# Table of Contents

**Linux - First Steps** ..... 1

**Basics** ..... 2
- Multiuser operating system ..... 2
- The command line ..... 2
- Working in a shell ..... 2
- The home directory - your personal folder ..... 2
- Essential shortcuts ..... 3

**Changing your password** ..... 4

**Folders and files** ..... 5
- Create a folder ..... 5
- Change directory ..... 5
- Change into your home directory ..... 5
- List files and folders ..... 5
- Rename files and folders ..... 6
- Move folders and files ..... 6
- Copy files and folders ..... 6
- Delete a file and folder ..... 7
- Take care of spaces in folder and file names ..... 7

**File and folder permissions** ..... 8
- Set file permissions ..... 8
- Some advanced examples ..... 9
- Setting your umask ..... 10
- Show my identity (username, groupnames) ..... 10
- Inherit group from parent folder ..... 10

**Text files** ..... 11
- View text files ..... 11
- Less command ..... 11
- Write the output into a text file ..... 11
- Sort text files ..... 12
- Search in files using grep ..... 12
- Using sed to edit files ..... 12

**Editors** ..... 14
- Emacs ..... 14
- Further text editors ..... 14

**Get info** ..... 15
- Free disk space in your home directory - Disk Quota ..... 15
- Shows the current location/folder ..... 15
- How much disk space is used by a folder or file ..... 15
- Show the usage of disk space ..... 15
- Get help for a command ..... 15
- Last entered commands ..... 15
- Monitor system resources ..... 16
  - top ..... 16
  - htop ..... 16
  - atop ..... 17
  - iotop ..... 17
  - ganglia ..... 17

# Table of Contents

**Get info**

xrestop....17
ps....18

**Processes / running jobs....19**

Show running processes....19
Start a job or program with lower priority....19
Determinate/kill a process....20
Determinate/kill a process "graphically"....20
Run a program and send it to the background....20

**Search....21**

Search a file in the current folder....21
Find files which have been changed during the last X days....21
Find files which have been changed recently....21

**Work on different systems....22**

Login to an other system....22
Copy files and folder to other systems....22

**Shell environment....24**

Which shell do I use?....24
Show shell variables....24
Show defined aliases....24
Define an alias....24
Customize your shell....24

**X Window System....25**

Restart X Server....25

**Problems and solutions....26**

I can no longer login to a Linux system....26
How can I find out the size of the folders in my home directory?....26
Checkout the Linux FAQ....26

# Linux - First Steps

# Basics

## Multiuser operating system

Linux is a multiuser operating system. Several users can work at the same time on a Linux system.

## The command line

To open a terminal (command line interface) run the command **xterm**, **konsole** or **gnome-terminal**. For example open the KDE menu and search for konsole. Or right click on your desktop, choose from the menu "Run Command...", and type **xterm**, **konsole** or **gnome-terminal**.

## Working in a shell

Type in the command including any options and parameters at the command line prompt. Press the return key to execute the command. **Please note:** Linux differentiates between small and capital letters!

**Auto completion**: The shell allows command completion using the TAB key. It allows you to complete the names of commands, file names, or directory names. An example:

```
ls /bo<TAB>
```

When you press the TAB key, /bo is automatically replaced with the value /boot. If you type

```
matla<TAB><TAB>
```

The first TAB will automatically complete to **`matlab`**. The second will show you all commands starting with **`matla`**.

**Re-execute commands:** At the command prompt press cursor up key and your last command will be shown. You can either re-execute this command or you can edit the command before executing it. Use cursor up and down to scroll through the most recent commands.

Type the command **`history`** to see your command history. Bash users can search in their history by pressing **`CTRL-r`**.

## The home directory - your personal folder

- Your home directory is a subfolder of the folder **`/home`**. The name of your home directory is equal to your username (**`/home/"username"`**).
- After login you are normally in your home directory.
- Your home directory is daily backed up.
- Disk space in your home directory is limited, see disk quota further down.
- For more information see LinuxHome

# Essential shortcuts

| Shortcut | Purpose |
|---|---|
| TAB | **Auto-complete** the command, if there is only one option, or else show all the available options |
| Ctrl+c | **Kill** the current process running in the terminal |
| Ctrl+d | Log out from the current terminal (does not work with tcsh) - use command **exit** instead |
| Ctrl+z | Send the current process to the background. The process will be stopped. Type **bg** to keep it running in the background |
| Ctrl+Alt+Esc | Mouse pointer will change to a cross. Kicking now on an application will kill the application |
| Ctrl+Alt+Backspace | Kills the graphical user interface (X windows). Has to be typed twice to take effect |
| Ctrl+Alt+F1 | Switch to text console 1. Text console 1-6 can be reached with Ctrl+Alt+F1 to Ctrl+Alt+F6 |
| Ctrl+Alt+F7 | Switch to the graphical user interface (X windows) |
| Middle Mouse Button | **Paste** the text which is currently highlighted somewhere else |
| Ctrl+s | Stop the transfer to the terminal. Terminal is locked. Press Ctrl+q to unlock! |
| Ctrl+q | Resume the transfer to the terminal. Try if your terminal mysteriously stops responding |

# Changing your password

Use the following webtool to change your ETH password:

https://password.ethz.ch

Please choose a password which is not easy to guess. Use small and capital letters, numbers and special characters like `,.$!`.

# Folders and files

## Create a folder

```
mkdir new_folder
```

The following command creates in one step a folder **`new_folder`** and a subfolder **`new_subfolder`** inside **`new_folder`**

```
mkdir -p new_folder/new_subfolder
```

## Change directory

```
cd new_folder
```

In one step decrease two levels

```
cd new_folder/new_subfolder
```

Move one folder up

```
cd ..
```

- `..` stands for the parent folder

## Change into your home directory

If you get let lost, change into your home directory with one of the following commands

```
cd
cd $HOME
cd ~
```

- **`~`** and **`$HOME`** are variables for your home directory

## List files and folders

Simple listing

```
ls
```

Show more details

```
ls -l
```

List also **hidden** files and folders. Hidden files and folders start with a dot (**`.hidden_file`**).

```
ls -la
```

Sort revers (-r) by modification time (-t)

```
ls -latr
```

## Rename files and folders

```
mv old_filename new_filename
```

## Move folders and files

Move the file **`file`** into the folder **`folder`**

```
mv file folder/
```

Move the file **`file`** out of the current folder into the parent folder (`..`)

```
mv file ../
```

## Copy files and folders

Create a copy of **`file1`** with the name **`file2`**

```
cp file1 file2
```

Copy a whole folder

```
cp -r folder1 folder2
cp -a folder1 folder2
```

- **-r**, **--recursive**: copy directories recursively
- **-a**, **--archive**: includes the option **-r**. In addition it will also preserve the mode, ownership and timestamps of files and folders

Copy a file **`/tmp/file`** into the current folder (`.`)

```
cp /tmp/file .
```

Copy the file **`file`** out of the current folder into the parent folder (`..`)

```
cp file ../
```

- `.` stands for the current folder
- `..` stands for the parent folder, i.e. the folder above the current folder

Use **wildcards** to copy several files at once.

```
cp file* folder/
```

Will copy all files having a filename starting with **`file`**.

```
cp file? folder/
```

Will copy all files matching the following pattern: file?, where ? can be any character. For example file1, file2, fileA, etc.

- **`?`** stands for a single character
- **`*`** stands for any number of characters including no character

# Delete a file and folder

```
rm file
rmdir folder
rm -r folder
```

- **`rmdir`** deletes the folder only in case it is empty
- **`rm -r`** deletes the folder recursively, including all its files and subfolders

**IMPORTANT:** Deleted files or folders can NOT be restored! Except from a backup, which maybe the case for files and folders in your home directory.

Delete all files in the current folder

```
rm *
```

Note, the above command will not delete hidden files (= **`.files`**) and folders.

Delete all files AND all folders in the current folder

```
rm -r *
```

Delete all files and folders in the current folder without asking you (-f = force)

```
rm -rf *
```

**WARNING:** Use the option **-f** with caution!

# Take care of spaces in folder and file names

Important: If the file or folder name contains spaces, your have to set the names in quotes

```
mv "My File" "My Backup File"
```

- Using spaces in folder or file names is NOT recommended. For better reading you may want use underscores instead of spaces (**`My_Backup_File`**)

# File and folder permissions

Only with the right permission you can access a file or change into a directory. Three different file permissions are known in Linux:

- **read** (**r**): File: Read and view. Directories: Read its content.
- **write** (**w**): Files: Write or edit. Directories: Can modify its content, i.e. creating/removing files or folders.
- **execute** (**x**): Files: Execute or run the file as a program. Directories: Can change into it.

The permissions can be set for the following user groups:

- **user**: The owner of a file or directory
- **group**: A user can be part of one or more groups.
- **other**: Defines the permission for all other users, not being the owner or belonging to the group.

**Note:**

- Directories have to be executable! Otherwise you get permission denied, if you want to change into it.
- In case you still get permission denied, check whether **all** directories in the path to the file are read and executable.

# Set file permissions

```
chmod g+w file          # gives write permission (w) for the group (g)
chmod o+rw file         # gives read/write permission (rw) for anybody/others (o)
chmod u+rwx file        # gives read/write/execute permission (rwx) to the user/owner (u)
chmod a+rwx file        # gives read/write/execute permission (rwx) to all (a)
```

- **u** user/owner, **g** group, **o** others, **a** all (= user+group+others)
- **r** read, **w** write, **x** execute
- **+** set permission, **-** remove permission

Beside the above modes, you can also use the **octal-mode** to change the permissions.

- 400 read by user
- 040 read by group
- 004 read by others
- 200 write by user
- 020 write by group
- 002 write by others
- 100 execute by user
- 010 execute by group
- 001 execute by others

The above numeric permissions can be added to set a certain permission. For example to give read/write by the owner and only read by everyone else (400+040+004+200 = 644) (**`-rw-r--r--`**)

```
chmod 644 file
```

A folder has to be executable and readable for everybody, but should be only writable by the owner (400+040+004+200+100+010+001 = 755) (**drwxr-xr-x**)

```
chmod 755 folder
```

Basically, there are three sets of bits, one each for user, group and other (in that order). Each set has three bits and each bit is an on/off switch for read, write and execute (in that order).

## Some advanced examples

Make all files and folders in the current directory only accessible for you. In other words remove (**-**) all rights (**rwx**) for the others (**o**) and the group (**g**)

```
chmod o-rwx *
chmod g-rwx *
```

With the above command only the permission of the files and folders in the current directory are changed. To change the permission recursively use the option **-R**

```
chmod -R o-rwx *
chmod -R g-rwx *
```

With the above command the permission of hidden files and folders (starting with a .) are not changed in the current directory. An other possibility to change permissions recursively can be done with the find command in combination with the option **-exec**:

```
find . -exec chmod o-rwx {} \;
find . -exec chmod g+rw {} \;
```

Assuming you would like to give to everybody (others) read permissions to the files and folders in the current directory

```
find . -type f -exec chmod o+r {} \;
find . -type d -exec chmod o+rx {} \;
```

An nice solution, if you want all files and directories inside /path/to/folder to be readable and accessible by everyone (a = all = owner, group and others)

```
chmod -R a+rX /path/to/folder
```

Assuming you would like to give to everybody (user, group and others) **rw** permission (but not **x**) to all files (**-type f**) and **rwx** permission to all directories (**-type d**) inside the current directory:

```
find . -type f -exec chmod ugo+rw {} \;
find . -type f -exec chmod ugo-x {} \;
find . -type d -exec chmod ugo+rwx {} \;
```

Same example as above, but others (**o**) should not gain write permission (only read permission):

```
find . -type f -exec chmod ugo+r,o-w,ug+w,ugo-x {} \;
find . -type d -exec chmod ugo+rwx,o-w {} \;
```

Sometimes it's maybe easier to use the octal-mode for chmod. The following commands do the same as the ones above

```
find . -type f -exec chmod 664 {} \;
find . -type d -exec chmod 775 {} \;
```

## Setting your umask

Your **`umask`** defines how permissions of new files and folders are set. Per default Linux has a umask of 0022, which means that you can read and write data and anyone else (group and others) can only read your data. In case you would like to exclude others from reading your data and only allow members of your group to read, set umask 0027. If you set umask 0077, only you can read and write your data.

Run the command umask in order to see your setting

```
umask
```

To set a new umask run

```
umask 0027
```

To permanently change your default umask add the above command to your shell profile file **`~/.cshrc`** (for tcsh) **`~/.bashrc`** (for bash).

## Show my identity (username, groupnames)

The command **`id`** prints your username id (**uid**) and your group ids (**gid**).

## Inherit group from parent folder

Set the group s-bit for the folder:

```
chmod g+s folder
```

New files or folders will inherit the group of parent folder.

---

# Text files

## View text files

```
less file.txt
more file.txt
cat file.txt
```

## Less command

```
less file
```

- The important keys in **`less`** are

| key | function |
|---|---|
| / | search, type in a pattern afterwards |
| n | Show next hit in the search mode |
| g | Jump to the beginning of the file |
| Shift+g | Jump to the end of the file |
| q | Quit less |

- Note, the same keys are also valid when viewing a manpage with the **`man`** command.

## Write the output into a text file

Write the current date (output of command **`date`**) into a text file **`time.txt`**

```
date > time.txt
```

- **`>`**: writes the standard output to the file. The file will be either created or, if already existing, overwritten.
- **`>>`**: appends the output to the file.

A simple example

```
echo "The current date:" > time.txt
date >> time.txt
```

**`cat time.txt`** will look like

```
The current date:
Mon Sep  8 12:12:34 CEST 2008
```

Write standard output AND standard error into the same text file

```
any_command > output_AND_error.txt 2>&1
```

- **`2>&1`**: **`2`** defines the standard error, which is written inot the same file as the standard output (= **`1`**)

# Sort text files

```
cat file | sort
cat file | sort > file_with_sorted_lines
cat file | sort | uniq
```

- **uniq** omits repeated lines

# Search in files using grep

```
grep <search_pattern> file
grep "Error" file
grep -i "error" file
grep "^Beginning of Line" file
grep "End of Line$" file
grep "pattern1\|pattern2" file
```

- **-i** Ignores case distinctions
- **^** matches the beginning of a line
- **$** matches the end of a line
- **\|** separates multiple patterns with **OR** condition

An example: How to show only the lines that are not comments in a file? Assuming comments start with a **#**, the regular expression is **^#**. Use the grep option **-v** to invert the sense of matching:

```
grep -v "^#" file
```

If you want in addition to suppress the output of empty lines, you can run

```
grep -v "^#" file | grep -v "^$"
```

More information about regular expression can be found for example at http://www.robelle.com/smugbook/regexpr.html.

A nice grep tutorial can be found at http://www.selectorweb.com/grep_tutorial.html.

Hint: For pdf files use **pdfgrep** for more info see

```
man pdfgrep
```

# Using sed to edit files

The stream editor **sed** can be used to edit files. If you use the option **-i** the file is edited in place. To prevent mistakes, it's recommended to run sed first without the option **-i** or you can run it with the option **-i.bak** which will create a backup of your file with extension **.bak**

To replace the name **Peter** with **Hans** in a file run:

```
sed "s/Peter/Hans/" file > newfile
```

This will create a new file with the name **newfile**. Or you can edit the file **file** in place with

```
sed -i "s/Peter/Hans/" file
```

If **Peter** appears more than once in a line and you want to replace all occurrences of **Peter**, you have to use the parameter **g** (=global)

```
sed -i "s/Peter/Hans/g" file
```

To delete (**d**) all lines which contains the word **Peter** run

```
sed -i "/.*Peter.*/d" file
```

**.*** is a placeholder for none or more characters.

More examples for sed can be found at http://www.cs.hmc.edu/tech_docs/qref/sed.html

---

# Editors

## Emacs

- Open file in emacs

```
emacs file
```

- Do not open emacs in windows mode (**nw** = no window mode)

```
emacs -nw file
```

- The important keys in emacs are

| key | function |
|---|---|
| Ctrl+x c | Quit Emacs |
| Ctrl+x s | Save |
| Ctrl+s | Search |
| Esc | Quit search mode |
| Esc+% | Search and replace |
| Ctrl+E | Jump to the end of the current line |
| Ctrl+A | Jump to the beginning of the current line |
| Ctrl+space | Set a mark |
| Ctrl+w | Cut the text between the last mark and the current position |
| Esc+w | Copy the text between the last mark and the current position |
| Ctrl+y | Paste |
| Ctrl+_ | Undo |

- There are many mores, see for example
  - http://www-pool.math.tu-berlin.de/doc/emacs/emacsintro.html

## Further text editors

- kwrite
- kate
- kedit
- vim
- gedit

# Get info

## Free disk space in your home directory - Disk Quota

```
homequota
```

For more information about your home directory see also LinuxHome.

## Shows the current location/folder

```
pwd
```

## How much disk space is used by a folder or file

```
du -chs file
du -chs folder
```

## Show the usage of disk space

```
df -h
df -h /lhome
df -h /net/atmos/data
```

## Get help for a command

```
"command" --help
"command" -h
man "command"
```

For example

```
ls --help
man ls
```

- To navigate inside a man page use the same key like navigating in **`less`**.

## Last entered commands

```
history
history | less
```

# Monitor system resources

## top

`top` is an alternative program to monitor CPU and memory usage.

```
top
```

If you want only to monitor some selected processes, give their process ID (PID) as an argument with to the option `-p`

```
top -p <PID>
top -p 9874 -p 30473
```

Or only monitor your processes (`$USER`)

```
top -u $USER
```

Per default top sorts the process list by CPU usage, press the following keys to change the sort order:

- `M`: sort processes by resident memory usage
- `P`: sort processes by CPU usage (default)
- `T`: sort processes by cumulative time
- `W`: save sorting state and open that way next time

Per default top shows only the command name in the COMMAND column, press `c` to show the full command line.

Sometimes the **load is high**, but `top` does not show any processes using a lot of CPU or memory. In order to show the "waiting" processes that are producing the load, run

```
top -i
```

## htop

`htop` is a tool to monitor CPU and memory usage. It is "newer" than top (see below) and you can scroll the list vertically and horizontally to see all processes and complete command lines. And htop supports mouse operations.

```
htop
```

Use the following commands:

- **Help**
  - `h` toggle help
- **Sort**
  - `Shift M` sort by memory usage
  - `Shift P` sort by CPU usage
  - `F5` show tree view of processes

- **Filter**
  - use `u` and arrows to show single users (alternatively `htop -u username`)
  - `Shift H` toggle user process threads

## atop

Use `atop` to get a nice overview which part of the system is the bottleneck (cpu, disk, memory, network). Watch for red text !

```
atop
```

Find red entries and check the leftmost column to see what causes the problem:

- `MEM` memory (RAM) -> memory
- `SWP` swapped memory -> memory
- `PAG` Paging frequency -> memory
- `LVM` logical volumes -> IO (more relevant than `DSK` below)
- `DSK` physical hard disks -> IO
- `NET` network -> too much network traffic (check the row at the bottom with `bond0`)

The bottleneck may also be on an other server! E.g. when reading data via `/net/` from another server!

## iotop

Use `iotop` to find out which users/ processes use high IO:

```
sudo iotop
sudo iotop -ao
```

- `-a` show accumulated I/O instead of bandwidth
- `-o` only show processes or threads actually doing I/O

`[nfsd]` means someone reads data from another server. There is no way to find out who/ which server.

## ganglia

Servers at IAC are monitored with **ganglia**: https://ganglia.iac.ethz.ch

## xrestop

Use `xrestop` to monitor server resources used by X11 clients

```
xrestop
```

## ps

Use `ps`. For example to list memory usage in percent (%) per user run

```
ps aux --no-headers | awk '{arr[$1]+=$4}; END {for (i in arr) {print i,arr[i]}}' | sort -g
```

Or to list CPU usage in percent (%) run (please note a system with 64 cores has in total 6400% CPU)

```
ps aux --no-headers | awk '{arr[$1]+=$3}; END {for (i in arr) {print i,arr[i]}}' | sort -g
```

Note: the command above also needs CPU, so sometime your number is therefore higher. Just run the command several times to get a good overview.

List number of processes running per user

```
ps aux | awk '{ print $1 }' | sort | uniq -c
```

# Processes / running jobs

## Show running processes

```
ps aux
ps aux | grep firefox
ps aux | grep ^$USER
```

Hint: If your username is longer than 8 characters, search (grep) for the first 7 characters of your username

```
ps aux | grep ^${USER:0:7}
```

## Start a job or program with lower priority

To start a job or program with lower CPU or Disk I/O priority use `nice` (`renice` for running jobs) and `ionice`. For example

```
nice -n 19 ionice -c 3 my_program
```

1. **nice -n 19** set CPU priority to lowest level (19)
2. **ionice -c 3**: set disk I/O priority to idle level (-c 3)
   idle level = a program running with idle I/O priority will only get disk time when no other program has asked for disk I/O

More examples for **ionice**

```
ionice -p PID                    # get priority of process with process number PID
ionice -c 3 -p 1004              # set process with PID 1004 as an idle io process
ionice -c 2 -n 0 my_script       # run my_script as a best-effort program with highest pri
ionice -c 3 -p $(pgrep -u $USER) # change IO priority of all your running processes to idl
```

More examples with **nice**

```
nice -n 13 my_script         # run my_script with niceness level 13
```

Note, niceness level 19 is the lowest priority. Niceness level 0 is the default

For already running jobs you need to use `renice`:

```
renice 19 -u $USER  # set all of your running jobs to the lowest priority
renice 19 -p PID    # set jobs with PID to the lowest priority
```

Note, niceness level 19 is the lowest priority. Niceness level 0 is the default

For more info please see

```
man nice
man ionice
man renice
```

# Determinate/kill a process

```
kill <PID>
kill 395
kill -9 356
```

- **`-9`** will kill the process hardly
- The **PID** is the process identification number, which is listed in **`ps aux`** or **`top`**

To kill **all** you running process, type

```
killall -u $USER
```

or stronger

```
killall -9 -u $USER
```

# Determinate/kill a process "graphically"

Type the command

```
xkill
```

The mouse pointer gets a cross or a skull. A mouse click will now determinate the program below the mouse pointer. Press the **`Ctrl+C`** to quit the "xkill" mode.

# Run a program and send it to the background

```
program &
emacs &
firefox &
```

Or

```
firefox
```

Press **Ctrl+z** will suspended the running program. Afterward type the command **`bg`**. This will resume the program in the background, as if it had been started with **`&`**.

```
bg
```

In some cases the program will be terminated when you close or exit the terminal from which you have started it - even if you have started the program in the background. The Linux tool **`screen`** is maybe a good solution to overcome this problem. For more info see LinuxUseScreen. However, screen only works for programs which run completely in a terminal window.

# Search

## Search a file in the current folder

```
find . -name index.html
find . | grep index
```

Apostrophs needed when using wildcards:

```
find . -name 'index.*'
```

## Find files which have been changed during the last X days

For example to find all files inside the current directory (.) and below that have be changed during the last 2 days (-mtime = modify time):

```
find . -type f -mtime -2
```

## Find files which have been changed recently

Create a list containing the modification data (%TY-%Tm-%Td %TH:%TM:%TS) and the filename (%f) and sort the list numerically (-n). Latest Files will be shown at the end of the list.

```
find . -type f -printf "%TY-%Tm-%Td %TH:%TM:%TS %f\n" | sort -n
```

# Work on different systems

## Login to an other system

```
ssh username@hostname
ssh beyerleu@firebolt.ethz.ch
ssh beyerleu@firebolt
```

## Copy files and folder to other systems

Use secure copy (**scp**):

```
scp file beyerleu@firebolt.ethz.ch:
scp file beyerleu@firebolt.ethz.ch:/home/beyerleu
scp -r folder beyerleu@firebolt.ethz.ch:/home/beyerleu
```

- The first two commands are equal
- Use **-r** to copy folders recursively

Instead of **scp** you can also use **rsync** over ssh:

```
scp -r     folder beyerleu@firebolt.ethz.ch:/data/beyerleu/
rsync -av  folder beyerleu@firebolt.ethz.ch:/data/beyerleu/
```

The advantage of **rsync** is that it only copies files which are not yet copied before.

In order to synchronize two folders use the option **--delete** which will delete extraneous files from the destination folder:

```
rsync -avz --delete folder beyerleu@firebolt.ethz.ch:/data/beyerleu/
```

Please use the option **--delete** with caution. It can delete all your new files, if you use it the wrong way round. Therefore use the option **-n** which makes rsync perform a trial run that doesn't make any changes and produces mostly the same output as a real run:

```
rsync -n -avz --delete folder beyerleu@firebolt.ethz.ch:/data/beyerleu/
```

More useful rsync options:

```
-z                      files are compressed before transfer, and uncompressed after trans
--remove-source-files   sender removes synchronized files, useful if you would like to del
-u, --update            skip files that are newer on the receiver
-c, --checksum          compare checksum (not only mod-time&size) if the same file exists
```

In case your connection is not stable, call rsync in a loop until rsync gives you a successful return code:

```
error=1
while [[ $error -ne 0 ]]
do
   rsync -av myfolder firebolt.ethz.ch:/path_to_data/
   error=$?
done
```

For more info about rsync, please see

```
man rsync
```

---

# Shell environment

## Which shell do I use?

```
echo $SHELL
```

## Show shell variables

```
set
```

## Show defined aliases

```
alias
```

## Define an alias

- Bash shell

```
alias ls_nice='ls -lag --color=tty'
```

- TCSH shell

```
alias ls_nice 'ls -lag --color=tty'
```

## Customize your shell

You can put personal shell settings like **aliases**, **environment variable** definitions, **path**, etc. into the personal initialization file of your shell.

- If you're using **bash** put them into **`~/.bashrc`**
- If you're using **tcsh** put them into **`~/.cshrc`**

---

# X Window System

The X Window System provides the graphical user interface. KDE is our default Windows manager.

## Restart X Server

To restart the X Server as user press **`Ctrl+Alt+Backspace`**. Please note, restarting the X Server will terminate your current login session.

# Problems and solutions

## I can no longer login to a Linux system

- You have exceeded your diskquota. Try to login over ssh and delete files.
- **`/tmp`** is 100% used. Try to login over ssh and delete files in **`/tmp`**.
- You have forget your password. The administrator can set a new password.
- Your account was disabled. Ask the administrator to enable it again.
- Quite often your Trash which is under **`$HOME/.local/share/Trash`** fills up the whole disk space. **Don't forget to delete your trash from time to time.**

## How can I find out the size of the folders in my home directory?

To print the size of your folders in the home directory type:

```
find $HOME -maxdepth 1 -type d -exec du -hs {} \;
```

## Checkout the Linux FAQ

For further help see Linux FAQs: LinuxFAQ

---

This topic: IT > LinuxFirstSteps
Topic revision: r70 - 02 Jun 2026 - 14:21:46 - UrsBeyerle

FOSWIKI Copyright © by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding Wiki? Send feedback