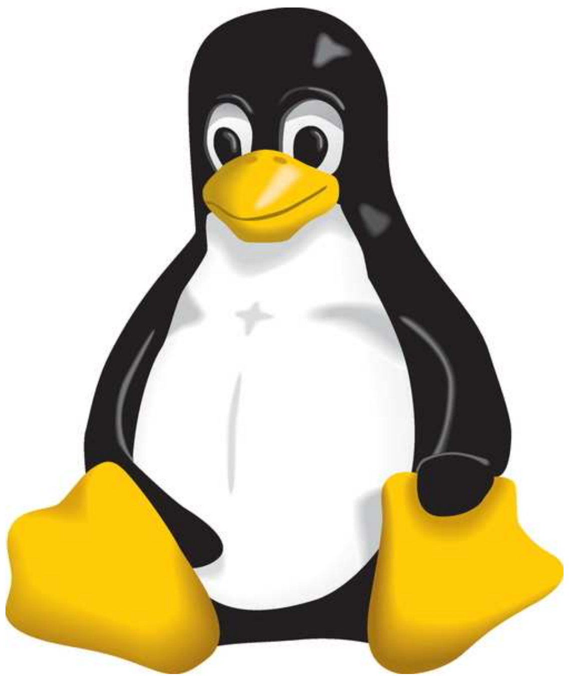
LINUX First Contact



Thomas Schanz
IAAT - University of Tuebingen

4. September 2019

Contents

1 Preamble

```
Hello?  Yes?  Megadodo Publications, home of The
Hitchhiker's Guide to the Galaxy, the most totally
remarkable book in the whole of the known Universe, can I
help you?

What?

Yes, I passed on your message to Mr Zarniwoop, but I'm
afraid he's too cool to see you right now.  He's on an
intergalactic cruise.

Yes, he is in his office, but he's on an intergalactic
cruise.[1]
```

LINUX is a Universe on your desktop.

Even after a livetime working with LINUX you will be surprised by discovering something new from time to time. LINUX is a Universe on your Desktop. If you really want to know how Computers work you should investigate LINUX further. LINUX is completely open. You can look into any detail of LINUX and you can contribute to its further development if you like. LINUX provides you with all tools and information for software development without any payment, all you have to spent is some time.

But also if you don't want to become a software developer, LINUX is worth a glimpse. LINUX is completely free and one of the most stable and secure operating systems in the world. It provides you with plenty of free tools and software suits and behind LINUX is no company that is spying on your privacy or collecting data for their own business benefit. You will recognize that the higher amount of time you may have to spent at the beginning is a very good investment after a second consideration.

This introduction is for those who will encounter LINUX the first time, may be just as a User or already as an Admin of their first own LINUX installation. It should be clear, that this manual is fully incomplete and just scratching the surface of this fascination operating system.

Have a lot of fun!

[1]Douglas Adams - The Hitchhikers Guide to the Galaxy

2 Introduction

power On a new adventure

LINUX is an open, network capable, multiuser, full-multitasking operating system that supports symmetric multiprocessing and real time execution. LINUX is a free version of the UNIX operating system which was invented in 1969 by the Bell labs (USA) to run on Mainframe- and Supercomputers and received contributions from various IT companies like: SCO, IBM, SGI, NEXT, DEC, APPLE, Sun-Microsystems and Hewlett-Packard since then.

... modern LINUX systems are 64 Bit. There are two major LINUX families which are spread widely in various distributions:

- BSD based (Berkeley) LINUX
  (Debian/Ubuntu/Xandros/Knoppix)
  dpkg - Package-Manager, apt- Advanced Package Tool, synaptic

- System V based LINUX
  (Fedora/RedHat/Suse/Mandrake/Scientific-LINUX)
  rpm - Package-Manager, yum - Yellowdog Updater Modified, yumex

besides that there are still some commercial UNIX systems available, like AIX (IBM), SOLARIS (ORACLE) and HP-UX (HP).

3 Operating Systems

The operating system of the computer is the interface between the software and the hardware of the computer. The architecture of the computer can be compared to a multi storey building as it is shown in Figure 1.

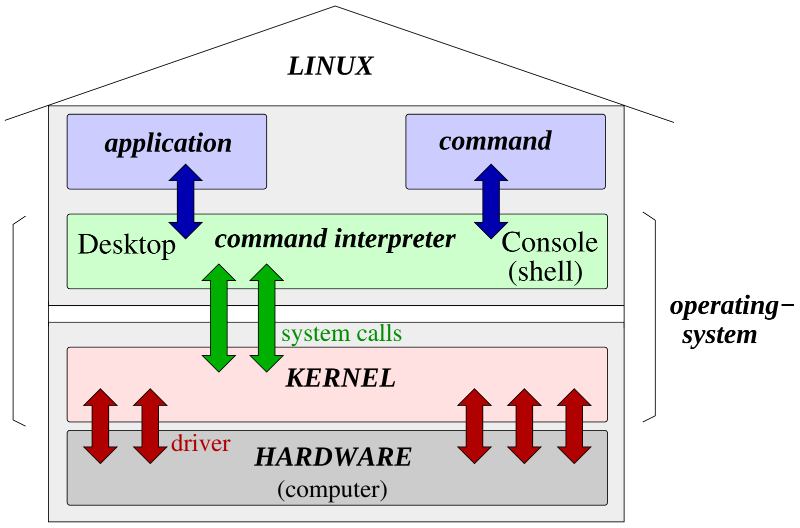


Figure 1: *The setup of the LINUX operating system is comparable to a multi floor building. The Hardware in the basement is isolated from the applications in the upper floors by the KERNEL. The KERNEL is the heart of the LINUX system. It controls the hardware alone and multiplexes the system calls of the various running commands and applications to access the systems hardware.*

The basement contains the hardware, the processing unit, the memory, disk drives, graphics hardware and so on. Above at ground floor, there is the operating system. Its purpose is to control the hardware layer below by so called 'drivers'. The upper floor holds the applications or programs that run on the system. The applications communicate by so called 'system calls' with the operating system. Their access to the hardware is always controlled by the layer of the operating system, applications can never have direct access to the hardware! The main tasks of the operating system are:

1. it defines a specified interface to applications for the coordinated access to the hardware of the computer. This is especially for programmers of particular importance
2. it allows the access to the hardware of the computer
3. it coordinates the access of simultaneously running applications to the hardware of the computer
4. it isolates the data of different users and limits their access to the system or to the data of other users
5. it provides a number of tools in order to manipulate files

In principle, computer could also run without any operating system. In that case however, the programmer would need to have exact knowledge of all the details of the particular hardware to make an application run stable. Because the hardware of different computers today varies in many aspects, this would be hardly a good approach for software development. As a further disadvantage only a single program could be running simultaneously on that kind of computer system. Small microcontroller systems built for a single purpose can still work today without any operating system. The applications in this cases have to be especially constructed for the particular microcontroller hardware.

- Modern operating systems like LINUX are **'multiuser capable'**. This means that more than one single User can be logged in to the machine simultaneously over the network. The applications this Users run will not conflicting in any way with the applications of any other User, nor with the data the Users are using or generating. Any User that has an account on the system will have his own private user directory (HOME-directory) which can not be accessed by any other User. The operating system coordinates the access of different Users to the machine and keeps their activities isolated from each other.
- A **'network capable'** operating system allows the Users to login over a remote network. Users can login via network from a remote computer or terminal to the host computer and start a session. The output of applications on the host computer is than directed and routed automatically to the remote terminal or displayed on the remote computer. The Users logged in over network will neither conflict with any other user login via network nor with a User that may be logged in at the Console of the host computer locally.
- A **'multitasking capable'** operating system can operate more than one single application on the same computer hardware at the same time. Multitasking is one of the main requirements for a multiuser operating system. But also one single User may want to run several applications simultaneously on the system. The Operating system will multiplex the hardware access of the applications to the computers hardware in a very fast time frame. This will normally not be recognized by the Users. Besides the Users, also the operating system itself will always run several applications on the system because the operating system is nothing else than a collection of applications.
- LINUX is **'multiprocessor capable'**. If the computer hardware offers more than one single processing unit, a multiprocessor capable operating system can distribute the processing of application on the number of available units in order to speed up the processing time. In case of multitasking the operating system will try to distribute the processing tasks equally to the available processing units (symmetric multiprocessing). Most modern computers provide multicore CPUs today which can be used by multiprocessor capable operating systems.
- Some LINUX systems are also **'realtime capable'**. This means that the applications have access to the hardware on a realtime basis. Normal operating systems process data in a sequential order where interrupts can happen at any time and delay the execution. For most applications this is no disadvantage. In general modern computers are so fast, the User will never recognize that the application was interrupted during execution. However there are cases where the execution of commands

is highly time critical and must be guarantied performed within a particular time interval. In many measurement applications in physics time is a crucial parameter and a signal is directly correlated to time. In such cases a 'realtime' operating system can be necessary where executions can be synchronized on specified clock cycles of the systems hardware. A realtime operating system provides a special realtime scheduler to force the execution of applications in realtime.

```
A computer lets you make more
   mistakes faster than any
    other invention - Mitch
          Ratcliffe
```

4 Login

After booting the LINUX system correctly, the Login-Manager is started, showing a login screen as shown in Figure 2.

In order to login you must type in your login name, the corresponding password and press 'Enter'.

The system will log you in with a graphical Window-Manager that provides the Desktop and the Console as interfaces to the system. This 'Desktop-Manager' can vary from LINUX distribution to LINUX distribution a little bit. There are at least a dozen different Desktop-Managers widely distributed and available for most LINUX distributions, GNOME, KDE, XFCE, LXDE, CDE are the most popular. Common to all Desktop-Managers is the capability to control the computer via mouse, they also provide a 'File-Manager' and an 'Application-Manager'. The File-Manager allows you the access and manipulation of files and folders via mouse: creating, naming, copying, moving and deleting. The Application-Manager provides an easy access to applications. Many installed applications can be selected for execution via the Application-Manager. This is however usually limited to those application that make use of a window based user interface. Console type programs are often not accessible via the Application-Manager because they don't generate graphical output.

The Desktop-Manager also provides functionality for system administration, software installation, user logout or for shutting down the computer system. When you logout from the system, all user files and desktop settings usually will be saved and made available again at the next login.

LINUX systems are multiuser capable! As a default every User gets his own work space on the Filesystem, the 'HOME'-directory. The HOME-directories contains all data, folders and files that belong to a particular User. No User can look into the data of any other User without permission, the HOME-directories are completely isolated from each other by default. The access to the data of other Users is in LINUX controlled by File-Permissions. File permissions can be changed by the User via Address-Flags if he wants to grant other Users the access to his data.

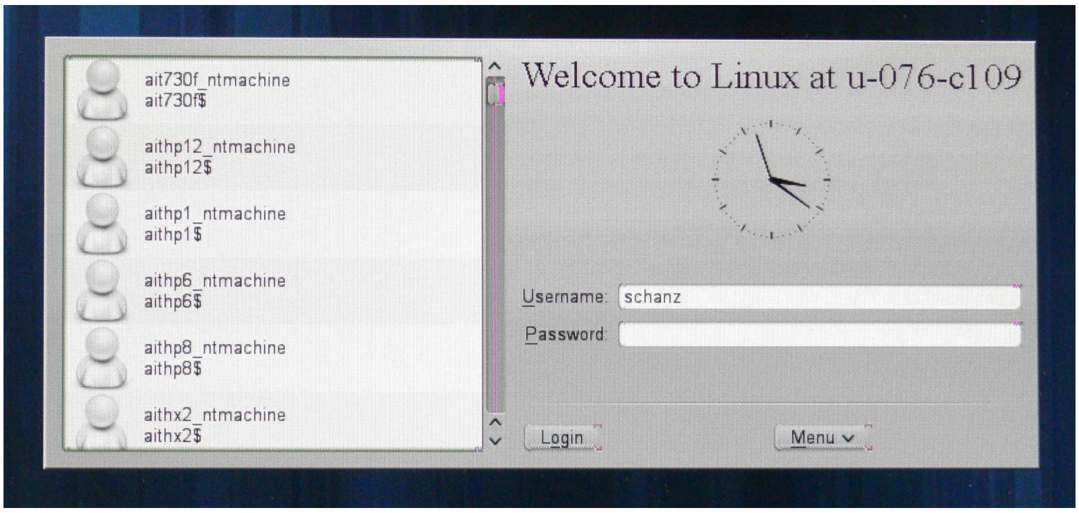


Figure 2: *How the Login-Manager looks like on your system depends on your distribution and your individual installation. LINUX provides a variety of different Login-Managers. Common to all is that they will request your Login-Name and a Password. Most Login-Managers also provide a menu where you can select the kind of Desktop-Manager (KDE/GNOME/CDE..) you want to launch for your session and some buttons to properly shutdown the system in case you cant login for some reason. The shutdown option however will not be offered on public systems where other Users may be working on the machine.*

```
Behind every great computer
 sits a skinny little geek
```

No User is allowed to change system data that belong to the LINUX operating system! There is no way for a normal User to damage or destroy the LINUX operating system neither by accident nor with purpose. Only the Super-User (system administrator) can change system files! However normal Users can destroy their own data if they are not careful enough. The strict separation of user-data and system-data makes LINUX relative secure against computer viruses or any kind of malicious software. A potential damage will be always limited to the Users own work area, the HOME-directory. If you run your own LINUX system with Super-User privileges you should be operating with extreme caution! With Super-User permissions you can harm and destroy the complete operating system including all user datas just by mistake or accident. For this reason you should never login with Super-User privileged when this is not necessary. Even if you are the systems administrator you should always login merely with your User account and change into Super-User privileges (su- command) only for the time frame this is really required.

Be aware, with Super-User privileges you can destroy the LINUX operating system completely (only the software, not the hardware)! If you delete the KERNEL for instance, lets say by accident, your system will never boot up again :-(.

LINUX systems you do not administrate yourself may not be allowed to be shutdowned by normal Users! There can be dozens of Users be logged in to the system, working on important issues. If a User could shutdown the system all other Users would be logged out immediately from their sessions without any warning! The shutdown of a system is therefore restricted to Super-User privileges only. But even if you are the Super-User you should be careful before you shutdown the system. If there are other Users working on the system you should consult them by time and grant them a couple of minutes or hours to end their sessions themselves before you shutdown.

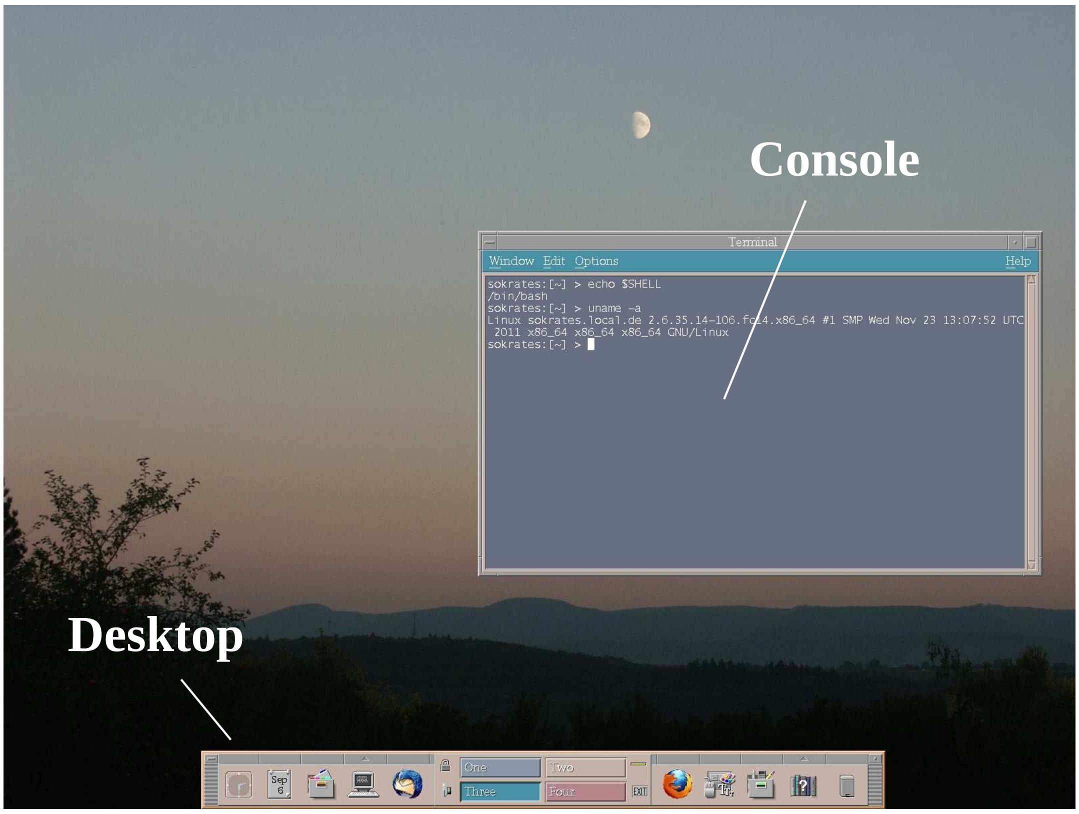
Even worse than an unannounced system shutdown is a power failure or the sudden power off of the computers hardware! You should never switch off the computer without shutting down the LINUX operating system properly before! As long as the operating system is running, many processes will cache their data in memory. Even if you save data to a disk drive, the operating system will not execute the storage command immediately and can delay the actual writing operation for some time, keeping the data in the systems volatile memory. If you switch off the computer without warning, the operating system is not able to flush the data down to disk drive just in time, you will produce a real data loss on the system and the Filesystem will become corrupted! LINUX will try to repair the Filesystem at the next booting of the system automatically. In most cases this works well, in some rare cases however, the damage to the Filesystem is severe, the operating system may stay damaged beyond repair and you need to install LINUX anew. Besides some continuous rumour: You can never destroy any hardware of the computer by switching it off deliberately, only the software of the operating system can be damaged or destroyed. You can always make a new installation!

So hands off the power switch! You should only shutdown LINUX systems that you own yourself and only switch off a running computer when you exactly know what you are doing!

```
At the source of every error
   which is blamed on the
 computer you will find at
   least two human errors,
   including the error of
 blaming it on the computer
```

5 The User-Interface

In order to start an application the User will need some kind of User-Interface to the computer. This can be either a graphical user environment, enabling the User to interact with the computer via mouse, or it is a Terminal emulation providing a Commandline interface where the computer is controlled by written commands. From here on, the graphical user environment is called the **’Desktop’** and the Commandline interface is called the **’Console’**. Both User-Interfaces possesses their pros and cons and are shown in Figure 3 for illustration. The Desktop is usually easier to operate for beginners. It is meant to be intuitive and mimics a kind of office environment for the operation of the computer. Most commands will be replaced by mouse moving and clicking. It is quite the right choice to consume the daily news magazine or for browsing through the Internet. In contrast to the Desktop is the Console. The Console is a much more powerful interface as the Desktop, it provides a programmable Shell which can automate the execution of applications or commands via pipelines and scripts. Commands are typed into the Console via the keyboard. Usually the operating system provides a few dozen commands as a baseline which can easily increase into the hundreds if additional software packages are installed. In principle there is no difference between a command and an application. All applications can also be called from the Commandline by typing their execution name. The great variety of commands and command options makes the reputation of the Commandline as complicated and less user friendly, but it is also the source which makes the Console such an incredible powerful interface. Most computer experts prefer the Console. A typical Console is shown in Figure 6. The complete interactions with the computer is performed by typing commands.



```
sokrates:[~] > echo $SHELL
/bin/bash
sokrates:[~] > uname -a
Linux sokrates.local.de 2.6.35.14-106.fc14.x86_64 #1 SMP Wed Nov 23 13:07:52 UTC
 2011 x86_64 x86_64 x86_64 GNU/Linux
sokrates:[~] >
```

Figure 3: *The two command interfaces of LINUX: Desktop and Console. In principle you could configure LINUX to launch only a Console without the Desktop, in former times this was once the default. Modern LINUX however will always launch the Desktop first. The Desktop provides the Console on a window system and permits the launch of several Consoles at once running in multitasking. The screenshot here shows a running CDE Desktop with a Dashboard that provides push buttons to launch the Console and other applications.*

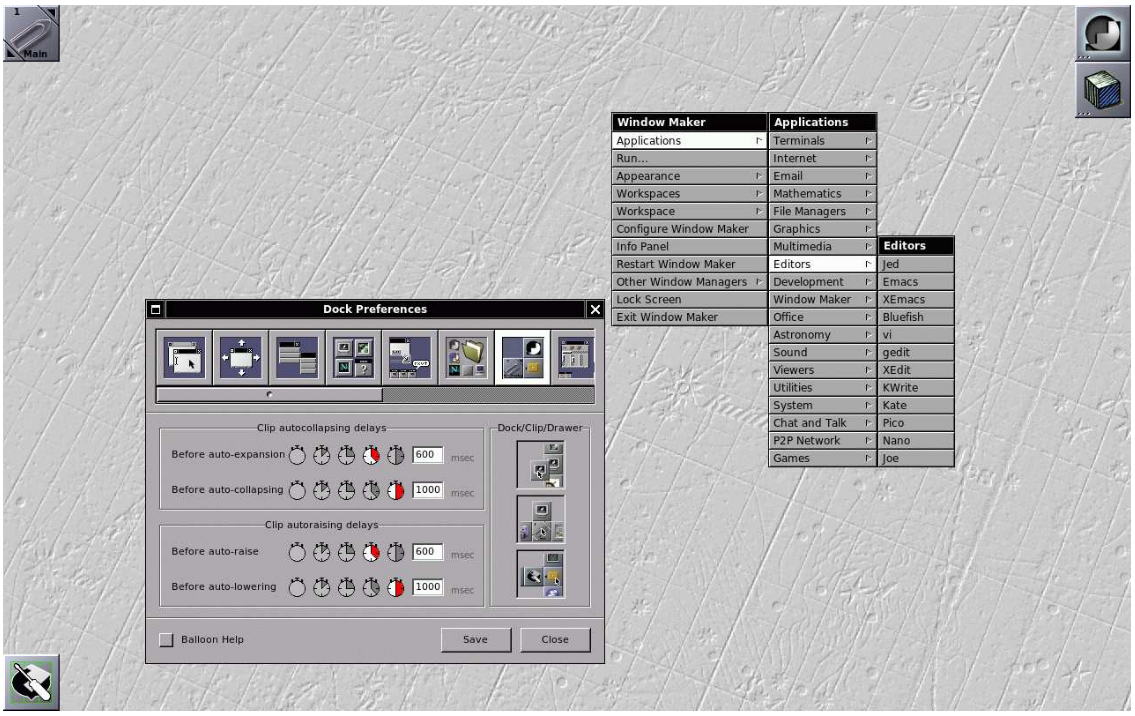
An example: Imagine you want to replace the string „AIT“ by the string „IAAT“ in all files with the file extension '.tex' which are spread over 600 subdirectories on your system. To perform this task with the Desktop you may open all 600 subdirectories by mouse click and open all '.tex'-files in those subdirectories with a Texteditor. In the Texteditor you use the search and replace function to make the changes. On the Console instead you can go to the corresponding 'root'-level directory and just type the following Commandline which will perform the task in a few seconds:

```
[~]> for i in 'find . | grep .tex' do; sed -i -e "s/AIT/IAAT/g" $i; done'.
```

The great power of the Console is, that it provides a programmable interface!

5.1 The Desktop

There are a number of different Desktops available for LINUX. Some imitate MICROSOFT WINDOWS and offer a menu driven way to launch applications. Others make use of a dashboard like shown in Figure 5 that provides direct access to the most vital daily required functions of the system. Common to all Desktops is that they provide a File-Manager to the Filesystem of the computer where files and directories can be accessed and manipulated via mouse movements and interactions. Applications are usually launched via a menu system or via an Application-Manager, both are operated by mouse also.



Clip autocollapsing delays
Before auto-expansion 600 msec
Before auto-collapsing 1000 msec
Clip autoraising delays
Before auto-raise 600 msec
Before auto-lowering 1000 msec
Balloon Help

Figure 4: *Screenshot of the running 'WINDOWMAKER' Window-Manager.*

There are plenty of different Desktops available for LINUX. The most common Desktops are:

- KDE - Desktop:
  very powerful and well equipped Window-Manager, slow and demands lots of resources
- GNOME - Desktop:
  The Fedora LINUX standard Window-Manager, powerful and well equipped, slow, demands lots of resources
- XFCE - Desktop:
  powerful and well equipped Window-Manager, light and fast
- LXDE - Desktop:
  well equipped Window-Manager, light and fast
- UNITY - Desktop:
  the Ubuntu LINUX standard Window-Manager
- MATE - Desktop:
  compact and fast Window-Manager
- CDE - Desktop:
  compact and fast Window-Manager, industry standard on UNIX for more than a decade
- FVWM - Desktop:
  very simplistic Window-Manager, fast and compact, from the early days of LINUX
- WINDOWMAKER - Desktop:
  very light and fast Window-Manager

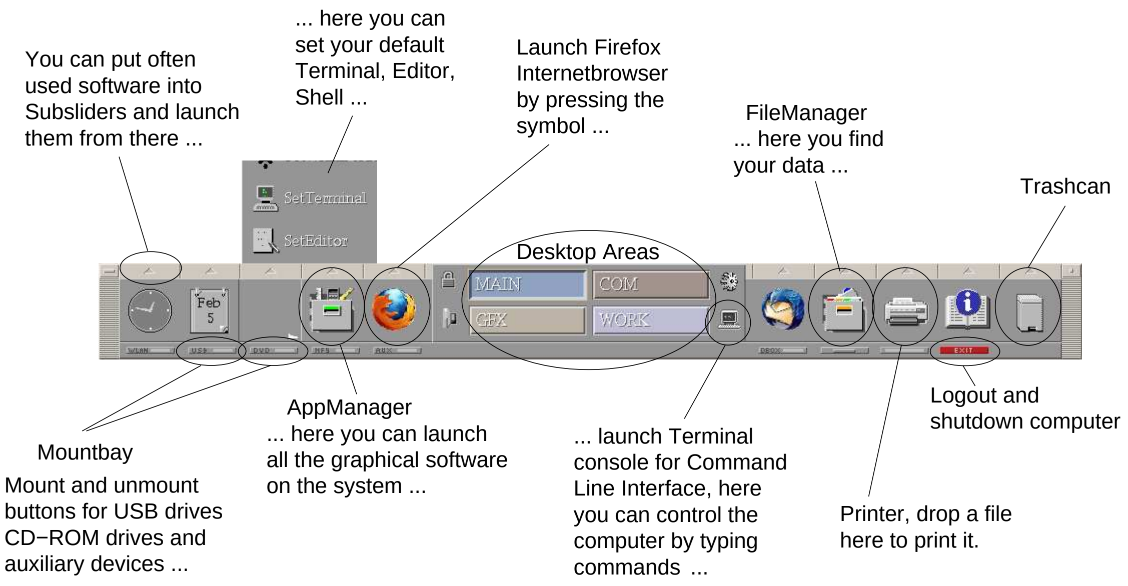

You can put often used software into Subsliders and launch them from there ...
... here you can set your default Terminal, Editor, Shell ...
Launch Firefox Internetbrowser by pressing the symbol ...
FileManager ... here you find your data ...
Trashcan
SetTerminal
SetEditor
Desktop Areas
MAIN
COM
GFX
WORK
Feb 5
EXIT
Mountbay
Mount and unmount buttons for USB drives CD−ROM drives and auxiliary devices ...
AppManager ... here you can launch all the graphical software on the system ...
... launch Terminal console for Command Line Interface, here you can control the computer by typing commands ...
Printer, drop a file here to print it.
Logout and shutdown computer

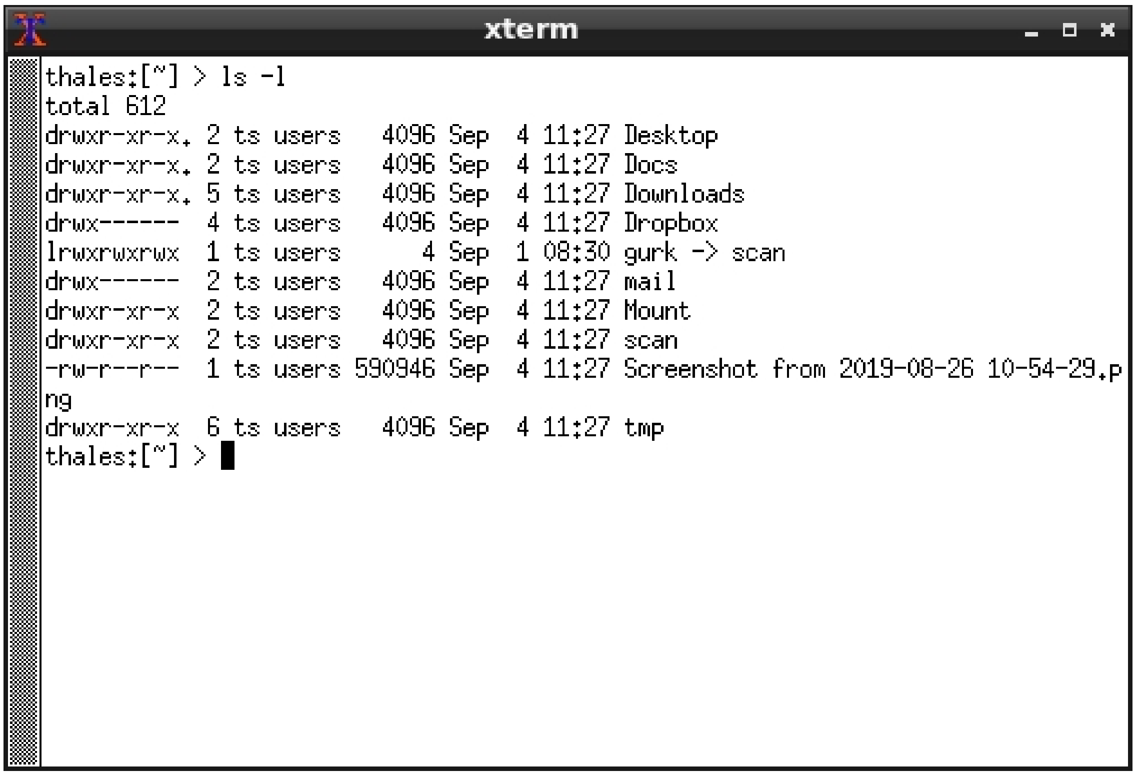
Figure 5: *The dashboard of CDE provides direct access to all vital functions of the system.*

The different looks and feels of different LINUX distributions is mainly caused by the choice of the Desktop. Most distributions have strong preferences to one Desktop or another. On the baseline,, different LINUX systems are very much alike, its the choice of the Desktop that make them appear such different from each other. Most Desktops make use of different window decorations and window widgets.

Usually the Desktop can be chosen freely by the User via a menu of the Login-Manager screen before logging in to the system.

5.2 The Console

The other User-Interface is the Console. On the level of the Console all LINUX distributions are almost identical! The Console is mainly influenced by the Terminal-Emulation and the Shell that is used to provide it, see also chapter 15 and 14. The Console is just a window that provides a Commandline Interface to the computer system, see also Figure 6.



```
thales:[~] > ls -l
total 612
drwxr-xr-x. 2 ts users   4096 Sep  4 11:27 Desktop
drwxr-xr-x. 2 ts users   4096 Sep  4 11:27 Docs
drwxr-xr-x. 5 ts users   4096 Sep  4 11:27 Downloads
drwx------  4 ts users   4096 Sep  4 11:27 Dropbox
lrwxrwxrwx  1 ts users      4 Sep  1 08:30 gurk -> scan
drwx------  2 ts users   4096 Sep  4 11:27 mail
drwxr-xr-x  2 ts users   4096 Sep  4 11:27 Mount
drwxr-xr-x  2 ts users   4096 Sep  4 11:27 scan
-rw-r--r--  1 ts users 590946 Sep  4 11:27 Screenshot from 2019-08-26 10-54-29.p
ng
drwxr-xr-x  6 ts users   4096 Sep  4 11:27 tmp
thales:[~] > 
```

Figure 6: *The Screenshot shows a typical Console. The Console here displays the result of the 'ls -l' command in the HOME-Directory of the User 'ts' on system 'thales'.*

```
 Build a system that even a
fool can use and only a fool
    will want to use it
```

6 System-Shutdown

How is a LINUX system shutdown properly?

Because LINUX is a multiuser operating system only the system administrator is allowed to shutdown the system, in general normal Users will not have the privileges! The first step to shutdown the system is to inform the logged in Users and wait until they have logged out and quit their sessions!

It is possible to configure LINUX in a way that normal Users can shutdown the system also. In this case Users can shutdown the system via some shutdown buttons of the Desktop-Manager. For standalone LINUX systems at home such an option is very useful. The permission to shutdown the system for certain normal Users can be configured by the Super-User by editing the ’sudo’-configuration file ’/etc/sudoers’ and adding the appropriate usernames. The specified Users can shutdown the system either by selecting the shutdown option from the Desktop-Manager or by typing the following Commandline to the Console:

```
[~]> sudo shutdown -h 0
```

(the password of the User may be requested again)

The letter ’-h’ means ’halt’, the system will definitely stop and power down, not reboot! The number ’0’ means that there is no shutdown delay granted. When you issue the shutdown command all Consoles of all Users will display a shutdown warning. The Users are ultimately told to logout. You can set the number to ’60’ and grant a logout time of 60 minutes. If you are not alone on the system you should always grant some time for the Users to stop their working and logout from the system by themselves. If you type in the number ’0’ the system will shutdown instantly. Only do that if you are alone on the system!

If you are the administrator on the system and know the Super-User password you can also acquire Super-User privileges by logging in as Super-User permanently. For this you type the command 'su -' into the Console, the Super-User password is required. If you can login as Super-User you can shutdown the system by typing the shutdown command directly:

```
[~]> shutdown -h 0
```

or

```
[~]> reboot -h
```

ATTENTION!
you must wait until the message:

```
Halted, you may now cycle power
```

appears. Do not cycle power before this message!

On modern computer systems LINUX makes the computer cycle power itself after the 'shutdown' command was completed.

ATTENTION!

Be careful always to shutdown the operating system properly, never cycle power before the shutdown has fully completed, a defective operating system and data loss may result otherwise!

```
   Brian Kernighan has an
 automobile which he helped
    design.  Unlike most
automobiles, it has neither
speedometer, nor gas gauge,
nor any of the numerous idiot
  lights which plague the
 modern driver.  Rather, if
the driver makes any mistake,
 a giant "?" lights up in the
  center of the dashboard.
"The experienced driver", he
  says, "will usually know
      what's wrong."
```

7 Processes

The Heart of the LINUX operating system is the 'KERNEL'. The KERNEL is a special program (e.g. vmlinuz-4.8.13-100.fc23.x86_64) in the directory '/boot' of the Filesystem, that is executed at system start. The KERNEL executes the complete control to the hardware of the computer and the so called 'process scheduling' which coordinates the execution and the access of applications to the hardware.

At boot time the LINUX system will start the KERNEL program first. The KERNEL will launch the 'init'-process which is the very first process running on the system. What the computer really executes are not programs, the computer executes 'processes', programs launch a number of processes! The very first process is the 'init'-process with the process-ID number '1'. Starting with this first process, all further processes will be forking from the 'init'-process as it is shown in Figure 7.

The 'init'-process is also called the 'Parent'-process of the system. All later processes are 'Child'-processes of the 'init'-process. 'Child'-processes will inherit all parameters that were used to call their 'Parent'-process. If a 'Parent-Process' is killed, all its 'Child'-processes will be ended also (different from real live). At boot time the 'init'-process is forked in a great number of 'Child'-processes which makes the operating system working.

As a last step at booting the system will start the Login-Manager. When the User is logging in, the user login-process will become the 'Parent'-process of all the Users future processes, it will fork again and again further. If the login-process would be killed accidentally, it would immediately quit also all of the Users Child-processes and the User would be logged out instantly.

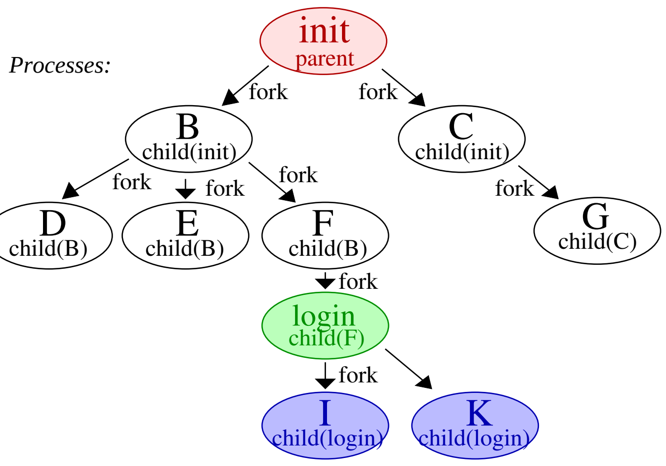


Figure 7: *At boot time the KERNEL is starting the 'init' process, it has the process-ID number '1'. All further processes will be launched out of the init-process by a method called forking. Forking means the init-process is the 'Parent'-process and will duplicate all its launch parameters to his 'Child'-processes. Using the command **'ps -fe'** you can display a list of all running processes on your system -> see below. The PPID is the Process-ID (PID) of the 'Parent'-process. Process-ID number '3730' is here the login-process of the user 'schanz' for instance. The colors refer directly to the illustration in Figure 1.*

As in certain cults it is possible to kill a process if you know its true name. – Ken Thompson and Dennis M. Ritchie

Every time a program or command is executed in the Console, the Parent-process of the Console is forking into one or several Child-processes. The Parent-process is blocked for the time frame the Child-processes are running and continues after the Child-processes have completed. In this case, such a Child-process is considered a 'Foreground'-process. This is the default.

It is possible to start processes as 'Background'-process. For this the particular application is called with a '&' sign behind the program- or command-name. The Parent-process will generate Child-processes also in this case but it will not wait until the Child-process has completed and will continue immediately. For instance:

```
[~]> firefox &
```

will start the 'Firefox'-Webbrowser as an Background-process. The Console will be immediately freed to receive and execute further commands. The Child-process (firefox) in this case will run as a Background-process to its Parent-process, the Console.

It is also possible to put a process into Background even if it was initially called as a Foreground-process. For this just go to the Console the Foreground-process was launched from and press **'ctrl z'** on the keyboard to suspend the Foreground-process. Subsequently you can type **'fg'** (foreground) to the Console in order to start the Foreground-process again (stop suspension) or you type **'bg'** (background) to the Console to put it into Background.

If you press **'ctrl c'** on the keyboard, the Console stops the execution of a Child-Fore ground-process immediately. This is often used to stop a program or command before normal completion.

The command **'ps'** (process status) or **'ps -fe'** will display a list of all running processes. This will give you an overview of the processes launched by applications, about the process-IDs and the Users who launched it. The program **'top'** displays a table of processes that have the highest processing demands on the system, you can use 'top' to identify the processes that do cause the highest load on the system.

```
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 08:54 ?        00:00:01 init [2]
:
root        1225     1  0 08:54 ?        00:00:00 udevd --daemon
daemon      2577     1  0 08:54 ?        00:00:00 /sbin/portmap
statd       2589     1  0 08:54 ?        00:00:00 /sbin/rpc.statd
root        2924     1  0 08:54 ?        00:00:00 /usr/sbin/rsyslogd -c3
root        2935     1  0 08:55 ?        00:00:00 /usr/sbin/acpid
root        3669  3662  0 08:55 ?        00:00:00 /usr/sbin/gdm
:
schanz      3730     1  0 08:55 ?        00:00:00 /usr/bin/gnome-keyring-daemon -d --login
schanz      3731  3669  0 08:55 ?        00:00:00 /bin/sh /usr/bin/startkde
:
```

Using the command **'kill'** you can force specified processes to be stopped and removed from further process scheduling. The command will be discussed in more detail later.

8 The Filesystem setup

All LINUX systems are using a hierarchical Filesystem setup as it is shown in this paragraph and illustrated in Figure 8. The entire operating system is organized by this Filesystem setup. For the User it is very useful to have an idea about the structure of the Filesystem and to know where certain files of the operating system are located. The Filesystem contains all files and folders of the operating system.

The Filesystem can be extended when you attache and mount additional disk drives like CDROM disks or USB sticks. Not all parts of the Filesystem however must be writable. CDROMs for instance can only be accessed for reading.

All files of the Filesystem are either accessible by command via Console or graphically via the File-Manager of the Desktop as it is shown in Figure 9. The File-Manager of the Desktop has the ability to mask such files their names start with a dot '.' . The file '.bashrc' for instance will not be displayed by the File-Manager of the Desktop by default.

The Filesystem contains many files or different filetypes for different purposes. Many filetypes are internally organized in ASCII and can be read by an usual Texteditor, most

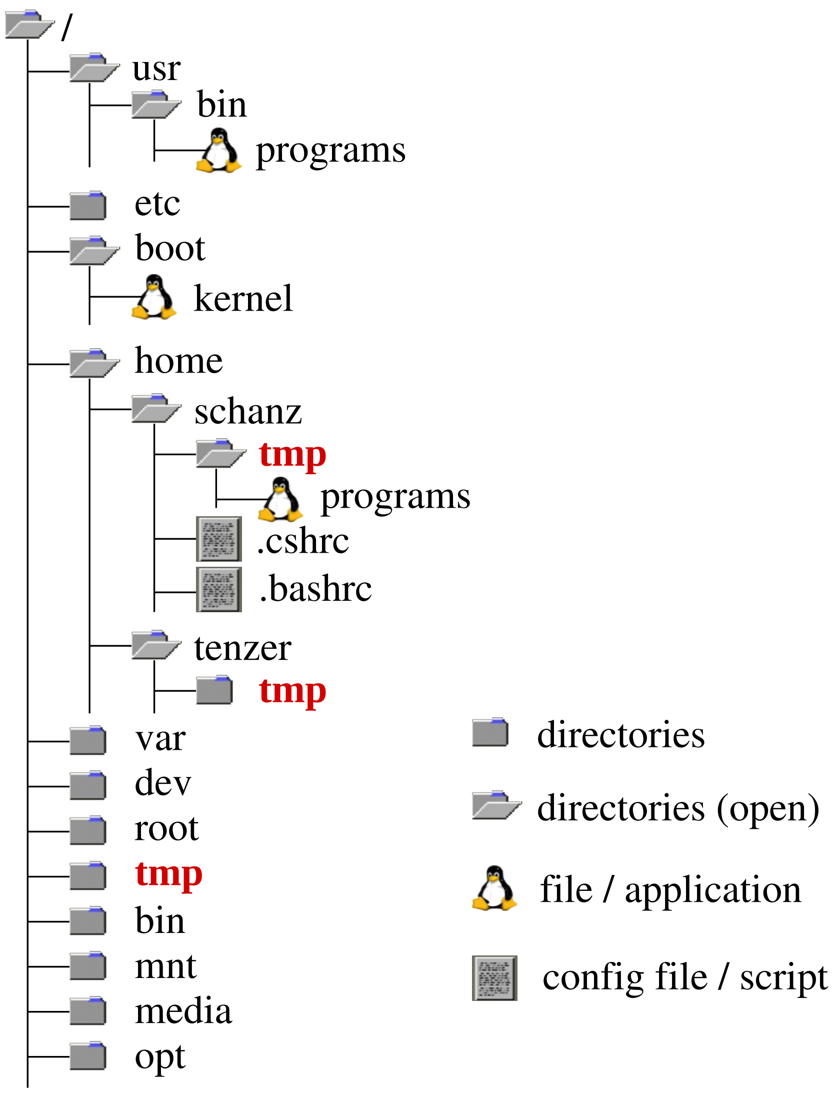
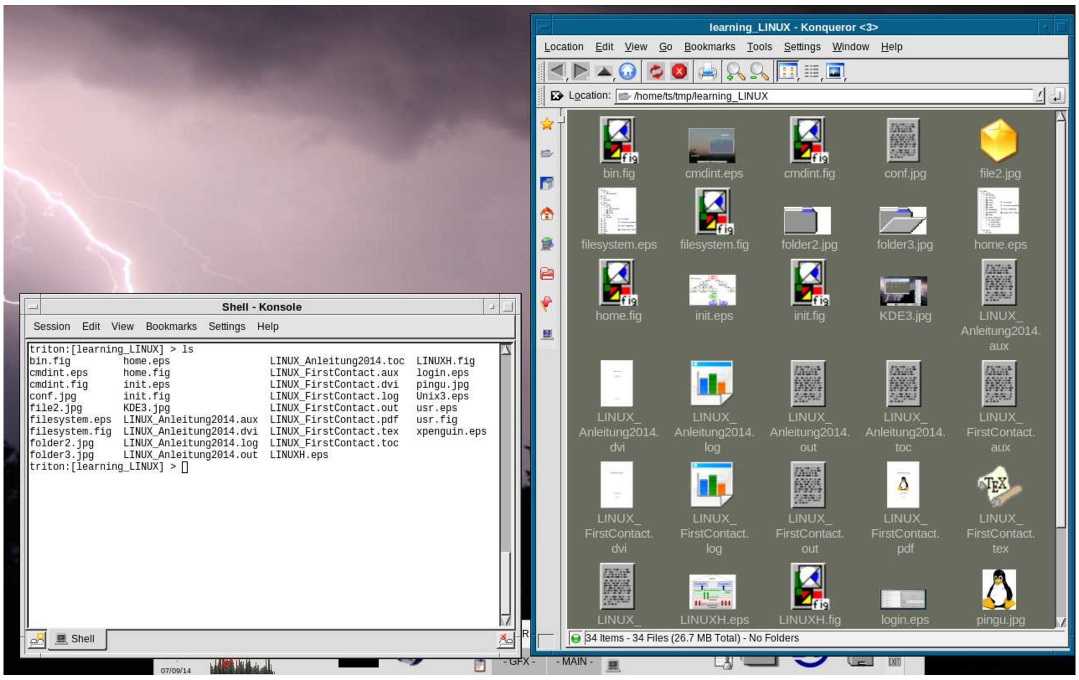


Figure 8: *LINUX uses a hierarchic Filesystem setup. The top level directory is called the 'root'-level directory '/' of the system, all subdirectories will branch from here. The Filesystem contains all files and directories of the LINUX system including all user data. File- or directory-names are called 'relative' as long the complete path from the 'root'-level down is not explicitly given. There can be different files or directories with the same 'relative' name be in existence in the Filesystem, like the* ***'tmp'****-directory in this example. In order to change the Command-Prompt into the directory 'tmp' of the User 'schanz' you either know where the prompt currently is, than you can address the change 'relative'. Assuming the Command-Prompt is at '/home/tenzer/tmp' the addressing for the change using relative addressing would be: 'cd ../../schanz/tmp/'. Otherwise you could always use 'absolute' addressing to perform the change by typing: 'cd /home/schanz/tmp/' which would work from everywhere in the Filesystem the same way.*

filetypes however are binary coded and only readable by a corresponding software.



```
triton:[learning_LINUX] > ls
bin.fig         home.eps                 LINUX_Anleitung2014.toc  LINUXH.fig
cmdint.eps      home.fig                 LINUX_FirstContact.aux   login.eps
cmdint.fig      init.eps                 LINUX_FirstContact.dvi   pingu.jpg
conf.jpg        init.fig                 LINUX_FirstContact.log   Unix3.eps
file2.jpg       KDE3.jpg                 LINUX_FirstContact.out   usr.eps
filesystem.eps  LINUX_Anleitung2014.aux  LINUX_FirstContact.pdf   usr.fig
filesystem.fig  LINUX_Anleitung2014.dvi  LINUX_FirstContact.tex   xpenguin.eps
folder2.jpg     LINUX_Anleitung2014.log  LINUX_FirstContact.toc
folder3.jpg     LINUX_Anleitung2014.out  LINUXH.eps
triton:[learning_LINUX] > 
```

```
34 Items - 34 Files (26.7 MB Total) - No Folders
```

Figure 9: *All files and directories of the Filesystem can either be accessed by the Console (left) or via a graphical File-Manager that usually belongs to the Desktop (right).*

There are filetypes for all kinds of applications, image-files, movie-files, document-files, sound-files and so on which have a different internal data structure. The File-Manager of the Desktop can distinguish between different filetypes and will show different pictograms according to the data the file is containing. The File-Manager will also link the filetypes to corresponding default applications that are attached to the particular filetype. Clicking the mouse on the filetype will then automatically launch the default application and load the file. Many File-Managers can also link secondary applications to a filetype that can be selected via mouse from a menu if required. Some File-Managers can display a preview of the file in the pictogram of the filetype. In general the File-Manager will recognize if the filetype contains any program data or is an executable application. If the file is an application, the File-Manager will launch it the moment you click the mouse on it.

Besides launching corresponding applications the File-Manager allows also the 'copy' or 'move' or 'remove' of any files. Usually this is executed by 'drag and drop' operations. When you drag and drop a file from a particular folder into another folder the File-Manager will execute the appropriate 'copy' or 'move' operation. You can also drag and drop a file on the Trashcan symbol of the Desktop in order to remove it from the Filesystem completely. There are plenty of additional functions concerned with the File-Manager of the system that are beyond the scope of this document and depends on the particular File-Manager. Please consult the documentations for the File-Manager that is installed on your system for further informations.

```
Computers are unreliable, but
    humans are even more
unreliable.  Any system which
 depends on human reliability
   is unreliable.  - Gilb
```

From here on we want to focus more on the functionality of the Console which is much more common and generalized for the different kinds of LINUX distributions, in fact its identical at any LINUX system.

The Filesystem can be easily accessed via the Commandline interface of the Console, which provides a number of commands for Filesystem navigation and file manipulation, see chapter 12. All files and folders of the LINUX system are rooted to the top level directory of the Filesystem which is called the ’root’-directory and shorted by the slash ’/’ symbol. Because the Filesystem is hierarchic there can be many layers below the ’/’-directory. When a file is located in a subdirectory of ’/’, the path to the file can be specified in an ’absolute’ manner if the name starts with the ’/’. The KERNEL ’vmlinuz-4.8.13-100.fc23.x86_64’ for instance may have the ’absolute’ path ’/boot/vmlinuz-4.8.13-100.fc23.x86_64’ because it is located in the subdirectory ’boot’ of the root-directory ’/’. The use of ’absolute’ pathnames makes the addressing of a file always unambiguous in the hierarchy of the Filesystem. In contrast to ’absolute addressing is ’relative’ addressing. For ’relative’ addressing you always need to know the location of the ’Command-Prompt’. The Command-Prompt is a pointer that defines from which Filesystem location the addressing of a filename will be started. The command 'cd /home/schanz/tmp' for instance will move the Command-Prompt of the Commandline into the subdirectory ’tmp’ of the HOME-directory of User ’schanz’. The Command-Prompt will ’change directory’ (cd) to ’/home/schanz/tmp’. The User ’schanz’ that is located in the HOME-directory ’/home/ schanz’ could achieve the same just by typing ’cd tmp’ because the Command-Prompt of User ’schanz’ will be pointing to the directory ’/home/schanz’ after login per default. Absolute pathnames are always unambiguous, ’relative’ pathnames require to know where the Command-Prompt is currently pointed. You can always find out where the Command-Prompt is currently pointed by typing the ’pwd’ command into the Commandline:

```
[~]> pwd
```

which will give you the current position, maybe ...

```
/home/schanz
```

The designation of the 'root'-directory depends on the context, don’t be confused! The 'root'-directory of the system is located at ’/’, the 'root'-directory of the User ’schanz’ is located at ’/home/schanz’ instead. In the future we will refer to this 'root'-directory of the

User as his 'HOME'-directory only. There may be 'root'-directories of some application software installations also. To make things worse, the Super-User of the system is called the 'root'-User! This is because in former UNIX days the HOME-directory of the Super-User was not located under '/home/root' or '/root' as it is today but directly in the root-directory '/' of the system. Can you still follow? :-)

The most important subdirectories of the 'root'-directory '/' are:

```
/usr    : contains most of the operating system and open source applications
/etc    : contains the configuration of the operating system, of services and applications
/boot   : contains the KERNEL and the Bootloader
/home   : contains the user-HOME-directories
/var    : contains log-files of the operating system and of applications
/dev    : contains device driver of the KERNEL
/root   : contains the configuration of the 'root'-User (Super-User)
/tmp    : contains temporary files and folders to run the system
/bin    : contains the executables of the core operating system
/sbin   : contains applications for devices (e.g. Filesystem operations)
/mnt    : contains mounted disk drives
/media  : contains mounted media systems of portable devices (USB-Sticks .),
          that will be mounted via automounter
/opt    : contains commercial applications and software
```

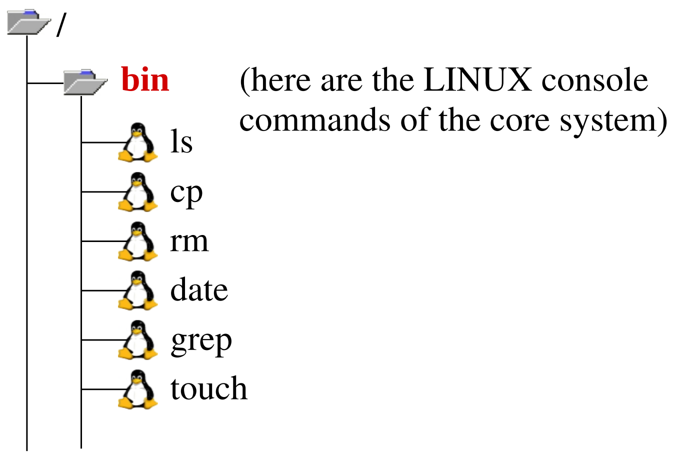


Figure 10: *The '/bin' directory contains the Console commands of the core operating system.*

The main part of the LINUX Console commands is located under '/bin' (core commands) or under '/usr/bin', illustrated in Figure 10 and 11. Also most applications will be installed under '/usr/bin'. The difference between commands and applications is marginal, in principle commands are also applications.

In the directory '/usr', most of the LINUX operating system is located. Besides system commands (bin) it contains also user applications, software libraries **(lib)**, definitions for software development **(include)**, the compiler and all the corresponding documentations of the software **(man)**. The software libraries are mostly so called 'shared libraries' (shared objects (.so)). They contain precompiled system calls that can be linked dynamically into applications and will be used by a large number of software applications together. This enables a fast and compact software development. The disadvantage, is

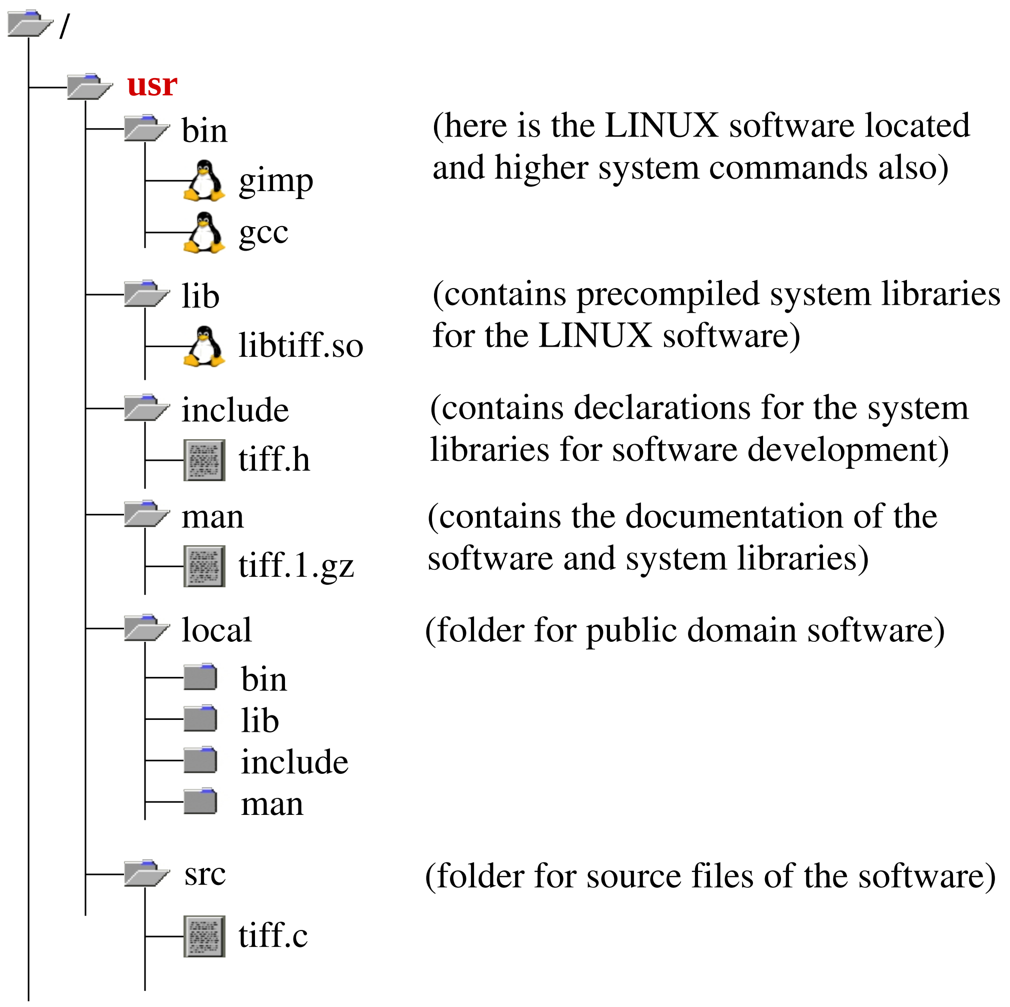

/
usr
bin
gimp
gcc
(here is the LINUX software located and higher system commands also)
lib
libtiff.so
(contains precompiled system libraries for the LINUX software)
include
tiff.h
(contains declarations for the system libraries for software development)
man
tiff.1.gz
(contains the documentation of the software and system libraries)
local
(folder for public domain software)
bin
lib
include
man
src
(folder for source files of the software)
tiff.c

Figure 11: *The '/usr' directory contains the largest part of the software that makes the LINUX operating system.*

one of the shared objects modified or upgraded, a large number of application might be affected. Typical LINUX systems use a number between 500 and 1500 shared objects, all located under '/usr/lib".
The distribution of software to the subdirectories 'bin', 'lib', 'include' and 'man' is typically for LINUX. You may find this structure also in other directories of the Filesystem besides '/usr'.

9 The HOME-directory

On LINUX systems every User has his own HOME-directory as it is illustrated in Figure 12. It has the name of the users login and is located under the directory '/home/<username>/'. The HOME-directory of the Users is per default secured against all other Users → see File-Permissions in chapter 13. No User is allowed to access the files of any other User. Besides the User himself, the Super-User is the only other one on the system that can look into the files of a user. You can trust, that the Super-User will not access your files without your explicit wishes! Every User is allowed to do in his own HOME-directory what ever he wants. He can generate an arbitrary number of files and folders as long he does not exceed the limits of disk space of the system. On some public systems the Super-User can grant Quotas for the disk space users are allowed to occupy. The Quotas shall guarantee that all Users get the same amount of disk space on the Filesystem available. They prevent that one single User will occupy all the available disk space for his own purpose.

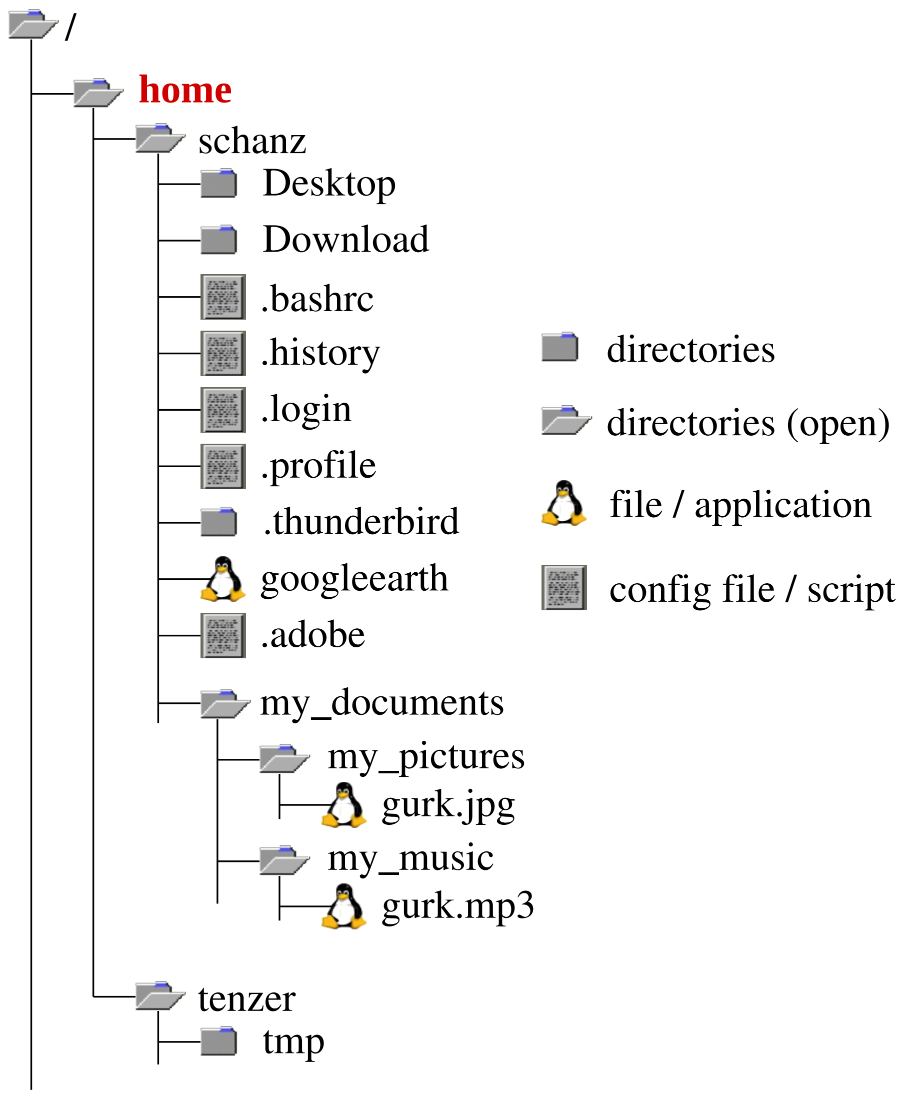


Figure 12: *The user-HOME-directory contains all the Users files and data including the local user-specific software configurations. Most of this configuration files start with a '.' in the filename and contain the private settings the User has set up in the specific software. The User is free to generate as many files and subfolder he want inside his own HOME-directory.*

The HOME-directory of a user will also contain his local configurations. This configurations affects the settings of the Desktop, the Console, the Internet Browser, the eMail

Client and also a number of applications (most of them located in '/usr/bin') the User has access to. The HOME-directories allows every user his own private configurations. The particular config files usually have the same name as the corresponding application but start with a '.' in the filename. Try the command 'ls -al' to display the hidden config files in your HOME-directory.

10 Execution PATH

A very frequent question of LINUX beginners is: „Where do I find the programs?"

This is truly not an easy question. On LINUX systems applications can be installed at very different locations. This can depend on the LINUX distribution, on the manufacturer of the software or on the preferences of the systems administrator. In my opinion this is truly one of the weaknesses of LINUX compared to its predecessor UNIX. The only common rule is that you will find the executables most probably in some subdirectory of the system that has the name 'bin', e.g '/bin', '/usr/bin', '/usr/local/bin' etc.

On UNIX-systems the application either belongs to the operating system and is located in „/usr/bin" of the Filesystem, or it is a commercial application and has its own installation subdirectory you will find under „/opt/". e.g.:

```
Oracle databases:                     /opt/oracle/bin
Adobe Reader:                         /opt/adobe/bin
Xilinx FPGA Development system:       /opt/xilinx/bin
... etc.
```

On LINUX systems this is unfortunately all a bit more anarchic. Operating system commands are usually located in „/usr/bin" of the Filesystem, same as on UNIX systems. Opensource applications meanwhile will also be installed to „/usr/bin/" mixing up with the commands of the operating system. Some LINUX distributions install the applications rather to „/usr/local/bin/" or „/usr/local/X11/bin/". Others use „/usr/X11R6/bin/" or „/usr/share/bin/", etc. The 'firefox' webbrowser installs itself for some reason under „/usr/lib64/", some applications still use the „/opt/" directory to put their stuff in and so on.

Is the install directory of some software not in the search path ('echo $PATH') of the Console, the system will not find the executable program when you type in the command name will reply with a 'command not found' error message. There is unfortunately no easy solution to find a particular software and its executable on the Filesystem when they are installed to a location that is not in the PATH of the Console. You can always use the 'find'-command if you know at least the name of the executable. Some software however uses names for the executable that deviate from the softwares package name, you will have a lot of fun about finding it :-).

If you can find the subdirectory that contains the executable of a certain software, you can add this location into the search PATH of the Console in order to launch the program in the future just by typing its name without any path. For this you edit the resource file of the Shell (most probably '.bashrc') in your HOME-directory and add the new location. The specific line in the resource file will probably somehow look like this:

```
PATH=/bin:/usr/bin:/usr/bin/X11:/usr/local/bin:/usr/local/bin/X11:/home/schanz/.dt/bin
:$PATH
```

Considering for instance, you want to add the location of NASAs HEASOFT executables that are located under ’/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/bin’ to the PATH, the modifications for the PATH variable will look like this:

```
PATH=/bin:/usr/bin:/usr/bin/X11:/usr/local/bin:/usr/local/bin/X11:/home/schanz/.dt/bin
:/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/bin:$PATH
```

Here is ’PATH’ used as an ’Environment-Variable’ that tells the Shell where to look for executable programs. PATH contains the directories just as a list separated by ’:’. You can display the PATH-variable if you type ’echo $PATH’ into the Commandline. After the modification above, the search PATH contains now the directories:

```
/bin
/usr/bin
/usr/bin/X11
/usr/local/bin
/usr/local/bin/X11
/home/schanz/.dt/bin
/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/bin
```

If you type in a command or an application name, the Shell will search in the directories shown above for the executable. The PATH mechanism of the Shell relieves you from knowing where the executable of a certain program actually is located, just type the name into the Commandline. If you don’t use the Bourne-Shell (bash) you have to modify or add the PATH variable in the corresponding resource file for the Shell you are using instead. You don’t know what Shell you are using? Type ’echo $SHELL’ into the Commandline of the Console. $SHELL is another Environment-Variable. This will give you an output like this:

```
/bin/bash
```

A similar mechanism is controlling the search PATH for documentations. The LINUX system offers manuals for each command and program installed on the system. These Man-Pages contain short descriptions how to operate the command or application including its options. You can open a specific Man-Page for a certain command by typing the following into the Commandline:

```
[~]> man <command/application name>
```

Try the Man-Page for the ’date’ command as an example:

```
[~]> man date
```

This will open the Man-Page how to use the 'date' command and will look like this:

```
DATE(1)                          User Commands                          DATE(1)


NAME
        date - print or set the system date and time

 SYNOPSIS
        date [OPTION]... [+FORMAT]
        date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

 DESCRIPTION
        Display the current time in the given FORMAT, or set the system date.

        -d, --date=STRING
               display time described by STRING, not 'now'

        -f, --file=DATEFILE
               like --date once for each line of DATEFILE

 .........
```

All Man-Pages follow a similar setup. In order to find the Man-Page for a given input the Shell uses an Environment-Variable again, called 'MANPATH'. The MANPATH tells the Shell where to look for Man-Pages. You can display the content of MANPATH by typing the following to the Commandline:

```
[~]> echo $MANPATH
```

this will generate an output like this:

```
/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/man:/usr/man:/usr/local/man:
/usr/share/man
```

The MANPATH variable contains a list of the directories separated by ':', where the Shell searches for Man-Pages.

The control of Shell functions and software via Environment-Variables is a very common mechanism on LINUX. You can display the Environment-Variables that are used on your system by typing the command 'env' to the Commandline. You can also add Environment-Variables of your own to control script functions at any time you want.

11 LINUX Configuration

The '/etc'-directory contains the global configurations of the LINUX operating system. This concerns the booting of LINUX the same way as the configuration of networking, the setup of the windowsystem (X11), the Secure-Shell (ssh), Filesystem configurations, the printing system or the setup of services like firewall, printer queues, name services, network Filesystems, dhcp and so on. Different from other popular operating systems, LINUX is not using a database to control the settings of software applications nor the configuration of the system. Instead each software has a bundle of its own config files hosted under the '/etc'-directory. This makes LINUX very stable and protected against attacks or software failures. The configuration of different softwares or services is independent from each other. A failure in the configuration of one application will not affect the configuration of another software or services. Systems other then LINUX, that depend on one single database mechanism for everything are much more fragile and unstable during operation and must be rebooted the moment the database has experienced any major changes. Figure 13 shows an illustrations of the '/etc'-directory including some of the most important config files of the system.

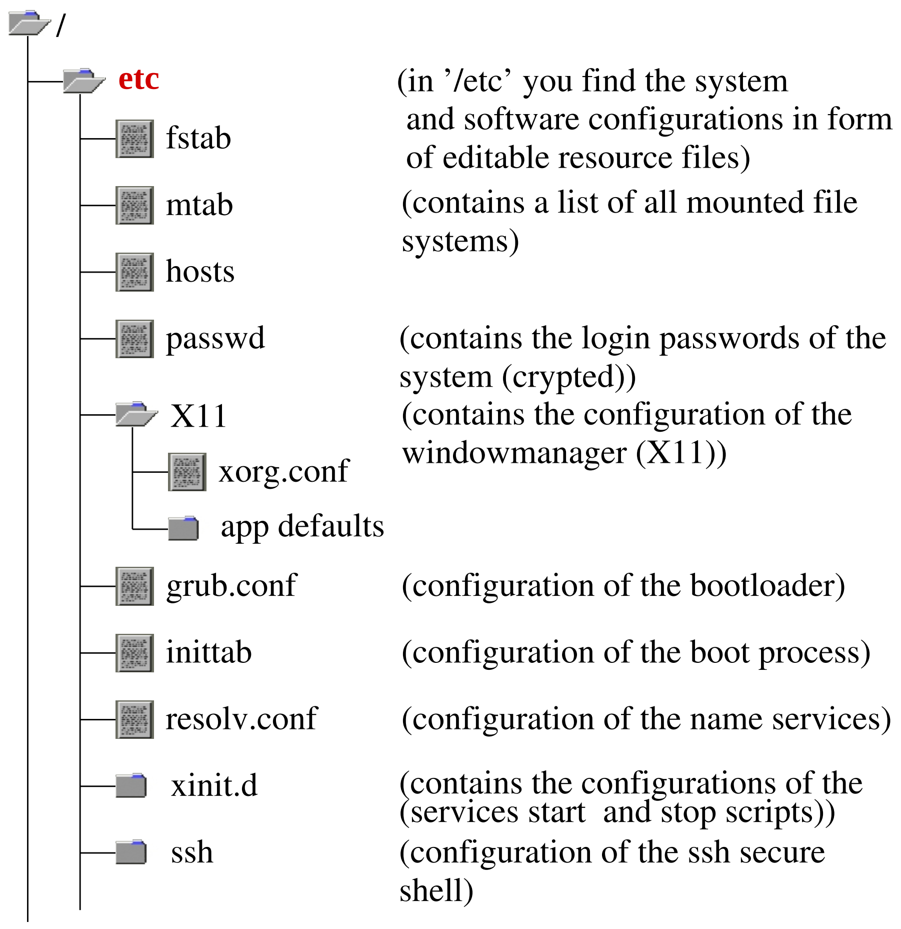

/
etc
(in '/etc' you find the system and software configurations in form of editable resource files)
fstab
mtab
(contains a list of all mounted file systems)
hosts
passwd
(contains the login passwords of the system (crypted))
X11
(contains the configuration of the windowmanager (X11))
xorg.conf
app defaults
grub.conf
(configuration of the bootloader)
inittab
(configuration of the boot process)
resolv.conf
(configuration of the name services)
xinit.d
(contains the configurations of the (services start and stop scripts))
ssh
(configuration of the ssh secure shell)

Figure 13: *The '/etc'-directory contains most of the configurations of the LINUX operating system.*

In order to change any of the global configurations of the LINUX operating system on the Commandline you will need Super-User privileges. For beginners this is not recommended! Better use one of the graphical setup tools of the Desktop if you want to change some settings on the system.

12 Console commands (most wanted)

This chapter will give you a short overview of the most important Console commands of the LINUX operating system. The list here is by far not complete! The LINUX commands are located at the folder '/bin' of the Filesystem. The Console command 'ls /bin' will show you the list of commands on your computer. Be aware that there might be additional commands also in other directories like '/sbin' or '/usr/bin'. Only a small fraction (the most important) of the actually installed commands will be discussed in this document. Most commands have a various number of options also. You can display a Man-Page for each command by typing 'man <commandname>' into the Commandline of the Console. Usually the commands in '/bin', '/sbin' and '/usr/bin' are all in the search PATH of the Console, so you can type the command names without any path directly into the Commandline for execution.

```
[It is] best to confuse only
  one issue at a time.  –
    Kernighan & Ritchie
```

12.1 ls

The **'ls'**-command ('list') gives you a sorted list of all files and folders in the current directory the Command-Prompt is pointing at. The kind of sorting and formatting of the output of 'ls' can be changed by various command options:

```
ls        : list           : list you the names of all files and folders
ls -l     : list long      : list you also the File-Permissions, the file size and the
                             creation date
ls -al    : list all long  : lists also hidden files and folders their names start with
                             a '.'
ls -ils   : list inodes    : lists also the 'inodes' (hardlinks) of particular files
ls -altr  : list all long timesort reverse : list is sorted by creation date, last one
                                             will be the newest
```

If you type 'ls -l' to the Commandline you will get an output like this:

```
total 8128
drwxr-xr-x. 2 schanz users    4096 Mar 30  2017 Desktop
drwxr-xr-x. 2 schanz users    4096 May 24 10:01 Docs
drwxr-xr-x. 5 schanz users    4096 Aug  3 17:56 Downloads
drwx------  4 schanz users    4096 Aug 23 16:09 Dropbox
drwxr-xr-x  2 schanz users    4096 Sep 15  2017 Mount
-rw-r--r--  1 schanz users  590946 Aug 26 10:54 Screenshot from 2019-08-26 10-54-29.png
-rw-r--r--  1 schanz users 7683180 Aug 26 11:03 cat.jpg
lrwxrwxr-x  3 schanz users    4096 Jul 15 14:53 gurk -> scan
drwx------  2 schanz users    4096 May  1  2016 mail
drwxr-xr-x  2 schanz users    4096 Aug 27 12:35 scan
drwxr-xr-x  2 schanz users    4096 Jul 15 14:53 tmp
......
|    |      |    |     |      |    |       |     |
| rights  links owner group bytes date    time  name
| (Access
|  Flags)
|
d = directory -> tmp is a folder, all others are files
l = link -> gurk is a soft link, pointing to the directory 'tmp', 'ls tmp' will list
            the content of 'tmp', also 'ls gurk'
```

The list above shows you the content of a typically user-HOME-directory. It contains some files the User has generated and a number of subdirectories that belong to the login. The call of 'ls' with the option '-l' shows you also the Access-Flags (File Permissions) of the files and directories, the 'Owner', the 'Group' it belongs to, the 'filesize' it is occupying in Bytes, the 'creation date' and the 'name'.

12.2 pwd

The **'pwd'**-command ('print working directory') shows you the working directory the Command-Prompt of the Shell is pointing at. The input of:

```
[~]> pwd
```

will produce an output like this:

```
/home/schanz
```

the Command-Prompt is pointing here to the 'root'-directory of the User 'schanz', his HOME-directory that is located at: '/home/schanz'.

If you are ever confused where in the Filesystem you are (the Command-Prompt), just type in 'pwd' to the Commandline.

```
  Computers don't actually
think.  You just think they
    think.  (We think.)
```

12.3 cd

The **'cd'**-command ('change directory') will change the directory where the Command-Prompt is pointing at. In a way you can move the Command-Prompt with 'cd' through the hierarchy levels of the Filesystem. You just give the 'cd'-command the new directory you want to change to as an argument. A few examples:

```
cd /home/schanz  : change the Command-Prompt into the HOME-directory of User 'schanz'
cd /             : change the Command-Prompt into the 'root'-directory of the
                   Filesystem
cd /usr/bin      : change the Command-Prompt into the '/usr/bin' directory
cd               : change the Command-Prompt into your own HOME-directory
cd .             : change the Command-Prompt into the current directory (no change)
cd ..            : change the Command-Prompt into the next upper directory
cd -             : change the Command-Prompt into the directory it was before the
                   last call of 'cd'
```

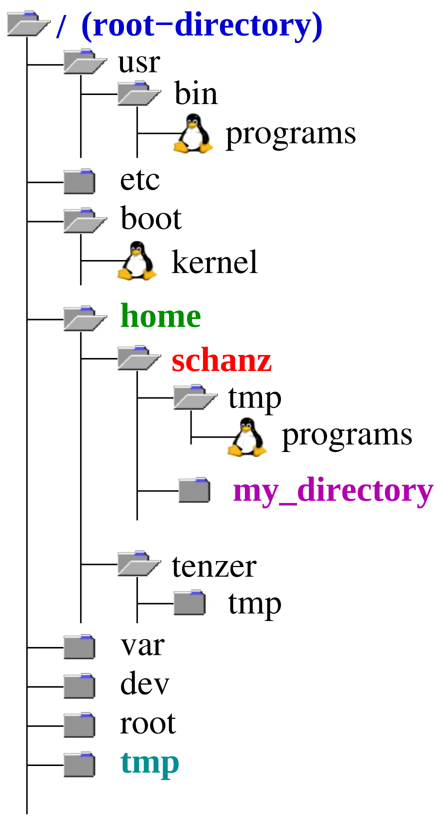


Figure 14: *Example for the Filesystem navigation utilizing the 'cd'-command: Considering you are User 'schanz' and your user-HOME-directory is '/home/schanz'. The call of the 'cd'-command alone without anything else will bring you (meaning the Command-Prompt of the Console) always back into your own user-HOME-directory '**/home/schanz**'. If you call 'cd ..' you will move one directory level up, in this case to '**/home**', if you repeat 'cd ..' again you will move up again to '**/**', this is the '**root-directory**' of the Filesystem. You can also use 'absolute' addressing like 'cd /home/schanz/my_directory'. This will bring you directly to '**my_directory**' anyway from which relative position you call it. If you call 'cd /tmp' next, you will go to '**/tmp**'. The call of 'cd -' will bring you back into the directory you were before the last 'cd', in this case '**my_directory**'*

12.4 cp

The **'cp'**-command ('copy') will copy files or directories of the Filesystem from one location to another:

```
cp gurk.txt wurg.txt      : makes a copy of the file 'gurk.txt' into the file named
                            'wurg.txt'.
cp gurk.txt /home/schanz/ : makes a copy of the file 'gurk.txt' into the directory
                            '/home/schanz' that will be named 'gurk.txt' and located
                            at '/home/schanz/gurk.txt'
cp gurk.txt gurk.txt      : makes a copy of the file 'gurk.txt' into the file
                            'gurk.txt'. This command call will fail because there can
                            not be two files with the same name in the same directory!
cp gurk.txt /             : makes a copy of the file 'gurk.txt' into the directory '/'
                            that will be named 'gurk.txt'. This command call will most
                            probably fail too, because normal Users have no write
                            permission to the 'root'-directory '/'! It will reply:
                            'cp: cannot create regular file 'gurk.txt': Permission
                            denied'
cp -r tmp neu_tmp         : makes a copy of the complete directory 'tmp' named
```

```
                              'neu_tmp'. The complete directory 'tmp' including all
                              files inside 'tmp' and all subdirectories inside 'tmp'
                              will me copied. The option '-r' means 'recursive'.
```

The 'copy'-command will generate a further instance of the file or directory, the old one stays in place.

12.5 mv

The command **'mv'** ('move') will move a file or a directory through the hierarchy levels of the Filesystem or rename it:

```
mv gurk.txt wurg.txt           : moves the file 'gurk.txt' into the file 'wurg.txt'.
                                 This is basically just a 'rename' of the file name!
mv schanz tenzer               : assuming 'schanz' is a directory, the renaming works
                                 also for directories, here into name 'tenzer'.
mv /tmp/gurk.txt /home/schanz/ : moves the file 'gurk.txt' from directory '/tmp' into
                                 the directory '/home/schanz'. The name of the file
                                 will be kept, the location inside the Filesystem is
                                 changed.
mv /home/schanz/gurk /tmp/     : assuming 'gurk' is a directory inside the
                                 user-HOME-directory of User 'schanz', the command will
                                 move the complete directory 'gurk' including everything
                                 inside to the directory '/tmp' of the Filesystem. After
                                 completing there will be a directory '/tmp/gurk' and no
                                 more '/home/schanz/gurk'!
```

The 'mv'-command will shift the file or directory to another location of the Filesystem. There will always be only one single instance.

12.6 rm

The **'rm'**-command ('remove') deletes files or directories from the Filesystem. You should handle 'rm' with care! There is no 'undo' for 'rm'. If you once have deleted something it is gone for ever!

```
rm gurk.txt              : deletes the file 'gurk.txt' from the current directory
rm gurk.txt wurg.txt     : deletes the files 'gurk.txt' and 'wurg.txt' from the current
                           directory
rm -r tmp                : deletes the directory 'tmp' recursively (all files and all
                           subdirectories in 'tmp') from the Filesystem.
rm *                     : deletes all files from the current directory of the Filesystem.
                           You should avoid this for safety reasons! If you are mistakenly
                           not in the directory you think, you will delete everything
                           somewhere else! At least you should check with the 'pwd'
                           command where your Command-Prompt actually is before you
                           execute an 'rm *'.
                           If you want to delete all files from a particular directory,
                           its saver to call the 'rm'-command stating the directory name
                           explicitly as an absolute path, e.g. 'rm -r /home/schanz/tmp'
                           or 'rm /home/schanz/tmp/*'
rm /home/schanz/tmp/*    : deletes all files in the directory of '/home/schanz/tmp/'
```

12.7 echo

The **'echo'**-command prints a string of letters or the content of an variable:

```
echo "Hello World"        : prints the string "Hello World" on the Console.
                            'echo' is usually used in scripts to print text onto the
                            Console
echo $PATH                : prints the content of the Environment-Variable PATH.
```

The 'echo'-command is mostly used in Scripts to produce a text output to the Console.

12.8 cat

The **’cat’**-command (’concatenate’) prints the content of a file onto ’stdio’ (standard I/O) of the Console. Example:

```
[~]> cat .bashrc
```

prints the content of the file ’.bashrc’ onto the Console.

```
An elephant is a mouse with
    an operating system
```

12.9 grep

The **’grep’**-command can be used to search for a particular string in a readable ASCII coded file. Assuming you want to display all files in your HOME-directory that contain your login name as a string (e.g. ’schanz’):

```
[~]> grep schanz *
```

The star ’*’ causes grep to look into every file of the current directory for the string ’schanz’. The output of ’grep’ will give you the filename of any file that contains the string together with the line in the file that contains the string to the Console.

If you want to know if a certain user account (e.g. ’root’) exists on your system you can grep the passwd file in the ’/etc’ directory of the system:

```
[~]> grep root /etc/passwd
```

Is there an entry for ’root’ inside the file ’/etc/passwd’, it will produce an output like this to the Console:

```
root:x:0:0:root:/root/:/bin/bash
```

or on more recent LINUX-Systems that don’t use a ’root’-login anymore for security reasons something more like this:

```
operator:x.11:0:operator:/root:/sbin/nologin
```

Try the last call of ’grep’ with your own login name if you like.

In general the ’grep’-command is used to search for specific word in a file or in a number of files.

12.10 more

The **’more’**-command displays the content of a file onto ’stdio’ (standard I/O) of the Console on a page basis. In contrast to the ’cat’-command the Console is not scrolling automatically when the output is larger than one single page. You can continue the output of ’more’ by the ’Enter’- or by the ’Space’-key of the keyboard:

```
[~]> more .bashrc
```

will display the content of the file ’.bashrc’ to ’stdio’ on a page basis.

In general ’more’ is a paging program to display extensive text outputs to the Console that provides some means of navigation.

12.11 less

The **’less’**-command displays the content of a file onto ’stdio’ (standard I/O) of the Console on a page basis. In contrast to the ’cat’-command the Console is not scrolling automatically when the output is more than one single page. You can continue the output of ’less’ by the ’Enter’- or by the ’Space’-key of the keyboard:

```
[~]> less .bashrc
```

will display the content of the file ’.bashrc’ to ’stdio’ on a page basis.

In general ’more’ is a paging program to display extensive text outputs to the Console that provides some

12.12 tail

The **’tail’**-command displays the last n-lines of a file. This can be very useful! Many files and log-files are very long and their output by ’cat’ or ’more’ or ’less’ is cumbersome because these commands always open the complete file. In particular logfiles are often only interesting in their last few entries. Using ’tail’ you can limit the output to the few interesting ’n’-lines at the end of the file. You can display the last 10 lines of the system logfile for instance:

```
[~]> sudo tail -10 /var/log/messages
```

The command above will display you the last 10 entries of the KERNEL-log-file ’/var/log/messages’. The KERNEL-log-file will grow with any new system call but only the last entry is of most interest and tells you if the last system call was successful. You will need here the command ’sudo’ also, because ordinary Users usually will not have any access to the KERNEL-log-file of the system.

```
[~]> sudo tail -f /var/log/messages
```

You can instruct ’tail’ to give you a running update of the logfile ’/var/log/messages’. Using the option ’-f’ tells ’tail’ to display only the last modified line of the logfile and keep it updated with any new line coming in. This is ideal to follow the running changes in a logfile.

12.13 touch

The **’touch’**-command will update and modify the timetag of a files or generate an empty file. This can be useful because some scripts test the occurrence of files or their last update date. The call of:

```
[~]> touch gurk.txt
```

updates the timetag of the file ’gurk.txt’. If ’gurk.txt’ does not exist, the file will be generated anew without any content (file size zero).

The ’touch’-command just touches the file, it does not modify anything besides the time tag.

```
Do you guys know what you're
   doing, or are you just
          hacking?
```

12.14 mkdir

The **’mkdir’**-command (’make directory’) will generate a new folder. For instance you can put the Command-Prompt in your HOME-directory (just type in ’cd’ to the Console will do the job) and generate there a new subdirectory named ’myfolder’:

```
[~]> mkdir myfolder
```

This will generate a directory with the name ’myfolder’ in your user-HOME-directory. Considering you are User ’schanz’ this will produce the directory ’/home/schanz/myfolder’.

You also can generate a complete hierarchy of directories and subdirectories with one single call of ’mkdir’ by using the option ’-p’:

```
[~]> mkdir -p folder1/folder2/folder3
```

Considering again that you are User ’schanz’ and you issue the command in the ’myfolder’ directory that was generated at the command before, you will generate the following hierarchy of directories: ’/home/schanz/myfolder/folder1/folder2/folder3’. If you issue the command just in the HOME-directory instead it will generate the folders: ’/home/schanz/folder1/folder2/folder3’. The position of the Command-Prompt determines where the directory will be generated.

12.15 rmdir

The **’rmdir’**-command (’remove directory’) will delete a directory from the Filesystem. Considering you are in your HOME-directory and want to remove the directory ’myfolder’, than you can call the command:

```
[~]> rmdir myfolder
```

You can also call the command with an absolute path in order to avoid any ambiguity. This will work from any position of the Command-Prompt it will have in the Filesystem:

```
[~]> rmdir /home/schanz/myfolder
```

Instead of ’rmdir’ you can also call the command ’rm’. The directory must be empty in order to remove it from the Filesystem. If you want to delete a directory including its content you have to issue the ’rm’-command with the option ’-r’ (recursive):

```
[~]> rm -rf /home/schanz/myfolder
```

This will remove everything inside the folder ’/home/schanz/myfolder/’ and at the end the folder ’myfolder’ itself.

12.16 find

Using the **'find'**-command you can search the Filesystem of the computer for a articular name. The command can be used in plenty of varieties. The normal 'find'-command will be called as follows:

```
[~]> find <where> <mode> <what> <output>
```

'Where' in the Filesystem will be searched, what kind of search **'mode'** will be performed (by name, by timetag ..), **'what'** will be searched (what namestring) and where will the **'output'** be directed to. A few examples:

```
find . -name gurk -print  : searches inside and down below the current directory '.' the
                            Command-Prompt is pointing at. It will search a file with the
                            'name' that is 'gurk'. The output will be directed to 'print'
                            that means into the Console (default).

                            The command will actually search the name 'gurk' in the list
                            of filenames in the current directory and all its
                            subdirectories and give the result as a list to the Console.
find /usr -name gurk      : will search in and below the directory '/usr' for the name
                            'gurk'
find /usr -name '*gurk*'  : will search in and below the directory '/usr' for a string'
                            that contains the name 'gurk' in some combination.
find .                    : will produce a list of all files and folders in the current
                            '.' directory and all its subdirectories.
find . -type f            : will only search for files and no directories. Will give a
                            list of filenames in and below the current directory.
find . -type d            : will only search for folders and not for files. Will give a
                            list of dirnames in and below the current directory.
find . -type d -name a*   : will find all directories that start with the letter 'a'
                            in and below the current directory.
find / -name "*.txt" -size +12000c : will find all files in and below '/' there name is
                                     ending with '.txt' and they are bigger than 12000
                                     Bytes.
find / -name core -exec rm -f '{}' \; : will find all files with the name 'core' in and
                                        below '/' (the complete Filesystem). The output
                                        list will be send to the command 'rm'. This
                                        means all files with the name 'core' will be
                                        searched and deleted in the complete Filesystem!
```

The way to combine the 'find' command with other command and to use it is virtually endless. There can be approximately 40 different search criteria applied to the 'find' command. Please look up the Man-Page of 'find' for further information.

```
If it's not in the computer,
      it doesn't exist
```

12.17 ps

The **'ps'**-command ('process status') will display a list of processes currently started on the Console:

```
[~]> ps
```

The list contains the Process-IDs (PID), the Terminal (TTY) from where the particular process was started, the consumed CPU processing time (TIME) and the name of the process (CMD):

```
  PID TTY          TIME CMD
 5684 pts/4    00:00:00 bash
 5709 pts/4    00:00:00 ps
```

If you call 'ps' with the option '-fe':

```
[~]> ps -fe
```

you will get a list like already shown in chapter 7. The list contains all currently started processes on the whole system.

12.18 kill

Using the **'kill'**-command you can force a process to quit before its normal execution will end. There are several priorities for 'kill', see also the Man-Page of 'kill': → 'man kill':

```
kill <PID>     : PID is the process number you want to kill. The PID you can find
                 by using the 'ps'-command. 'kill' will ask the program to quit.
                 In this priority the program will be granted enough time for a
                 ordered shutdown and to flush out data to disk if necessary.
kill -9 <PID>  : The priority '-9' will force the process to quit immediately.
                 There will be no time for the program to save data, data loss can
                 be the result! You normally will use this priority to kill a
                 software that has been crashed already, hence data loss will be
                 involved anyway.
```

The 'kill'-command is generally used to kill processes that have hang up or generate an abnormal high amount of CPU load.

Computer programmers never
die, they just get lost in
the processing

12.19 chown

The **'chown'**-command ('change owner') will change the owner of a file or directory.

Each file and directory of the Filesystem has his 'owner' which is normally one of the Users of the system -> see also chapter 13 File-Permissions. The commands 'chown', 'chgrp' and 'chmod' allows you to set and modify the Access-Flags that control the File-Permissions to a particular file or directory. This determines who else on the computer has the right to 'read', 'write' or 'execute' the file or directory. For instance:

```
[~]> chown schanz gurk.txt
```

will change the 'owner' of the file 'gurk.txt' to the owner 'schanz'.

12.20 chgrp

The **’chgrp’**-command (’change group’) will change the group a file or directory belongs to.

Each file and directory of the Filesystem belongs to a specific ’group’ which may include a number of users -> see also chapter 13 File-Permissions. The commands ’chown’, ’chgrp’ and ’chmod’ allows you to set and modify the Access-Flags that control the File-Permissions to a particular file or directory. This determines who else on the computer has the right to ’read’, ’write’ or ’execute’ the file or directory. For instance:

```
[~]> chgrp users gurk.txt
```

will change the ’group’ belongings of the file ’gurk.txt’ to the group ’users’.

12.21 chmod

The **’chmod’**-command (’change mode’) will change the access mode of a file or directory. Each file and directory of the Filesystem is using a number of Access-Flags to control the access by other Users of the system -> see also chapter 13 File-Permissions. The flags determine who else on the computer has the right to ’read’, ’write’ or ’execute’ a file or directory. For instance:

```
[~]> chmod 744 gurk.txt
```

will change the flags for the access of the file ’gurk.txt’ to the new mode ’744’ -> see also chapter 13 File-Permissions for more details.

12.22 uname

The **’uname’**-command displays information about the release version of the LINUX operating system:

```
[~]> uname -a
```

will produce a Console output like:

```
Linux atlas.local.de 4.8.13-100.fc23.x86_64 #1 SMP Fri Dec 9 14:51:40 UTC 2016 x86_64
x86_64 x86_64 GNU/Linux
```

This will tell you that the system has the computer name ’atlas’ with the domain ’local.de’, the Kernel has the release number ’4.8.13-100’, is a ’Fedora 23’ operating system for a ’x86’ CPU architecture that is ’64’ bit wide. The system will support ’SMP’ (symmetric multi processing) and was compiled on ’Fri Dec 9 14:51:40 UTC 2016’.

12.23 man

The **’man’**-command will display the Man-Page for a specified command. Each command that is installed on the system has a user manual that describes how the command will be operated and what options the command has to offer. It is useful to make yourself acquainted with the commands on your system by reading this manuals. There are far more commands and options on the system than this document here can present you!

```
[~]> man man
```

will produce the following output to the Console:

```
MAN(1)                        Manual pager utils                        MAN(1)

NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man  [-c|-w|-tZ] [-H[browser]] [-T[device]] [-X[dpi]] [-adhu7V] [-i|-I]
       [-m system[,...]] [-L locale] [-p  string]  [-C  file]  [-M  path]  [-P
       pager]  [-r  prompt]  [-S  list] [-e extension] [--warnings [warnings]]
       [[section] page ...] ...
       man -l [-7] [-tZ] [-H[browser]] [-T[device]] [-X[dpi]] [-p string]  [-P
       pager] [-r prompt] [--warnings[warnings]] file ...
       man -k [apropos options] regexp ...
       man -f [whatis options] page ...

DESCRIPTION
       man  is  the  system's manual pager. Each page argument given to man is
       normally the name of a program, utility or function.  The  manual  page
       associated  with each of these arguments is then found and displayed. A
       section, if provided, will direct man to look only in that  section  of
       the  manual.   The  Default action is to search in all of the available
       .....
```

Using the option '-k' you can search among all Man-Pages on the system for a special search string. This can be very useful when you don't know exactly where a special topic is discussed and what commands are involved. For instance if you want to know which commands and Man-Pages are concerned with 'tapes' on your system you can type in:

```
[~]> man -k tape
```

This will produce an output like below, containing all Man-Pages that have something to do with the 'tape' subject.

```
amanda-taperscan (7) - Amanda Taperscan Algorithms
amcheckdb (8)        - check Amanda database for tape consistency
amfetchdump (8)      - extract backup images from multiple Amanda tapes.
amflush (8)          - flush Amanda backup files from holding disk to tape
amlabel (8)          - label an Amanda tape
amrmtape (8)         - remove a tape from the Amanda database
amtape (8)           - Control Amanda changers
amtapetype (8)       - generate a tapetype definition by testing the device directly
smbtar (1)           - Shell script for backing up SMB/CIFS shares directly to UNIX tape
                       drives
st (4)               - SCSI tape device
tapelist (5)         - The list of Amanda volumes in use
tar (5)              - format of tape archive files
```

We are drowning in
information but starved for
knowledge. - John Naisbitt,
Megatrends

12.24 df

The **’df’**-command (’disk free’) will show you the current usage of the Filesystem and how much space is still left on the diskdrive:

```
[~]> df
```

will show you an output like:

```
Filesystem              1K-blocks       Used Available Use% Mounted on
/dev/sda5                80431944   46051704  30294480  61% /
tmpfs                     1009936          0   1009936   0% /lib/init/rw
udev                        10240        884      9356   9% /dev
tmpfs                     1009936         12   1009924   1% /dev/shm
/dev/sda3                  198337      34907    153190  19% /boot
192.168.0.1:/mnt/dsk2/  3845710848 3476797440 173562880  96% /mnt/dsk2
```

The list will show you the attached Filesystems, the currently amount of space that is already occupied and the space that is still free and also the corresponding hardware devices and mount points. In the example shown above the ’root’ directory of the Filesystem is stored on the device ’/dev/sda5’ and mounted to ’/’. Its usage is at 61%. The device ’/dev/sda3’ hosts the boot partition and the KERNEL with a usage of 19% and the mount point ’/boot’. The last entry contains a network Filesystem that is mounted from another computer (Server - 192.168.0.1) to the directory ’/mnt/dsk2’. It has a usage of 96%, however the disk has a size of 3845710848 kByte that is approximately 4 TByte of disk space. The network Filesystem is linked into the local Filesystem and can be accessed exactly the same way as the Filesystems that are hosted on the local diskdrives.

12.25 top

The **’top’**-command will give you a list of processes that have the highest CPU usage on the system. When you type:

```
[~]> top
```

into the Console it will produce an output like:

```
top - 14:35:23 up  6:08,  5 users,  load average: 0.12, 0.10, 0.18
Tasks: 223 total,   2 running, 221 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.3 us,  0.8 sy,  0.0 ni, 95.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1991500 total,   724244 free,   223524 used,  1043732 buff/cache
KiB Swap:  3905532 total,  3905532 free,        0 used.  1492352 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2334 schanz    20   0  588020  57776  42100 S   2.3  2.9   0:02.27 konsole
 2165 schanz     9 -11  454840  10632   9164 S   1.3  0.5   8:39.30 pulseaudio
 1688 schanz    20   0  304372  24296  15488 S   0.7  1.2   4:29.11 Xorg
 2370 schanz    20   0   55056   4216   3460 R   0.7  0.2   0:00.21 top
18120 schanz    20   0  587468  19064  15628 S   0.7  1.0   0:22.17 konsole
   21 root      20   0       0      0      0 S   0.3  0.0   0:02.36 rcuos/1
 8789 schanz    20   0  454800   2968   2864 S   0.3  0.1   0:01.74 sd_espeak
    1 root      20   0  128256   4676   3360 S   0.0  0.2   0:04.40 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.07 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   0:05.64 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:04.15 rcuos/0
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.04 migration/0
   12 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   13 root      rt   0       0      0      0 S   0.0  0.0   0:00.04 watchdog/0
```

From the list above you can learn that the current CPU usage of the system is at 0.12 (load average: 0.12 : 12%), there are 223 processes launched on the system, two process are running, 221 sleeping. The machine has 2 GByte of memory (1991500 total) from which 224 MByte (223524 used) are in use and 1.04 GByte allocated for cache. Approximately 700 MByte (724244) are still ’free’ for usage. So far the machine does not make any use of swap memory (0 used). The list below gives you the momentary rank of processes in terms of CPU load. The issues of ’top’ are only running snapshots of the system and will change from second to second. For further information, see also the Man-Page of ’top’, type ’man top’.

```
The goal of Computer Science
 is to build something that
  will last at least until
 we’ve finished building it.
```

12.26 reboot

The **’reboot’**-command is used to restart the computer from scratch. The command will stop all running programs immediately, logout all Users and reboot the system. You should never reboot a system if other Users are logged in to the system. The ’reboot’ command requires Super-User privileges and can not be launched by normal Users, however standalone systems like laptops are often configured to permit also normal Users to ’reboot’ the system. See also chapter 6 for further information.

12.27 shutdown

The **’shutdown’**-command is used to shutdown the computer and switch off power. The command will stop all running programs, logout all Users and shutdown and power off the system. You should never shutdown a system if other Users are logged in to the system. The ’shutdown’ command requires Super-User privileges and can not be launched by normal Users, however standalone systems like laptops are often configured to permit also normal Users to ’shutdown’ the system. See also chapter 6 for further information.

12.28 which

Using the **’which’**-command you can find out the location of a command or an application in the search PATH of the Commandline:

```
[~]> which reboot
```

will show you the location of the ’reboot’-command:

```
/usr/bin/reboot
```

The ’reboot’-command is located at ’/usr/bin’. You can find out from where a command is executed. This can be important when you are using a local software version of a particular command that has the same name as the system installed command. Using ’which’ you can find out which of the installed versions is actually launched by the Commandline of the Console.

12.29 ln

The **’ln’**-command (’link’) is used to link different names to a file or directory. This can be very useful if you want to offer a certain file or directory for different Users by different names without producing several instances. For example: You are the Admin of the system and want to offer news-magazines on your system to different Users. The news-magazines are pdf-files in the directory ’/dsk2/magazines’. Using ’ln’ you can offer the files to different Users in a very easy way, you just link the files into the Users corresponding HOME-directories:

```
[~]> ln -s /dsk2/magazines /home/schanz/magazines
[~]> ln -s /dsk2/magazines /home/tenzer/magazines
```

This will map the folder ’magazines’ from ’/dsk2/magazines’ to ’/home/schanz/magazines’ and ’/home/tenzer/magazines’. The folders ’/home/schanz/magazines’ and ’/home/tenzer/magazines’ are actually only pointers to ’/dsk2/magazines’. This linking avoids the duplications of the folder ’/dsk2/magazines’ and the files it is containing. When User ’schanz’ is calling the ’ls -l’ command in his HOME-directory the link to the ’magazines’ folder is displayed as:

```
lrwxrwxr-x 1 root root       21 2014-09-07 19:22 magazines -> /dsk2/magazines
```

Simply by calling the command:

```
[~]> cd magazines
```

in his HOME-directory User ’schanz’ can change into the ’magazines’-folder and will have immediate access to the magazine pdf-files. This will be identical for the User ’tenzer’ also. All the linked folders are giving access to the same files without wasting disk space by multiple instances.

12.30 tar

The **’tar’**-command (’tape archive reader’) will archive a number of files or directories into one single archive. Originally ’tar’ was used to archive or disarchive files and directory via tapedrive onto a magnetic tape. Since magnetic tapes are obsolete for home usage today, ’tar’ is nowadays used to archive great number of files and folders into one single archive-file. This can be very useful for instance to encapsulate a number of files that belong to a certain project into one single file (tar-archive) that can be used for storage or for easy transportation. The tar-archive can further be compressed with the ’gzip’- or ’bzip2’-command for efficient file transfer also. Tar-archives are very common under LINUX as containers for file-projects, picture-collections, or for backup purposes. In principle you can archive a complete Filesystem that contains some 10 thousand files and folders into one single tar-archive-file. The complete Filesystem hierarchy will stay intact inside the archive. After expanding the archive, the original hierarchy of files and folders will be reproduced exactly and reinstalled again. The ’tar’-command is used to archive and to disarchive tar-archive-files as well. Consider you want to archive your local ’tmp’-directory in your user-HOME-directory you can call the command as follows:

```
[~]> tar cvf tmp.tar tmp
```

In this example the 'tmp'-directory including all its files and subfolders will be archived into the single tar-archive-file 'tmp.tar'. The options 'cvf' means 'create', 'verbose', 'file'. It will 'create' an archive-'file' (no tape) and display ('verbose') all files and folders that are archived on the Console during packing.

After archiving, you can easily put the tar-archive-file onto an USB-Stick or send it somewhere via email attachment. The receiver can disarchive (expand) the tar-archive-file again simply by calling the 'tar' command with a different options:

```
[~]> tar xvf tmp.tar
```

The option 'x' means 'expand'. The command call above will expand the tar-archive-file ('tmp.tar') into the original folder 'tmp' containing all files and subfolders in the original hierarchy structure as it was before.

The question of whether computers can think is just like the question of whether submarines can swim. – Edsger W. Dijkstra

12.31 gzip

The **'gzip'**-command can compress files of any kind, it is mostly used to compress and decompress tar-archive-files. The compression by 'gzip' is without any loss and reversible, it is just shrinking the filesize and spares a lot of diskspace. The 'gzip'-command is used for compression and decompression as well, depending on the options you are using:

```
[~]> gzip -9 tmp.tar
```

In this case the file 'tmp.tar' will be compresses into the file 'tmp.tar.gz' which is often also just named as 'tmp.tgz'. The file extension '.tgz' tells you that it is a compressed tar-archive-file. The 'tmp.tgz' file can be largely reduced in files size compared to the original uncompressed file 'tmp.tar'. The amount of compression depends on the internal setup of the file, in this case on the contents of the tar-archive.

The decompression of a compressed '.gz'-file will be executed by calling the '-d' option of the 'gzip'-command:

```
[~]> gzip -d tmp.tar.gz
```

This will decompress the file 'tmp.tar.gz' into the original file 'tmp.tar' again.

12.32 bzip2

The **’bzip2’**-command can compress files of any kind, it is mostly used to compress and decompress tar-archive-files. The compression by ’bzip2’ is without any loss and reversible, it is just shrinking the filesize and spares a lot of diskspace. The ’bzip2’-command is used for compression and decompression as well, depending on the options you are using. The compression rate of ’bzip2’ is higher than that of the ’gzip’ compressor:

```
[~]> bzip2 -9 tmp.tar
```

In this case the file ’tmp.tar’ will be compresses into the file ’tmp.tar.bz2’ which is often also just named as ’tmp.tbz’. The file extension ’.tbz’ tells you that it is a compressed tar-archive-file. The ’tmp.tbz’ file can be largely reduced in files size compared to the original uncompressed file ’tmp.tar’. The amount of compression depends on the internal setup of the file, in this case on the contents of the tar-archive.

The decompression of a compressed ’.bz2’-file will be executed by calling the ’-d’ option of the ’bzip2’-command:

```
[~]> bzip2 -d tmp.tar.bz2
```

This will decompress the file ’tmp.tar.bz2’ into the original file ’tmp.tar’ again.

12.33 date

The **’date’**-command is used to display or modify the current system time. If you have Super-User permissions you can use ’date’ also to set the system time. If you call ’date’ you will get just an output like this:

```
Wed Aug 28 17:02:01 CEST 2019
```

You can also use the command **’cal’** that will plot you a small calendar into the Console.

12.34 who, w, finger

The **’who’**-command will show you who else is working on the system. The command gives you a list of all logged in users and tells you how (from where) these Users are logged in to the system. The call of ’who -a’ will produce an output like:

```
           system boot  2014-07-08 08:51
           run-level 5  2014-07-08 08:51
LOGIN      tty3         2014-07-08 08:51              1855 id=3
LOGIN      tty2         2014-07-08 08:51              1853 id=2
LOGIN      tty6         2014-07-08 08:51              1861 id=6
LOGIN      tty4         2014-07-08 08:51              1857 id=4
LOGIN      tty5         2014-07-08 08:51              1859 id=5
a107     - tty7         2014-07-08 08:52  old         2046 (:0)
a107     + pts/0        2014-07-09 17:22  old         8389 (:0.0)
root     + pts/1        2014-09-09 10:35   .          5482 (triton.local.de)
schanz   + pts/2        2014-09-09 10:37   .          5514 (phobos.local.de)
           pts/3        2014-08-11 09:18             28188 id=ts/3  term=0 exit=0
           pts/4        2014-08-12 11:20              3678 id=ts/4  term=0 exit=0
```

This list shows you the 'login-name', the 'terminal emulation', the 'time of login' and the 'location' from where the User has logged in.

The command **'w'** and the command **'finger'** will produce comparable outputs.

12.35 sudo

The **'sudo'**-command ('substitute user do') allows you to launch commands or programs with the identity of another User, considering you know the password of this other User.

If you don't give a specific user name the command will give you Super-User privileges temporary if you are listed in the 'sudo-configuration-file' - ask your Admin. You will acquire the other privileges only temporary for the single command you call by the 'sudo'. Usually you call 'sudo' together with the command you want to execute with Super-User privileges like: 'sudo <command>'. For instance if you want to mount a disk drive this could be done like:

```
[~]> sudo mount /dev/sdb1 /mnt/dsk3
```

The command above will mount a disk partition under the device name '/dev/sdb1' into the directory '/mnt/dsk3' of the Filesystem.

As Super-User you can free specified Users from the password request when you add them to the sudoers list under '/etc/sudoers'. Be very careful to open Super-User permissions to normal Users, they may have a lack of experience or responsibility to use it properly!

12.36 su

The **'su'**-command ('substitute user') will give you permanently the privileges of another User or of the Super-User assuming you know the password of the particular User or the Super-User. If you call the command 'su' into the Commandline the Console becomes permanently a Super-User-Console. Some recent LINUX distributions don't allow this mechanism anymore for security reasons, they only permit the call of commands with Super-User privileges via the 'sudo' mechanism. Of course you can modify any system to allow the 'su' again if you know how to do it.
Assuming you are User 'schanz', you can become the User 'tenzer' just by typing the 'su' command into the Console like:

```
[~]> su - tenzer
```

The system will ask you the password of User 'tenzer'. If you type in the correct password you will become the User 'tenzer' permanently in the current running Console. Your HOME-directory will change from '/home/schanz' to '/home/tenzer' and the resource files of the User 'tenzer' will be executed. You should be prepared that the User 'tenzer' may have configured his account differently than you have configured your own account (other Terminal, other Shell, other Editor ...). If you type the command **'exit'** you will drop out and become yourself (User 'schanz') again.

If you call the ’su’ command as ’su -’ without any user name you will acquire full Super-User privileges and become the ’root’-User of the system. The system will request the Super-User password! In this case your HOME-directory will change to ’/root’ and the resource files hosted under ’/root’ will be executed. Be prepared that the ’root’-User may have another Shell as the normal User and some things may work differently. As ’root’-User you can read, write and execute any file on the system including private files of any Users. Even if possible, the reading of private user files is an absolute NO GO. Its considered a violation of trust and will never be executed by any responsible administrator nor it will be tolerated at most companies or public organizations! You will getting in great trouble if you do it!
You should only stay ’root’-User as long it is required. When you call the command **’exit’** in the Console you will drop out from ’root’-User and become your normal User again.

```
The trouble with computers is
  that they do what you tell
 them, not what you want.  -
          D. Cohen
```

12.37 passwd

The **’passwd’**-command allows you to change your password. The system will ask you for your old password first and then for the new password. The system administrator can setup constraints for passwords and an expiring date when the password will become invalid.

12.38 env

The **’env’**-command (’environment’) will display all Environment-Variables of your login. When you call the ’env’-command in the Console it will give you a list like this:

```
MANPATH=/usr/man:/usr/contrib/X11R6/man:/usr/local/man:/net/usr/man:/opt/wabi/man:/opt
        /man:/usr/share/man
SSH_AGENT_PID=1817
HOSTNAME=atlas.local.de
XANBIN=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22
TERM=xterm-256color
SHELL=/bin/bash
HISTSIZE=500
PGPLOT_RGB=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/lib/rgb.txt
PGPLOT_FONT=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/lib/grfont.dat
HEADAS=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22
WINDOWID=54525957
FTOOLS=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22
LD_LIBRARY_PATH=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/lib
LHEASOFT=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22
USERNAME=ts
PAGER=/bin/more
PATH=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/bin:/home/ts/.dt/bin:/bin:/
```

```
     usr/bin:/sbin:/usr/sbin:/usr/bin/X11:/usr/local/bin:/usr/local/bin/X11:/usr/contr
     ib/bin:/usr/contrib/bin/X11:/opt/Xilinx/14.7/ISE_DS/ISE/bin/lin64:/usr/dt/bin:/op
     t/fv5.4:/usr/local/bin
BROWSER_PATH=/bin/firefox
EDITOR=/bin/vi
LANG=C
....
```

Many settings of the user-account are controlled by Environment-Variables. You can display the content of any Environment-Variable to the Console when you type in the command 'echo $<environment-variable>' to the Commandline.

There is is no reason for any individual to have a computer in their home. - Ken Olsen (President of Digital Equipment Corporation), Convention of the World Future Society, in Boston, 1977

12.39 ssh

The **'ssh'**-command ('Secure-Shell') allows you to login to a remote system that is attached to your computer via network or via the Internet. This will only be successful if the remote host has an ssh-daemon running and you have a valid user-account on the remote system. Considering your login on the remote system is 'schanz' and the name of the remote system is 'demokrit'. Then you can launch an 'ssh' session by typing:

```
[~]> ssh schanz@demokrit
```

The remote system, in this case 'demokrit' will reply with a password request:

```
schanz@demokrit's password:
```

At this point you have to type in your password you have on the remote system 'demokrit' (not the local system). If the login is successful the remote system will reply something like this:

```
Last login: Wed Mar 27 18:02:44 2013 from triton.local.de
demokrit:[~]>
```

Considering you are now logged in as User 'schanz' on the remote system 'demokrit'. You can use the remote system via Console the same way as you are using your local system. Most of the commands are common to most LINUX systems, they will also run on the remote system. This is however restricted to Console commands and applications. If you want to start a software that may want produce a graphical output you must initiate the ssh connection by using the option '-X' or '-Y' in the first place. For instance:

```
[~]> ssh -X schanz@demokrit
```

This will start an ’ssh’ session that directs any graphical output to the window-server of your local system. The windows will open on your local computer, the program generating the windows is running on the remote computer.
You can end the ’ssh’ session just by typing ’logout’ or ’exit’ to the remote Console. Of course you can also just close the remote Console window on your local computer.

If the name of the remote system is not known at your local system (e.g. ’demokrit’), you can also use the IP-address (here 192.168.0.215) of the remote system to initiate the ’ssh’-session:

```
[~]> ssh -X schanz@192.168.0.215
```

If you login at your IAAT-account at the University of Tuebingen via ’ssh’, then you should use the domain name ’astro.uni-tuebingen.de’. Considering you are User ’schanz’ this would look like this:

```
[~]> ssh -X schanz@astro.uni-tuebingen.de
```

the Secure-Shell (ssh) will initiate an encrypted connection to the remote host that is considered secure for data interchange.

12.40 scp

The **’scp’**-command (’Secure Copy’) works similar to the ’ssh’ command. The ’scp’ command allows you to copy files and other data from your local system to a remote system or from a remote system to your local system. If you want to copy files from your local system to the remote system you type something like:

```
[~]> scp /home/schanz/tmp/* schanz@demokrit:/home/schanz/
```

This would copy all data from your local directory ’/home/schanz/tmp/’ to the ’schanz’ HOME-directory on the remote system ’demokrit’. The remote system ’demokrit’ will request the password of the User ’schanz’ on ’demokrit’.

```
schanz@demokrit's password:
```

If you want not only copy files but also directories via the ’scp’-command, you must use the ’-r’ option (recursive) of the ’scp’ command. During the copy the ’scp’-command will keep the complete directory hierarchy of the copied directories intact, exactly like it is on the local system.

```
[~]> scp -r /home/schanz/tmp/* schanz@demokrit:/home/schanz/
```

During copy ’scp’ will issue a running list of all files and directory that are copied. The output may look somehow like this:

```
filesystem.fig                              100% 6940      6.8KB/s   00:01
LINUX_FirstContact.aux                      100% 2979      2.9KB/s   00:00
LINUX_Anleitung2014.log                     100%   24KB  24.4KB/s   00:00
.....
```

If you want copy files from the remote system to the local system you initiate the ’scp’-command like this:

```
[~]> scp -r schanz@demokrit:/home/schanz/tmp/* /home/schanz/
```

The command above copies all files and directories of the folder ’/home/schanz/tmp/’ of the remote system to the local directory ’/home/schanz/’ on your local system, keeping the folder structure intact.

12.41 script

The **’script’**-command can be very useful if you want a complete record of the commands that you typed into the Console during your login session. ’Script’ will log everything you type into the Console in the file ’typescript’ that is hosted in the directory where you have launched the ’script’ command in the first place. Just type into the Console:

```
[~]> script
```

A process is started that will record everything you type to the Console into the file ’typescript’ until you press **’ctrl d’** on the keyboard and quit this recording. You can open the file ’typescript’ in order to analyze your session in detail afterwards.

```
 LINUX was not designed to
stop you from doing stupid
things, because that would
 also stop you from doing
clever things.  - Doug Gwyn
```

13 File-Permissions

The File-Permissions control the access to files and directories on your LINUX system. There are three categories for which File-Permissions are defined for the files and directories of the Filesystem. These three categories are:

```
OWNER  GROUP  OTHER
```

The File-Permissions are controlled by ’Access-Flags’ and can be displayed by the command ’ls -l’ and manipulated by the commands **'chown'**, **'chgrp'** and **'chmod'**. The command ’ls -al’ will produce an extended output like:

```
dr-xr-xr-x   2   root         sys       1024   Nov 17 14:02     bin
dr-xr-xr-x   7   root         sys       1024   Nov 28 23:12     config
-rwxr-xr--   1   schanz       users      400   Dec 12 00:10     gurk
|   |        |     |            |         |         |              |
| rights   links  owner       group     Bytes      date            name
| Access
| Flags
|
Typ: d = directory; l = link; b/c = devices
```

Every file and directory of the system has an ’owner’ and belongs to a ’group’. The Access-Flags control the access to the files and directories by the ’owner’, by ’group’ members or by ’others’ that means anybody else on the system.

```
     d   r w x   r w x   r w x      <- Access-Flags
     |     |       |       |

    Typ   owner   group   others (rest)
```

The letters refer to: **’r’** (readable), **’w’** (writeable) and **’x’** (executable). Using the flags ’rwx’ you can specify for each of the three categories owner/group/others the rights you want to grant for the particular file or directory. For instance:

```
- rwx r-x r--  1  schanz  users  400  Dec 12 00:10  gurk
```

This would mean for the File-Permissions to the file ’gurk’ for instance:

```
owner: r w x :   the owner of the file, User 'schanz', can read, write and execute
                 the file
group: r - x :   members of the group 'Users' can read and execute the file, but can
                 not write the file, means they can not delete it also!
others:r - - :   all others can read the file but have no permission to write nor to
                 execute the file.
```

The Access-Flags can be modified by the command 'chmod'. The numerical argument of the ’chmod’-command encodes the new settings of the Access-Flags. The command:

```
[~]> chmod 744 gurk
```

will change the status of the Access-Flags of the file ’gurk’ into:

```
- rwx r-- r--  1  schanz  users  400  Dec 12 00:10  gurk
```

Why is that so? Why does the value ’744’ modify the Access-Flags into ’rwxr--r--’?

The numbers in the argument of the ’chmod’-command (here ’744’) are treated as hex-number and translated into a binary number representation. In this case the hexnumber ’744’ is converted into ’111 100 100’. This bitcode is combined with the Access-Flag-patter ’rwx rwx rwx’ and multiplied on a bit basis into the resulting Access-Flag-pattern ’rwx r– r–’. Some examples:

hexadecimal	binary	Access-Flags
0	000	---
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Launch a Console and change the Command-Prompt into the directory ’/home’:

```
[~]> cd /home
```

Call the command ’ls -l’, you will see something like this:

```
total 32
drwx------  51 a107   users  4096 Aug 29 11:09 a107
drwx------  33 cde    users  4096 Jul  9 17:09 cde
drwx------. 25 tenzer users  4096 May  2 18:28 tenzer
drwxrwx---   2 prakt  ait    4096 Aug 12  2010 prakt
drwxr-xr-x   2 root   root   4096 Aug 30  2012 printer
drwxr-xr--  25 schanz users 12288 Jun 26 19:22 schanz
```

You see the user-HOME-directories (the accounts) on your system. Most of the Users in this example belong to the Group ’users’. Like usual the Access-Flags of most Users (a107, cde, tenzer) is set to hexadecimal ’700’. This is the default and means it is ’rwx — —’. Only the User himself has access (read, write, execute) to his corresponding HOME-directory! For all Users of the Group ’users’ and for all ’others’ Users the File-Permissions to the HOME-directories of these User is limited to (-, -, -), hence they have no access at all!

However, User ’schanz’ has changed his Access-Flags into ’754’. This means, he self has the File-Permissions (read, write, execute), all members of the Group ’users’, where he belongs to, will have the File-Permissions (read, -, execute) and all Others users on the system have the File-Permissions (read, -, -) to the HOME-directory of User ’schanz’. Therefore all Users on the system that belong to the Group ’users’ can access the HOME-directory of ’schanz’ for reading and executing files but not for writing or deleting files. Users outside the Group ’users’, all Others, can still access the HOME-directory of ’schanz’ for reading files, but they can neither execute a file from there nor they can write or delete one. The modification of the Access-Flags for files and directories, explicitly for the HOME-directory can be only changes by the User himself. You can isolate your HOME-directory completely or grant limited access for other Users to certain files.

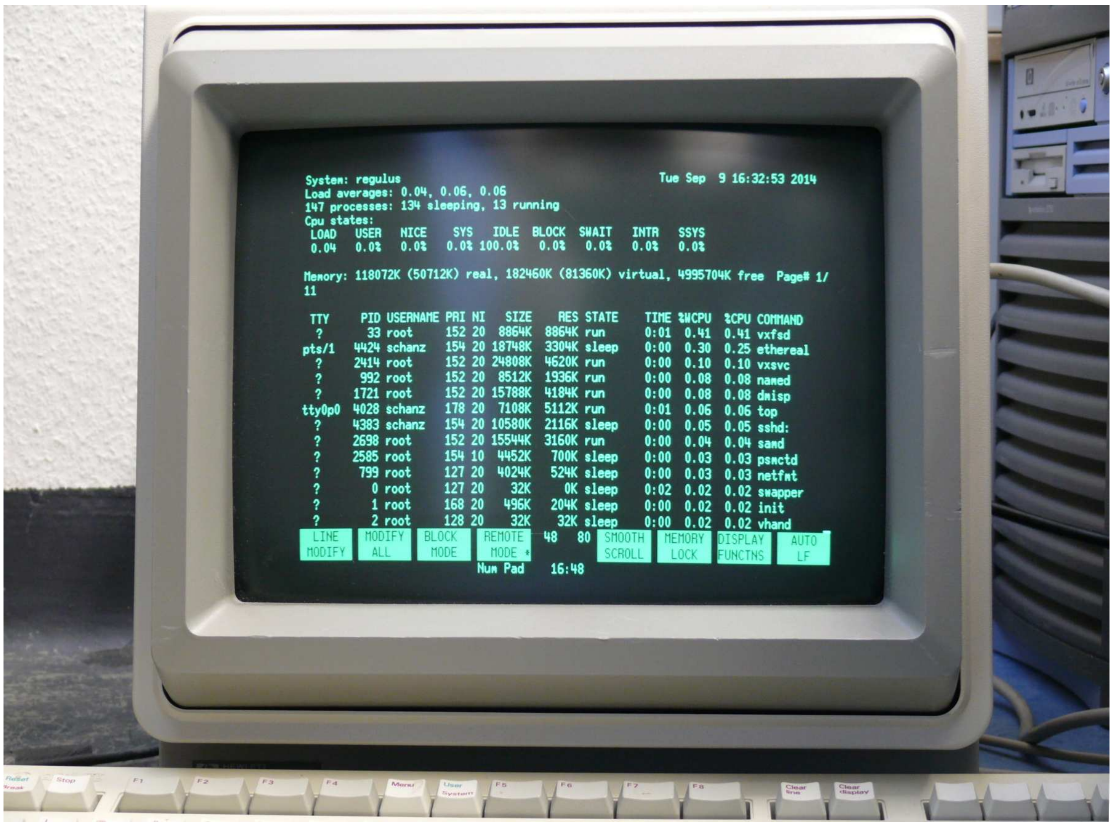
```
   Never make anything simple
  and efficient when a way can
  be found to make it complex
         and wonderful
```

As you can see there is also a user ’printer’ in the example. The ’Owner’ of ’printer’ is the ’root’-User (Super-User) and ’printer’ belongs to the Group ’root’. The Super-User has File-Permissions to ’read’, ’write’ and ’execute’ files in the HOME-directory of ’printer’, members of the Group ’root’ can still ’read’ and ’execute’ files in the HOME-directory of ’printer’, same as all Other users on the system that do not belong to the Group ’root’. That is the meaning of the Access-Flags ’755’ (rwx r-x r-x) set for the HOME-directory of ’printer’.

The Access-Flags of LINUX grant every User on the system exactly the amount of privacy that he wants to have. You can isolate your data completely from any other User, grant some Users of special groups some limited access or even open your account to everybody and everything else on the system completely. The last however would not be recommendable, considering that opening your account completely to others would also mean that everybody else on the system can delete your data. Make sure you be aware what levels of access you grant to other people and how the Access-Flags for your account are set.

14 Terminals

The Console provides a Commandline Interface to control and program your computer. In the early days of UNIX, computers didn't have graphical displays nor a Desktop, they were completely operated via Console. These Console was usually a Terminal that was attached to one of the serial ports of the computer like shown in Figure 15. A Terminal is an electronic device capable to communicate with the computer on a serial cable. It has no own processing unit, mass memory or disk drives but provides a display for written output and a keyboard to type in commands. The Users were usually connected via Terminal to the computer, directly or via a network. Using the Terminal the Users could access all computer functions by command, writing and executing programs, writing or receiving emails, etc, short, everything that did not require a pixel based high resolution graphical output. The Terminal was the only interface to the computer. Today, many servers, mainframes and super-computers that have no need for graphics still work on that basis.



```
System: regulus                                          Tue Sep  9 16:32:53 2014
Load averages: 0.04, 0.06, 0.06
147 processes: 134 sleeping, 13 running
Cpu states:
 LOAD   USER   NICE    SYS   IDLE  BLOCK  SWAIT   INTR   SSYS
 0.04   0.0%   0.0%   0.0% 100.0%   0.0%   0.0%   0.0%   0.0%

Memory: 118072K (50712K) real, 182460K (81360K) virtual, 4995704K free  Page# 1/
11

 TTY     PID USERNAME PRI NI   SIZE    RES STATE    TIME %WCPU  %CPU COMMAND
  ?       33 root     152 20  8864K  8864K run      0:01  0.41  0.41 vxfsd
pts/1   4424 schanz   154 20 18748K  3304K sleep    0:00  0.30  0.25 ethereal
  ?     2414 root     152 20 24808K  4620K run      0:00  0.10  0.10 vxsvc
  ?      992 root     152 20  8512K  1936K run      0:00  0.08  0.08 named
  ?     1721 root     152 20 15788K  4184K run      0:00  0.08  0.08 dmisp
tty0p0  4028 schanz   178 20  7108K  5112K run      0:01  0.06  0.06 top
  ?     4383 schanz   154 20 10580K  2116K sleep    0:00  0.05  0.05 sshd:
  ?     2698 root     152 20 15544K  3160K run      0:00  0.04  0.04 samd
  ?     2585 root     154 10  4452K   700K sleep    0:00  0.03  0.03 psmctd
  ?      799 root     127 20  4024K   524K sleep    0:00  0.03  0.03 netfmt
  ?        0 root     127 20    32K     0K sleep    0:02  0.02  0.02 swapper
  ?        1 root     168 20   496K   204K sleep    0:00  0.02  0.02 init
  ?        2 root     128 20    32K    32K sleep    0:00  0.02  0.02 vhand
```

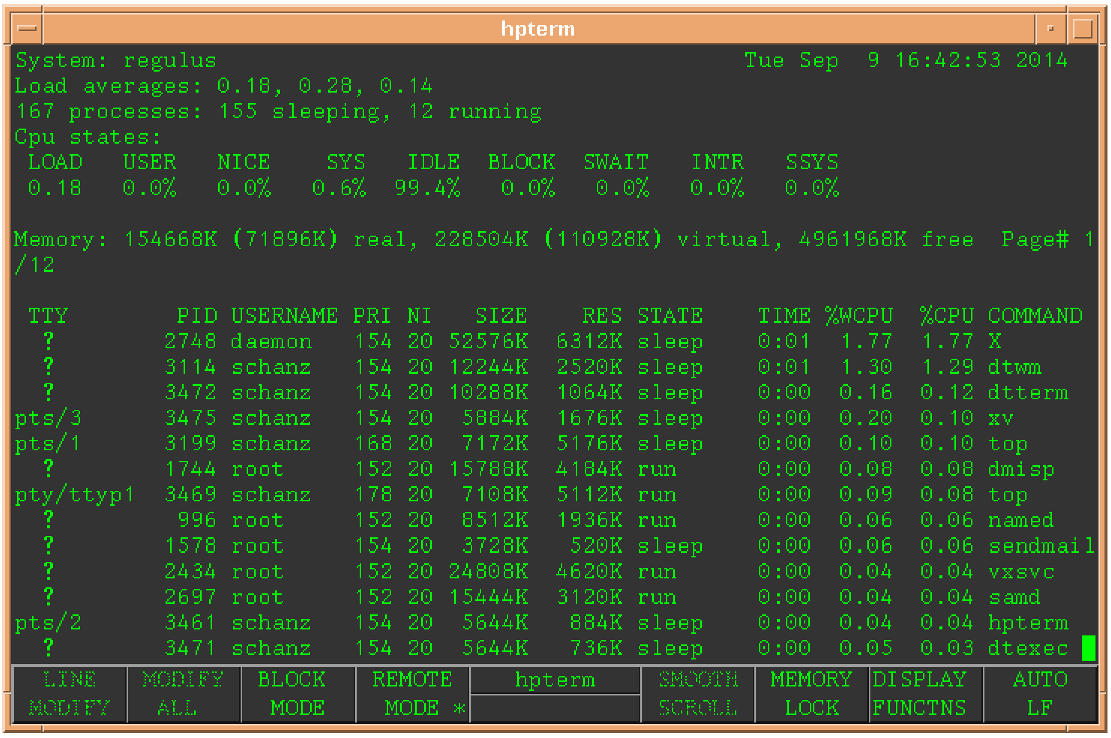
Figure 15: *A Terminal of the Hewlett-Packard Company. Such Terminals were used to communicate to a host computer. The display shows the output of the 'top'-command on the Console, launched by User 'schanz' on the system 'regulus'.*

When it comes to the Console, LINUX is still using Terminals today! These Terminals are no longer electronic devices connected to the serial port of the computer but emulations that run on the Console or in a matter provide the Console. Of course there were different

Terminals in the past. Every big company tried to make their own Terminals the new computer industries standard. Most of this different Terminal types survived until today as Terminal-emulations added also by a few generic Terminals that had never a physical hardware counterpart. Different LINUX distributions may advertise different Terminals, fortunately most Terminals are very similar to each other. As a LINUX User you don't need to bother very much about it. Figure 16 shows such a Terminal emulation running the 'top'-command on a LINUX system.

Nevertheless here is a short list of some popular Terminal emulations on LINUX:

(1)	konsole	(default at KDE)
(2)	xterm	(x11 default)
(3)	gnome-terminal	(default at GNOME)
(4)	rxvt	
(5)	dtterm	(default at CDE)
(6)	hpterm	(only HP-UX systems)
(7)	aterm	



```
System: regulus                                         Tue Sep  9 16:42:53 2014
Load averages: 0.18, 0.28, 0.14
167 processes: 155 sleeping, 12 running
Cpu states:
 LOAD   USER   NICE    SYS   IDLE  BLOCK  SWAIT   INTR   SSYS
 0.18   0.0%   0.0%   0.6%  99.4%   0.0%   0.0%   0.0%   0.0%

Memory: 154668K (71896K) real, 228504K (110928K) virtual, 4961968K free  Page# 1
/12

 TTY          PID USERNAME PRI NI   SIZE    RES STATE    TIME %WCPU  %CPU COMMAND
  ?          2748 daemon   154 20 52576K  6312K sleep    0:01  1.77  1.77 X
  ?          3114 schanz   154 20 12244K  2520K sleep    0:01  1.30  1.29 dtwm
  ?          3472 schanz   154 20 10288K  1064K sleep    0:00  0.16  0.12 dtterm
pts/3        3475 schanz   154 20  5884K  1676K sleep    0:00  0.20  0.10 xv
pts/1        3199 schanz   168 20  7172K  5176K sleep    0:00  0.10  0.10 top
  ?          1744 root     152 20 15788K  4184K run      0:00  0.08  0.08 dmisp
pty/ttyp1    3469 schanz   178 20  7108K  5112K run      0:00  0.09  0.08 top
  ?           996 root     152 20  8512K  1936K run      0:00  0.06  0.06 named
  ?          1578 root     154 20  3728K   520K sleep    0:00  0.06  0.06 sendmail
  ?          2434 root     152 20 24808K  4620K run      0:00  0.04  0.04 vxsvc
  ?          2697 root     152 20 15444K  3120K run      0:00  0.04  0.04 samd
pts/2        3461 schanz   154 20  5644K   884K sleep    0:00  0.04  0.04 hpterm
  ?          3471 schanz   154 20  5644K   736K sleep    0:00  0.05  0.03 dtexec
```

Figure 16: *A Terminal emulation of an HP-Terminal running the 'top'-command for User 'schanz' on a the system 'regulus'. If you compare the Picture to Figure 15, you will recognize the similarity of the emulation to the original hardware Terminal.*

If it happens once, it’s a bug. If it happens twice, it’s a feature. If it happens more than twice, it’s a design philosophy

15 The Shell

The Shell provides a kind of programming language and the command interpreter that runs on the Terminal. There are of course more than one Shell interpreters available on LINUX, one for every taste and every programming language. Why one Shell if you can have seven? The most common Shells under LINUX are:

```
(1) sh    /bin/sh    Bourne-Shell        classical LINUX/UNIX-Shell

(2) bash  /bin/bash  Bourne again Shell  improved bourne ’sh’, most common Shell on LINUX

(3) csh   /bin/csh   C-Shell             a Shell that uses C language syntax

(4) tcsh  /bin/tcsh  Tenex C-Shell       improved C-Shell

(5) ksh   /bin/ksh   Korn-Shell          very popular among administrators

(6) zsh   /bin/zsh   Z-Shell             very much like ’ksh’, powerful

(7) xonsh /bin/xonsh Python-Shell        a Shell that uses Python language syntax
```

Each of these Shells has its own resource files that are stored in the HOME-Directory of the User and commonly starts with a ’.’: '.bashrc' for the 'bash' or '.csh' for the 'csh' etc. In the resource files you can define the behaviour of the Shell, Environment-Variables like $PATH and command aliases for instance.

Cutout of the '.bashrc', the resource file of the ’bash’-Shell:

```
############ bashrc ##########

# Set up finder
      function findx ()
      { grep < /.log/findDB $* | more;
      }

# Set up pager
      PAGER=/bin/more
      export PAGER

# include dot in search path, but not for root !

# Set up the search paths:
PATH=$HOME/.dt/bin:/bin:/usr/bin:/sbin:/usr/sbin:/usr/bin/X11:/usr/local/bin:/us
r/local/bin/X11:/usr/contrib/bin:/usr/contrib/bin/X11:/usr/dt/bin:$PATH
export PATH

PATH=$HOME/.dt/bin:$PATH
export PATH

# Set up the Manpath:
```

```
MANPATH=/usr/man:/usr/local/man:/net/usr/man:$MANPATH
export MANPATH

# Set prompt:
if [ -O /etc/passwd ]; then
        export PS1="\$system:\[\W] # "
else
        export PS1="\h:[\W] > "
fi

# Aliase
alias ll='ls -l'
alias cls=clear
alias h=history
alias md=mkdir
alias rd=rmdir
alias dir=ls
alias copy=cp
alias m=more
alias del=rm
alias da='ls -al | more'
alias rename=mv
alias et=$HOME/.dt/bin/defedit

stty erase "^H" kill "^U" intr "^C" eof "^D" susp "^Z" hupcl ixon ixoff tostop

EDITOR=/usr/bin/vi
export EDITOR

BROWSER_PATH=/usr/local/bin/firefox
export BROWSER_PATH
```

Most Shells maintain a list of the last inputted commands in a so called 'history'. When you type in the command **'history'** (if you are using the 'bash') it will show you the list of the last 500 commands. You can look up the call of earlier commands and just recall them by the 'arrow'-keys (up and down) on the keyboard or by calling the appropriate history number. Most Shells also use command completion. Just type the first few letters of a command and then press the 'tab'-key of the keyboard. The Shell will try to guess which command you meant to have next and makes suggestions if there is more than one possible command choice available. This function is very useful for the daily use of the Commandline, it speeds up Commandline interactions tremendously. The following gives you an example about how such a history list can look like:

```
 :    :
492  man man
493  uname -a
494  ps
495  man ps
496  su -
497  ls -al
498  ssh root@rigelD
499  grep root /etc/passwd
500  vi .bashrc
501  history
```

Most LINUX commands will send their output to „stdio“ (Standard I/O), that means directly into the Console. This can be problematic if the output of a command is very extensive because the Terminal will start to scroll and its hard to follow.

Therefore you can directing the output of any command directly into a file. This file you can open by an Editor later for further analysis. In order to direct the 'stdio' output of a command into a file, you just add the '>' sign together with a filename for the logfile to the command call:

```
[~]> history > mylogfile.log
```

This will direct the output of the command (in this case 'history' – 500 lines) into the logfile 'mylogfile.log' which will be newly generated for this purpose. You can search further for a special string in this 'mylogfile.log' by using the 'grep'-command for instance:

```
[~]> grep jpg mylogfile.log
```

Using the 'logfile' mechanism above, every new call of the command will generate the 'mylogfile.log' anew, the former 'mylogfile.log' will be overwritten. If you want to add the output of a particular command to a logfile without overwriting its content, you can make the command call by using the '>>' sign instead.

```
[~]> history >> mylogfile.log
```

This will add the output of 'stdio' of the command ('history' in this case) just at the end of the already existing logfile 'mylogfile.log' without overwriting the old part. You can concatenate the command output of several 'stdio' outputs into one single logfile.

```
Any sufficiently advanced bug
 is indistinguishable from a
  feature.  - Rich Kulawiec
```

16 Pipes

Pipes '|' can connect the output of one command to the input of another. This is a clever way to construct pipelines. The classical way would be to write a script that calls several commands one after another and connect the output and inputs of these commands via temporary files or variables. Pipes instead are a much more elegant way to connect and combine commands with each other.

Considering you want to know how many pictures of the filetype 'jpg' in your HOME-directory are? So lets go to your HOME-directory first, type:

```
[~]> cd
```

Using the 'ls'-command you can display the files in your HOME-directory, but probably there are no pictures on the 'root'-level of your HOME-directory. Maybe there are some 'jpg'-pictures in one of the subdirectories or in one of the hidden directories (those which start with a '.'). You could use the 'find' command to search all subdirectories below your HOME-directory. Try:

```
[~]> find .
```

This will give you a long list of all files below your HOME-directory, not only the picture-files of the 'jpg'-type. To filter out only the 'jpg'-files you could use the 'grep'-command and connect it by using a PIPE with the output of the 'find .' command, try:

```
[~]> find . | grep .jpg
```

The ’grep .jpg’-command will filter out only the lines that contain the ’.jpg’ string and display them. If you want, you can sort the output of this commands alphabetically with the ’sort’ command. Just add the ’sort’-command by using another PIPE, try:

```
[~]> find . | grep .jpg | sort
```

This will give you a sorted list of the files below your HOME-directory that contain the string ’.jpg’. You can direct this list into a file if you want it for later:

```
[~]> find . | grep .jpg | sort > my_sorted_jpg_list.log
```

If you want to know how many files of the type ’.jpg’ are there, you can let it count by the ’wc’ (word count) command. Just add the ’wc’-command by using another PIPE, try:

```
[~]> find . | grep .jpg | sort | wc -l
```

The output of this pipeline will give you only one single number, the number of ’jpg’-files below your HOME-directory. You want to know how many ’.jpg’-files are in the complete Filesystem of your computer? No problem, type:

```
[~]> find / | grep .jpg | sort | wc -l
```

Here we have replaced the ’find .’-command by ’find /’. The command ’find /’ will search everything below the ’root’-level directory ’/’ instead. You will see a lot of ’Permission denied’ if you try this, because as a normal User you will not have permissions to look into most of the directories of the system. Hence the call will only count ’.jpg’-files that can be seen by normal Users! It will also run quite a while, because its scanning the complete Filesystem. Doing it on my file server I found the number: 211096

You can use the PIPE mechanism also to direct the command output to a paging program. Considering the ’history’-command from above again. The output of ’history’ is quite extensive and makes the Terminal scrolling away. You can direct the output of the ’history’-command very easily by using the PIPE mechanism into a paging program like ’more’ or ’less’. Try:

```
[~]> history | less
```

The PIPE will direct the output into the paging program ’less’ where you can scroll the list by the cursor keys of the keyboard. You can also try the pager program ’more’ instead:

```
[~]> history | more
```

You should always use a paging program and the PIPE mechanism when the output of a command is extensive. Usually even the call of the command ’ls -al’ in your own HOME-directory will produce more output than the Terminal can display at once. Try;

```
[~]> ls -al | less
```

Using pipelining you can produce very powerful system command functions just by combining the various commands that come with your LINUX distribution just for free. Together with PIPE the commands of the operating system build a kind of construction set that can produce amazingly outputs!

I think there's a world market for about five computers. - attr. Thomas J. Watson (Chairman of the Board, IBM), 1943

17 Editors

Whats an Editor?

An Editor is a software that allows you to open and manipulate the content of any file on your computer. The kind of Editor you will need most is the Texteditor. The Texteditor can open 'readable' text files of any kind. The LINUX-files are usually divided in 'readable' files that contain ASCII signs, and 'unreadable' files that contain binary code for machine execution. The terms 'readable' and 'unreadable' just refers here to the use with Texteditors. Files that can be read by Texteditors are usually plain text-files, readme-files, log-files, configuration-files and so on. Machine coded files that are unreadable by Texteditors are usually program-executables, function-libraries, image-files, sound-files, compressed-document-files and so on.

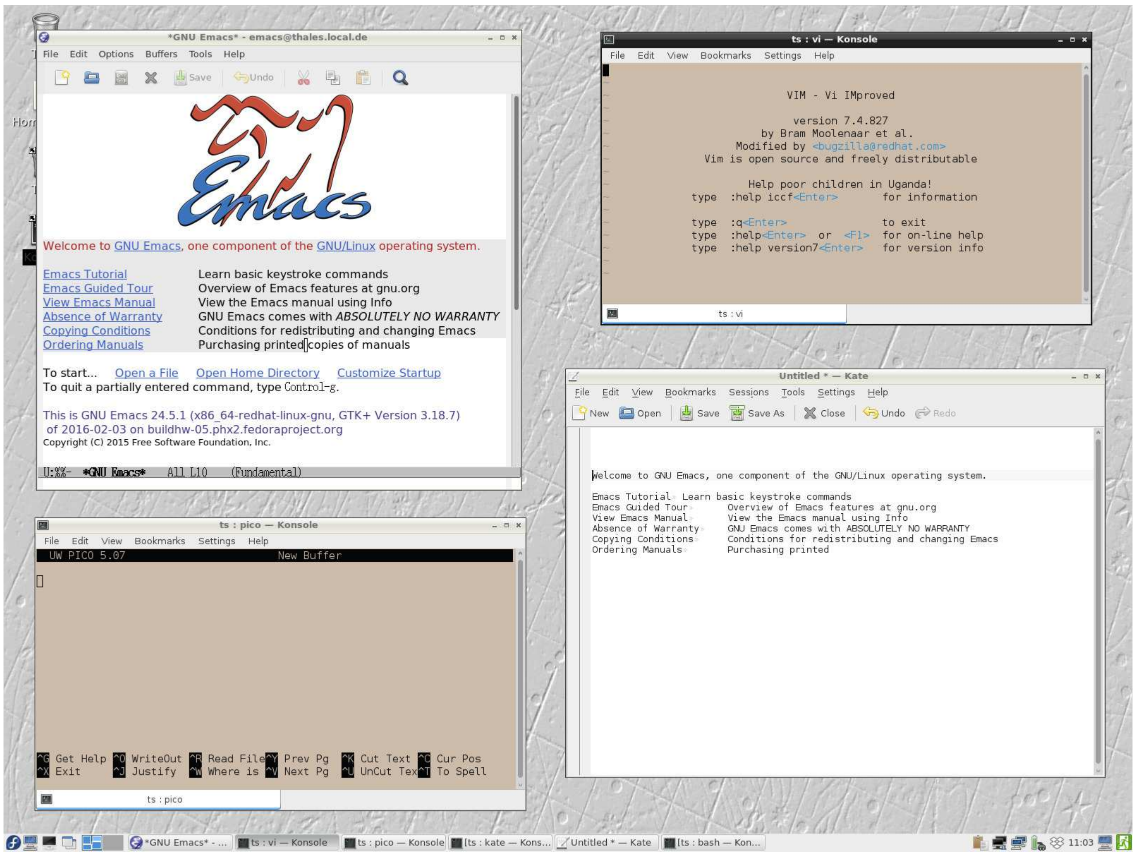
The most common use for Texteditors on a LINUX system is the setup of configuration-files. Most global configuration files of the system are located in the '/etc'-directory, the User dependent configuration files however usually are hosted in the user-HOME-directory of the User, commonly they start with a '.' in the filename. You can look up many of this local configuration files when you go to your user-HOME-directory ('cd') and type in the command 'ls -a' or 'ls -al | less'. Most files that you can see starting with a '.' will be used for configuration. In order to open or modify these configurations, you need some Texteditor.

Some most common Editors under the LINUX system are:

```
(1) 'vi'       : very simplistic and powerful Editor for system administration,
                 it is often considered cryptic and complicated by beginners. A
                 Console based Editor for professionals.

(2) 'emacs'    : can run on the Console as well as in a window based mode, very
                 extended functionality, very common among software developers.

(3) 'nedit'    : simple window based (graphical) Editor, easy to use also for
                 inexperienced users.

(4) 'gedit'    : The GNOME Editor, window based graphical Editor, easy to use.

(5) 'kate'     : very powerful graphical Editor, slow and big.
```

```
(6) 'dtpad'     : simple graphical Editor, available only on CDE systems

(7) 'pico'      : simple Console based Editor
```

You should prefer an Editor that can also run in a Console mode. Suppose you want editing a file on a system that does not support any graphical output like embedded systems, routers, micro controllers, the Raspberry-Pi or you have a secure shell connection to a remote system without graphics or you just have a damaged system after a crash that can not start graphics temporarily. You will still be able of editing files and repair the system using a Console based Editor, a graphical Editor will not be of any use in such circumstances! Figure 18 shows a screenshot of four running Editors on the 'LXDE'-Desktop: 'emacs', 'vi', 'pico' and 'kate'.



Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial   Learn basic keystroke commands
Emacs Guided Tour   Overview of Emacs features at gnu.org
View Emacs Manual   View the Emacs manual using Info
Absence of Warranty   GNU Emacs comes with *ABSOLUTELY NO WARRANTY*
Copying Conditions   Conditions for redistributing and changing Emacs
Ordering Manuals   Purchasing printed copies of manuals

To start...   Open a File   Open Home Directory   Customize Startup
To quit a partially entered command, type Control-g.

This is GNU Emacs 24.5.1 (x86_64-redhat-linux-gnu, GTK+ Version 3.18.7)
of 2016-02-03 on buildhw-05.phx2.fedoraproject.org
Copyright (C) 2015 Free Software Foundation, Inc.

```
U:%%-  *GNU Emacs*    All L10    (Fundamental)
```

```
                 VIM - Vi IMproved

                 version 7.4.827
              by Bram Moolenaar et al.
         Modified by <bugzilla@redhat.com>
      Vim is open source and freely distributable

             Help poor children in Uganda!
 type  :help iccf<Enter>       for information

 type  :q<Enter>               to exit
 type  :help<Enter>  or  <F1>  for on-line help
 type  :help version7<Enter>   for version info
```

```
  UW PICO 5.07                        New Buffer

^G Get Help  ^O WriteOut  ^R Read File^Y Prev Pg   ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where is ^V Next Pg   ^U UnCut Tex^T To Spell
```

```
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial  Learn basic keystroke commands
Emacs Guided Tour       Overview of Emacs features at gnu.org
View Emacs Manual       View the Emacs manual using Info
Absence of Warranty     GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions      Conditions for redistributing and changing Emacs
Ordering Manuals        Purchasing printed
```

Figure 17: *Screenshot of some running LINUX Editors. Upper left: 'Emacs', upper right: 'vi', lower left: 'pico', lower right: 'kate'. The 'copy' and 'paste' via the Clipboard is just done by marking the textfield you want to copy via the left mouse button (here seen in the emacs window -> gray) and drop it at the position you want to paste it via the middle mouse button (here in the kate window).*

In order to open a file, lets say the config file of the Bourne-Shell '.bashrc' in your HOME-directory, you just call the name of the Editor together with the file you want to open, try it!

```
[~]> gedit .bashrc
```

I've finally learned what "upward compatible" means. It means we get to keep all our old mistakes. – Dennie van Tassel

18 The Clipboard

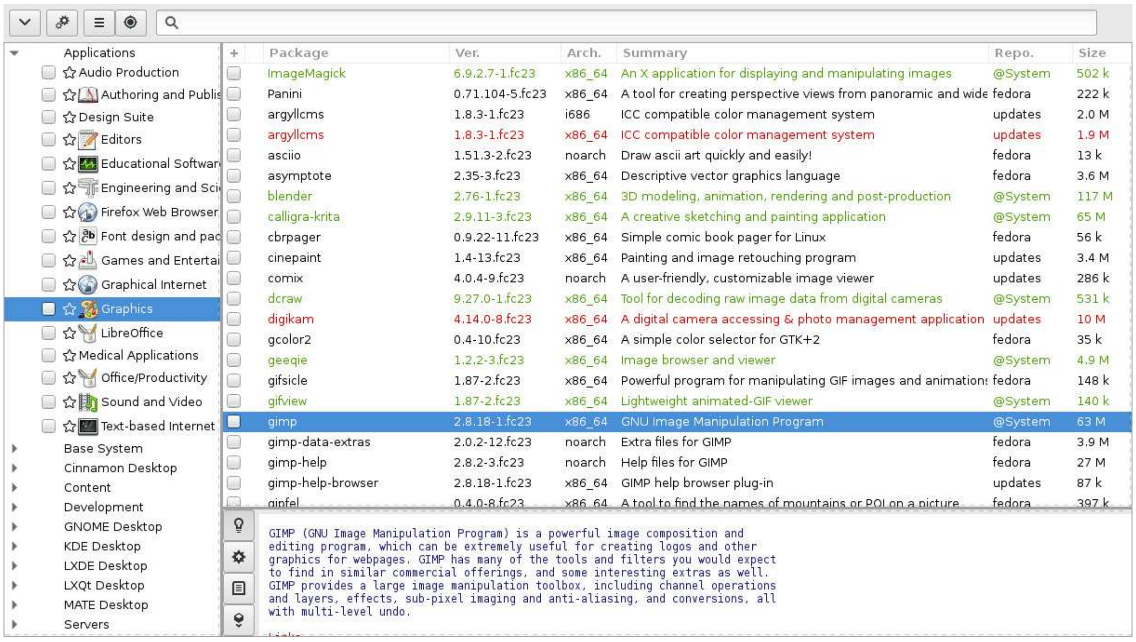
The 'Clipboard' mechanism allows you to 'copy' and 'paste' textfields from one application into another just by marking them with the mouse of the computer. The Clipboard of LINUX works different than you may know it from Microsoft WINDOWS. In LINUX you can 'copy' a textfield from the Console or from a document you have opened via Editor just by framing it with the left mouse button. For 'paste' you point the mouse to the receiving window and press the middle mouse button in order to drop the textfield. The Microsoft WINDOWS based 'copy' and 'paste' mechanism using 'ctrl c', 'ctrl v' will not work on most LINUX applications! In contrary, 'ctrl c' is usually used to stop an application, try 'ctrl c' on a running command. For instance you can start the 'emacs' Texteditor from the Console, just type:

```
[~]> emacs
```

If you click now into the Console and press 'ctrl c' on the keyboard, the Emacs Editor will be stopped immediately again, 'ctrl c' stops the execution of the Child process on the Console. If you press 'ctrl d' you will quit the Console as well! There are some exceptions like OpenOffice which where originally developed for WINDOWS that can also use 'ctrl -c' 'ctrl -v' for the Clipboard like it is common on WINDOWS systems.

19 Applications

The mechanism for the installation of software depends on your LINUX distribution. Currently there are two major LINUX branches that dominate the LINUX market. These two branches are 'Debian' and 'Fedora' and they use different software for package management and installation. The Debian systems are using the Commandline based 'dpkg' Package-Manager and the 'apt' installation software. As a graphical installation software 'synaptic' is used. In contrast, the Fedora based systems have the 'rpm' Package-Manager and the 'dnf' Commandline installation software. As graphical interface, the software 'yumex-dnf' is used. Older Fedora based systems still use 'yum' as Commandline installer instead of 'dnf', and 'yumex' as graphical interface instead of 'yumex-dnf'. The graphical interfaces are very easy to use because they present you a complete list of the available software from various repositories in the internet. You can install new and additional software just by clicking with the mouse on it. Console based installers like 'apt' and 'dnf' are a bit more difficult to handle but more powerful also. For further information, please consult the Man-Page, type 'man dnf' or 'man apt'. I personally recommend you to install a graphical Package-Manager software like 'synaptic' or 'yumex-dnf' on your system!



Package	Ver.	Arch.	Summary	Repo.	Size
ImageMagick	6.9.2.7-1.fc23	x86_64	An X application for displaying and manipulating images	@System	502 k
Panini	0.71.104-5.fc23	x86_64	A tool for creating perspective views from panoramic and wide	fedora	222 k
argyllcms	1.8.3-1.fc23	i686	ICC compatible color management system	updates	2.0 M
argyllcms	1.8.3-1.fc23	x86_64	ICC compatible color management system	updates	1.9 M
asciio	1.51.3-2.fc23	noarch	Draw ascii art quickly and easily!	fedora	13 k
asymptote	2.35-3.fc23	x86_64	Descriptive vector graphics language	fedora	3.6 M
blender	2.76-1.fc23	x86_64	3D modeling, animation, rendering and post-production	@System	117 M
calligra-krita	2.9.11-3.fc23	x86_64	A creative sketching and painting application	@System	65 M
cbrpager	0.9.22-11.fc23	x86_64	Simple comic book pager for Linux	fedora	56 k
cinepaint	1.4-13.fc23	x86_64	Painting and image retouching program	updates	3.4 M
comix	4.0.4-9.fc23	noarch	A user-friendly, customizable image viewer	updates	286 k
dcraw	9.27.0-1.fc23	x86_64	Tool for decoding raw image data from digital cameras	@System	531 k
digikam	4.14.0-8.fc23	x86_64	A digital camera accessing & photo management application	updates	10 M
gcolor2	0.4-10.fc23	x86_64	A simple color selector for GTK+2	fedora	35 k
geeqie	1.2.2-3.fc23	x86_64	Image browser and viewer	@System	4.9 M
gifsicle	1.87-2.fc23	x86_64	Powerful program for manipulating GIF images and animation	fedora	148 k
gifview	1.87-2.fc23	x86_64	Lightweight animated-GIF viewer	@System	140 k
gimp	2.8.18-1.fc23	x86_64	GNU Image Manipulation Program	@System	63 M
gimp-data-extras	2.0.2-12.fc23	noarch	Extra files for GIMP	fedora	3.9 M
gimp-help	2.8.2-3.fc23	noarch	Help files for GIMP	fedora	27 M
gimp-help-browser	2.8.18-1.fc23	x86_64	GIMP help browser plug-in	updates	87 k
gipfel	0.4.0-8.fc23	x86_64	A tool to find the names of mountains or POI on a picture	fedora	397 k

```
GIMP (GNU Image Manipulation Program) is a powerful image composition and
editing program, which can be extremely useful for creating logos and other
graphics for webpages. GIMP has many of the tools and filters you would expect
to find in similar commercial offerings, and some interesting extras as well.
GIMP provides a large image manipulation toolbox, including channel operations
and layers, effects, sub-pixel imaging and anti-aliasing, and conversions, all
with multi-level undo.
```

Figure 18: *The YUMEX graphically installation software offers a simple interface to software repositories and all the available software that can still be installed on your Fedora (rpm) based LINUX system.*

On a Debian based system just type 'apt-get install synaptic' to the Commandline of the Console and the 'synaptic' software will be downloaded and installed automatically. If you need 'root'-User privileges, type: 'sudo apt-get install synaptic'.

If you are using a Fedora based system you type 'dnf install yumex-dnf' instead to download and install the 'yumex-dnf' graphical package management software automatically. If you need 'root'-User privileges, type: 'sudo dnf install yumex-dnf'.

WARNING!

The installation of software usually affects and concerns all Users of a system, therefore it requires Super-User privileges! Normal Users are not allowed to install software to the system or remove software from the system by using the package management system. If you need a special software that is not installed on the system or a software upgrade → ask your Admin to install it!

In a limited way Users can also install and run additional software into their own HOME-directories. In general this is not supported by the Package-Manager but can be done manually by the User if he knows how to do it ($\rightarrow$ configure $\rightarrow$ make $\rightarrow$ make install). Larger software installations however will most probably not fit in the user space and will require a global installation.

```
If builders built buildings
 the way programmers wrote
  programs, then the first
  woodpecker to come along
would destroy civilization
```

20 Administration

The administration of the LINUX system is certainly beyond the scope of this manual. Unfortunately there is still no common graphical administration software for LINUX so far. Most LINUX distribution cook their own solutions in order to configure and administrate the system. Many distributions have no graphical admin tools at all and rely on the Commandline skills of their Users and Admins. Under the terms of administration are the following tasks:

(1) setup and deleting of user accounts
(2) diskdrive management, installation, formatting, mounting, Logical Volume management
(3) network configuration
(4) system backup
(5) installation and configuration of new hardware
(6) language and keyboard setup
(7) system time and date setup
(8) firewall configurations
(9) printer configuration and the management of printer-queues
(10) configuration of the graphical system and the display resolution
(11) configuration of services (cron/automounter/cups/nfs/samba/sshd etc ...)
(12) configuration of daemons and servers
(13) installation and upgrade of software
etc....

In general you can always admin a LINUX system completely over the Commandline of the Console. Most of the configuration will be done by config files that are hosted somewhere below the '/etc'-directory of the system. In many cases however these tasks require expert knowledge. I recommend to consult an user or admin forum in the internet that is related to your particular LINUX distribution. Be careful not to exercise instructions that do not exactly match your LINUX release or distribution. The administration of LINUX can vary a lot between different versions and distributions!

On Fedora based distributions there exists a number of small graphical admin tools for many purposes. These tools all start with the word 'system-...'. You can type 'system' on the Console and then press the 'tab'-key of the keyboard. The command completion will show you a list of commands that start with the word ’system’:

```
system-config-boot
system-config-users
system-config-firewall
system-config-network
system-config-services
...
```

Using this tools even a beginner can admin many aspects of the Fedora-LINUX system.

The probably best and most complete admin tool among LINUX systems is the 'yast' software of the SUSE-Distribution (Fedora based). Yast provides a detailed Administration-Manager where you can admin nearly every function of the system via mouse by a graphical user interface, see also Figure 19. Such a complete admin environment you normally find only among commercial UNIX operating systems like IBMs AIX ('smit') or HP-UX by Hewlett-Packard ('sam').

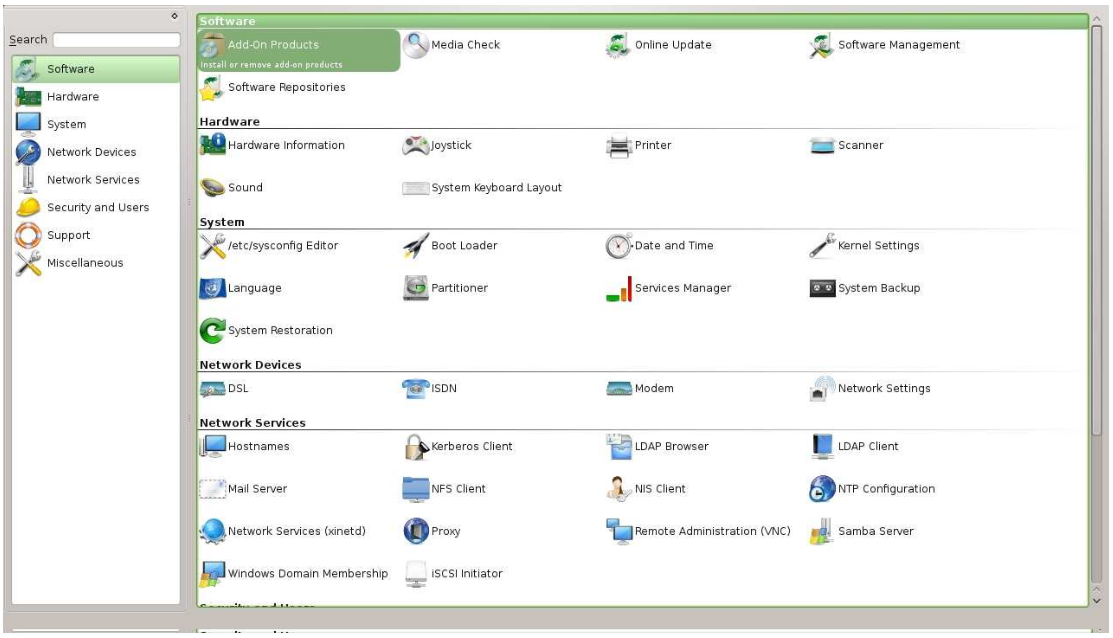


Figure 19: *Screenshot of the YAST system Administration-Manager on SUSE-Linux (OpenSUSE). YAST gives a good example how system administration can be simplified on a graphical interface.*

On any other LINUX distribution besides SUSE, you can still admin the system by using the graphical admin tools that are provided by certain Desktop-Managers like KDE or GNOME. These Desktop-Managers usually integrate a lot of admin functions directly in their graphical setup. The ’Settings’-Manager of GNOME or ’system settings’ of KDE will provide graphical tools for most administration tasks you usually have to do on a LINUX system and they are independent of the particular LINUX distribution.

```
There are never any bugs you
      haven't found yet
```

21 Installation

Finally some hints for the LINUX initial system installation. Normally LINUX today comes on a live-CD/DVD where you can boot a LINUX from the CD/DVD and than decide if you want continue with a fully LINUX installation. Most of the files will then be installed from a repository in the internet. If you install LINUX yourself, make sure that you also install the development packages for your distribution! This will allow you later to compile and install additional software or to modify and rebuild the KERNEL for special purposes. Especially for people working in science or in software development, a working compiler is a mandatory requirement on the system! You can check if the development package is installed just by typing:

```
[~]> gcc -v
```

to the Commandline of the Console. If your system responds with a 'Command not found' than you should install the development packages for your distribution. You should also make sure you have OpenGL support for the graphical window system. Many recent programs have extended visualization properties which you cant use properly without OpenGL drivers. If applications like 'GOOGLE EARTH' or 'STELLARIUM' are very slow you most probably are missing the correct drivers for your graphics system. You can try to find and install the proper drivers from a repository for your system as discussed in the chapter 20.

If you install LINUX the first time, maybe for curiosity reasons, you should consider to do it on a separate computer and not on your main working computer with all your emails and the layout for the new hospital you have worked on for the past 6 month. Of course it is possible to install LINUX along with WINDOWS on the same system, making it a dual boot system. The best way to investigate LINUX is to use an old computer that you don't need for anything else anymore and install LINUX there. You will find out that most LINUX distributions run much faster and take less resources than most commercial operating systems. Hence a LINUX installation can often still run well also on an older computer. If you are new in LINUX be prepared to kill the system by accident, that's pretty normal at the beginning! When you have Super-User privileges and delete the KERNEL by chance, the system will never booting up again ever. Its better you create a desktop computer lab in your own office to investigate the LINUX Universe in safety!

This document was fully created on a LINUX system (Fedora 23)! The software involved was:

```
(1)  the 'vi' Editor
(2) 'texlive' the LaTeX document preparation system
(3) 'xlatex' LaTeX control interface
(4) 'xfig' vector graphics design software
(5) 'gimp' gnu image processing software
(6) 'xv' image processing software
(7) 'convert' Commandline image converting software
(8) 'ispell' syntax and spell checker software
(9) 'firefox' internet browser
```

`The program isn't debugged`
`until the last user is dead`