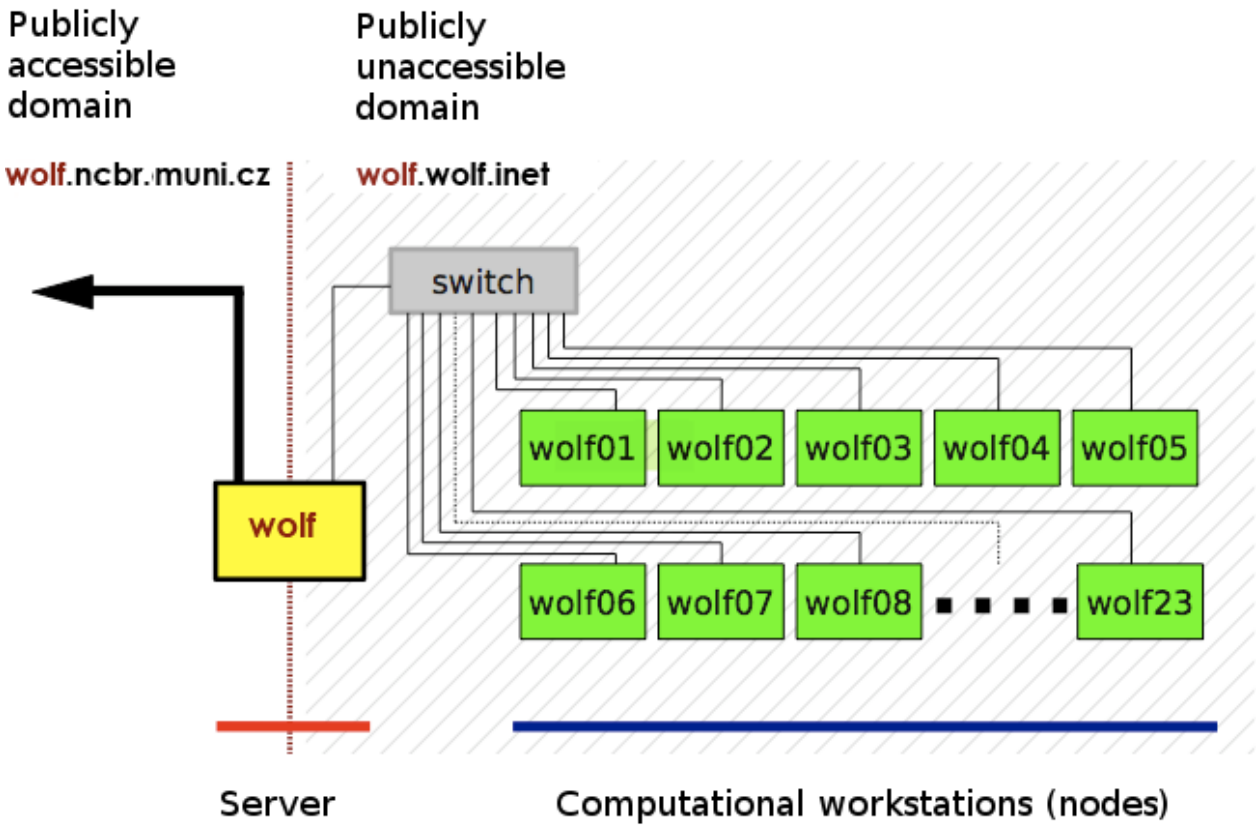
Introduction to Unix

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

- Head administrator: Mgr. Jakub Štěpán
- Scientific software administrator: RNDr. Petr Kulhánek, PhD.



Graphical interfaces

- Unity
  - Native Ubuntu environment
- Gnome
- Xterm
  - Mac OS-like looking environment
- KDE
  - Windows-like looking environment
- Several text terminals
  - Accessed with CTRL+ALT+F1-F6

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

Terminal welcome output

- Useful information: Highly advisable to read
- Contains:
  - System statistics
  - Last login of user and IP address
  - Active site
  - User and Host info
  - **Site documentation and support**

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

- General way of use is:
  - mnovak@wolf:~$ *command* [argument1, [argument2, ...]]
- Switches
  - Alter output of commands
  - Short notation: *command* -a -b -C
  - Long notation: *command* --alpha --beta --Gamma
  - Use switch “-h” or “--help” for general help
- Manual pages
  - Help for basic commands
  - Accessed via command *man*
  - Close help with “q” key

Work in terminal II

- 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!

Example

- Run the following commands and compare results
  - $ *ls*
  - $ *ls* -a
  - $ *ls* -l
  - $ *ls* -h
  - $ *ls* -lah

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

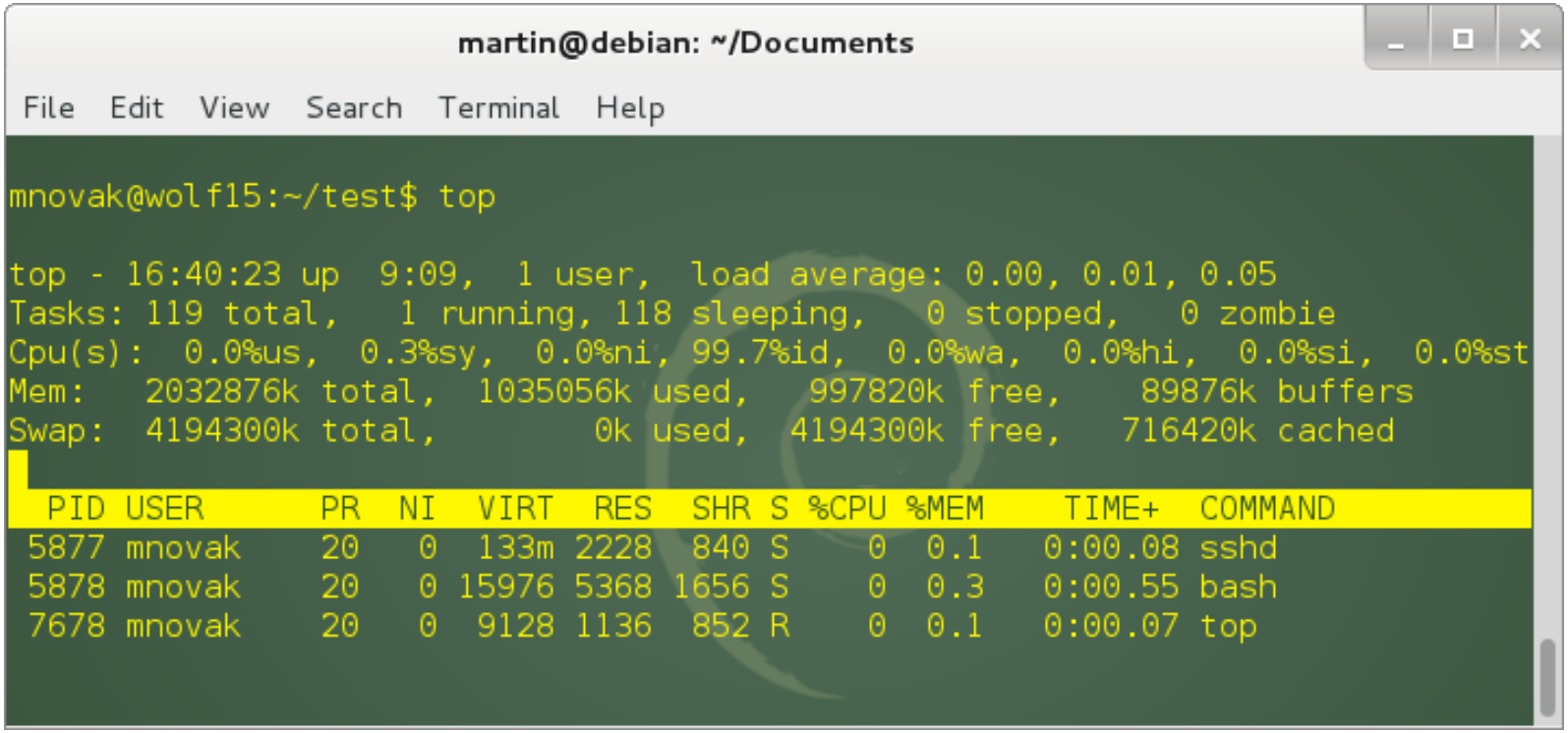
- Example:
- $ *ls* a*[0-2].??[df] This command will print all files which:
  - Start with “a”
  - Then they have any string of characters
  - Then there is either 0, 1, or 2
  - Followed by a dot
  - Then any two characters
  - Last character is either “d” or “f”
- All conditions must be satisfied

Running jobs in background

- Terminal is still usable for other tasks
- Two ways to achieve:
  - $ *command* &
  - Once job is running in terminal:
    - CTRL+z # Stops current task
    - $ *bg* # Puts all jobs into background

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)



Password change

- $ passwd
- Insert current password
- Enter new password twice (check for typos)
- **No characters are printed in terminal during typing**

A valid password should be a mix of upper and lower case letters, digits, and other characters. You can use an 8 character long password with characters from at least 3 of these 4 classes, or a 7 character long password containing characters from all the classes. An upper case letter that begins the password and a digit that ends it do not count towards the number of character classes used.

Scientific software

- System of modules located on server
- Modules must be imported prior to running
- Some modules available only at selected nodes
- $ ***module* avail** # Lists all available modules
- $ ***module* add moduleName** # Adds selected module to path
- $ ***module* remove moduleName** # Removes selected module
- Full description of imported module:
  - moduleName:version:arch:build
  - It is always a good practice to include version
- Full documentation at website:

`https://lcc.ncbr.muni.cz//whitezone/development/infinity/`

Example

- Run following commands and see the results:

  $ *gabedit*
  $ *module* add gabedit:2.3.0
  $ *gabedit*

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

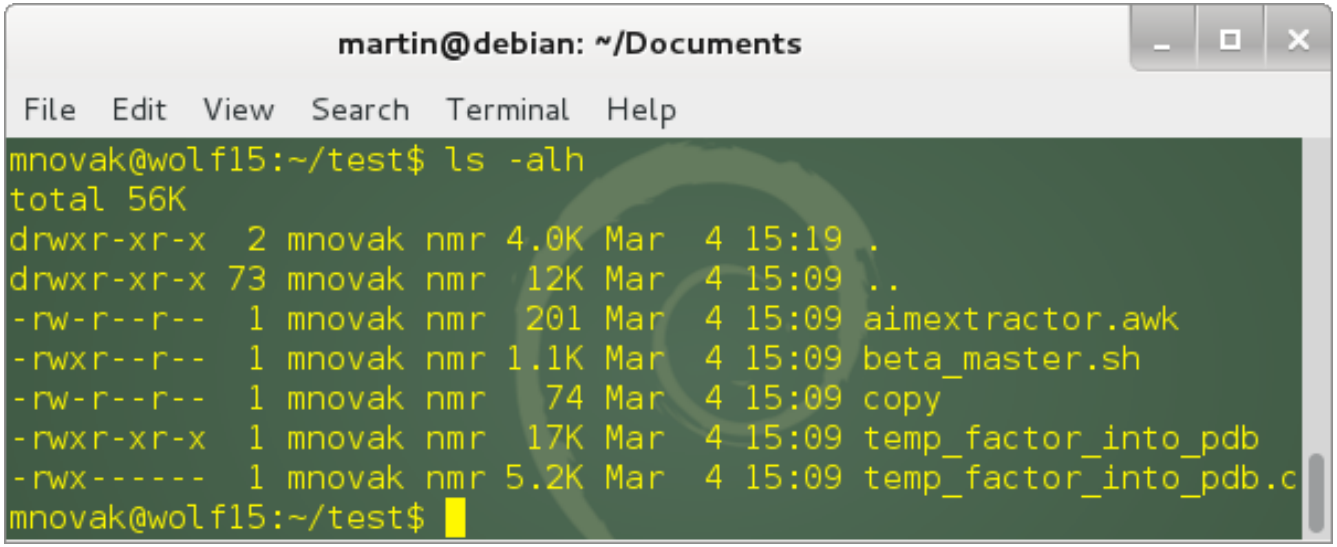
- $ *scp* source target
  - Source and/or target can be on remote machine:
  - mnovak@wolf12:~$ scp text.dat wolf13:/scratch/mnovak/
  - mnovak@wolf12:~$ scp -r wolf13:/scratch/mnovak/ directory/
- $ *mc*
  - Midnight commander - same as in Windows/Mac machines
  - “Graphical interface”
- $ *gftp*
 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

Variables

- Variables are used for storing values
- Various languages/interpreters use different name schemes
  - bash/tcsh: CAPITAL_LETTERS
  - awk: lowercase_letters
  - c++: almostEveryFirstLetterCappitalized
  - ...
- Variable names follow in general the same rules as filenames
- Variables in bash/tcsh accessible via $ sign, e.g. $RANDOM, for example

Scripts in bash

- Scripts are executable files which serve as storage of commands
- They are read line by line by interpreter and commands are run
- User does not have to type everything by hand
- Script must have executable permission
- Running scripts: mnovak@wolf:~$ *./myscript.sh*
- Comments after hash “#” sign

mnovak@wolf:~

```
#!/bin/bash
# This is a script which makes a directory, enters it and, lists all files

mkdir test;    # This commands makes the directory

cd test;       # This command enters it
ls -alh;       # This command lists all files inside
```

Useful script in bash

mnovak@wolf:~

```
#!/bin/bash
# This script loads gaussian module and performs calculations

module add gaussian:09.A2;  # Loads the module

g09 input.com;              # Performs the calculations
echo "All work done!";      # Prints info into terminal
return 0;                   # Return code for the script can be accessed via '$?' variable
```