

`h`, `j`, `k`, `l` move cursor left, down, up, right

`$` Move to end of line

`gg` Move cursor to first line

`Xgg` Move cursor to line X

`G` Move cursor to last line

`w` Jump forward to next word

`b` Jump backward to previous word

`%` Jump to matching character (default pairs: `()`, `{}`, `[]`)

COMMON VIM COMMANDS (EDITING)

HPC.NRW

- `u` Undo last change
- `<Ctrl-r>` Redo last change
- `r<X>` Replace current character with `<X>`
- `R` Enter “Replace mode” (multi character replacement)
- `cc` Change (replace) current line
- `C` Change (replace) to end of line
- `cw` Change (replace) to end of word (`ciw` for entire word)
- `.` Repeat last command

COMMON VIM COMMANDS (CUT AND PASTE)

HPC.NRW

yy **or** **Y** Yank (copy) entire line

yw Yank (copy) to end of word

y$ Yank (copy) to end of line

COMMON VIM COMMANDS (CUT AND PASTE)

HPC.NRW

`yy` **or** `Y` Yank (copy) entire line

`yw` Yank (copy) to end of word

`y$` Yank (copy) to end of line

`dd` Delete (cut) entire line

`dw` Delete (cut) to end of word

`d$` **or** `D` Delete (cut) to end of line

`x` Delete character

COMMON VIM COMMANDS (CUT AND PASTE)

HPC.NRW

`yy` **or** `Y` Yank (copy) entire line

`yw` Yank (copy) to end of word

`y$` Yank (copy) to end of line

`dd` Delete (cut) entire line

`dw` Delete (cut) to end of word

`d$` **or** `D` Delete (cut) to end of line

`x` Delete character

`p` Paste after cursor

`P` Paste before cursor

COMMON VIM COMMANDS (SEARCHING)

HPC.NRW

`/pattern`	Forward search for regular expression
`?pattern`	Backward search for regular expression

COMMON VIM COMMANDS (SEARCHING)

HPC.NRW

`/pattern`	Forward search for regular expression
`?pattern`	Backward search for regular expression
`n`	Repeat last search
`N`	Repeat last search in opposite direction

COMMON VIM COMMANDS (SEARCHING)

HPC.NRW

`/pattern`	Forward search for regular expression
`?pattern`	Backward search for regular expression
`n`	Repeat last search
`N`	Repeat last search in opposite direction
`%s/old/new/`	Replace `old` pattern with `new` pattern on current line
`%s/old/new/g`	Replace `old` pattern with `new` pattern in entire file

COMMON VIM COMMANDS (EXIT)

HPC.NRW

`:w` Write (save) file

`:wq` **or** `:x` **or** `ZZ` Write (save) file and quit

`:wqa` **or** `:xa` Write (save) all files and quit

`:q!` Close file without saving

`:qa!` Close all files without saving

- Most common vim problem: forgetting which mode you are in
  - Run commands when you meant to type text
  - Remember `u` for undo

  When in doubt: keep pressing Esc

- When to use vim:
  - Either only for simple things
  - Or commit to learning it (worth it in the long run)

  Otherwise, you will spend a lot of time looking up commands

ALTERNATIVE TEXT EDITORS

HPC.NRW

If all else fails, vim usually still works

→ **Knowing vim basics is important for all Linux users**

However I don’t blame you if you look for something simpler for everyday use

ALTERNATIVE TEXT EDITORS

HPC.NRW

If all else fails, vim usually still works
→ **Knowing vim basics is important for all Linux users**
However I don't blame you if you look for something simpler for everyday use

- Most Linux computers have at least one text editor in addition to vim
  - `gedit` (requires X window connection)
  - `nano`
  - `emacs` (also very powerful and hard to master)
  - Not on cluster but common: `kate` (graphical)
  - MobaXTerm: built-in text editor

SHELL SCRIPTS

HPC.NRW Competence Network

INTRODUCTION TO LINUX

- Interaction with Linux: just a series of commands
  - Commands can be put into a text file
  - Text file is fed to console
  - Console runs commands one after the other

- Interaction with Linux: just a series of commands
  - Commands can be put into a text file
  - Text file is fed to console
  - Console runs commands one after the other

- Advantage: very easy automation

- Interaction with Linux: just a series of commands
  - Commands can be put into a text file
  - Text file is fed to console
  - Console runs commands one after the other
- Advantage: very easy automation
- Shell script: execute like a program
  - Remember “execute” permissions

EXECUTING SHELL SCRIPTS

HPC.NRW

- Command to run script
  - Full script name (including location)
  - Commonly: `./scriptname.sh`

EXECUTING SHELL SCRIPTS

HPC.NRW

- Command to run script
  - Full script name (including location)
  - Commonly: `./scriptname.sh`
- Why not only script name?
  - Linux only looks up commands in specific folders (listed in environment variable `$PATH`)
    - Safety feature (not everyone can run everything)

EXECUTING SHELL SCRIPTS

HPC.NRW

- Command to run script
  - Full script name (including location)
  - Commonly: `./scriptname.sh`
- Why not only script name?
  - Linux only looks up commands in specific folders (listed in environment variable `$PATH`)
    - Safety feature (not everyone can run everything)
- File needs execute permissions
  - Another safety feature
  - Remember `chmod` command (e.g. `chmod u+x`)

EXAMPLE SHELL SCRIPT

HPC.NRW

```
#!/bin/bash

# This is a comment line.
echo "Hello world."

ls -l
sleep 3s
ls \
 -l
```

EXAMPLE SHELL SCRIPT

HPC.NRW

```
#!/bin/bash

# This is a comment line.
echo "Hello world."

ls -l
sleep 3s
ls \
 -l
```

So-called “shebang”

- Always has to be first line
- Comment plus exclamation point
- Specifies interpreter (here bash)
- Does not have to be Linux console (/usr/bin/python)

EXAMPLE SHELL SCRIPT

HPC.NRW

```
#!/bin/bash

# This is a comment line.
echo "Hello world."

ls -l
sleep 3s
ls \
 -l
```

So-called “shebang”

- Always has to be first line
- Comment plus exclamation point
- Specifies interpreter (here bash)
- Does not have to be Linux console (/usr/bin/python)

Comment symbol

- Line comments only
- Sometimes meta-commands

EXAMPLE SHELL SCRIPT

HPC.NRW

```
#!/bin/bash

# This is a comment line.
echo "Hello world."

ls -l
sleep 3s
ls \
 -l
```

So-called “shebang”

- Always has to be first line
- Comment plus exclamation point
- Specifies interpreter (here bash)
- Does not have to be Linux console (/usr/bin/python)

Comment symbol

- Line comments only
- Sometimes meta-commands

Echo command

- Common command
- Debugging, logging

EXAMPLE SHELL SCRIPT

HPC.NRW

```
#!/bin/bash

# This is a comment line.
echo "Hello world."

ls -l
sleep 3s
ls \
 -l
```

So-called “shebang”

- Always has to be first line
- Comment plus exclamation point
- Specifies interpreter (here bash)
- Does not have to be Linux console (/usr/bin/python)

Comment symbol

- Line comments only
- Sometimes meta-commands

Echo command

- Common command
- Debugging, logging

List of commands

- Same as when entered manually

EXAMPLE SHELL SCRIPT

HPC.NRW

```
#!/bin/bash

# This is a comment line.
echo "Hello world."

ls -l
sleep 3s
ls \
 -l
```

So-called “shebang”

- Always has to be first line
- Comment plus exclamation point
- Specifies interpreter (here bash)
- Does not have to be Linux console (/usr/bin/python)

Comment symbol

- Line comments only
- Sometimes meta-commands

Echo command

- Common command
- Debugging, logging

List of commands

- Same as when entered manually

Line break

- Backslash as last character

- Use command line arguments: `$0` - `$9`, `${10}`
  - Example: script was called with `script.sh -f 5.0`
  - Then: `$0=script.sh`, `$1=-f`, `$2=5.0`

SHELL SCRIPTS: ADDITIONAL TIPS

HPC.NRW

- Use command line arguments: `$0` - `$9`, `${10}`
  - Example: script was called with `script.sh -f 5.0`
  - Then: `$0=script.sh`, `$1=-f`, `$2=5.0`
- Parantheses: output of command `$(echo "example")`

SHELL SCRIPTS: ADDITIONAL TIPS

HPC.NRW

- Use command line arguments: `$0` - `$9`, `${10}`
  - Example: script was called with `script.sh -f 5.0`
  - Then: `$0=script.sh`, `$1=-f`, `$2=5.0`
- Parantheses: output of command `$(echo "example")`
- Loops and if statements, similar to most programming languages

```
for file in $( ls ); do
    echo item: $file
done

if [ -e $filename ]; then
    echo "$filename exists."
fi
```

- Shell scripts are good for running series of commands
  - Not so good for more complex programming
    - Loops, ifs etc. are an afterthought
    - I don’t know of an IDE or debugger
    - Can delete wrong file(s) very easily
  - Better: “proper” scripting language (e.g. Python)

- Shell scripts are good for running series of commands
  - Not so good for more complex programming
    - Loops, ifs etc. are an afterthought
    - I don’t know of an IDE or debugger
    - Can delete wrong file(s) very easily
  - Better: “proper” scripting language (e.g. Python)

- Default shell in most Linux systems (e.g. Ubuntu, CentOS): `bash`
  - Many alternatives: C-Shell(`csh`), Z Shell(`zsh`), Fish(`fish`)
    - Often completely different syntax
    - Prefer portable shell programming where possible

- Store output of commands
- Assignment via `=` (equal sign)
  - Example: `var="value"`
  - Important: no spaces around `=`
  - Always text

- Store output of commands
- Assignment via `=` (equal sign)
  - Example: `var="value"`
  - Important: no spaces around `=`
  - Always text
- Retrieve with `$` sign

  `$var`
  - Example: `echo $var` prints value to screen
- By default: variable only in current console

- “Environment”: which variables are defined and available
  - To a process
  - Within a shell

- “Environment”: which variables are defined and available
  - To a process
  - Within a shell
- Avoids hardcoding varying information
- Example: current user’s home directory

```
HOME=/home/Schulung12
```

ENVIRONMENT

HPC.NRW

- “Environment”: which variables are defined and available
  - To a process
  - Within a shell
- Avoids hardcoding varying information
- Example: current user’s home directory

```
HOME=/home/Schulung12
```

- Helpful to provide configuration scripts
  - Change information in a single location
  - Keep business logic apart from config

- Many environment variables already defined
  - By system (e.g. `$USER`)
  - By installed software

- Many environment variables already defined
  - By system (e.g. `$USER`)
  - By installed software
- Command `env` to show all currently defined variables
  - Convention: usually capital letters

- Many environment variables already defined
  - By system (e.g. `$USER`)
  - By installed software
- Command `env` to show all currently defined variables
  - Convention: usually capital letters
- Passing on environment variables:

```
export MY_VAR="value"
```

  - Available in child processes

SYSTEM CONFIGURATION FILES

HPC.NRW Competence Network

INTRODUCTION TO LINUX

SYSTEM INFORMATION FILES

HPC.NRW

- Files in `/proc` are not regular files
  - Text containing system information
  - E.g. `/proc/cpuinfo`, `/proc/meminfo`
  - Display with `cat` or similar
  - Cannot be edited

- Environment variable `PATH`
  - List of directories (separated by `:`)
  - Console will look for command names
    - Command may be in multiple directories: first hit is used
  - Own commands: add directory to path
- Core concept of operating system
  - Same principle in Windows console
- Also used by other software
  - Example `PYTHONPATH`

EXCURSION: ENVIRONMENT MODULES

HPC.NRW

- Cluster: different environments for different people
  - Admins cannot predict who needs what
  - Different version of same software: collision of environment variables!

- Solution: make it easy to switch environments
  - Environment modules: sets of environment settings
  - Not limited to clusters

EXCURSION: ENVIRONMENT MODULES

HPC.NRW

- Use of modules covered in Cluster Introduction Course
  - Now: what actually happens when module is loaded?
- Each module has a definition file
  - Actually a LUA script
- Let’s examine a module file:
  - OpenMPI module (compiled with GCC)

```
$ module show openmpi/gcc/64/1.10.3
```

EXCURSION: ENVIRONMENT MODULES

HPC.NRW

- Three things
  - Description what module does
  - Prepend to path and other variables
  - Add new variables

- Anyone want to guess why it prepends rather than appending?

- Problem: long command, has to be typed often
  - One option: script (but overkill)
- Built into the shell: aliases
  - Define with `alias name='command'`
  - List with `alias` (no arguments)
- Common aliases:

```
alias ll='ls -l'
alias cd..='cd ..'
```

HPC.NRW

- Console settings usually temporary
  - Environment variables, aliases etc.
  - Adding a directory to PATH
  - Disappear when you close console/disconnect SSH

- Making them permanent: put settings into configuration file
  - Specific files that are read when console is started
  - Examples for Bash:

    `~/.bashrc`

    `~/.bash_profile`

- Other configuration files
  - Example: `~/.vimrc`
- CAUTION WHEN EDITING THESE FILES
  - Breaking `.bashrc` can make it impossible to log in
- Applying changes:
  - Type `source <filename>`
  - Alternative: log out and back in

- Linux determines language and keyboard settings with a so-called locale
- Dictionary definition:

  “Locale (noun): a place or locality, especially with reference to events or circumstances connected with it”
- Grouped into various settings
- See and set with `locale` command
- Sometimes causes weird problems

EXAMPLE LOCALE OUTPUT

HPC.NRW

```
$ locale
LANG=de_DE.UTF-8            # Default for all below variables that are not explicitly set
LC_CTYPE="de_DE.UTF-8"      # Printable characters, used by some C functions
LC_NUMERIC="de_DE.UTF-8"    # Number format (e.g. decimal point or comma)
LC_TIME="de_DE.UTF-8"       # Date and time format
...
LC_ALL=                     # Hard override for all variables above (e.g. for testing)
```

- Output from HorUS cluster
  - Some settings omitted for brevity
  - de_DE.UTF-8
    - German language
    - German region (as opposed to e.g. Austria)
    - UTF-8 character encoding

VARIOUS TIPS

HPC.NRW Competence Network

INTRODUCTION TO LINUX

- You should make backups regularly.
- It is recommended that you back up important files at regular intervals.
- If you break something, you can restore it from a backup that you should have made earlier.
- When working with Linux, be prepared for mistakes, which may require you to use the backups that you hopefully made.
- When you save a copy of a file to a different location regularly with the purpose of copying that file back to the original location to replace the original file after the original file became unusable, that is called a backup, and you should do that.

- Useful commands: `du`
  - Shows disk usage
  - Common options: `-h` (human-readable) `-s` (Show total), `-c` (Show individual files)
  - Example: `du -sch .`

- Counterpart: `df`
  - Disk free

- Useful commands: `histroy`
  - Lists previous commands (same as Up-Arrow/Down-Arrow)
  - Text file in your home directory: `~/.bash_history`
  - Advantage: searchable
  - Example: `history | grep <commandname>`
    - When you forget what options you used

- Useful commands: `ln -s`
  - Creates a symbolic link
  - Similar to Windows links
  - Visible with `ls -l` or `which`
  - Usage: `ln [Option] <Target> <Link name>`
  - Example: `ln -s myfile.txt mylink`
  - Also possible: “hard links” (not covered here)

- Useful commands: `watch`
  - Runs target command every 2 seconds
  - Any target command possible
  - Interval modifiable
  - Example: `watch tail mylog.txt` will show what is written to log file
  - Leave with Ctrl+C

- Useful commands: calculator `$(( ))`
  - For simple integer math
  - Example: `echo $(( 5 + 3 ))`

- Stream editor `sed`
  - For simple text operations (e.g. replacing text)
  - Example: `sed -i "s/old/new/g" example.txt`
    - `-i` Edit in place
    - `s` Replace (followed by three-slash syntax)
    - Search text “`old`”, replace with “`new`”
    - `g` Repeat for all occurrences in file
  - Similar purpose and idea, but more powerful: `awk`
  - Both commonly used, I cannot recommend them due to complexity

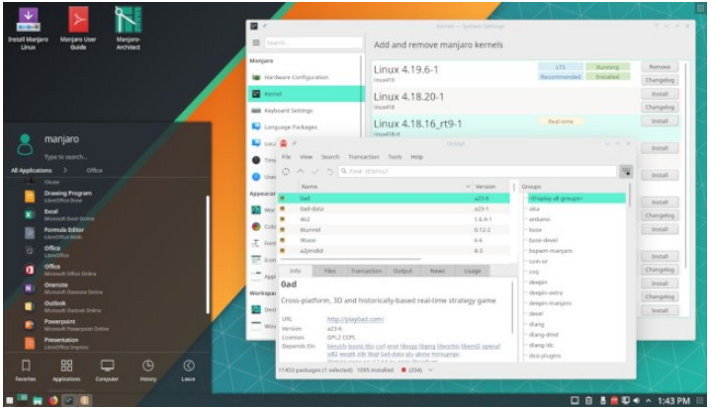
BEYOND THE CLUSTER

HPC.NRW Competence Network

INTRODUCTION TO LINUX

- Key components
  - X Window System
  - Desktop, Graphical User Interface (GUI)
    - Gnome (Ubuntu), KDE, XFCE (Mint)
    - Look is highly distro-dependent
- When remote: X server
  - Displays windows from other computer (cluster)
  - Careful with wording: server is on your machine, client is program that runs on cluster

EXAMPLE DESKTOP (KDE ON MANJARO)



Most features should look familiar to users of Windows and other OSes

- Software is often installed as packages
  - Organized in internet repositories
- Distro-dependent
  - Often maintain their own repository
- Not possible on cluster (exception: inside of application, e.g. Python, R)
- In general, three different package managers:
  - `apt-get` (Debian family), package format `.deb`
  - `yum` (Red Hat family), package format `.rpm`
  - `zypper` (Suse), package format `.rpm`

Thank you for your attention