Introduction to Unix

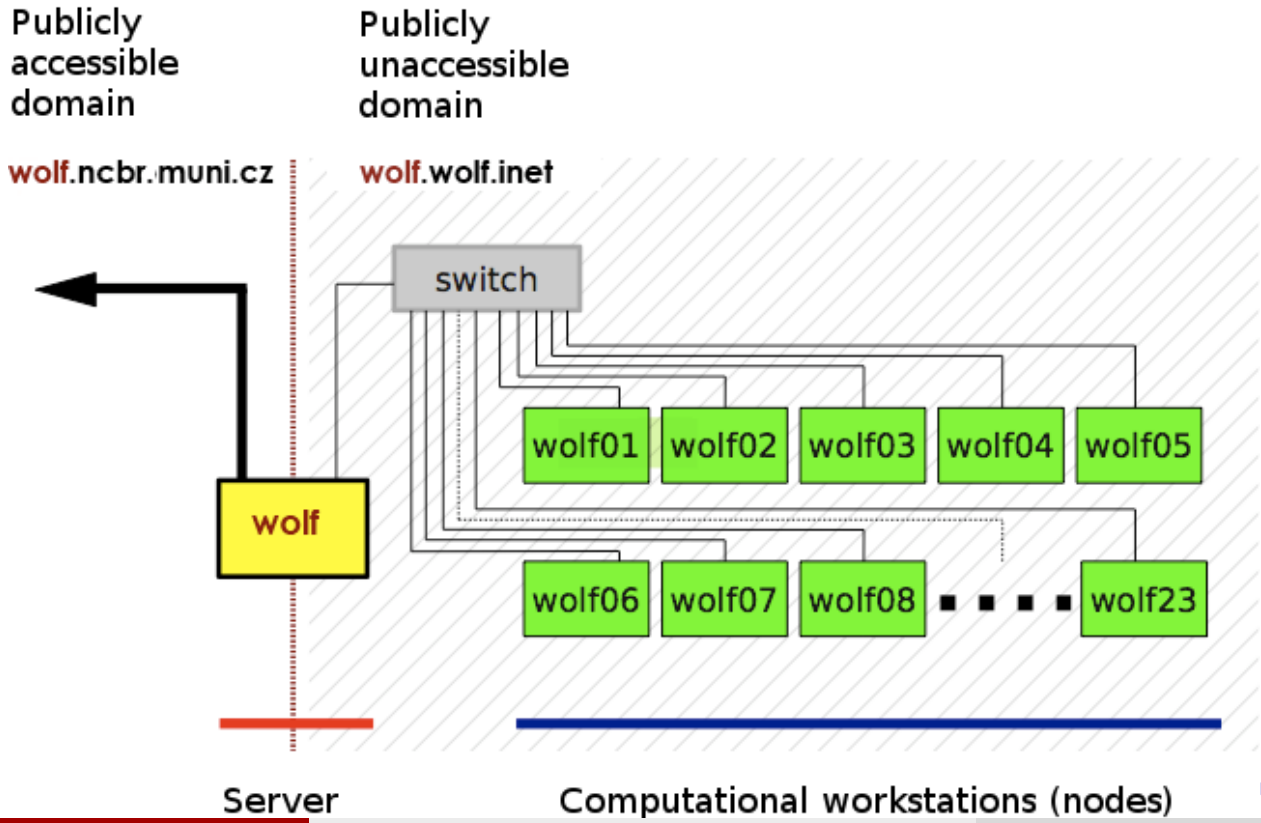
Unix

- Developped in 1970s in C language
- Open source code
- Multiuser system
- **Case-sensitive system**
- Many distributions developped since:
  - **Ubuntu**
  - Debian
  - BSD
  - Fedora
  - ...

Cluster Wolf

- Scientific software administrator: RNDr. Petr Kulhánek, PhD.

https://einfra.ncbr.muni.cz/whitezone/root/index.php?lang=enaction=ncbr show=wolf



User system

- Superuser
  - Administrative privileges
  - Can edit system files
- User
  - Cannot edit system files
  - Only selected items are editable/accessible
  - Belongs to certain groups with respective rights (hardware/software access...)

Filesystem

- No “Windows-like” discs
- Everything mounted under “/” (root) directory
- Slash sign is used as separator between directories
- Important paths:
  - /home/*username*/ or “~”: Quota 1.5 GB, backed-up
  - /scratch/*username*/: No quota, NOT backed-up
  - /media/*filesystem*/: USB sticks, DVD discs...
- Everything is either *file* or *process*
- Arbitrary suffixes for files

Directories and filenames

General advices aka “Good-To-Follow” rules:

- Case-sensitive system
- Do NOT use spaces in filenames (use underscore or dash)
- Good characters:
  - Alphanumerics
  - _ . − +
- Forbidden characters:
  - Any kind of diacritics
  - Quotation marks
  - Brackets
  - # % ? ! , * ˆ & @ / ~...

Text terminal

- Found in Applications → Accessories → Terminal
- Shell interpreter translating written commands into actions
- *Cygwin*, *PuTTY*: Terminal emulators for Windows machines
- Pros:
  - Fast and effective way of work
  - Directly visible output from operation
  - Error tracking
  - No GUI needed
- Cons:
  - Need of memorizing commands

Useful commands I

Command	Action
cd foo	Change current working directory to “foo”
ls	List files in directory
cp source target	Copy source file to target file
cp -r source target	Copy source directory recursively into target
mv source target	Move source file to target file
mkdir foo	Create “foo” directory
rmdir foo	Remove[a] “foo” directory (only if empty)
rm foo	Remove[a] “foo” file
rm -r foo	Remove[a] “foo” directory recursively
cat foo	Print content of a “foo” file into terminal
grep foo file	Print only line containing “foo” keyword in “file”
top	See currently running processes

[a] Removing means deleting from the disc. **NOT** moving into trash.

Useful commands II

Command	Action
head -n number foo	Print first “number” rows of “foo” file
tail -n number foo	Print last “number” rows of “foo” file
echo foo	Prints “foo” into terminal
printf	Similar to *echo* but handles formatted text
chmod switch foo	Changes rights of “foo” file according to switch
quota	Prints current quota of user and disc usage
ssh user@host	Remote access to host machine
exit	Logout from the terminal
who	Prints all users logged into machine
passwd	Change current pasword
kill PID	Kill the process with number “PID”
ps	Print all current processes running in terminal
module	Accessing the scientific software

Work in terminal

- Use ArrowUp and ArrowDown for searching the command history
- **Use Tabulator for word completion**
- Copy/Paste from terminal using mouse (CTRL+c/CTRL+v does **NOT** work here)

Will terminate current command!

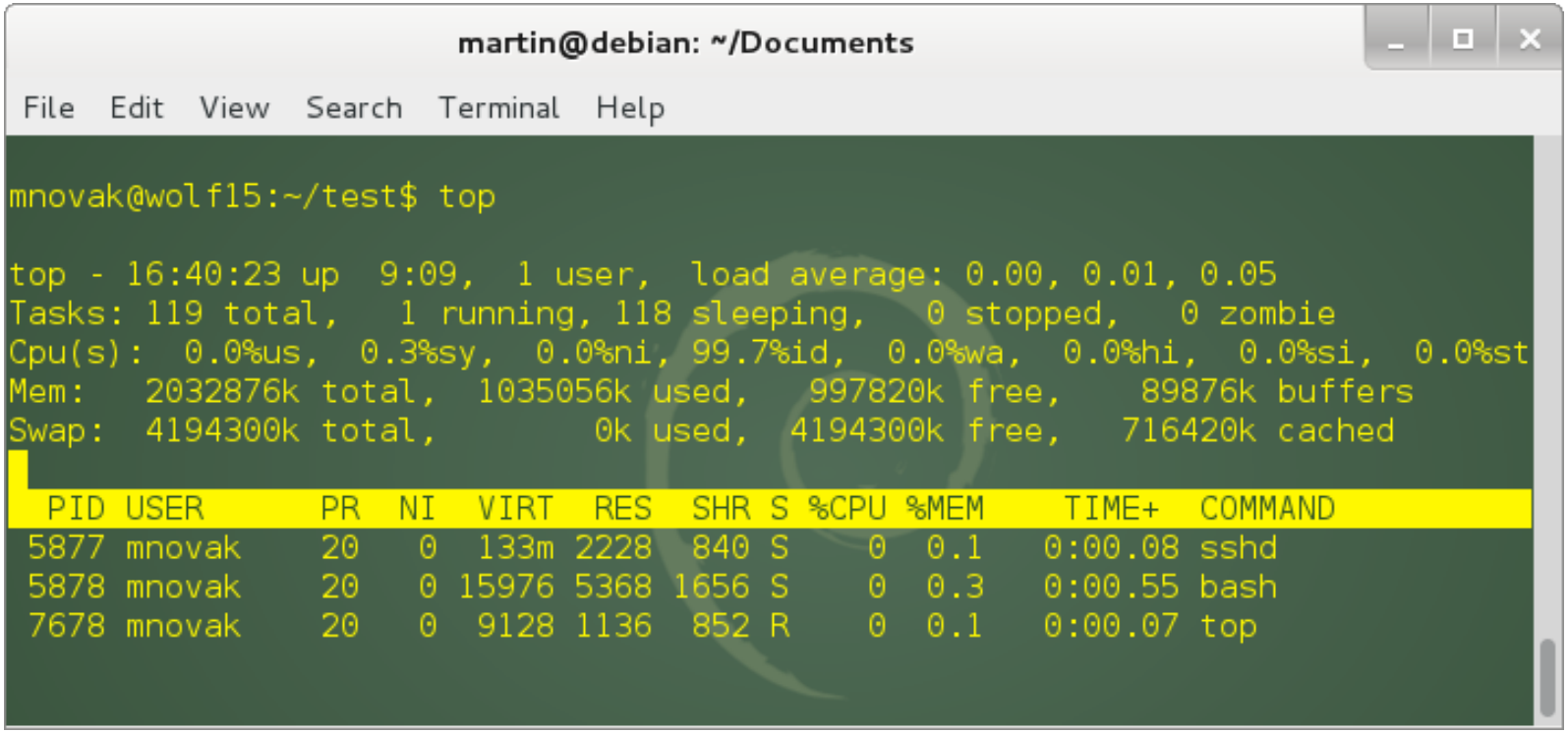
Wild characters

Notation	Matches
*	Any string of characters including empty string
?	Any single character
[jklm.]	Single character j, k, l, m or a dot
[a-m]	Single character from range a to m
[2-9]	Single number from range of 2 to 9

- Example:
- $ *ls* a*[0-2].??[df] This command will print all files which:
  - Start with “a”
  - Then they have any string of characters
  - Then there is either 0, 1, or 2
  - Followed by a dot
  - Then any two characters
  - Last character is either “d” or “f”
- All conditions must be satisfied

Listing and killing processes

- Once *command* is run, it obtains a unique process ID (PID)
- $ *top* # Displays currently running jobs in real time
- $ *kill* PID # Kills process with a given PID
- $ *kill* -9 PID # Kills process (Signal cannot be blocked)



Text editors

- With graphical interface:
  - gedit
  - kate
  - kwrite
  - gvim
- Without graphical interface (editing in terminal):
  - vi / vim
- Programmed to highlight keywords of many languages/source codes

Editor vi

- Fast and effective way to edit files in remote machine
- 3 modes:
  - Command mode
  - Edit mode
  - Visual mode
- Enter command mode via ESC key
- Enter edit mode via Insert or “i” key
- Visual mode for editing blocks of text:

`http://vimdoc.sourceforge.net/htmldoc/visual.html#Visual`

Commands of editor vi

Command	Action
:w	Save document
:w filename	Save document as “filename”
:q	Quit document
:q!	Quit without saving
:wq	Save and quit
:u	Undo
i / insert	Enter edit mode
R	Enter replace mode
gg	Go to the beginning of the document
G	Go to the end of the document
dd	Delete current line
25D	Delete next 25 lines
dG	Delete all lines starting from cursor
/keyword	Search for keyword

Vi tutorial

- Writing a plain text file:

$ vi test.dat	Open ‘test.dat’ file for editing
i / insert	Enter editing mode
Write some text	
ESC	exit editing mode and enter command mode
:w	Write text to file
gg	Go to first line
2D	Delete two lines
:u	Undo last change
:wq	Write and quit
$ rm test.dat	Remove file

Remote access

- Accessing remote machine via ethernet or internet
- *ssh* command:
- $ *ssh* [username@]hostmachine
- username does not have to be specified if same as current login
- If X applications should be exportable, use “-X” switch

Example

- Access the wolf node next to yours with X server export enabled
- Find out who is logged in there
- Exit from this computer
- Help: here

Passwordless authentication within cluster

- No password required for access the host machine
- Should be used with great care only on local networks
- Procedure:

```
$ cd .ssh
$ ssh-keygen
<enter>
<enter>
$ cat id_rsa.pub » authorized_keys
```

- Try to remotely access the same machine

Copying files between machines

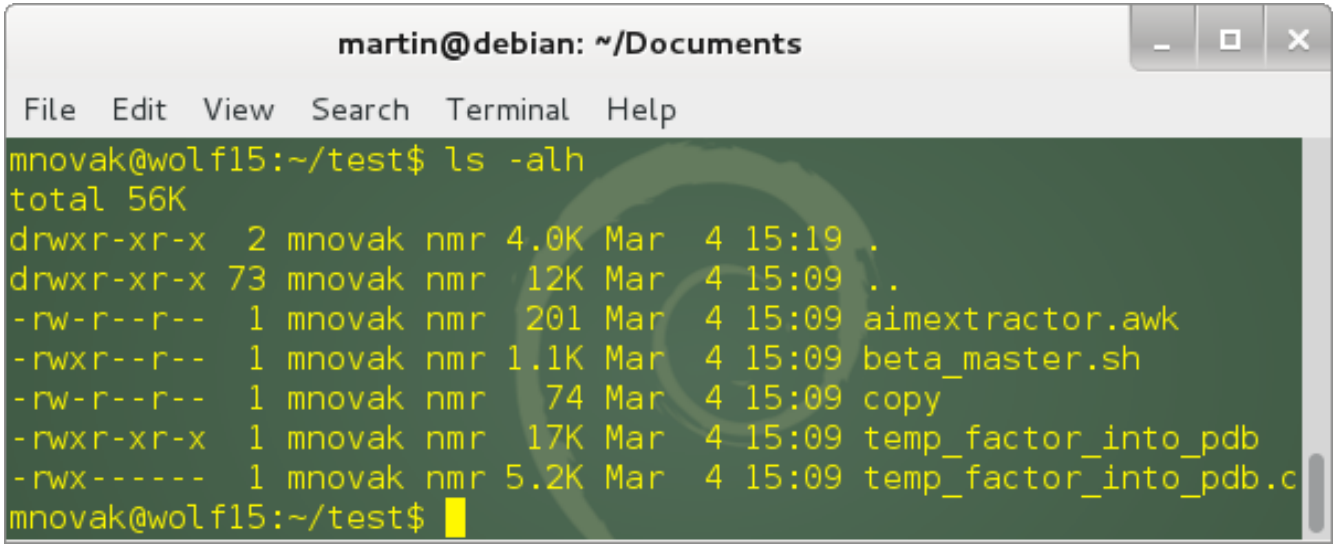
- $ *scp* source target
  - Source and/or target can be on remote machine:
  - mnovak@wolf12:~$ scp text.dat wolf13:/scratch/mnovak/
  - mnovak@wolf12:~$ scp -r wolf13:/scratch/mnovak/ directory/
- $ *mc*
  - Midnight commander - same as in Windows/Mac machines
  - “Graphical interface”
- $ *gftp*
  - “Real” graphical interface

Absolute versus Relative paths

- Absolute path:
  - Total path from the root directory
  - /scratch/mnovak/test
  - ~/Documents/
- Relative path:
  - ./ # Current directory
  - ../ # Parent directory
  - ../../../data/test/

Access permissions

- Each file has permissions for **Owner**, **Group** and **Others**
- drwxrwxrwx
  - d – Directory
  - r – Read
  - w – Write
  - x – Execute
  - - – Permission not granted



Change permissions

- $ *chmod* switch file
- examples of switches:

u+x	User can execute file
go+w	Group members and others can write to file
a-r	Remove right to read for all users
o-rwx	Remove right to read, write and execute to others

Excercise

- Create in your home folder directory **folder01**
- Copy current pdf presentation and *.tex from address **wolf01:/share/ivavik/novotnyj/teaching** to your newly created directory, try to open it from terminal using evince, make it readable for all users
- Using vi editor create a plain text file called **prop.txt** and insert inside complete info about the pdf file based on ls output
- please store all subsequent working commands in this prop.txt file (use another terminal window for easier copying)

Excercise

- Study the manual info about *pdfjam* tool for manipulating pdf files and generate a new pdf file containing first 4 slides in landscape orientation (**pres4.pdf**)
- run simple command in terminal and inspect its function: *for ((i=1; i<30; i++)); do head -n$i 01.tex | tail -1 > $i.tex; done*
- remove all .tex files whose index ends 0 or 5
- create folder **your_username** a move there .tex files and prop.txt with inserted commands for the entire excercise
- copy recursively the folder **your_username** to **wolf01:/share/ivavik/novotnyj/teaching**