Workshop on Terminal Basics

Johannes‘ Guide for Beginners

Prerequesites

- This guide is valid only for Unix (Linux or Max) users. Windows users use with care!
- It assumes you have an MPP account and know your username and password
- You need some python version callable from the terminal. Open the terminal, type python, if it doesn‘t print an error exit with Ctrl+d

Motivation

You have a great idea!

You just need to do some calculations on your PC!

The calculations take longer than expected. You let your PC run over night in your room.

You check the next morning and it broke down for some reason!

My code ran all night and broke down! ☹

What machines now? And what are cores?

Oh noo! That sucks! How many cores of our computing machines did you use?

Wtf?!

This does not have to be you!

Fingerübungen

- Open the terminal! (e.g. cmd+space, enter „terminal“)
- Make yourself comfortable:
  - Where am I (what‘s the path of the current directory)? **pwd**
  - What‘s in the current directory? **ls**
    - Commands can have options. Common ones for this one are **–l** and **–a** (or **–la** combined). Try to understand what they do!
    - Most important option (for most advanced commands): **-h** (show help page)
  - I wanna go to a different directory! **cd**
    cd DIR → go to subdirectory called DIR (or to absolute path DIR, if it starts with slash „/“; **cd** or **cd ~** → go back to home directory; **cd ..** → go to parent directory; **cd .** → That‘s where you are already!
  - Hit **Enter** and **Arrow up/down** a couple of times
- We want to have a place for our exercises, so go to your favourite directory and create a subdirectory for the workshop:
  mkdir TerminalWorkshop
  cd TerminalWorkshop

Vimerübungen

- If you are working with textfiles (or mathematica .m files) and just want to have a quick look or edit some details vim may be useful:
  - Create a new file/ open existing file with **vim VimTest1**
  - Write „Hi there!“ into your new textfile. That‘s weird, right? To edit text you have to press **i** for insert first. To exit insert mode, press **Esc**
  - Now leave and save your file. But… Oh no! How do I get out again?! Press **Esc** and type **:q** and see what happens. Press **Esc** and type **:wq** to actually leave
  - Create a second text file called VimTest2, exit with saving, type **vim V** and press **tab** a couple of times to appreciate autocompletion
- We don‘t really need these files however. So try removing them with **rm VimT***. The wildcard * tells the terminal to execute the command on all files fitting the criteria.

Danger Zone!

ssh

- I prepared a small python script for you that we can use. I put it on the MPP system, so we need to get it from there. But first we want to go there and have a look! Connect with the MPP system via **ssh USERNAME@th247.mpp.mpg.de**. You may need to accept the connection and enter your password. th247 is a specific desktop PC in my office. Use the number written on the PC below your desk in the future.
- Go to **/remote/ceph/user/d/diehl** . Ceph is great for storing Big Data (1GB and above) or sharing with colleagues. You will find a file called **pythonscript.py** there.
- Terminate the connection via **Ctrl+d** or typing **exit**.

scp

- We‘re back on our own PCs now! And we know where to get the file from. So let‘s get it! Type
  scp USER@th247.mpp.mpg.de:/remote/ceph/user/d/diehl/TerminalWorkshop/pythonscript.py .
  To copy the file pythonscript.py to the current directory. To copy it to a different directory use its relative or absolute path.
- th247 is just one of the theory computers. You should in general use the machine on which your home directory is located (or that your IT admin suggests)

The dot is Important!

Python Section

- If you do not have Python installed on your computer, copy it to your home directory on the MPP system instead. You can now execute the file with **python pythonscript.py**
- You can see, it does some calculations that should take a couple of seconds. You do not need to understand what the script is doing.
- Edit the script using vim. Change the 3 in the first line to a 9, save and exit.

screen

- Let‘s get all of this into your home on the MPP system!

Danger Zone!

cd ~

scp –r TerminalWorkshop USER@th247.mpp.mpg.de:~/

scp –r can overwrite existing directories. Without the / you replace your whole home directory!

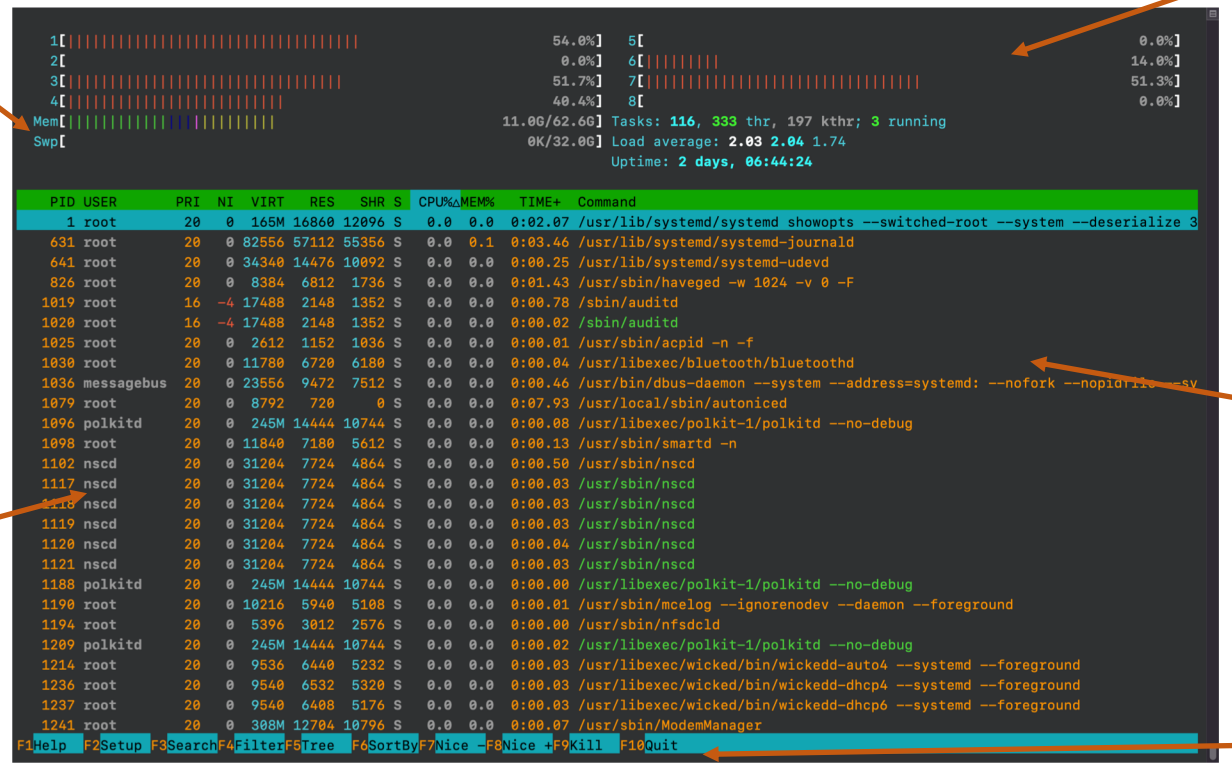
- Now, enter the MPP system, navigate to the TerminalWorkshop folder and type **screen**. To give the screen a name use **screen -R NAME**.
- Start the python script. When starting other scripts also use **> a.out** or **&** to suppress output!
- Type **screen –d**. The program will continue running in the background. You can also exit with **Ctrl**[hold]**+a,d**.
- To terminate the screen type **exit** or use **Ctrl+d**.
- Log out. Log back in. Type **screen –r** to see what the status of your program is.

htop

- If the program is still running we can learn about the last thing for today. Type htop. The result should look something like this:

Memory. To e.g. multiply an array by 3 your PC has to remember it, i.e. put it into memory. Memory is limited, stuff that is not used can be moved to Swp (swap). Full memory means slow program.

Cores. Using only one is slow. Using all on certain machines may summon an angry IT guy.



List of all active processes

Name of the user that will come shouting at you when you type **pkill PID**

Quit with **q** or **F10**

parallelization

- I showed you why it can be great
- Speed up depends on
  - Computer and its number of cores
  - You knowing what you are doing
  - Parallelizability of your program
- Multithreading vs Multiprocessing
  - Threads have shared memory, processes not
  - Multiprocess usually requires more coding and computational overhead
  - Good article with more details: https://www.indeed.com/career-advice/career-development/multithreading-vs-multiprocessing

Other tools you might want to check out

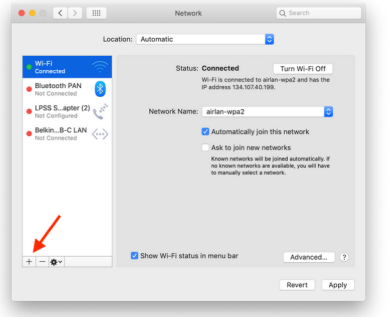
- My program takes hours! Or my program takes 5 mins and debugging is a pain! → look into parallelization in detail
- I‘m using the terminal often and want to make highly used commands shorter → edit your .bash_rc or .zprofile (for ideas look at the file /remote/ceph/user/d/diehl/TerminalWorkshop/.zprofile)
- I want to install something and make sure it works, also in the terminal, without tweaking it for 2h → homebrew (Mac), apt-get (Linux)
- I want to scp big data or want to easily make backups → rsync
- I always want to code on my local machine, but always want to let it run on a remote (and I‘m not using Mathematica) → try the editor called VSCode

VPN for access in homeoffice

- If you cannot follow from home, try following the guide on the right, which(together with Windows and Linux version) also can be found at https://max.mpg.de/sites/mpp/IT/Seiten/VPN-setup.aspx

Here you can see how to set up a VPN on your computer using Mac assuming, of course, that you already have a working Internet connection.

1. Under System Preferences open the "Network" panel and click on "+" (red arrow) to add a new VPN connection.

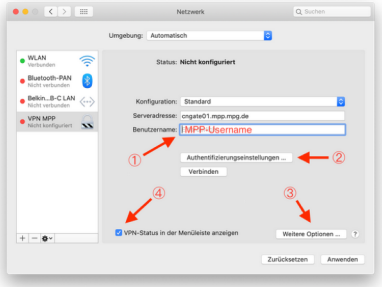


2. Interface is "VPN" and VPN Type is "L2TP over IPSec". Service Name can be chosen freely, the VPN connection will be shown under this name in the list of interfaces later.

3. Set Server Address "cngate01.mpp.mpg.de" and put your MPP username as Account Name (arrow 1). Click on "Authentication Settings..." (arrow 2).

4. Put your MPP password and "Jebek12" as Shared Secret.

5. Go to "Advanced..." (arrow 3).



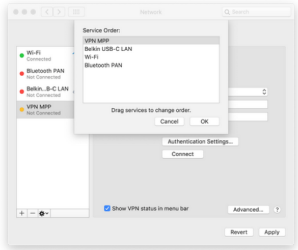
6. Check "Send all traffic over VPN connection".

7. Optionally check "Show VPN status in menu bar" (arrow 4).

8. The VPN Status Icon will appear in the menu bar from where you can connect/disconnect easily.

9. Click on the cogwheel icon and select "Set Service Order...".

10. Drag the VPN connection to the top of the list.



You just made your first steps towards being a terminal wizard!