Get Ready for Open Source: SUSE Linux Enterprise Desktop Book 2

Novell Training Services

2007

Acknowledgements

Michael Eicks (Novell Inc.)

Contents

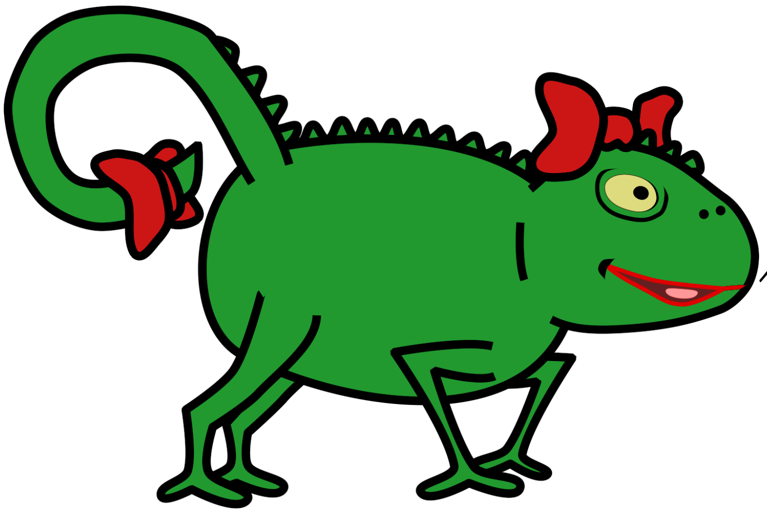
Hello, my name is *Suzie*…

…and I will help Geeko guide you through this course. The course is called “Get Ready for Open Source: SUSE Linux Enterprise Desktop – Book 2.”

In the last course, you learned a lot about open source software and Linux. You learned about important applications on Linux that can help you at school.

In this course, you will get experience as a software developer. But first you have to learn some important administration tasks.

This course is based on Novell SUSE Linux Enterprise Desktop 10. You will find all the software you need on the course CD.

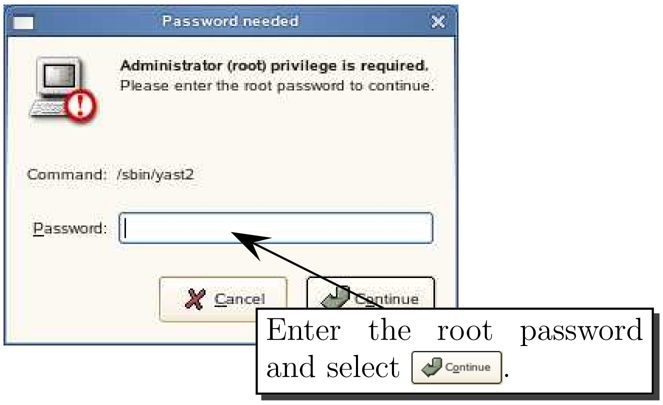


1 Administer Linux with YaST

You can use the tool YaST to complete many configuration tasks on your SUSE Linux Enterprise Desktop. YaST stands for "Yet another Setup Tool."

To start the graphical user interface, select the `YaST` icon in the `System` group of the Application Browser.

Then, enter the root password.



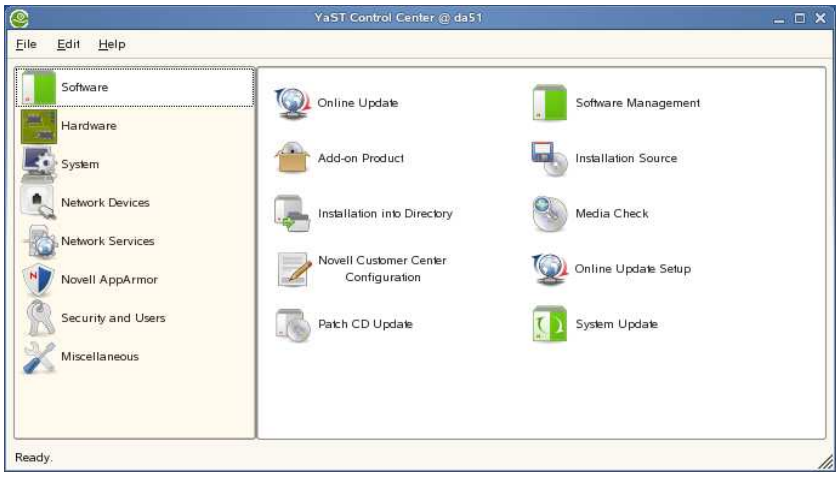
Administrator (root) privilege is required.
Please enter the root password to continue.

Command: /sbin/yast2

Password:

Enter the root password and select Continue.

The YaST Control Center dialog appears.



In the left frame of the dialog there is a list of module categories. If you select an item in this list, you can see the modules in the right frame.

There are a lot of modules available. In this manual, you will learn the modules to perform the following tasks:

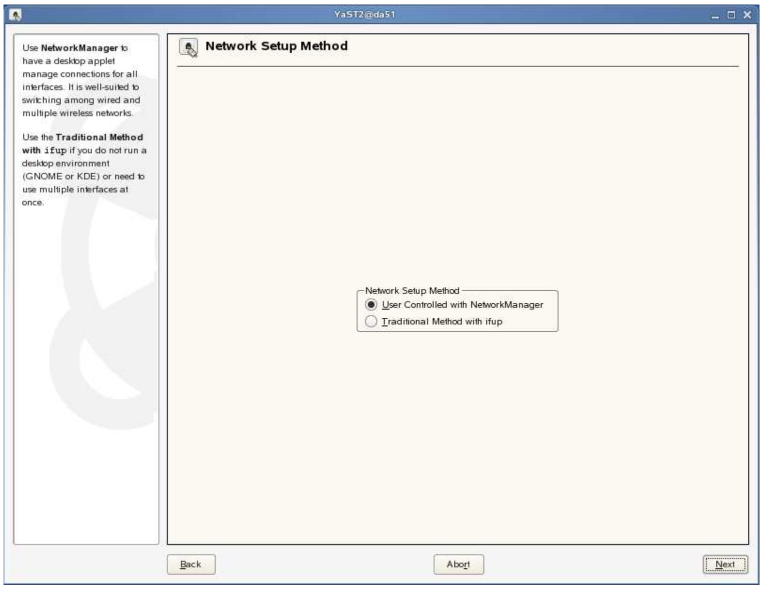
- Manage the network
- Manage users and groups
- Install and remove software
- Configure a printer

1.1 Manage the Network

To start the network configuration module of YaST, select `Network Devices` → `Network Card`.

Network Card

In the first dialog, select the setup method.



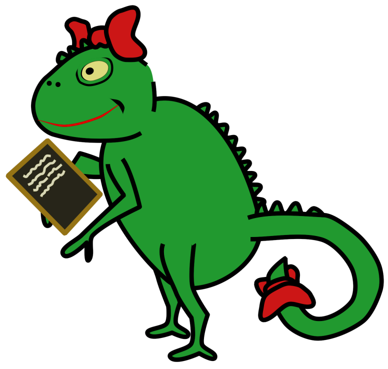
YaST2@da51

Use **NetworkManager** to have a desktop applet manage connections for all interfaces. It is well-suited to switching among wired and multiple wireless networks.

Use the **Traditional Method with `ifup`** if you do not run a desktop environment (GNOME or KDE) or need to use multiple interfaces at once.

Network Setup Method

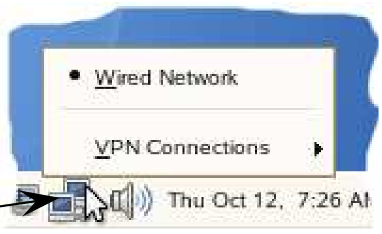
Network Setup Method
[✓ User Controlled with NetworkManager]
Traditional Method with ifup



You can choose between the following network setup methods:

`User Controlled with NetworkManager.` By default, only the system administrator root is allowed to change the network configuration. If you are running SUSE Linux Enterprise Desktop on a laptop computer and often changing networks (e. g., home network, school network, cafe network), you often have to switch to user root and back to normal user. To avoid this, use the NetworkManager.

With the NetworkManager method a normal user is allowed to change the network configuration. To do this, you have to use a desktop applet that manages the connections for all network interfaces.

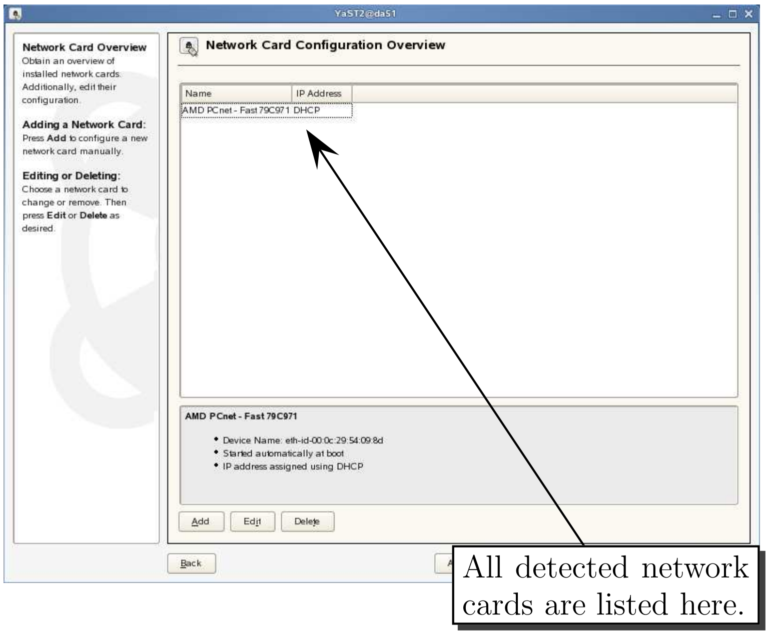


The NetworkManager applet in the GNOME panel.

`Traditional Method with ifup.` The traditional method uses the command `ifup` to set up a network interface. To change the network configuration, you have to have root permission.

The traditional method is recommended if you run your SUSE Linux Enterprise Desktop on a desktop machine, because you do not have to change the network configuration very often on such a machine.

Select Next to go to the Network Card Configuration Overview.



Network Card Overview
Obtain an overview of installed network cards. Additionally, edit their configuration.

Adding a Network Card:
Press **Add** to configure a new network card manually.

Editing or Deleting:
Choose a network card to change or remove. Then press **Edit** or **Delete** as desired.

Network Card Configuration Overview

Name	IP Address
AMD PCnet - Fast 79C971	DHCP

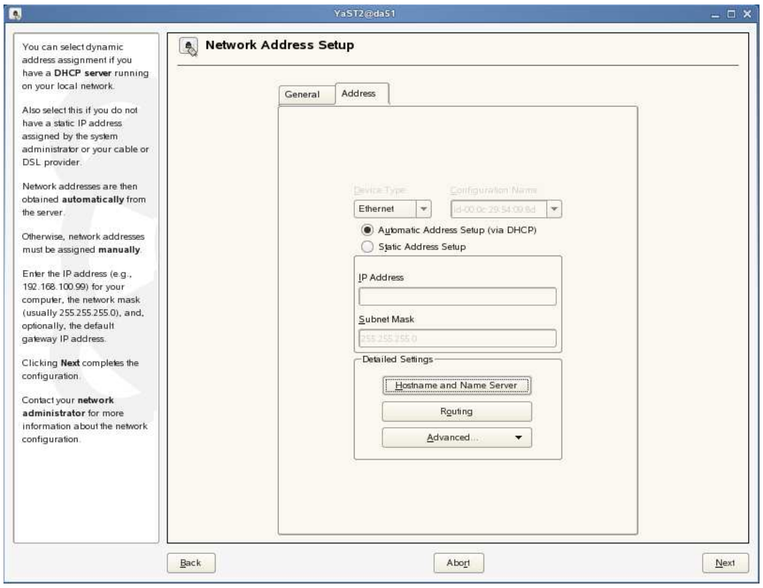
AMD PCnet - Fast 79C971

- Device Name: eth-id-00:0c:29:54:09:8d
- Started automatically at boot
- IP address assigned using DHCP

All detected network cards are listed here.

If you want to use the default settings, select Next. Otherwise, select the card you want to configure; then select Edit.

The Network Address Setup dialog appears.



You can select dynamic address assignment if you have a **DHCP server** running on your local network.

Also select this if you do not have a static IP address assigned by the system administrator or your cable or DSL provider.

Network addresses are then obtained **automatically** from the server.

Otherwise, network addresses must be assigned **manually**.

Enter the IP address (e.g., 192.168.100.99) for your computer, the network mask (usually 255.255.255.0), and, optionally, the default gateway IP address.

Clicking **Next** completes the configuration.

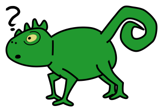
Contact your **network administrator** for more information about the network configuration.

Network Address Setup

Device Type: Ethernet
Configuration Name: id-00:0c:29:54:09:8d

[✓ Automatic Address Setup (via DHCP)]
Static Address Setup

IP Address:
Subnet Mask: 255.255.255.0



In this dialog, enter the following information to integrate the network device into an existing network:

`Automatic Address Setup (via DHCP).` Select this option if the network card should receive an IP address from a DHCP server. This server must be available in the network.

`Static Address Setup.` If you choose this option, you need to enter the IP address of the network interface or of the computer in the network under `IP Address`.

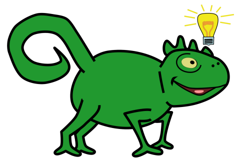
Each computer in the network has at least one address for each network interface, which must be unique in the entire network. According to the currently valid standard (IPv4), this address consists of a sequence of four bytes, separated by dots (such as “10.10.0.69”).

When choosing the IP address, you need to know if the computer will be directly connected to the Internet. In this case, use an assigned official IP address. Otherwise, use an address from a private address space.

`Subnet Mask.` The network mask (referred to as subnet mask in YaST), determines in which network an IP address is located. The mask divides the IP address into a network part and a host part, thus defining the number of possible computers in a network. All computers within the same network can reach each other directly (without a router in between).

`Hostname and Name Server.` Computers in the network can be addressed directly by using their IP addresses or a unique name. A name server (DNS) has to be there to convert names into IP addresses and vice versa.

`Routing.` If the computer should only reach computers in the same subnet, you need not enter any routes.

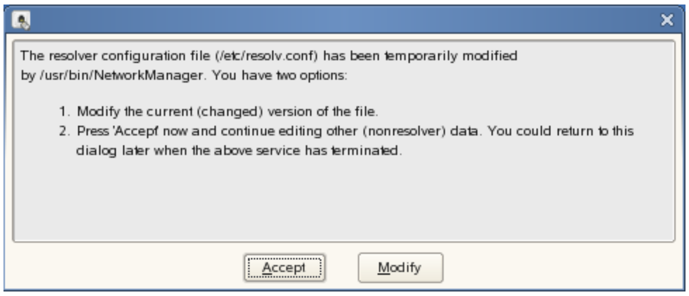


Exercise: Understand IP Addresses

Search the internet for “network class” and “network mask” to understand how an IP address is composed.

1.1.1 Change Hostname Domain and DNS Information

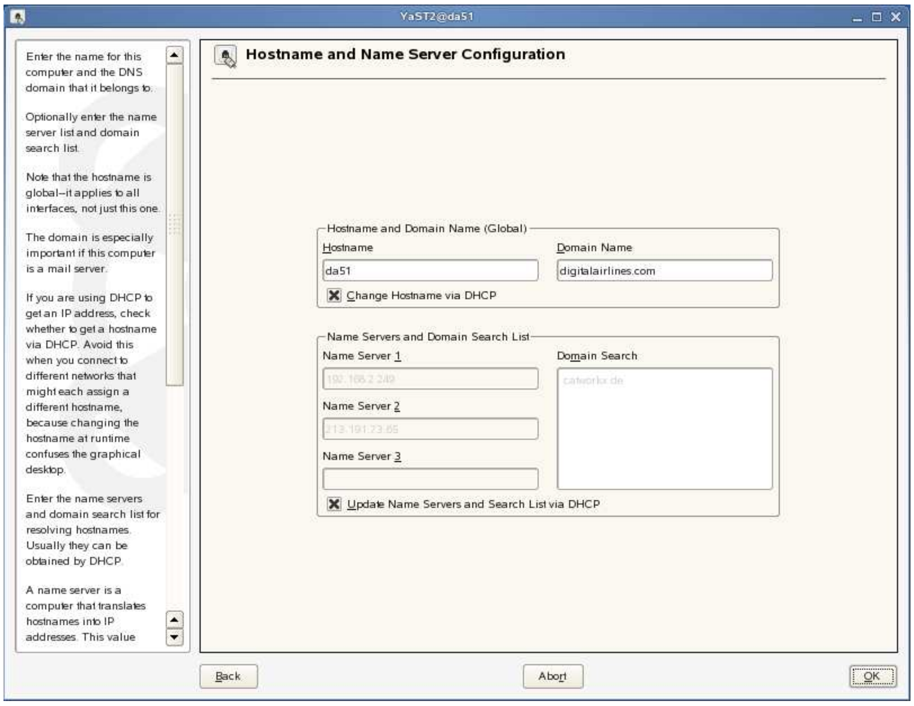
If you are using DHCP and you select Hostname and Name Server, the following appears:


The resolver configuration file (/etc/resolv.conf) has been temporarily modified by /usr/bin/NetworkManager. You have two options:

1. Modify the current (changed) version of the file.
2. Press 'Accept' now and continue editing other (nonresolver) data. You could return to this dialog later when the above service has terminated.

If you want to change data delivered by DHCP (e. g., IP number), select Modify. But if you want to modify other information, select Accept.

The Hostname and Name Server Configuration dialog appears.



Enter the name for this computer and the DNS domain that it belongs to.

Optionally enter the name server list and domain search list.

Note that the hostname is global–it applies to all interfaces, not just this one.

The domain is especially important if this computer is a mail server.

If you are using DHCP to get an IP address, check whether to get a hostname via DHCP. Avoid this when you connect to different networks that might each assign a different hostname, because changing the hostname at runtime confuses the graphical desktop.

Enter the name servers and domain search list for resolving hostnames. Usually they can be obtained by DHCP.

A name server is a computer that translates hostnames into IP addresses. This value

Hostname and Name Server Configuration

Hostname and Domain Name (Global)

Hostname: da51

Domain Name: digitalairlines.com

[✓] Change Hostname via DHCP

Name Servers and Domain Search List

Name Server 1: 192.168.2.249

Name Server 2: 213.191.23.65

Name Server 3:

Domain Search: caworkx.de

[✓] Update Name Servers and Search List via DHCP

In this dialog, you can enter the following:

`Hostname.` Enter a name for the computer. This name should be unique within the network.

`Domain Name.` This is the name of the DNS domain the computer belongs to. Domains help to divide networks. All computers in a defined organizational area normally belong to the same domain.

A computer can be addressed uniquely by giving its FQDN (Fully Qualified Domain Name). This consists of the host name and the domain name, such as da51.digitalairlines.com. In this example, the domain is digitalairlines.com.

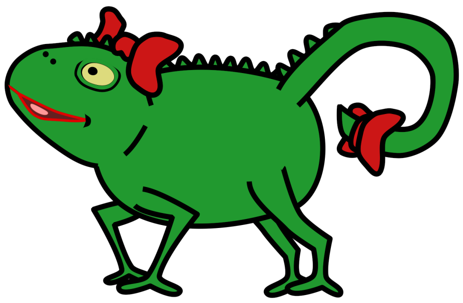
`List of name servers.` To address other computers in the network with their host names, identify the name server, which converts computer names into IP addresses and vice versa.

You can specify a maximum of three name servers.

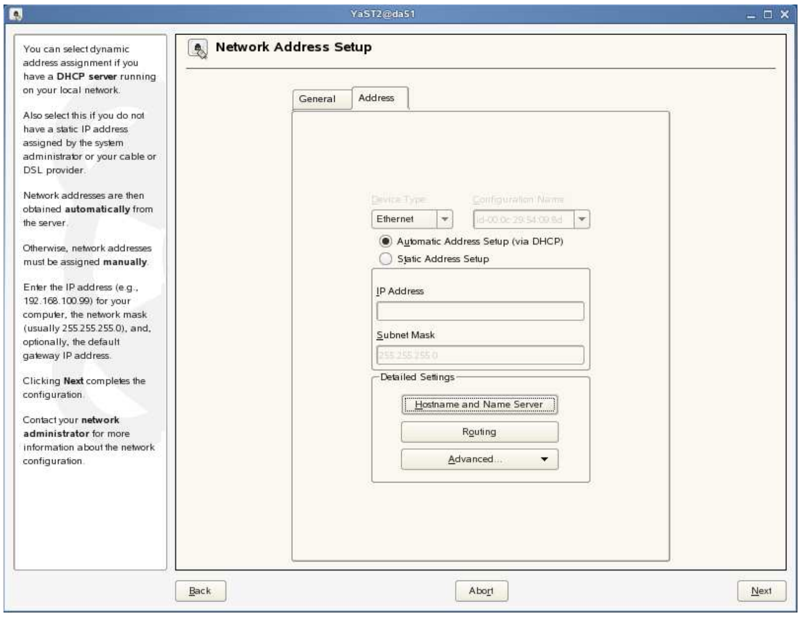
`Domain search list.` In the local network, you can address other hosts with their host names. You do not have to use the FQDN, because the domain search list specifies the domains used by the system to expand the host name to the FQDN.

This FQDN is then passed to the name server to be converted. For example, da51 is completed by using digitalairlines.com to the FQDN da51.digitalairlines.com. Then this name is passed to the name server to be converted.

If the search list contains more than one domain, the completion takes place one after the other. The FQDN is passed to the name server until an entry returns an associated IP address. Separate the domains with a comma or white space.



Select OK to go back to the Network Address Setup dialog.



You can select dynamic address assignment if you have a **DHCP server** running on your local network.

Also select this if you do not have a static IP address assigned by the system administrator or your cable or DSL provider.

Network addresses are then obtained **automatically** from the server.

Otherwise, network addresses must be assigned **manually**.

Enter the IP address (e.g., 192.168.100.99) for your computer, the network mask (usually 255.255.255.0), and, optionally, the default gateway IP address.

Clicking **Next** completes the configuration.

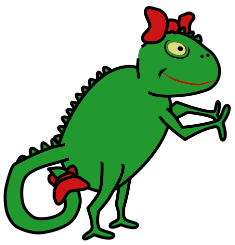
Contact your **network administrator** for more information about the network configuration.

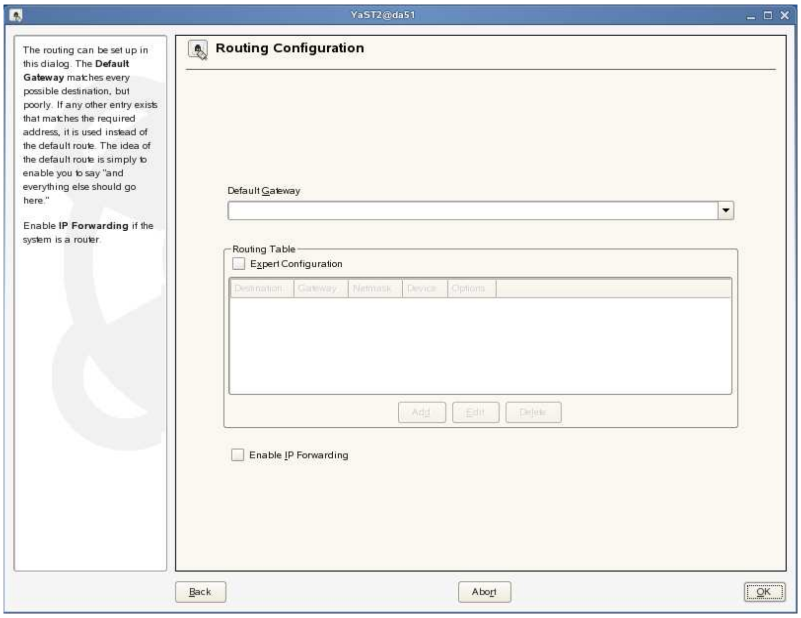
Network Address Setup

Device Type: Ethernet
Configuration Name: id-00:0c:29:54:09:6d
(●) Automatic Address Setup (via DHCP)
( ) Static Address Setup
IP Address:
Subnet Mask: 255.255.255.0

1.1.2 Change Routes

If you need to enter a default gateway (a computer that forwards information from one network to other networks) or create a routing table, select Routing in the Network Address Setup dialog. The following appears:





The routing can be set up in this dialog. The **Default Gateway** matches every possible destination, but poorly. If any other entry exists that matches the required address, it is used instead of the default route. The idea of the default route is simply to enable you to say "and everything else should go here."

Enable **IP Forwarding** if the system is a router.

Routing Configuration

Default Gateway:

Routing Table
[ ] Expert Configuration

Destination	Gateway	Netmask	Device	Options

[ ] Enable IP Forwarding

You can define the following:

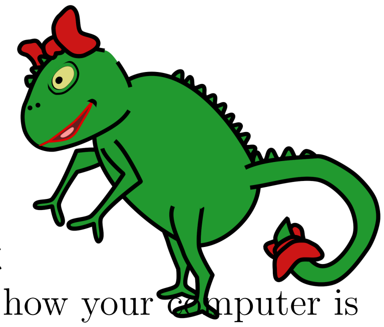
`Default Gateway`. If the network has a gateway, its address can be specified in the network configuration.

All data not addressed to the local network is then forwarded directly to the gateway.

`Routing Table`. You can create entries in the routing table of the system after selecting `Expert Configuration`.

`Enable IP Forwarding`. If you select this option, IP packages that are not dedicated for your computer are routed.

Select OK to go back to the Network Address Setup dialog.



Exercise: Manage the Network

Use the YaST network module to view how your computer is connected to the network.

❶ Does it use DHCP or does it have a static IP address?

____

❷ What is your current IP address?

____

❸ What is your DNS server?

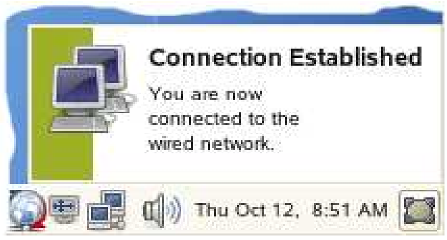
____

❹ What is your gateway?

____

1.1.3 View your Network Settings

After you have saved the configuration with YaST, you can use the ethernet card in your computer. If you are using the NetworkManager method, the NetworkManager applet in GNOME panel informs you when the network connection is established successfully.



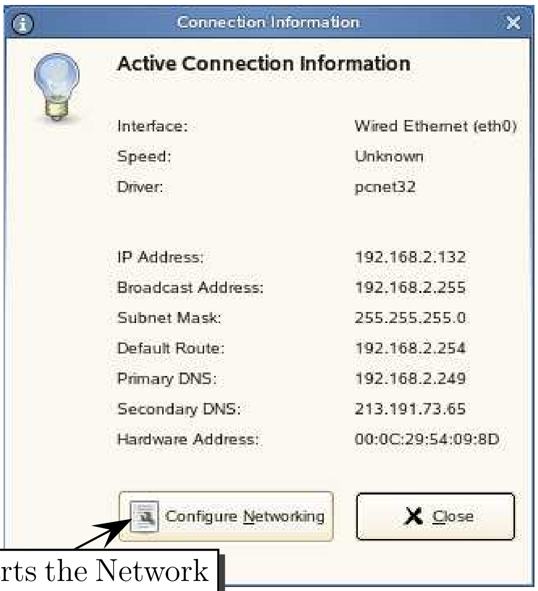
Connection Established

You are now connected to the wired network.

Thu Oct 12, 8:51 AM

To see the data of your network connection, right-click the NetworkManager icon in GNOME panel and select `Connection Information` from the pop-up menu.

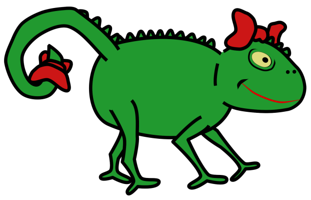
A dialog appears:



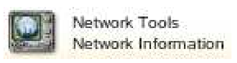
Active Connection Information

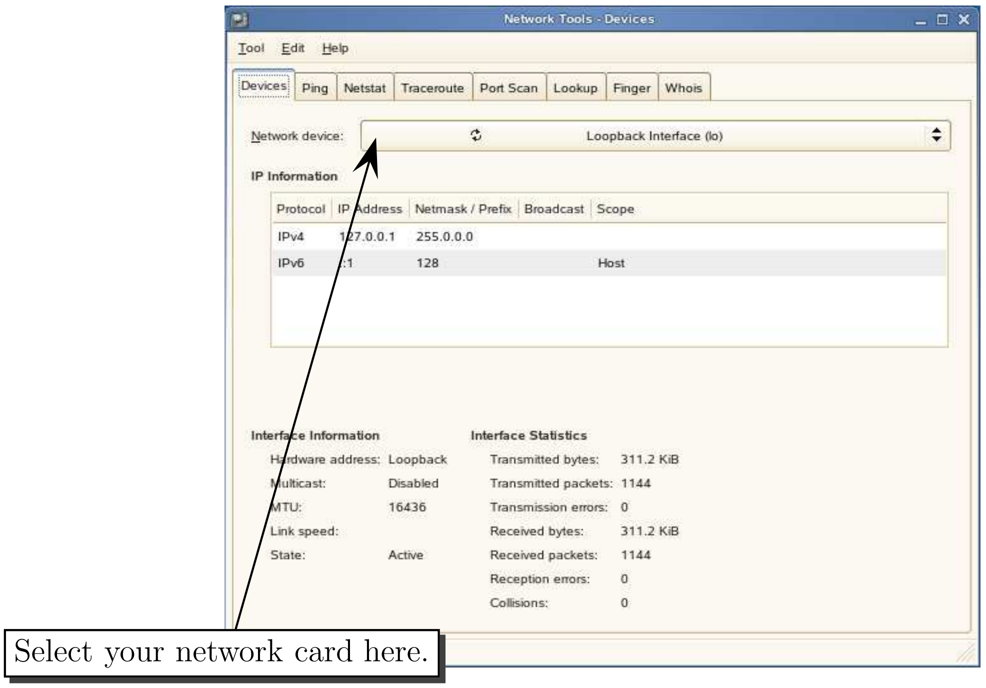
Interface:	Wired Ethernet (eth0)
Speed:	Unknown
Driver:	pcnet32
IP Address:	192.168.2.132
Broadcast Address:	192.168.2.255
Subnet Mask:	255.255.255.0
Default Route:	192.168.2.254
Primary DNS:	192.168.2.249
Secondary DNS:	213.191.73.65
Hardware Address:	00:0C:29:54:09:8D

This button starts the Network Card module of YaST directly.



If you do not use NetworkManager, you can get the data of your network connection by selecting `Network Tools` in the `System` group of the Application Browser.




Network Tools - Devices
Tool Edit Help
Devices Ping Netstat Traceroute Port Scan Lookup Finger Whois
Select your network card here.

Network device: Loopback Interface (lo)

IP Information

Protocol	IP Address	Netmask / Prefix	Broadcast	Scope
IPv4	127.0.0.1	255.0.0.0		
IPv6	::1	128		Host

Interface Information

Hardware address:	Loopback
Multicast:	Disabled
MTU:	16436
Link speed:	
State:	Active

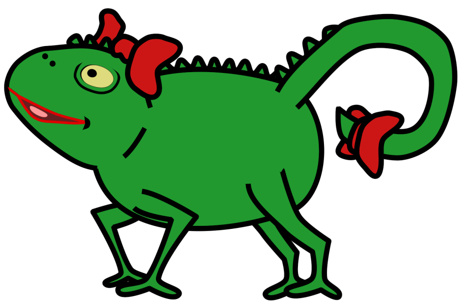
Interface Statistics

Transmitted bytes:	311.2 KiB
Transmitted packets:	1144
Transmission errors:	0
Received bytes:	311.2 KiB
Received packets:	1144
Reception errors:	0
Collisions:	0

The information on your network connection is written in the `Devices` tab. If you have problems with your connection, you can use the other tabs to test your network.

Exercise: View Your Network Settings

Verify the information from the last exercise by using the Network Tools.



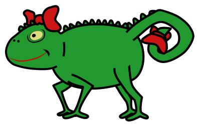
1.2 Manage Users and Groups

A Linux operating system handles several users at the same time (multiuser) and allows these users to perform several tasks on the same computer simultaneously (multitasking).

For this reason, every user has to log in before using the system. To log in, you need the following information:

- A username
- A password

As the operating system can operate numbers much better than strings, users are handled internally as numbers. This number is called UID (User ID).



Every Linux system has a privileged user, called *root*, the administrator of the system. This user always has the UID 0. Users can be grouped according to characteristics or activities. For example:

- Normal users are usually in the group *users*.
- All users who create web pages can be grouped in the group *webedit*.

The file permissions for the directory containing the web pages has to be set in the way the group webedit is able to write (save files).

Each group is also handled by the system with a number, called the GID (Group ID). The following group types are available:

- Normal groups (GID >= 100)
- Groups used by the system (GID < 100)
- The root group (GID = 0)

YaST has a module to manage users and groups. You can access YaST user and group account administration in the following ways:

User administration. From the YaST Control Center, select `Security and Users` → `User Management`.

User Management

Group administration. From the YaST Control Center, select `Security and Users` → `Group Management`.

Group Management

Exercise: Manage Users and Groups

Consider six groups at your school that need different permission.

❶ ________________________________

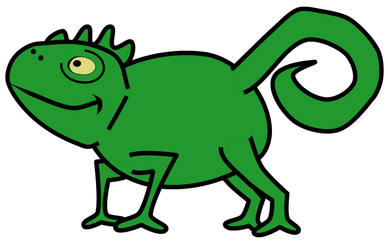
❷ ________________________________

❸ ________________________________

❹ ________________________________

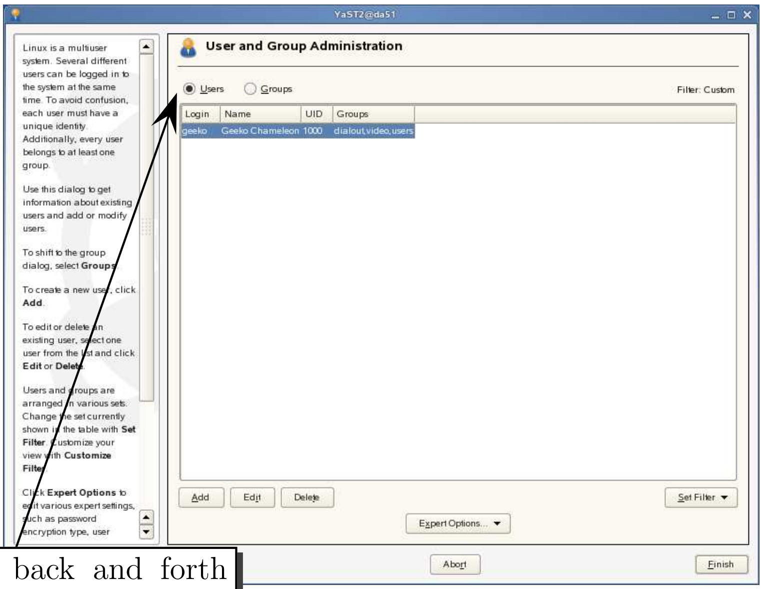
❺ ________________________________

❻ ________________________________



1.2.1 User Administration

The user account management window lists the existing user accounts (as in the following):



Linux is a multiuser system. Several different users can be logged in to the system at the same time. To avoid confusion, each user must have a unique identity. Additionally, every user belongs to at least one group.

Use this dialog to get information about existing users and add or modify users.

To shift to the group dialog, select **Groups**.

To create a new user, click **Add**.

To edit or delete an existing user, select one user from the list and click **Edit** or **Delete**.

Users and groups are arranged in various sets. Change the set currently shown in the table with **Set Filter**. Customize your view with **Customize Filter**.

Click **Expert Options** to edit various expert settings, such as password encryption type, user

Login	Name	UID	Groups
geeko	Geeko Chameleon	1000	dialout,video,users

You can switch back and forth between administering users and administering groups by selecting these radio buttons.

A list of users (accounts on your server) appears with information such as login name, full name, UID, and groups each user belongs to.

If you want to change the users listed, select `Set Filter`; then select one of the following:

`Local Users.` User accounts you have created on your local computer for logging into the computer

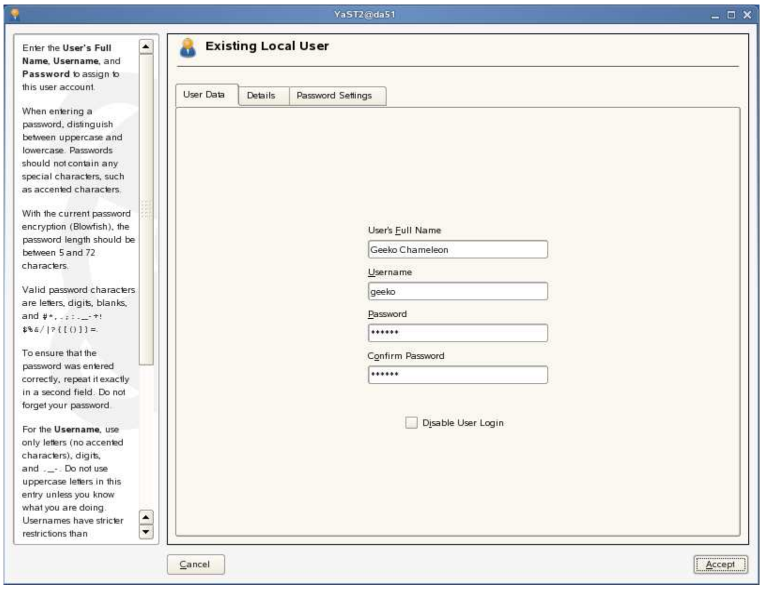
`System Users.` User accounts created by the system to run services

`Custom.` A customized view of users based on the settings configured with `Customize Filter`

`Customize Filter.` This option lets you combine user sets (such as `Local Users` and `System Users`) to display a customized view (with `Custom`) of the user list

Additional sets of users are added to the `Set Filter` drop-down list as you configure and start services on your server.

You can create a new user account or edit an existing account by selecting Add or Edit. The following appears:



Enter the **User's Full Name**, **Username**, and **Password** to assign to this user account.

When entering a password, distinguish between uppercase and lowercase. Passwords should not contain any special characters, such as accented characters.

With the current password encryption (Blowfish), the password length should be between 5 and 72 characters.

Valid password characters are letters, digits, blanks, and #*,.;:._-+! $%&/|?{[()]}=.

To ensure that the password was entered correctly, repeat it exactly in a second field. Do not forget your password.

For the **Username**, use only letters (no accented characters), digits, and ._-. Do not use uppercase letters in this entry unless you know what you are doing. Usernames have stricter restrictions than

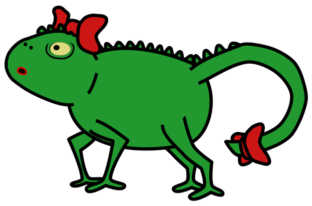
User's Full Name: Geeko Chameleon

Username: geeko

Password: ******

Confirm Password: ******

☐ Disable User Login



Enter or edit information in the following fields:

`User's Full Name.` Enter a user name (such as Geeko Chameleon).

`Username.` Enter a user name that is used to log in to the system (such as geeko).

`Password` and `Confirm Password.` Enter (twice) a password for the user account. When entering a password, distinguish between uppercase and lowercase letters.

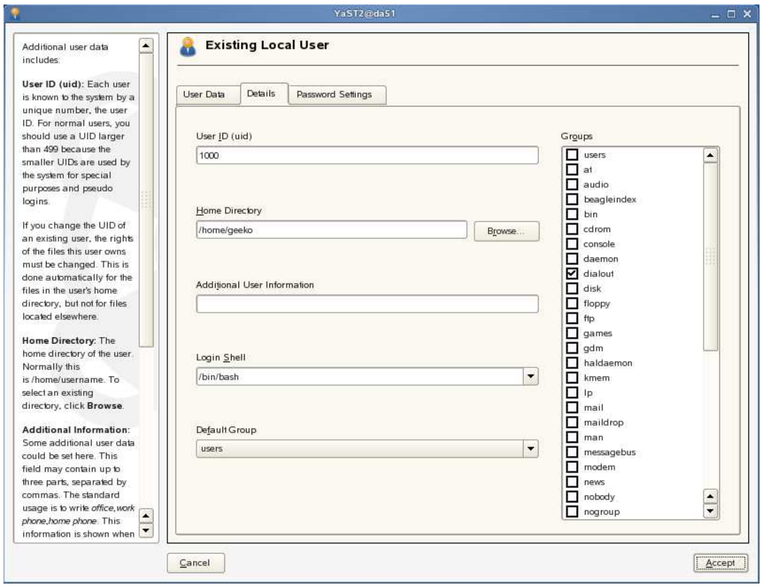
Valid password characters include letters, digits, blanks, and `#*,.;:._-+!$%&/|?{[()]}=`.

The password should not contain any special characters (such as accented characters), as you might find it difficult to type these characters on a different keyboard layout when logging in from another country.

With the current password encryption (Blowfish), the password length should be between 5 and 72 characters.

Disable User Login. If you select this option, the user is not able to login anymore. But the user account is not deleted. This option is useful if you want to disable a user account temporarily.

To set the properties of the user (such as the UID, the home directory, the login shell, group affiliation, and additional user account comments), select the **Details** tab. The following appears:



Additional user data includes:

User ID (uid): Each user is known to the system by a unique number, the user ID. For normal users, you should use a UID larger than 499 because the smaller UIDs are used by the system for special purposes and pseudo logins.

If you change the UID of an existing user, the rights of the files this user owns must be changed. This is done automatically for the files in the user's home directory, but not for files located elsewhere.

Home Directory: The home directory of the user. Normally this is /home/username. To select an existing directory, click **Browse**.

Additional Information: Some additional user data could be set here. This field may contain up to three parts, separated by commas. The standard usage is to write *office,work phone,home phone*. This information is shown when

User ID (uid): 1000

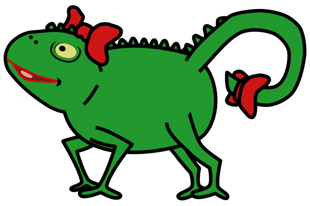
Home Directory: /home/geeko

Additional User Information:

Login Shell: /bin/bash

Default Group: users

Groups:
- [ ] users
- [ ] at
- [ ] audio
- [ ] beagleindex
- [ ] bin
- [ ] cdrom
- [ ] console
- [ ] daemon
- [✓] dialout
- [ ] disk
- [ ] floppy
- [ ] ftp
- [ ] games
- [ ] gdm
- [ ] haldaemon
- [ ] kmem
- [ ] lp
- [ ] mail
- [ ] maildrop
- [ ] man
- [ ] messagebus
- [ ] modem
- [ ] news
- [ ] nobody
- [ ] nogroup



Enter or edit information in the following fields:

User ID (uid). For normal users, you should use a UID greater than 999, because the lower UIDs are used by the system for special purposes and pseudo logins.

If you change the UID of an existing user, the permission of the files of this user must be changed, too. This is done automatically for the files in the user's home directory, but not for files located elsewhere.

Home Directory. The home directory of the user. Normally this is `/home/`*`username`*.

You can select an existing directory by selecting Browse... .

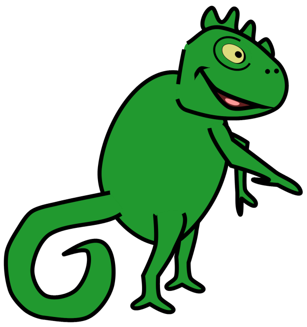
Additional User Information. This field contains up to three parts separated by commas. It is often used to enter such information as office, work phone, or home phone.

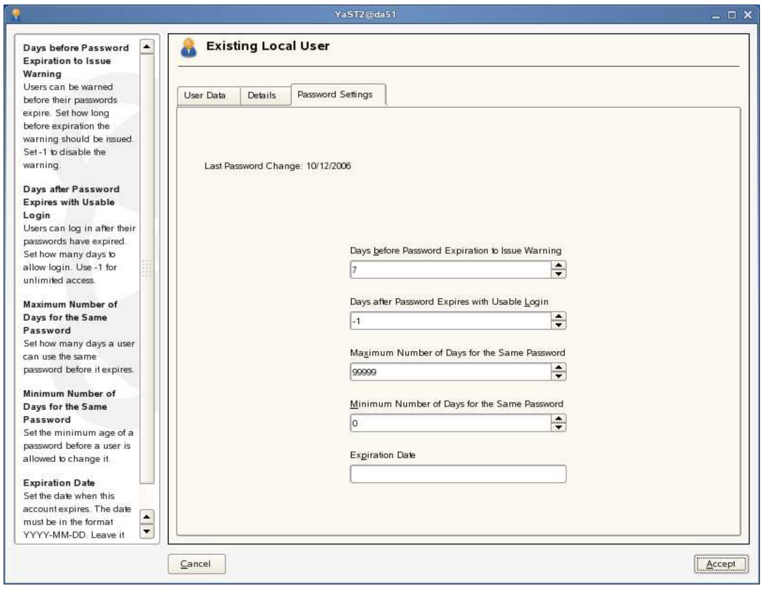
`Login Shell`**.** From the drop-down list, select the default login shell for this user.

`Default Group`**.** This is the group the user belongs to. Select a group from the list of all groups configured on your system.

`Groups`**.** Select all additional memberships you want to assign to the user.

To set various password parameters (such as duration of a password), select the `Password Settings` tab. The following appears:





Days before Password Expiration to Issue Warning
Users can be warned before their passwords expire. Set how long before expiration the warning should be issued. Set -1 to disable the warning.

Days after Password Expires with Usable Login
Users can log in after their passwords have expired. Set how many days to allow login. Use -1 for unlimited access.

Maximum Number of Days for the Same Password
Set how many days a user can use the same password before it expires.

Minimum Number of Days for the Same Password
Set the minimum age of a password before a user is allowed to change it.

Expiration Date
Set the date when this account expires. The date must be in the format YYYY-MM-DD. Leave it

Existing Local User

Last Password Change: 10/12/2006

Days before Password Expiration to Issue Warning: 7

Days after Password Expires with Usable Login: -1

Maximum Number of Days for the Same Password: 99999

Minimum Number of Days for the Same Password: 0

Expiration Date:

Enter or edit information in the following fields:

`Days before Password Expiration to Issue Warning`**.** Enter the number of days before the password expires that the user is informed about it.

Enter `-1` to disable the warning.

`Days after Password Expires with Usable Login`**.** Enter the number of days after the password expires that users can continue to log in.

Enter `-1` for unlimited access.

`Maximum number of days for the same password`**.** Enter the number of days a user can use the same password before it expires.

`Minimum number of days for the same password`**.** Enter the minimum age of a password before a user can change it.

`Expiration date.` Enter the date when the account expires. The format is *`YYYY-MM-DD`*.

Leave the field empty if the account never expires.

Save the settings for the new or edited user by selecting Accept.

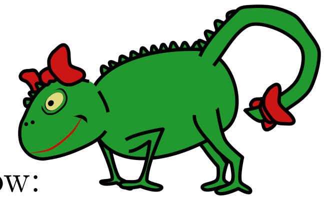
A new user appears in the list.

Configure your server with the new settings by selecting Finish.

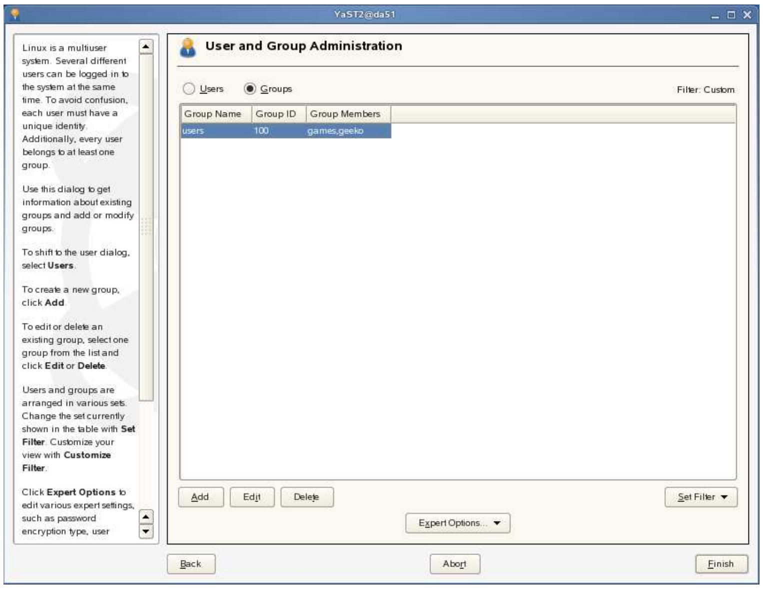
Exercise: Create a New User

Create a new user account on your local computer. This account is needed for exercises later in this book.

1.2.2 Administer Groups



You can administer groups from the following window:



YaST2@da51

User and Group Administration

Linux is a multiuser system. Several different users can be logged in to the system at the same time. To avoid confusion, each user must have a unique identity. Additionally, every user belongs to at least one group.

Use this dialog to get information about existing groups and add or modify groups.

To shift to the user dialog, select **Users**.

To create a new group, click **Add**.

To edit or delete an existing group, select one group from the list and click **Edit** or **Delete**.

Users and groups are arranged in various sets. Change the set currently shown in the table with **Set Filter**. Customize your view with **Customize Filter**.

Click **Expert Options** to edit various expert settings, such as password encryption type, user

Users [✓ Groups]   Filter: Custom

Group Name	Group ID	Group Members
users	100	games,geeko

A list of groups appears with information such as Group Name, Group ID (GID), and Group Members.

To change the groups listed, select `Set Filter`; then select one of the following:

`Local Groups.` Groups created on your local server to provide permissions for members assigned to the group.

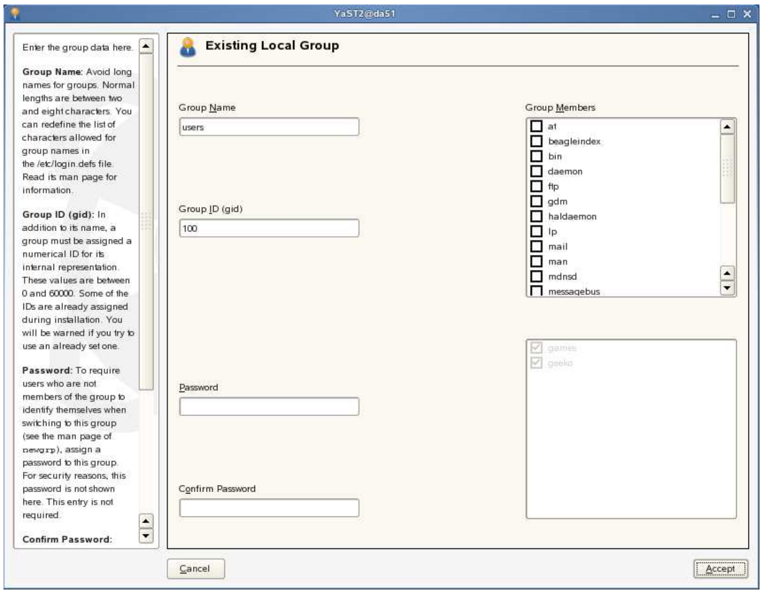
System Groups. Groups created by the system to run services.

Custom. A customized view of groups based on the settings configured with `Customize Filter`.

Customize Filter. This option lets you combine group sets (such as `Local Groups` and `System Groups`) to display a customized view (with `Custom`) of the groups list.

Additional sets of groups are added to the `Set Filter` drop-down list when you configure and start services on your server.

You can create a new group or edit an existing group by selecting Add or Edit. If you select Edit, the following dialog appears.



Enter the group data here.

Group Name: Avoid long names for groups. Normal lengths are between two and eight characters. You can redefine the list of characters allowed for group names in the /etc/login.defs file. Read its man page for information.

Group ID (gid): In addition to its name, a group must be assigned a numerical ID for its internal representation. These values are between 0 and 60000. Some of the IDs are already assigned during installation. You will be warned if you try to use an already set one.

Password: To require users who are not members of the group to identify themselves when switching to this group (see the man page of `newgrp`), assign a password to this group. For security reasons, this password is not shown here. This entry is not required.

Confirm Password:

Existing Local Group

Group Name: users

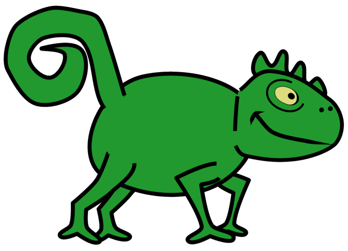
Group ID (gid): 100

Password:

Confirm Password:

Group Members:
- ☐ at
- ☐ beagleindex
- ☐ bin
- ☐ daemon
- ☐ ftp
- ☐ gdm
- ☐ haldaemon
- ☐ lp
- ☐ mail
- ☐ man
- ☐ mdnsd
- ☐ messagebus

- ✓ games
- ✓ geeko



The fields allow you to enter or edit information:

`Group Name`. The name of the group. Avoid long names. Default name lengths are between two and eight characters.

`Group ID (gid)`. The number assigned to the group. The number must be between 0 and 60000. GIDs up to 99 represent system groups. GIDs beyond 99 can be used for normal groups. YaST warns you if you try to create a GID that already exists.

`Password`. (Optional) If you want the members of the group to identify themselves while switching to this group (see `man newgrp`), assign a password.

For security reasons, the password is represented by dots.

`Confirm Password`. Enter the password once again to avoid typing errors.

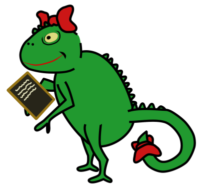
`Group Members`. Select which user should be a group member.

A second list may appear (if you select Edit). It shows for which users this group is the primary group. This list cannot be edited by YaST.

After you have finished entering or editing the group information, select Accept. You are returned to the Group Administration dialog. Save the configuration settings by selecting Finish.

The information you enter when creating or editing users and groups with YaST is saved in the following user administration files:

- /etc/passwd
- /etc/shadow
- /etc/group



Exercise: Administer Groups

How many local groups and system groups are available on your computer?

1.3 Install and Remove Software

After performing a standard SUSE Linux Enterprise Desktop 10 installation, you will often have to install additional software.

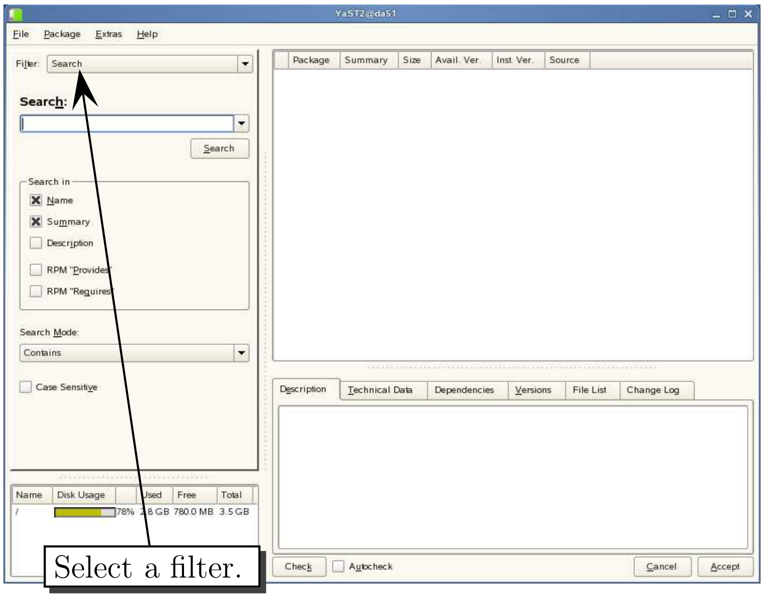
RPM (*RPM Package Management*) is the default file type for software to install. It contains the software and a lot of information (e. g., dependencies to other software packages).

Start the YaST module `Software` → `Software Management`.

Software Management

The installed packages and the packages available on the installation media are analyzed, and dependencies between packages are checked.

After this, the following dialog for searching packages appears:



Filter: Search

Search in: [✓ Name] [✓ Summary] Description, RPM "Provides", RPM "Requires"

Search Mode: Contains

Case Sensitive

Autocheck

Name	Disk Usage		Used	Free	Total
/		78%	2.8 GB	780.0 MB	3.5 GB

To find the software you want to install, you can select a filter listed in the drop-down list labeled `Filter` in the top left corner of the window. The following filters are available:

`Patterns`. Displays all software that is available on the known installation media. It is grouped in predefined sets of packages that logically belong together.

`Package Groups`. Displays all software that is available on the known installation media. It is grouped by category.

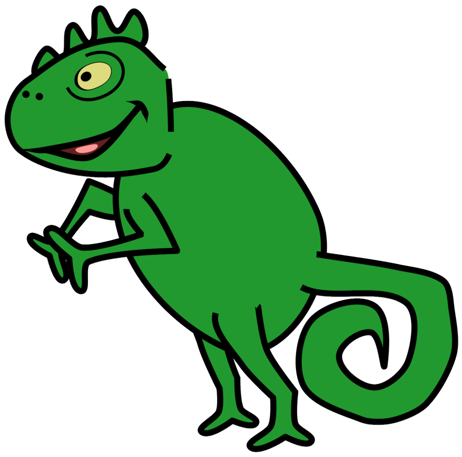
`Languages`. Displays all language related files (e. g., spell checker, help)

`Installation Sources`. Lists all registrated installation sources and displays the available packages of these sources.

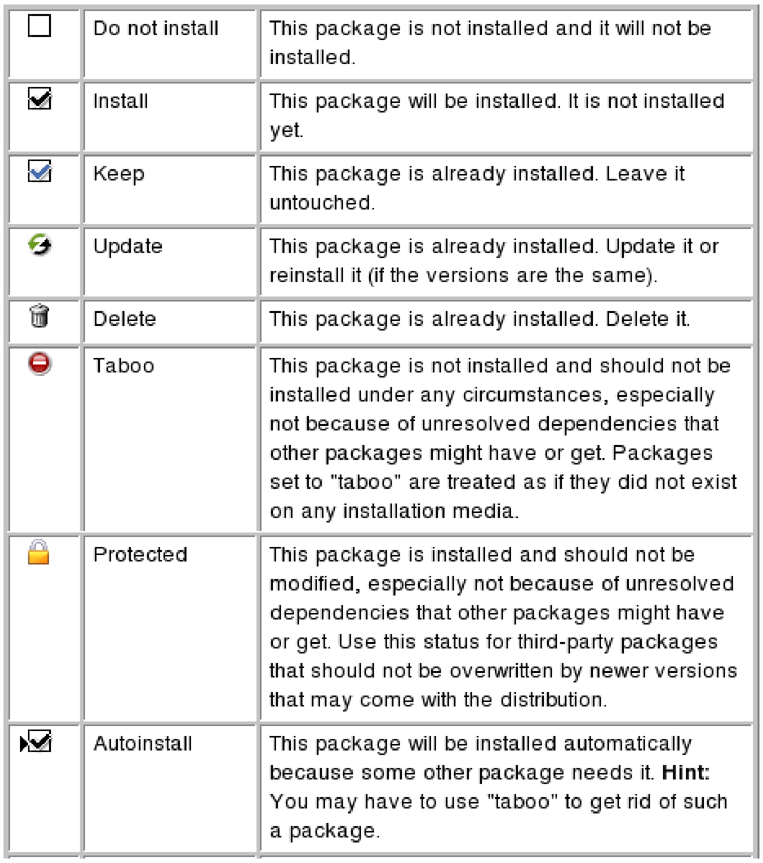
`Search`. Enter a search term and define where you want YaST to search for the software package.

`Installation Summary`. Displays all the packages with a marked status.

To find a package, select the `Search` filter and enter the package name, parts of the package name, or some keywords in the `Search` field; then select Search.



The matched packages are listed in the area on the right side. The installation state is shown by a small symbol in front of the package name. The most commonly used symbols include the following:



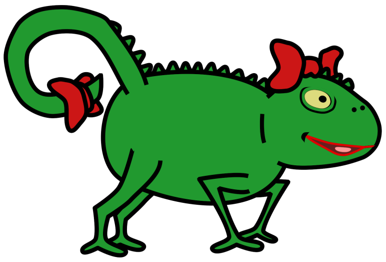
☐	Do not install	This package is not installed and it will not be installed.
☑	Install	This package will be installed. It is not installed yet.
☑	Keep	This package is already installed. Leave it untouched.
⟳	Update	This package is already installed. Update it or reinstall it (if the versions are the same).
🗑	Delete	This package is already installed. Delete it.
⛔	Taboo	This package is not installed and should not be installed under any circumstances, especially not because of unresolved dependencies that other packages might have or get. Packages set to "taboo" are treated as if they did not exist on any installation media.
🔒	Protected	This package is installed and should not be modified, especially not because of unresolved dependencies that other packages might have or get. Use this status for third-party packages that should not be overwritten by newer versions that may come with the distribution.
▶☑	Autoinstall	This package will be installed automatically because some other package needs it. **Hint:** You may have to use "taboo" to get rid of such a package.

To view a list of all possible symbols, select `Help` → `Symbols`.

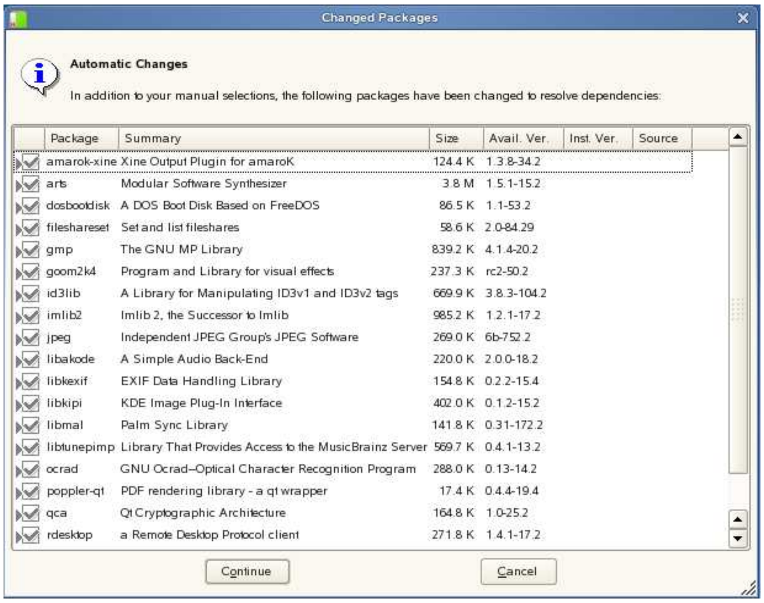
Select the symbol of the package you want to install several times until the install symbol appears; then select Accept.

Exercise: Install Software – Part I

Install the package `findutils-locate`. This tool is introduced later in this book.



You might see a dialog indicating that dependencies between the packages cannot be solved and some other packages need to be installed, too. In most cases, you can confirm this dialog.



Changed Packages

Automatic Changes

In addition to your manual selections, the following packages have been changed to resolve dependencies:

	Package	Summary	Size	Avail. Ver.	Inst. Ver.	Source
✓	amarok-xine	Xine Output Plugin for amaroK	124.4 K	1.3.8-34.2		
✓	arts	Modular Software Synthesizer	3.8 M	1.5.1-15.2		
✓	dosbootdisk	A DOS Boot Disk Based on FreeDOS	86.5 K	1.1-53.2		
✓	fileshareset	Set and list fileshares	58.6 K	2.0-84.29		
✓	gmp	The GNU MP Library	839.2 K	4.1.4-20.2		
✓	goom2k4	Program and Library for visual effects	237.3 K	rc2-50.2		
✓	id3lib	A Library for Manipulating ID3v1 and ID3v2 tags	669.9 K	3.8.3-104.2		
✓	imlib2	Imlib 2, the Successor to Imlib	985.2 K	1.2.1-17.2		
✓	jpeg	Independent JPEG Group's JPEG Software	269.0 K	6b-752.2		
✓	libakode	A Simple Audio Back-End	220.0 K	2.0.0-18.2		
✓	libkexif	EXIF Data Handling Library	154.8 K	0.2.2-15.4		
✓	libkipi	KDE Image Plug-In Interface	402.0 K	0.1.2-15.2		
✓	libmal	Palm Sync Library	141.8 K	0.31-172.2		
✓	libtunepimp	Library That Provides Access to the MusicBrainz Server	569.7 K	0.4.1-13.2		
✓	ocrad	GNU Ocrad–Optical Character Recognition Program	288.0 K	0.13-14.2		
✓	poppler-qt	PDF rendering library - a qt wrapper	17.4 K	0.4.4-19.4		
✓	qca	Qt Cryptographic Architecture	164.8 K	1.0-25.2		
✓	rdesktop	a Remote Desktop Protocol client	271.8 K	1.4.1-17.2		

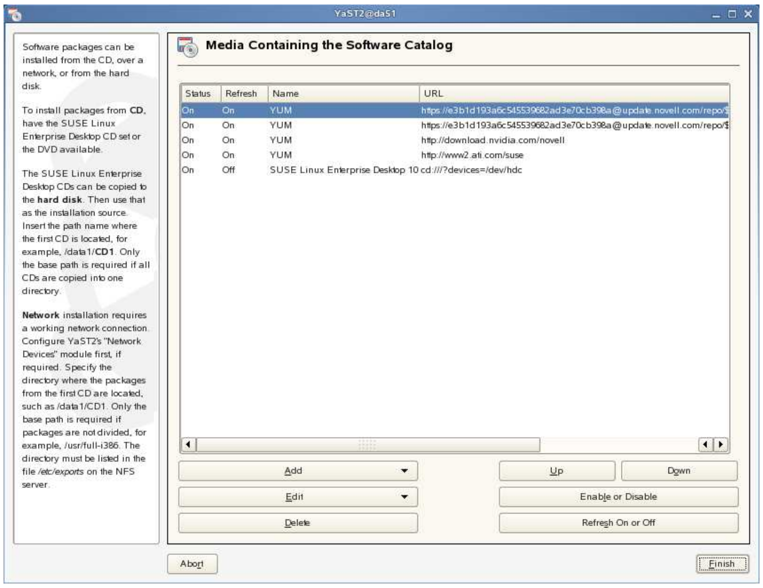
Exercise: Install Software – Part II

Install the package `ksh`. This is a Linux shell like the bash and introduced later in this book.

If the wrong CD or DVD is in your drive, a warning appears.

The software installation dialog lists only the packages that are on the current installation media. If you want to add more installation sources, select `Software` → `Installation Source` from the YaST Control Center.

The following appears:



Software packages can be installed from the CD, over a network, or from the hard disk.

To install packages from **CD**, have the SUSE Linux Enterprise Desktop CD set or the DVD available.

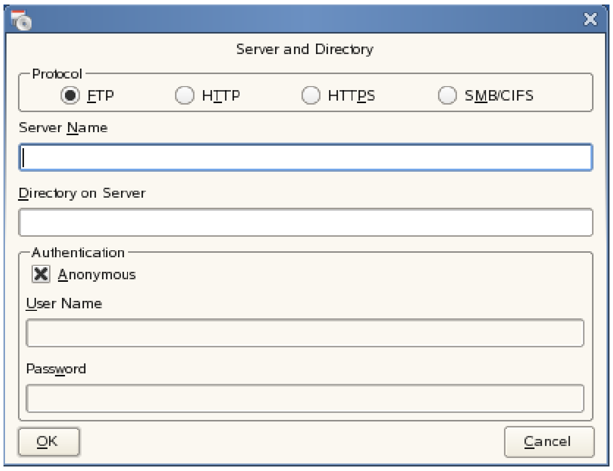
The SUSE Linux Enterprise Desktop CDs can be copied to the **hard disk**. Then use that as the installation source. Insert the path name where the first CD is located, for example, /data1/**CD1**. Only the base path is required if all CDs are copied into one directory.

Network installation requires a working network connection. Configure YaST2's "Network Devices" module first, if required. Specify the directory where the packages from the first CD are located, such as /data1/CD1. Only the base path is required if packages are not divided, for example, /usr/full-i386. The directory must be listed in the file */etc/exports* on the NFS server.

Status	Refresh	Name	URL
On	On	YUM	https://e3b1d193a6c545539682ad3e70cb398a@update.novell.com/repo/$
On	On	YUM	https://e3b1d193a6c545539682ad3e70cb398a@update.novell.com/repo/$
On	On	YUM	http://download.nvidia.com/novell
On	On	YUM	http://www2.ati.com/suse
On	Off	SUSE Linux Enterprise Desktop 10	cd:///?devices=/dev/hdc

To add a new source, select the `Add` drop-down list and select the type of installation source.

Depending on the type of source, you might have to provide additional information (such as the IP address of an installation server). For example:



To edit the configuration of an existing installation source, select the source in the list; then select `Edit`.

If you want to disable an installation source temporarily, select the source in the list; then select `Enable` or `Disable`.

To remove an installation source permanently, select the source from the list; then select `Delete`.

YaST uses the first installation source in the list that has the software package you want to install. To change the order of a source in the list, select the source and `Up` or `Down`.

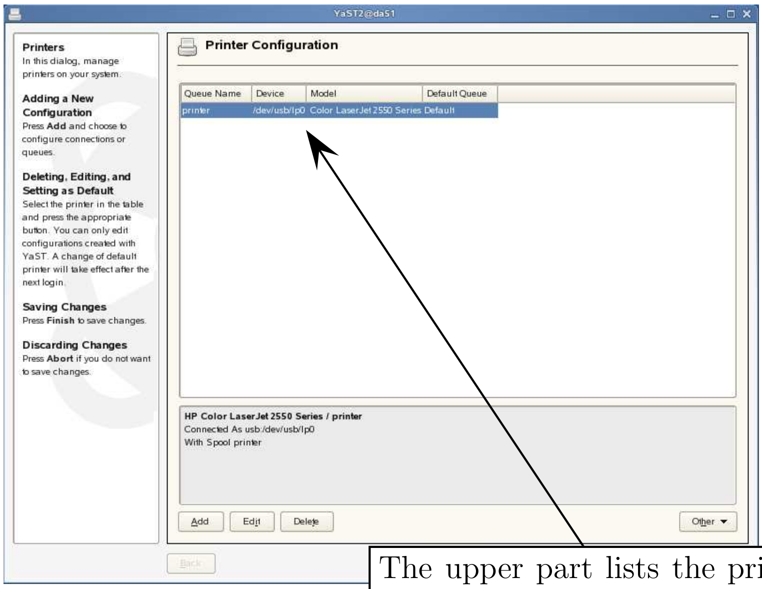
1.4 Configure a Printer

The software that manages your printers and print jobs is called CUPS (*Common Unix Printing System*). It is installed by default.

YaST provides a module for printer installation and configuration functionality. The connected printers are detected and configured automatically during the installation. You can start the module manually by selecting `Hardware` → `Printer`.

Printer

The Printer Configuration dialog appears:



Printers
In this dialog, manage printers on your system.

Adding a New Configuration
Press **Add** and choose to configure connections or queues.

Deleting, Editing, and Setting as Default
Select the printer in the table and press the appropriate button. You can only edit configurations created with YaST. A change of default printer will take effect after the next login.

Saving Changes
Press **Finish** to save changes.

Discarding Changes
Press **Abort** if you do not want to save changes.

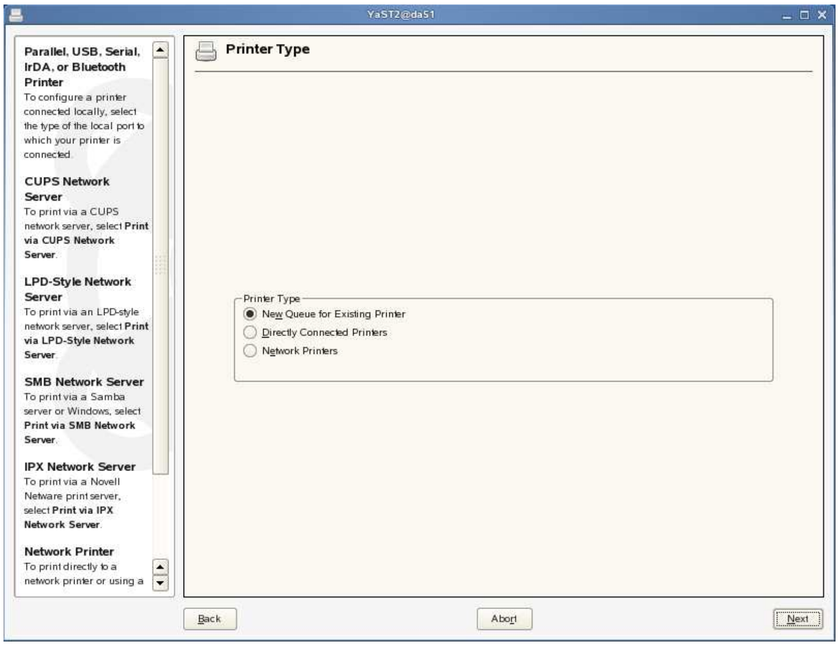
Queue Name	Device	Model	Default Queue
printer	/dev/usb/lp0	Color LaserJet 2550 Series	Default

HP Color LaserJet 2550 Series / printer
Connected As usb:/dev/usb/lp0
With Spool printer

The upper part lists the printers that have already been configured and al the automatically detected printers.

If there are printers listed in the upper part, the lower part of the dialog shows details of the selected printer.

To add a printer that is not listed in the upper part of the dialog (e. g., a network printer), select Add. The following appears:



YaST2@da51

Parallel, USB, Serial, IrDA, or Bluetooth Printer
To configure a printer connected locally, select the type of the local port to which your printer is connected.

CUPS Network Server
To print via a CUPS network server, select **Print via CUPS Network Server**.

LPD-Style Network Server
To print via an LPD-style network server, select **Print via LPD-Style Network Server**.

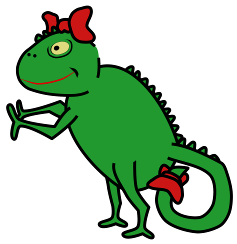
SMB Network Server
To print via a Samba server or Windows, select **Print via SMB Network Server**.

IPX Network Server
To print via a Novell Netware print server, select **Print via IPX Network Server**.

Network Printer
To print directly to a network printer or using a

Printer Type

Printer Type
[✓ New Queue for Existing Printer]
Directly Connected Printers
Network Printers



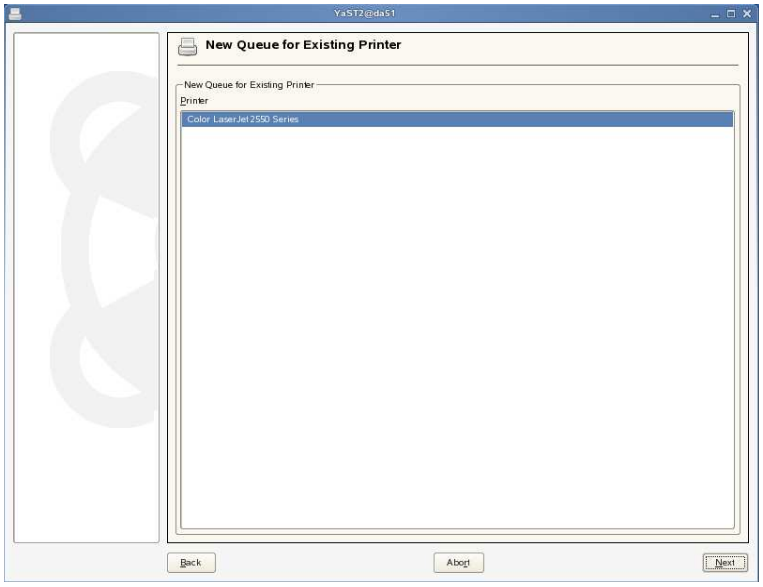
Depending on your selection here, the next dialog offers more specific option. The possible selections are:

- `New Queue for Existing Printer`
- `Directly Connected Printers`
- `Network Printers`

1.4.1 New Queue for Existing Printer

This option appears if there is already a printer configured. Adding a queue to an existing printer is useful if you want to use, for instance, a different resolution or print quality by printing to a different queue.

Selecting `New Queue for Existing Printer` and pressing Next opens a dialog where you select the printer you want to add a queue.



New Queue for Existing Printer

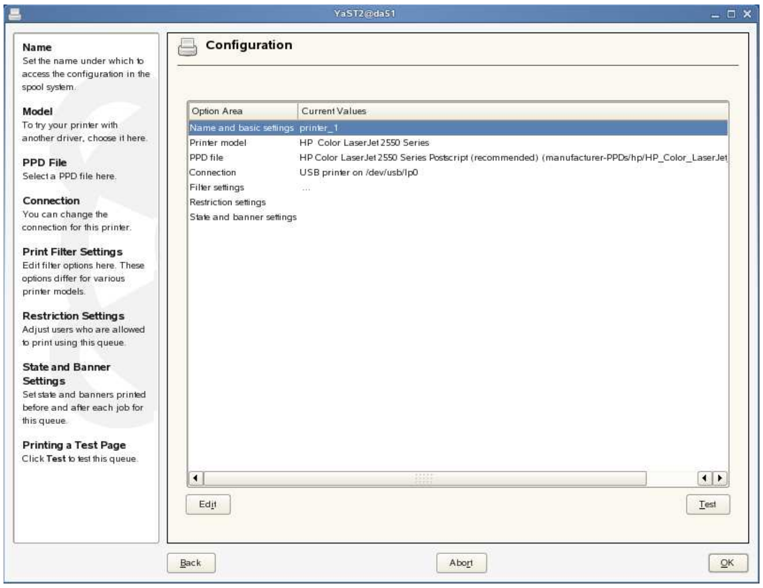
New Queue for Existing Printer

Printer

Color LaserJet 2550 Series

Select a printer; then select Next.

In the following Configuraton dialog, select an entry; then Edit.



Name
Set the name under which to access the configuration in the spool system.

Model
To try your printer with another driver, choose it here.

PPD File
Select a PPD file here.

Connection
You can change the connection for this printer.

Print Filter Settings
Edit filter options here. These options differ for various printer models.

Restriction Settings
Adjust users who are allowed to print using this queue.

State and Banner Settings
Set state and banners printed before and after each job for this queue.

Printing a Test Page
Click **Test** to test this queue.

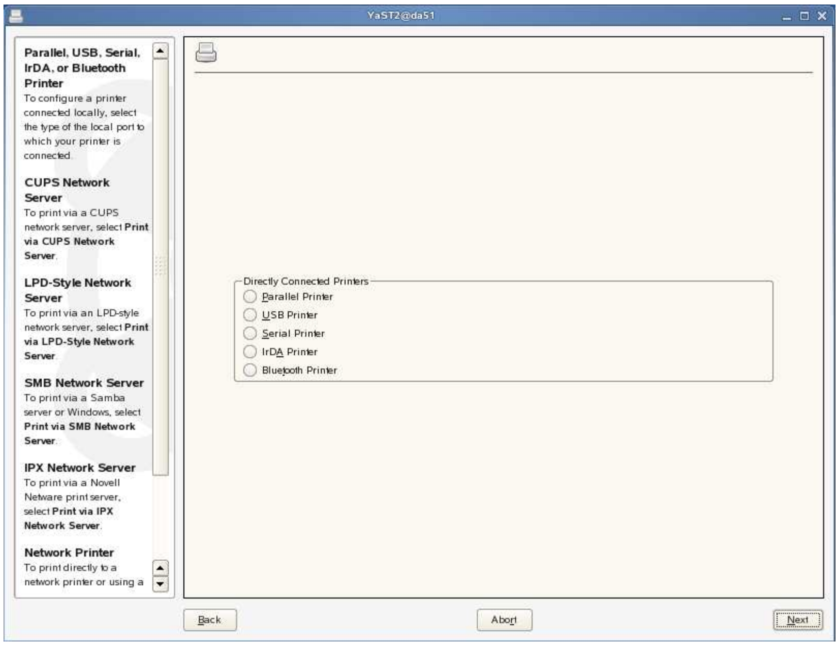
Configuration

Option Area	Current Values
Name and basic settings	printer_1
Printer model	HP Color LaserJet 2550 Series
PPD file	HP Color LaserJet 2550 Series Postscript (recommended) (manufacturer-PPDs/hp/HP_Color_LaserJet
Connection	USB printer on /dev/usb/lp0
Filter settings	...
Restriction settings	
State and banner settings	

Define the queue according to your needs. You can test your printer configuration by selecting Test. OK creates the queue and returns you to the Printer Configuration dialog.

1.4.2 Directly Connected Printers

Most directly connected printers are detected automatically. If not, select the connection type:



Parallel, USB, Serial, IrDA, or Bluetooth Printer
To configure a printer connected locally, select the type of the local port to which your printer is connected.

CUPS Network Server
To print via a CUPS network server, select **Print via CUPS Network Server**.

LPD-Style Network Server
To print via an LPD-style network server, select **Print via LPD-Style Network Server**.

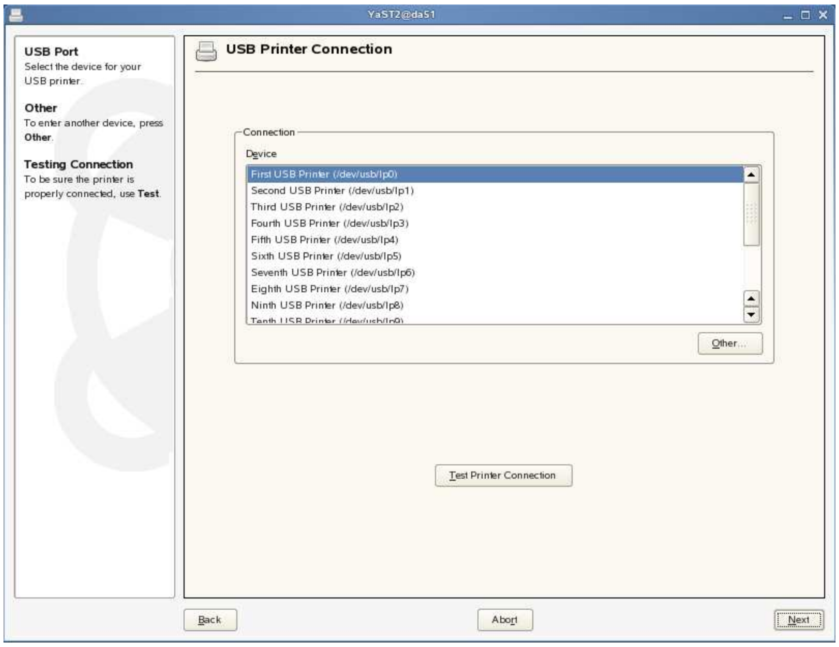
SMB Network Server
To print via a Samba server or Windows, select **Print via SMB Network Server**.

IPX Network Server
To print via a Novell Netware print server, select **Print via IPX Network Server**.

Network Printer
To print directly to a network printer or using a

Directly Connected Printers
- ◯ Parallel Printer
- ◯ USB Printer
- ◯ Serial Printer
- ◯ IrDA Printer
- ◯ Bluetooth Printer

In the next dialog select the interface your printer is connected with.



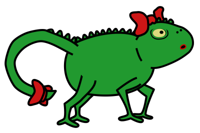
USB Port
Select the device for your USB printer.

Other
To enter another device, press **Other**.

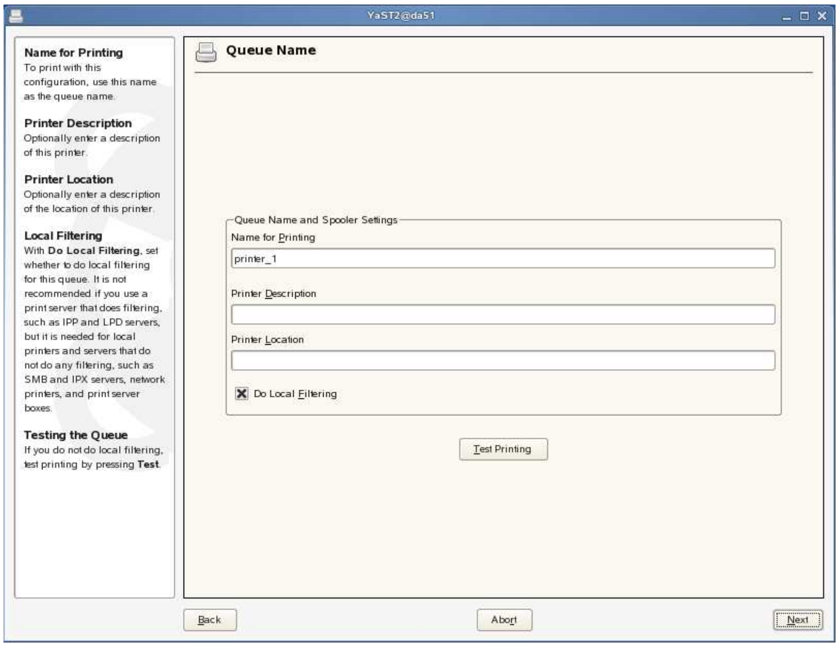
Testing Connection
To be sure the printer is properly connected, use **Test**.

USB Printer Connection

Connection
Device
- First USB Printer (/dev/usb/lp0)
- Second USB Printer (/dev/usb/lp1)
- Third USB Printer (/dev/usb/lp2)
- Fourth USB Printer (/dev/usb/lp3)
- Fifth USB Printer (/dev/usb/lp4)
- Sixth USB Printer (/dev/usb/lp5)
- Seventh USB Printer (/dev/usb/lp6)
- Eighth USB Printer (/dev/usb/lp7)
- Ninth USB Printer (/dev/usb/lp8)
- Tenth USB Printer (/dev/usb/lp9)



After selecting Next, enter a name for the new print queue. You can also enter a description of the printer and the printer's location.



Name for Printing
To print with this configuration, use this name as the queue name.

Printer Description
Optionally enter a description of this printer.

Printer Location
Optionally enter a description of the location of this printer.

Local Filtering
With **Do Local Filtering**, set whether to do local filtering for this queue. It is not recommended if you use a print server that does filtering, such as IPP and LPD servers, but it is needed for local printers and servers that do not do any filtering, such as SMB and IPX servers, network printers, and print server boxes.

Testing the Queue
If you do not do local filtering, test printing by pressing **Test**.

Queue Name

Queue Name and Spooler Settings

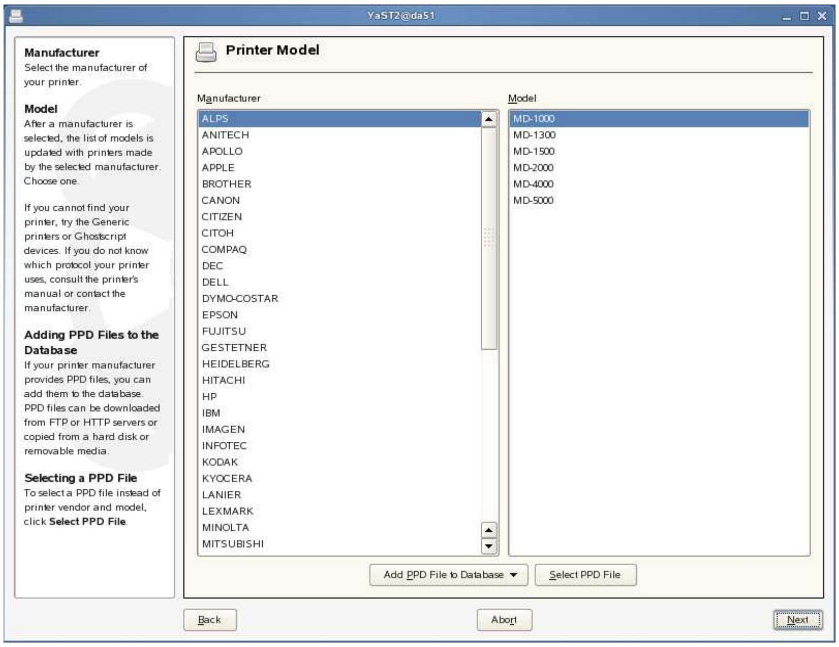
Name for Printing: printer_1

Printer Description:

Printer Location:

[✓ Do Local Filtering]

In the next dialog, select the printer model. First, in the left frame, select a manufacturer; then, in the right frame, select a model.



Manufacturer
Select the manufacturer of your printer.

Model
After a manufacturer is selected, the list of models is updated with printers made by the selected manufacturer. Choose one.

If you cannot find your printer, try the Generic printers or Ghostscript devices. If you do not know which protocol your printer uses, consult the printer's manual or contact the manufacturer.

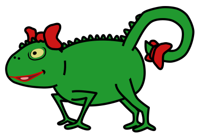
Adding PPD Files to the Database
If your printer manufacturer provides PPD files, you can add them to the database. PPD files can be downloaded from FTP or HTTP servers or copied from a hard disk or removable media.

Selecting a PPD File
To select a PPD file instead of printer vendor and model, click **Select PPD File**.

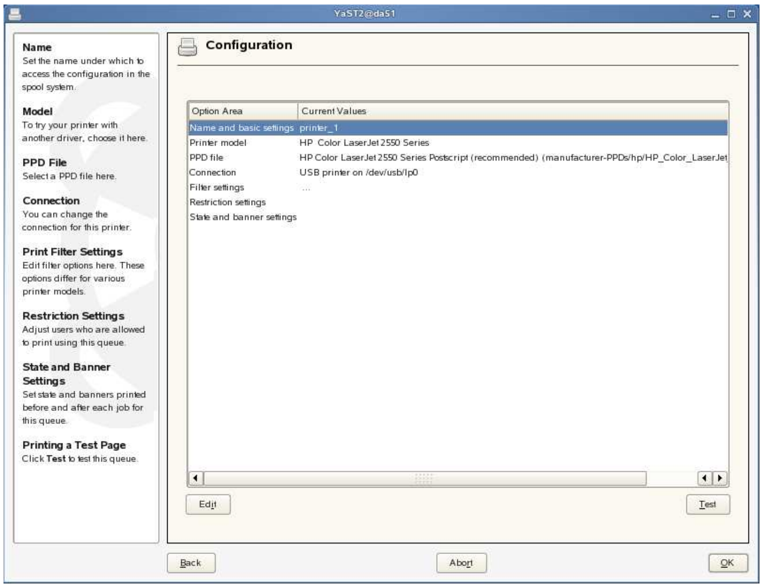
Printer Model

Manufacturer: [✓ ALPS], ANITECH, APOLLO, APPLE, BROTHER, CANON, CITIZEN, CITOH, COMPAQ, DEC, DELL, DYMO-COSTAR, EPSON, FUJITSU, GESTETNER, HEIDELBERG, HITACHI, HP, IBM, IMAGEN, INFOTEC, KODAK, KYOCERA, LANIER, LEXMARK, MINOLTA, MITSUBISHI

Model: [✓ MD-1000], MD-1300, MD-1500, MD-2000, MD-4000, MD-5000



Finally, the Configuration dialog—you already know it—is shown.



Name
Set the name under which to access the configuration in the spool system.

Model
To try your printer with another driver, choose it here.

PPD File
Select a PPD file here.

Connection
You can change the connection for this printer.

Print Filter Settings
Edit filter options here. These options differ for various printer models.

Restriction Settings
Adjust users who are allowed to print using this queue.

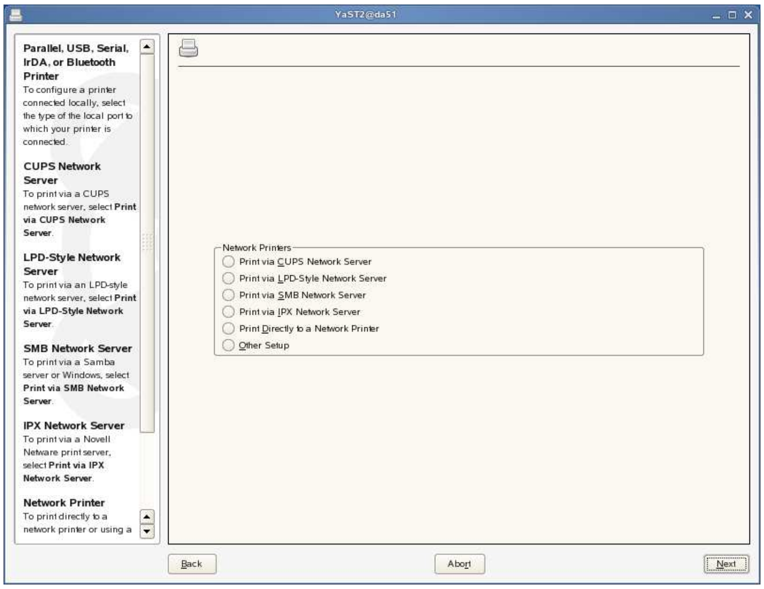
State and Banner Settings
Set state and banners printed before and after each job for this queue.

Printing a Test Page
Click **Test** to test this queue.

Option Area	Current Values
Name and basic settings	printer_1
Printer model	HP Color LaserJet 2550 Series
PPD file	HP Color LaserJet 2550 Series Postscript (recommended) (manufacturer-PPDs/hp/HP_Color_LaserJet
Connection	USB printer on /dev/usb/lp0
Filter settings	...
Restriction settings	
State and banner settings	

1.4.3 Network Printers

CUPS supports a couple of network protocols to communicate with a print server:



Parallel, USB, Serial, IrDA, or Bluetooth Printer
To configure a printer connected locally, select the type of the local port to which your printer is connected.

CUPS Network Server
To print via a CUPS network server, select **Print via CUPS Network Server**.

LPD-Style Network Server
To print via an LPD-style network server, select **Print via LPD-Style Network Server**.

SMB Network Server
To print via a Samba server or Windows, select **Print via SMB Network Server**.

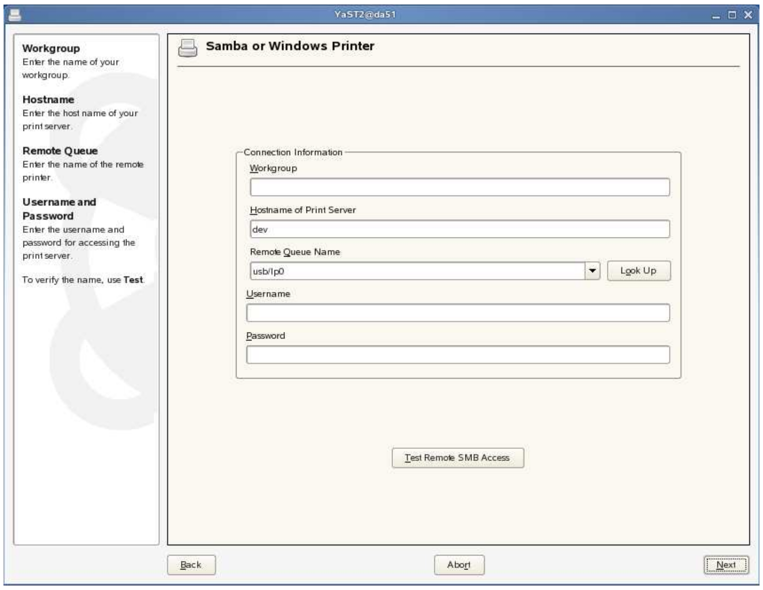
IPX Network Server
To print via a Novell Netware print server, select **Print via IPX Network Server**.

Network Printer
To print directly to a network printer or using a

Network Printers
- ○ Print via CUPS Network Server
- ○ Print via LPD-Style Network Server
- ○ Print via SMB Network Server
- ○ Print via IPX Network Server
- ○ Print Directly to a Network Printer
- ○ Other Setup

Ask the admistrator of your network what protocol is used by the print server. Enter more detailed information (e. g., server name, queue name) in the following dialog. Depending on the selected protocol, this dialog varies.

For example, the dialog for SMB printers look like this:



YaST2@da51

Samba or Windows Printer

Workgroup
Enter the name of your workgroup.

Hostname
Enter the host name of your print server.

Remote Queue
Enter the name of the remote printer.

Username and Password
Enter the username and password for accessing the print server.

To verify the name, use **Test**.

Connection Information

Workgroup:

Hostname of Print Server: dev

Remote Queue Name: usb/lp0

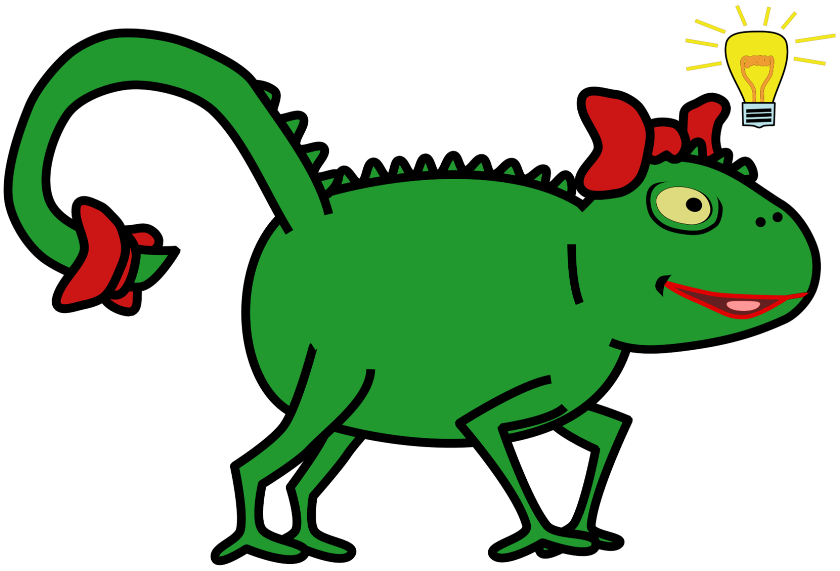
Username:

Password:

It is out of the scope of this course to describe how to connect to a network printer.

Exercise: Configure a Printer

Use YaST to view your printer configuration. Create a second queue for your current printer. This new queue should use another (lower) resolution.



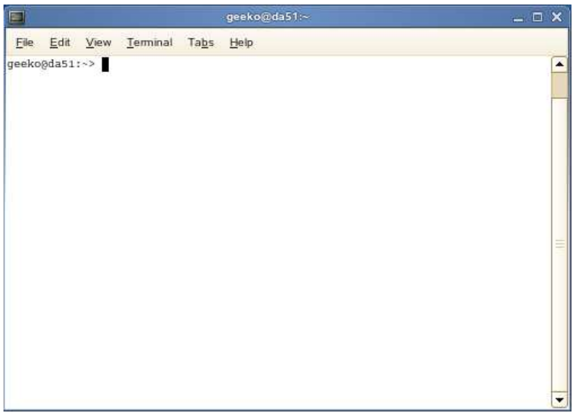
2 The Linux Shell and Tools

You cannot communicate directly with the operating system kernel. This is why you need to use a program that acts as an interface between the user and the operating system.

In the operating systems of the UNIX family, this program is called *shell*.

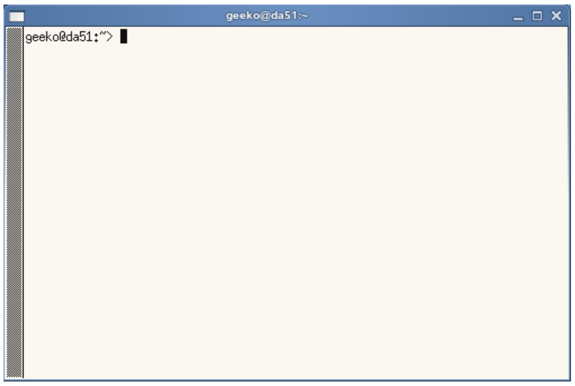
To have a look at the shell, select `Gnome Terminal` or `X Terminal` from the main menu.

Gnome Terminal



```
geeko@da51:~>
```

X Terminal



```
geeko@da51:~>
```

You also have up to six virtual terminals (F1 – F6) running on your computer. By pressing `Ctrl` + `Alt` + `Fx`, you can switch between individual terminals. By pressing `Ctrl` + `Alt` + `F7`, you can switch back to your graphical user interface.

The shell takes entries that users make, interprets them, converts them to system calls, and delivers system messages back to the user. This is why it is called a "command interpreter." UNIX has a couple of shells, most of which Linux provides.

The following are examples of some popular shells:

- ☞ The Bourne shell (linked to Bourne Again shell)
- ☞ The Bourne Again shell
- ☞ The Korn shell
- ☞ The C shell (linked to Tenex C shell)
- ☞ The Tenex C shell

The various shells provide different functions and have different syntaxes.

The Bourne shell (sh) and the C shell (tsh) are not used anymore. The Bourne Again shell (bash) and the Tenex C shell (tcsh) are more user-friendly.

To show syntax differences between the Bourne Again shell (on the left) and the Tenex C shell (on the right), we use simple shell programs that display the numbers 1 to 10 on the screen.

```
#!/bin/bash

declare -i maximum=10
declare -i current=0

while :; do
  if [ $current -eq $maximum ]; then
    break
  fi

  current=$current+1
  echo $current
done
```

```
#!/bin/tcsh

@ maximum = 10
@ current = 0

while (1)
  if ( $current == $maximum ) then
    break
  endif

  @ current = $current + 1
  echo $current
end
```

The most important tasks of the shell are:

1. To interpret the command line. The shell takes the user's input and interprets it.
2. To configure the user environment. When a shell is started, configuration files are read. Variables or aliases can be set.
3. Shell programming. You can simplify or automate complex tasks.

The standard Linux shell is bash. Every shell can be started like a program and you can switch to a different shell at any time. For example, you can switch to the C shell by entering `tcsh` and you can switch to the Korn shell by entering `ksh`. A new bash is started with the command `bash`.

A shell is started at a text console right after a user logs in. This is called the *login shell*. The shell that is started is determined by the user database.

Unlike most other programs, the shell does not shut itself down. You need to enter the command `exit` to return to the previous shell.

Exercise: Understand the Linux Shells

The prompt is sightly different in each shell. Which prompt belongs to which shell?

A – Korn Shell

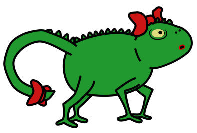
B – Bourne Again Shell

C – Tenex C Shell

❶
```
geeko@da51:~>
```

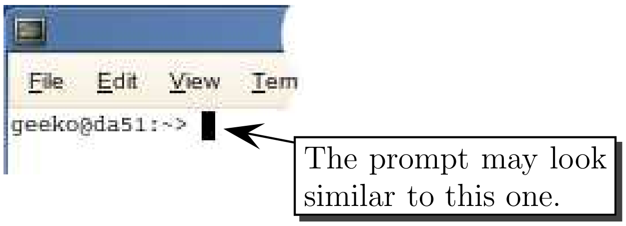
❷
```
/home/geeko>
```

❸
```
geeko@da51:/home/geeko>
```



2.1 Organize Files at the Command Line

When you start a shell, a prompt indicates that the shell is ready to receive your commands.



```
geeko@da51:~>
```

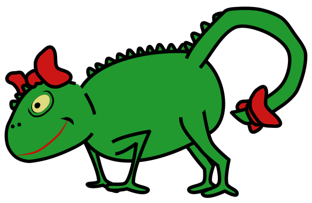
The prompt in this example contains the following information:

geeko	– Current user's login
da51	– Computer's hostname
~	– Current position in the file system

The path for the user's home directory can be abbreviated to the tilde ("~").

If you are logged in as user root, the login is not shown. The prompt looks like this:

```
da51:~ #
```



2.1.1 Move in the File System

You can use the command `cd` (change directory) to change between directories.

Some examples include the following:

- `cd Documents`

  Change to the subdirectory Documents

  ```
  geeko@da51:~> cd Documents
  geeko@da51:~/Documents>
  ```

- `cd /tmp`

  Change directly to the directory `/tmp` (absolute path)

  ```
  geeko@da51:~> cd /tmp
  geeko@da51:/tmp>
  ```

- `cd`

  Change from any directory to the home directory

  ```
  geeko@da51:/tmp> cd
  geeko@da51:~>
  ```

☞ `cd ..`

Move one directory level higher

```
geeko@da51:~> cd ..
geeko@da51:/home>
```

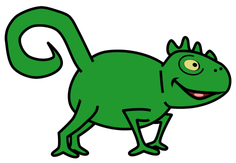
☞ `cd ../..`

Move two directory levels higher

☞ `cd -`

Move to the last valid directory

```
geeko@da51:/home> cd /tmp
geeko@da51:/tmp> cd -
/home
geeko@da51:/home>
```

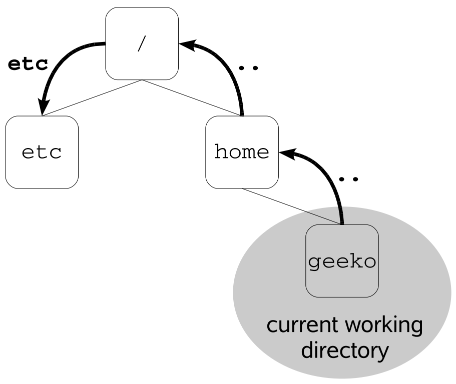


The directory names in a path are separated by a slash ("/"). The path can be specified in two ways:

☞ As a relative path starting from the current directory

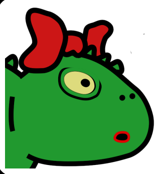
☞ As an absolute path starting from the root of the entire file system tree

The absolute path always begins with a slash ("/"), the symbol for the root directory.



In this figure, the current position in the file system is geeko's home directory. To change into the `/etc` directory, you can use the following commands:

- ☞ absolute: `cd /etc`
- ☞ relative: `cd ../../etc`

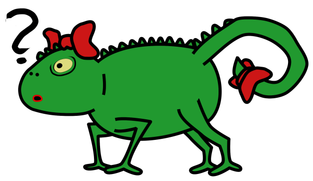


The length of the path cannot be more than 4096 characters, including the slashes.

✍ Exercise: Move in the File System

Write down to which directory you switch after entering the following commands:

Current Position	Command	Final Position
/var/log/	`cd cups`	
/usr/X11R6/	`cd bin`	
/usr/X11R6/	`cd /bin`	
/usr/share/	`cd ..`	
/etc/postfix/	`cd /`	
~	`cd ..`	
/home	`cd ..`	
/media/cdrom/	`cd ../../lib`	
/sbin	`cd`	



2.1.2 List Directory Content

The command `ls` (list) lists directory contents. You can specify the directory without entering an option the contents of the current directory are listed.

```
geeko@da51:~> ls
bin  Desktop  Documents  public_html
geeko@da51:~>
```

The following are the most important options you can use with `ls`:

- ☞ Without any option

  Displays the contents of the current directory in several columns (file and directory names only)

  ```
  geeko@da51:~> ls /var
  adm  cache  games  lib  lock  log  mail  opt  run  spool  tmp  X11R6  yp
  geeko@da51:~>
  ```

- ☞ `-a`

  Also displays hidden files (such as `.bashrc`)

  ```
  geeko@da51:~> ls -a /var
  .   adm    games  lock  mail  run    tmp    yp
  ..  cache  lib    log   opt   spool  X11R6
  geeko@da51:~>
  ```

- ☞ `-F`

  After each name, a character indicates the file type. ("`/`" for directories, "`*`" for executable files, "`|`" for FIFO files, "`@`" for symbolic links).

  ```
  geeko@da51:~> ls -F /var
  adm/    games/  lock/  mail@  run/    tmp/    yp/
  cache/  lib/    log/   opt/   spool/  X11R6/
  geeko@da51:~>
  ```

- ☞ `-d`

  Lists directory entries instead of contents.

  ```
  geeko@da51:~> ls -d /var
  /var
  geeko@da51:~>
  ```

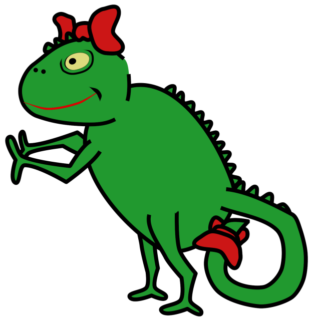
- ☞ `-l` ("long list")

  Gives a detailed list of all the files. For each file name, information on permission, modification time, and size is included.

```
geeko@da51:~> ls -l /var
insgesamt 4
drwxr-xr-x  8 root  root   224 2006-09-04 10:21 adm
drwxr-xr-x 12 root  root   304 2006-09-04 10:26 cache
drwxrwxr-x  2 games games 2816 2006-09-04 10:37 games
drwxr-xr-x 39 root  root   968 2006-09-04 10:42 lib
drwxrwxr-t  4 root  uucp   128 2006-09-04 11:23 lock
drwxr-xr-x 11 root  root   936 2006-09-04 11:30 log
lrwxrwxrwx  1 root  root    10 2006-09-04 09:59 mail -> spool/mail
drwxr-xr-x  3 root  root    72 2006-09-04 10:06 opt
drwxr-xr-x 19 root  root   952 2006-09-04 11:30 run
drwxr-xr-x 11 root  root   296 2006-09-04 10:15 spool
drwxrwxrwt  3 root  root    80 2006-09-04 11:38 tmp
drwxr-xr-x  5 root  root   152 2006-09-04 10:13 X11R6
drwxr-xr-x  3 root  root   104 2006-09-04 10:15 yp
geeko@da51:~>
```

Have a look at the first ten characters of the output ("`-rw-r--r--`"). The first character indicates the type of file:

Character	File Type
–	Normal file
d	Directory
l	Link



The remaining nine characters show the file permission.

You can assign the following three kinds of permission to a file or directory:

Read (r) This permission allows the file to be read or the contents of a directory to be listed.

Write (w) This permission allows a file to be modified. It allows files to be created or deleted within a directory.

Execute (x) This permission allows a file to be executed. It allows you to make changes to this directory.

If permission is set, the character is shown. Otherwise a "–" appears.

The permission characters are grouped ("`rwx rwx rwx`"):

Character 1 to 3 Represent the permission of the file owner.

Character 4 to 6 Represent the permission of the owning group.

Character 7 to 9 Represent the permission of all other users.

The special permission SUID, SGID, and Sticky are shown by the characters “`s`” and “`t`”.

SUID Set user ID

```
geeko@da51:~ > ls -l /usr/bin/passwd
-rwsr-xr-x 1 root shadow 79765 2006-03-24 12:19
/usr/bin/passwd
geeko@da51:~ >
```

SGID Set group ID

```
geeko@da51:~ > ls -l /usr/bin/wall
-rwxr-sr-x 1 root tty 10192 2006-03-22 05:24 /usr/bin/wall
geeko@da51:~ >
```

Sticky Sticky bit

```
geeko@da51:~ > ls -ld /tmp
drwxrwxrwt 15 root root 608 2006-04-06 12:45 /tmp
geeko@da51:~ >
```

The number next to the permission shows the number of hard links to that file. Hard links are explained on page 55 in this chapter.

The name of the file owner is shown next to the number of hard links. The name of the owning group is shown next to the file owner.

Then, the date and time of the changes in the file and the filename follow. For symbolic links, the file name of the target is also shown.

☞ `-t`

Files are sorted by date of alteration.

```
geeko@da51:~> ls -t /var
tmp  run  log  lock  lib  games  cache  adm  spool  yp  X11R6  opt  mail
geeko@da51:~>
```

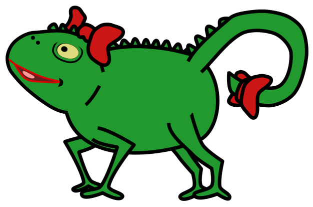
Combined with the option `-r`, the files are listed the other way round (the latest file is displayed at the end).

☞ `-R`

Output is recursive, including all subdirectories.

```
geeko@da51:~> ls -R /var
/var:
adm
cache
games
lib
lock
log
```

```
mail
opt
run
spool
tmp
X11R6
yp

/var/adm:
autoinstall
backup
fillup-templates
mount
perl-modules
SuSEconfig

/var/adm/fillup-templates:
group.aaa_base
passwd.aaa_base
...
```



☞ `-u`

The files are sorted by date of last access.

```
geeko@da51:~> ls -u /var
adm  cache  games  lib  lock  log  opt  run  spool  tmp  X11R6  yp  mail
geeko@da51:~>
```

You also can combine the `ls` options.

```
geeko@da51:~> ls -la /var
insgesamt 5
drwxr-xr-x 14 root  root   360 2006-09-04 10:04 .
drwxr-xr-x 22 root  root   520 2006-09-04 11:19 ..
drwxr-xr-x  8 root  root   224 2006-09-04 10:21 adm
drwxr-xr-x 12 root  root   304 2006-09-04 10:26 cache
drwxrwxr-x  2 games games 2816 2006-09-04 10:37 games
drwxr-xr-x 39 root  root   968 2006-09-04 10:42 lib
drwxrwxr-t  4 root  uucp   128 2006-09-04 11:23 lock
drwxr-xr-x 11 root  root   936 2006-09-04 11:30 log
lrwxrwxrwx  1 root  root    10 2006-09-04 09:59 mail -> spool/mail
drwxr-xr-x  3 root  root    72 2006-09-04 10:06 opt
drwxr-xr-x 19 root  root   952 2006-09-04 11:30 run
drwxr-xr-x 11 root  root   296 2006-09-04 10:15 spool
drwxrwxrwt  3 root  root    80 2006-09-04 11:38 tmp
drwxr-xr-x  5 root  root   152 2006-09-04 10:13 X11R6
drwxr-xr-x  3 root  root   104 2006-09-04 10:15 yp
geeko@da51:~>
```

Exercise: List Directory Content – Part I

How many subdirectories are inside the following directories?

Directory	Number of Subdirectories
`/`	
`/sbin`	
`/var/log/`	
`/etc/init.d/`	
`/usr/share/doc/`	
`/bin`	

Exercise: List Directory Content – Part II

Write down the owner and the owning group of the following files and directory. What are the permission of the following files and directories?

File/Directory	User	Group	Permission
`~/.nautilus/`			
`/tmp`			
`/var/spool/mail/`			
`/etc/init.d/autofs`			
`/usr/bin/passwd`			
`/etc/passwd`			
`/etc/shadow`			
`/usr/bin/wall`			

2.1.3 Move a File

You can use the command `mv` (move) to move or rename files. The syntax is

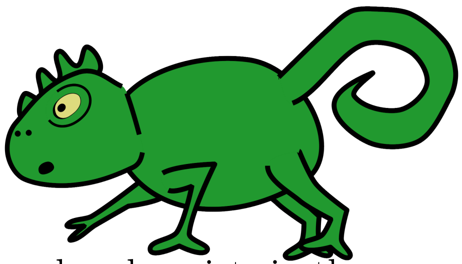
```
mv source destination
```

For example, to move one file (`index.html`) to another directory (`/tmp`), the command looks like this:

```
geeko@da51:~> mv index.html /tmp
geeko@da51:~>
```

Attention! The move has not been confirmed as successful. Normally, the bash does not confirm that a task has been completed successfully. Only error messages are shown.

```
geeko@da51:~> mv index.html /tmp
mv: cannot stat ‘index.html’: No such file or directory
geeko@da51:~>
```



Attention! If a file with the same name already exists in the target directory, it is replaced without consulting you.

You can also use the command `mv` to rename a file, for example:

```
geeko@da51:~> mv index.html index_en.html
geeko@da51:~>
```

The file index.html is renamed to index_en.html.

The difference between these two examples is: in the first example, the second parameter is a directory name and in the second example, the second parameter is a filename.

If you want to move more than one file, you can use the wildcards “?” (for any character) and “*” (for none, one, or several characters) in the first parameter.

```
geeko@da51:~> mv *.html /tmp
geeko@da51:~>
```

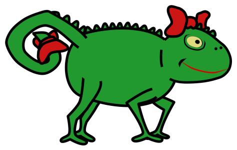
All files with the suffix “`.html`” in the current directory are moved into the `/tmp` directory.

You can use `mv` with some options. The most two important are:

☞ `-i`

Consults you before moving or renaming a file. This prevents existing files with the same name from being replaced.

```
1 geeko@da51:~> mv -i index.html /tmp
2 mv: overwrite ‘/tmp/index.html’? n
  geeko@da51:~> mv index.html /tmp
4 geeko@da51:~>
```



☞ `-u`

Only moves files that are newer than the target files of the same name.

```
 1 geeko@da51:~> ls -l index.html
 2 -rw-r--r-- 1 geeko users 26432 2005-12-08 15:48 index.html
   geeko@da51:~> ls -l /tmp/index.html
 4 -rw-r--r-- 1 geeko users 864 2006-09-04 11:26 /tmp/index.html
   geeko@da51:~> mv -u index.html /tmp/
 6 geeko@da51:~> ls -l index.html
   -rw-r--r-- 1 geeko users 26432 2005-12-08 15:48 index.html
 8 geeko@da51:~> ls -l /tmp/index.html
   -rw-r--r-- 1 geeko users 864 2006-09-04 11:26 /tmp/index.html
10 geeko@da51:~> mv index.html /tmp/
   geeko@da51:~> ls -l /tmp/index.html
12 -rw-r--r-- 1 geeko users 26432 2005-12-08 15:48 /tmp/index.html
   geeko@da51:~> ls -l index.html
14 /bin/ls: index.html: No such file or directory
   geeko@da51:~>
```

Explanation: The file `~/index.html` was modified in December 2005 and is 26432 bytes (lines 1–2). The file `/tmp/index.html` was modified in September 2006 and is 864 bytes (lines 3–4). After moving the file `~/index.html` to the directory `/tmp` (line 5) using the option `-u`, the file `~/index.html` still exists (lines 6–7) and the destination file still has the same data (864 bytes, lines 8–9). That means, the file was not moved. Without `-u` (line 10), the command `mv` overwrites the destination file although it is newer than `~/index.html` (lines 11–12). The file `~/index.html` does not exist anymore (lines 13–14).

2.1.4 Copy a File

You can copy files and directories with the command `cp` (copy). The syntax is

`cp` *`source destination`*

When using the command `cp`, please do not forget:

- ☞ The command `cp` replaces existing files without consulting you.
- ☞ If you only want to copy the contents of a directory (without the directory itself), the target directory must already exist, for example, to back up a directory using a different name.

Without any option, `cp` copies files. If you want to copy a directory, you have to use the option `-r`.

```
geeko@da51:~> cp proposals/ /tmp
cp: omitting directory ‘proposals/’
geeko@da51:~> cp -r proposals/ /tmp
geeko@da51:~>
```

In this example, the directory `~/proposals/` with all its subdirectories is copied to the directory `/tmp`. The result is a directory `/tmp/proposals/`.

To only copy the contents of the directory `proposals/` (including hidden files and subdirectories) to the directory `/tmp`, enter the following:

```
geeko@da51:~ > cp -r proposals/. /tmp
geeko@da51:~ >
```

If you do not want to copy the hidden files, enter the following:

```
geeko@da51:~ > cp -r proposals/* /tmp
geeko@da51:~ >
```

Besides `-r`, the most important options of `cp` are:

- ☞ `-a`, `--archive`

  Copies a directory and subdirectories (like `-r`); symbolic links, file permission, owners, and time stamps are not changed.

```
geeko@da51:~> ls -a index.html
-rw-r--r-- 1 geeko users 864 2006-09-04 11:26 index.html
geeko@da51:~> cp -a index.html /tmp/
geeko@da51:~> ls -l /tmp/index.html
-rw-r--r-- 1 geeko users 864 2006-09-04 11:26 /tmp/index.html
geeko@da51:~> cp index.html /tmp/
geeko@da51:~> ls -l /tmp/index.html
-rw-r--r-- 1 geeko users 864 2006-09-04 13:35 /tmp/index.html
geeko@da51:~>
```

Explanation: In this example, the last change of the file ~/`index.html` was at 11:26 (lines 1–2). After copying the file with the `-a` option (line 3), the modification time of the copy is 11:26, too (lines 4–5). When not using `-a` (line 6), the modification time is set to the current time (here 13:35, lines 7–8).

`-i`, `--interactive`

Asks before overwriting.

```
geeko@da51:~> cp -i index.html /tmp
cp: overwrite '/tmp/index.html'? n
geeko@da51:~> cp index.html /tmp
geeko@da51:~>
```

`-s`, `--symbolic-link`

Creates symbolic links instead of copying.

```
geeko@da51:~> cp -s /tmp/index.html .
geeko@da51:~> ls -l index.html
lrwxrwxrwx 1 geeko users 15 2006-09-04 13:37 index.html -> /tmp/index.html
geeko@da51:~>
```

`-u`, `--update`

Copies a file only when the source file is newer than the destination file or when the destination file is missing.

```
 1 geeko@da51:~> ls -l index.html
 2 -rw-r--r-- 1 geeko users 26432 2005-12-08 15:48 index.html
   geeko@da51:~> ls -l /tmp/index.html
 4 -rw-r--r-- 1 geeko users 864 2006-09-04 13:40 /tmp/index.html
   geeko@da51:~> cp -u index.html /tmp/
 6 geeko@da51:~> ls -l /tmp/index.html
   -rw-r--r-- 1 geeko users 864 2006-09-04 13:40 /tmp/index.html
 8 geeko@da51:~> cp index.html /tmp/
   geeko@da51:~> ls -l /tmp/index.html
10 -rw-r--r-- 1 geeko users 26432 2006-09-04 13:42 /tmp/index.html
   geeko@da51:~>
```

Explanation: The file ~/`index.html` was modified in December 2005 and is 26432 bytes (lines 1–2). The file `/tmp/index.html` was modified in September 2006 and is 864 bytes (lines 3–4). After copying the file ~/`index.html` to the directory `/tmp` (line 5) using the option `-u`, the destination file is still the same (864 bytes, lines 6–7). Without `-u` (line 8), the command `cp` overwrites the destination file, although it is newer than ~/`index.html` (lines 9-10).

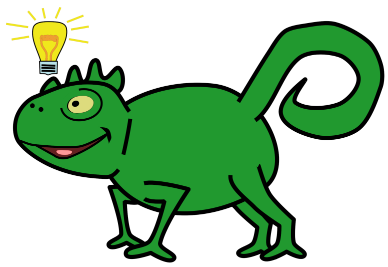
Exercise: Move and Copy Files

Do the following:

❶ Copy the file `/etc/init.d/syslog` into your home directory. How does the file permission change?

❷ Rename the file ~/`syslog` into `testfile`.

3. Copy the file `grep-demo` from the examples directory of the book CD into your home directory.



2.1.5 Create a Directory

You can use the command `mkdir` (make directory) to create new directories.

```
geeko@da51:~> mkdir Proposal
geeko@da51:~>
```

With the option `-p`, you can create a complete path, as in the following:

```
 1 geeko@da51:~> ls
 2 bin  Desktop  Documents  public_html
   geeko@da51:~> mkdir Invoice/Customer
 4 mkdir: cannot create directory ‘Invoice/Customer’: No such file or directory
   geeko@da51:~> mkdir -p Invoice/Customer
 6 geeko@da51:~> ls
   bin  Desktop  Documents  Invoice  public_html
 8 geeko@da51:~> ls Invoice/
   Customer
10 geeko@da51:~>
```

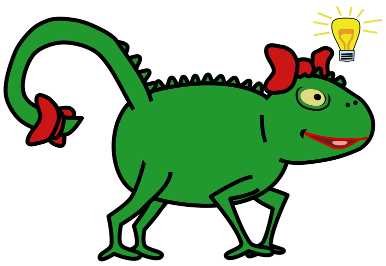
Explanation: It is possible to create a new directory Invoice including a directory Customer (lines 3–4). Use the option `-p`: `mkdir -p` (line 5).

Exercise: Create a Directory

Do the following:

1. Create a new directory `Marketing` in your home directory.
2. Move the file `testfile` into the `Marketing` directory.
3. Create a new directory `School` that includes a subdirectory `Mathematics` in the `/tmp` directory.
4. Copy the Marketing directory from your home directory into the directory `/tmp/School/Mathematics`.

❺ Create a new directory `German` in the directory `/tmp/School/`.



2.1.6 Delete Directories and Files

You can use the `rmdir` (remove directory) command to delete the directory or directories (e. g., `rmdir Proposal`).

```
geeko@da51:~> rmdir Proposal
geeko@da51:~>
```

The directory or directories must be empty before you can delete them.

```
geeko@da51:~> rmdir Invoice
rmdir: Invoice: Directory not empty
geeko@da51:~> ls Invoice/
Customer
geeko@da51:~>
```

You can use the command `rm` (remove) to delete files, as in the following:

```
geeko@da51:~> rm file
geeko@da51:~>
```

If you use `rm` with wildcards, be careful that only the files are deleted you want to delete.

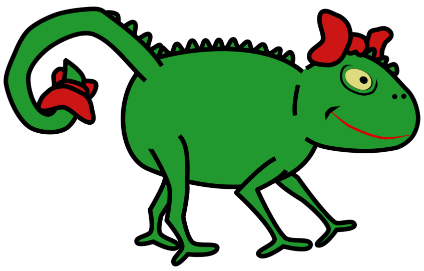
```
geeko@da51:~> ls
bin      Documents  file2  file4  file6  Invoice
Desktop  file1      file3  file5  file7  public_html
geeko@da51:~> rm file*
geeko@da51:~>
```

This example deletes all files, that begin with `file`, in the current directory without asking for confirmation.

❢ ***Attention!*** Files deleted with the command `rm` cannot be restored.

If the user does not have the permission to delete a file, this file is ignored and an error message is printed.

```
geeko@da51:~> rm /etc/vimrc
rm: remove write-protected regular file ‘/etc/vimrc’? y
rm: cannot remove ‘/etc/vimrc’: Permission denied
geeko@da51:~>
```



The following are some important options you can use with `rm`:

☞ `-i`

Consults you before deleting.

```
geeko@da51:~> rm -i file
rm: remove regular empty file ‘file’? y
geeko@da51:~>
```

☞ `-r` (recursively)

Allows full directories to be deleted.

```
geeko@da51:~> rmdir Invoice
rmdir: Invoice: Directory not empty
geeko@da51:~> rm -r Invoice
geeko@da51:~>
```

☞ `-f` (force)

By default, `rm` consults you if the file that should be deleted is read-only. Using this option, the files are deleted without consulting you.

```
1 geeko@da51:~> ls -l file
2 -r--r--r-- 1 geeko users 0 2006-09-04 11:44 file
  geeko@da51:~> rm file
4 rm: remove write-protected regular empty file ‘file’? n
  geeko@da51:~> rm -f file
6 geeko@da51:~>
```

Explanation: The file `file` is read-only (lines 1–2). When you use `rm` without any option the command prompts for confirmation (lines 3–4). (This can be annoying if you delete a couple of files using wildcards.) Using `-f`, the files are deleted without consulting you (line 5).

Exercise: Delete Directories and Files

Do the following:

❶ Delete the file /tmp/School/Mathematics/Marketing/testfile.
❷ Delete the empty directory /tmp/School/German/.
❸ Delete the directory School with all its content from the /tmp directory.

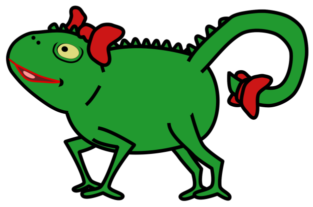
2.1.7 Link Files

File system formats in Linux keep data and administration information separately. The data are organized according to the respective file system format.

Each file is described by an inode (index node or information node). To see the inode number, enter `ls -i`.

```
geeko@da51:~> ls -li
insgesamt 0
84800 drwxr-xr-x 2 geeko users 48 2006-09-22 11:20 bin
84811 drwxr-xr-x 2 geeko users 80 2006-09-22 11:20 Desktop
84801 drwx------ 2 geeko users 80 2006-09-22 11:20 Documents
84808 drwxr-xr-x 2 geeko users 80 2006-09-22 11:20 public_html
geeko@da51:~>
```

Each of these inodes is 128 bytes and contains all the information on this file except the file name. This includes details of the owner, access permission, size, various time details (time of modification, of access, of modification of the inode), and the links to the data blocks of this file.



The command `ln` creates a link. A link is a reference to a file. With a link, you can access a file from anywhere in the file system using different names for it. This means that the file itself exists only once on the system, but it can be accessed by using different names.

In Linux, two kinds of links are available:

- ☞ Hard links
- ☞ Symbolic links

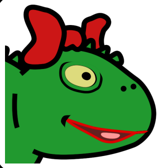
Use the command `ln` to create a hard link. This link points to the inode of a file. Thereafter, the file can be accessed by using both names—by the name of the file itself and by the name of the link. That means you are no longer able to tell the difference between the original file and the link.

The following is an example of using the command `ln`:

```
1 geeko@da51:~/Proposal > ls -li
2 total 4
  88658 -rw-r--r-- 1 geeko users 82 2004-04-06 14:21 old
4 geeko@da51:~/Proposal > ln old new
  geeko@da51:~/Proposal > ls -li
6 total 8
  88658 -rw-r--r-- 2 geeko users 82 2004-04-06 14:21 old
8 88658 -rw-r--r-- 2 geeko users 82 2004-04-06 14:21 new
  geeko@da51:~/Proposal >
```

Explanation: `ls -i` displays the file inode (lines 1–3). In this example, the inode of the file `old` is 88658. With the `ln` command, you can create a hard link (line 4). The new link has the same inode as the original file (lines 5–8).

Hard links can only be used when both the file and the link are in the same file system (on the same partition), because inode numbers are only unique within the same file system.

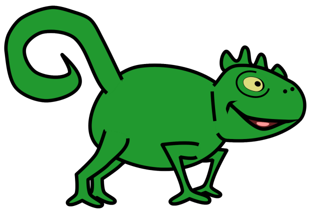

You can see the number of hard links pointing to an inode in the `ls` output after the permission.

A symbolic link is assigned its own inode—the link refers to a file, so a distinction can always be made between the link and the actual file. You can create a symbolic link with the command `ln` and the option `-s`.

The following is an example of creating a symbolic link:

```
1 geeko@da51:~/Proposal > ls -li
2 total 4
  88658 -rw-r--r-- 1 geeko users 82 2004-04-06 14:21 old
4 geeko@da51:~/Proposal > ln -s old new
  geeko@da51:~/Proposal > ls -li
6 total 4
  88658 -rw-r--r-- 1 geeko users 82 2004-04-06 14:21 old
8 88657 lrwxrwxrwx 1 geeko users 3 2004-04-06 14:27 new -> old
  geeko@da51:~/Proposal >
```

Explanation: In this example, the inode of the file `old` is 88658 (lines 1–3). With the `ln -s` command you can create a soft link (line 4). In this example, the link has the inode 88657 (lines 5–8).



The disadvantage of a symbolic link is that it can point to a non-existing object if the object no longer exist.

If you delete the file `old` in the example above, `new` will point to a non-existing file. The common `ls` does not display that the link is "broken":

```
geeko@da51:~/Proposal > rm old
geeko@da51:~/Proposal > ls -li
total 0
88657 lrwxrwxrwx 1 geeko users 3 2004-04-06 14:27 new -> old
geeko@da51:~/Proposal >
```

With symbolic links, you can also create links to directories.

Exercise: Link Files

Do the following:

❶ How many links does the following directory include?
`/boot/boot/boot/boot/boot/boot/boot/boot/boot/`

❷ Create a symbolic link named `symblink` in your home directory that points to the file `Marketing/testfile`.

❸ Create a hard link named `hardlink` in your home directory that points to the file `Marketing/testfile`.

❹ Remove the file `Marketing/testfile`.

❺ Test links with the command `cat` ***`filename`***. The command `cat` displays the content of a text file.

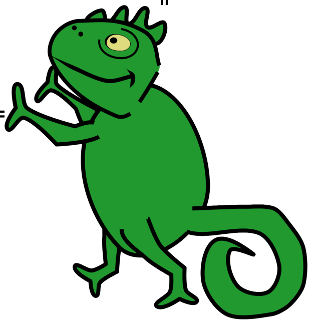
2.1.8 Find Files

The shell provides commands to find files. The most important commands are:

- find
- locate

Example (Suppose the following files exist):

File	File1a	File2a
file	File1b	FILE
File1	File2	MyFile



Find Files with `file`

To search for files on the command line, use the command **`find`**. The syntax is:

`find` *`path criterion action`*

For example:

```
geeko@da51:~> find . -name "File?"
./File1
./File2
geeko@da51:~>
```

The command has a lot of options. Some of them are explained in the following:

`path` The part of the file system to search (the specified directory and all its subdirectories). If nothing is specified, the file system below the current directory is used.

`criterion` The properties the file should have. They refer to the following:

- `-ctime [+/-]`*`days`*

  Searches for files whose last change took place no later than (or no earlier than) a specified number of days ago.

- `-group` *`name`*

  Searches for files owned by the group *`name`*. Instead of a name, you can use the numeric GID.

- `-name` *`pattern`*

  Searches for files whose names contain the given pattern. If the pattern contains wild cards, it must be enclosed by quotation marks. Otherwise, it will be interpreted by the shell and not by the `find` command.

- `-size [+/-]`*`size`*

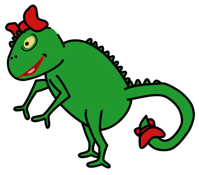
  Matches files that are bigger or smaller than a certain size. As an argument, the size (in blocks of 512 bytes) is given. The suffix "`c`" switches to byte (e. g., `-size -260c`) and "`k`" to blocks of 1024 bytes (e. g., `-size +600k`). A preceding "`+`" stands for all larger files and a "`-`" for all smaller files.

- `-type` *`file_type`*

  Searches for a file type. A file type can be one of the following: "`d`" for a directory, "`f`" for a file, or "`l`" for a symbolic link.

- `-user` *`name`*

  Searches for files owned by user *`name`*. Instead of a name, you can use the numeric UID.



`action` Options that influence the following conditions or control the whole search, such as the following:

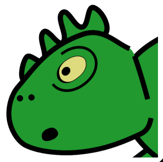
- `-print` (default)
- `-exec` *`command`*

With the option `-exec`, you can call up another command. This option is frequently used to link `find` and `grep` (see section 2.3.2 on page 90) as in the following:

```
geeko@da51:~ > find ~ -name "letter*" -type f -exec grep appointment {} \;
appointment for next meeting: 23.08.
/home/geeko/letters/letter_Smith
geeko@da51:~ >
```

Explanation: In this example, the command `find` searches for files whose names start with `letter`, and then passes the names of the files found with `-exec` to the following command (in this case `grep appointment {}`).
The two brackets {} are placeholders for the file names that are found and passed to the command `grep`. The semicolon closes the `-exec` instruction. As the semicolon is a special character, it is masked by a backslash.

When you use `grep` alone, it searches for a specific expression in a given file. If you use `grep` with `find`, you can search for a specific expression in a file whose location is not known.



Exercise: Find Files

Please answer the following questions:

☞ How many files in the `/bin` directory are larger than 1024kByte?

____

☞ How many files in the directory `/etc` belong to the group lp?

____

☞ Where in the filesystem is the command `grep` located?

____

Locate Files with `locate`

Attention! The command `locate` is not available in a standard installation. Before you can use this command, you have to install the package `findutils-locate`.

The command `locate` is equivalent to `find -name`.

As the command `find` searches through the selected part of the file system, this process might be quite slow. `locate` searches through a database that was created especially for this purpose (/var/lib/locatedb). That quicken the search.

On SUSE Linux Enterprise Desktop, the database is automatically created and updated every day. If you change something after this update, you have to update manually using the command updatedb.

Attention! After installing findutils-locate, create the database by using the command updatedb (as user root). Otherwise, you will get an error message:

```
geeko@da51:~> locate File?
locate: /var/lib/locatedb: No such file or directory
```

You can execute a command at the command line as user root by using the su command. If you want to execute just one command, use the option -c *command*. To authenticate, enter the root password.

```
geeko@da51:~> su -c updatedb
Passwort:
geeko@da51:~>
```

The following example shows the output of locate:

```
geeko@da51:~> locate File?
/home/geeko/File1
/home/geeko/File1a
/home/geeko/File1b
/home/geeko/File2
geeko@da51:~>
```

locate displays all files whose names contain the search string.

To learn more about locate, enter man locate.

Exercise: Locate Files

Make sure that the package findutils-locate is installed and update the database that is used by locate.

Use locate to find the following files:

- updatedb: ________________________________
- XF86Config: ________________________________
- smb.conf: ________________________________
- hardlink: ________________________________

☞ `boot.log`: ______________________________

☞ `installkernel`: ______________________________

Find Executable Files with `which`

The command **`which`** searches all paths listed in the variable `PATH` for the specified command and returns the full path of the command.

A variable is a labeled memory space, where you can store information.

In the variable `PATH`, the most important directories where the shell looks for executable files are listed.

To see the content of a variable, use the command **`echo`** and add a "**$**" in front of the variable's name.

To see the content of the variable PATH, enter **`echo $PATH`**.

```
geeko@da51:~> echo $PATH
/home/geeko/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/gnom
e/bin:/opt/kde3/bin:/usr/lib/jvm/jre/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin
geeko@da51:~>
```

If several versions of a command are in different directories and you want to know which version is executed when you enter it without specifying a path, use the command **`which`**. The following is an example of using the command **`which`**:

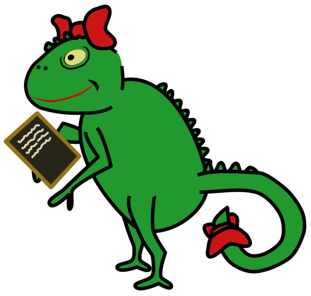
```
geeko@da51:~ > which find
/usr/bin/find
geeko@da51:~ > which cp
/bin/cp
geeko@da51:~> which locate
/usr/bin/locate
geeko@da51:~ >
```

For more information on **`which`**, enter **`man which`**.

Exercise: Find Executable Files

Use `which` to locate the following commands:

- `grep`: ______________________________
- `which`: ______________________________
- `mkdir`: ______________________________
- `nautilus`: ______________________________
- `OOo-writer`: ______________________________
- `ktab`: ______________________________
- `fvwm2`: ______________________________



Specify the Type of a Command

With the command `type`, you can find out what kind of command is executed when the command is entered:

- a shell built-in command (an essential command that is hardcoded in the shell)
- an external command (called by the shell)
- an alias
- a function

The option `-a` displays all instances of a command with this name in the file system. The following is an example of using the command `type`:

```
geeko@da51:~ > type type
type is a shell built in
geeko@da51:~ > type grep
grep is /usr/bin/grep
geeko@da51:~ > type -a grep
grep is /usr/bin/grep
grep is /bin/grep
geeko@da51:~ >
```

Exercise: Specify the Type of a Command

What kind of files are the following:

☞ `echo`

- ❒ A shell built in
- ❒ An external command: ______________________
- ❒ An alias to: ______________________________
- ❒ A function

☞ `ls`

- ❒ A shell built in
- ❒ An external command: ______________________
- ❒ An alias to: ______________________________
- ❒ A function

☞ `which`

- ❒ A shell built in
- ❒ An external command: ______________________
- ❒ An alias to: ______________________________
- ❒ A function

☞ `_which`

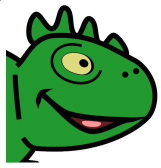
- ❒ A shell built in
- ❒ An external command: ______________________
- ❒ An alias to: ______________________________
- ❒ A function

☞ `test`

- ❒ A shell built in
- ❒ An external command: ______________________
- ❒ An alias to: ______________________________
- ❒ A function

2.1.9 Archive Files

`tar` (tape archiver) is the most commonly used tool for data backup. It archives files in a special format, either directly on a corresponding medium (such as magnetic tape or formatted floppy disk) or to a so-called archive file. By default, the data is *not* compressed.


Names of archive files have the suffix `.tar`. If archive files are compressed (usually with the command `gzip`; see page 66), the suffix is either `.tar.gz` or `.tgz`.

The syntax is: `tar` *`option files`*

The command first expects an option (which is why it can also be used without a minus sign) and the name of the directory to be backed up. All directories and files beneath this are also saved. One very important option is `-f` *`filename`* which is used to specify the name of the archive to be written (or the device file).

Directories are typically backed up with:

```
geeko@da51:~ > tar -cvf /tmp/backup.tar /home/geeko
```

Explanation: This command backs up the complete contents of the user's home directory and stores it in the file `/tmp/backup.tar`.

The option `-c` (*create*) creates the archive. The option `-v` (*verbose*) provides a detailed list containing the name of each file being stored. After the option `-f` (*file*), specify the name of the archive to be created.

When an archive is created, absolute paths turns to relative and that is the first "/" in the path is removed.

```
tar: Removing leading '/' from member names
```

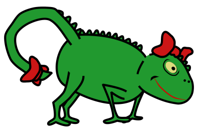
To unpack files of an archive, enter:

```
geeko@da51:~ > tar -xvf /tmp/backup.tar
```

Explanation: This command unpacks all files stored in the archive into the current directory. The directory structure is created according to the relative path specifications in the `tar` archive.

If you only want to unpack one file, specify the filename:

```
geeko@da51:~ > tar -xvf /tmp/backup.tar home/geeko/.bashrc
```



More useful options are:

☞ `-C`

Unpacks to a specific directory using the option `-C` followed by the directory name.

```
geeko@da51:~ > tar -xvf /tmp/backup.tar -C /data
```

☞ `-d`

Compares files in the archive with those in the file system.

```
geeko@da51:~ > tar -dvf /tmp/backup.tar -C /home/geeko
```

☞ `-j`

Compresses or decompresses the tar archive using `bzip2` (see page 67).

```
geeko@da51:~ > tar -cjvf /tmp/backup.tar.bz2 /home/geeko
```

☞ `-r`

Adds files to an archive.

```
geeko@da51:~ > tar -vf /tmp/backup.tar -r new_file
```

☞ `-t`

Displays the contents of an archive:

```
geeko@da51:~ > tar -tvf /tmp/backup.tar
```

☞ `-u`

Only adds files to an archive tha are newer that the file version in the archive (*update*).

```
geeko@da51:~ > tar -uvf /tmp/backup.tar -C /home/geeko
```

☞ `--exclude`

The given files are not included when a directory is archived.

```
geeko@da51:~ > tar -cvf /tmp/backup.tar /home/geeko --exclude="*.iso"
```

Explanation: The home directory of user geeko is archived without all ISO files in his home directory.

☞ `-z`

Compresses or decompresses the `tar` archiv using `gzip` (see page 66).

```
geeko@da51:~ > tar -czvf /tmp/backup.tar.gz /home/geeko
```

Exercise: Create a tar Archive

Create the `tar` archive `/tmp/bin.tar` containing your entire `/bin` directory. Then go to the directory `/tmp/` and unpack only one file, e. g., the file `chown`. Compare owner and time stamp of the unpacked and the original file.

To compress `tar` archives, use the options `-j` or `-z`. In Linux, you can also use some tools to compress and decompress data.

You can use the command `gzip` to compress and decompress data. The compressed data has the suffix `.gz`. This command is only useful for compressing individual files.

The syntax of gzip is:

`gzip [`*`options`*`]` *`file`*

For example:

```
geeko@da51:~> ls
backup.tar  bin  Desktop  Documents  public_html
geeko@da51:~> gzip backup.tar
geeko@da51:~> ls
backup.tar.gz  bin  Desktop  Documents  public_html
geeko@da51:~>
```

To save several files or entire directories in a compressed file, use the command `tar`.

The following are some useful options of `gzip`:

- `-c`

  Compresses the file without modifying the original file. The result is written to the standard output (usually the screen).

- `-d`

  Decompresses the specified file instead of compressing it.

- `-1` to `-9`

  Controls the compression speed. `-1` compresses faster but produces larger files. `-9` requires more computing time but produces smaller files. The default setting is `-6`.

☞ `-r`

Also compresses and decompresses files in all subdirectories.

Instead of `gzip -d`, you can also use the command `gunzip`. This command decompresses a file compressed by `gzip` and removes the suffix `.gz`.

```
geeko@da51:~> ls
backup.tar.gz  bin  Desktop  Documents  public_html
geeko@da51:~> gunzip backup.tar.gz
geeko@da51:~> ls
backup.tar  bin  Desktop  Documents  public_html
geeko@da51:~>
```

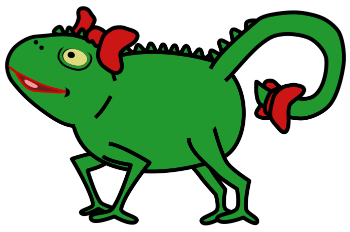
Exercise: Compress a File – Part I

Compress the archive `/tmp/bin.tar` using `gzip` with three different compression speeds. Write down the file size of the compressed file here:

-1 ______________________________

-6 ______________________________

-9 ______________________________



`bzip2` also compresses files. It creates files that are usually about twenty to thirty percent smaller than the files compressed by `gzip`. But `bzip2` needs more computing time to compress. The compressed data has the suffix `.bz2`.

The syntax of `bzip2` is:

`bzip2 [`*`options`*`]` *`file`*

For example:

```
geeko@da51:~> ls
backup.tar  bin  Desktop  Documents  public_html
geeko@da51:~> bzip2 backup.tar
geeko@da51:~> ls
backup.tar.bz2  bin  Desktop  Documents  public_html
geeko@da51:~>
```

Some important options are:

☞ `-c`

Compresses the file without modifying the original file. The result is written to the standard output (usually the screen).

☞ `-d`

Decompresses the specified file instead of compressing it.

☞ `-1` to `-9`

Controls the compression speed. `-1` compresses faster but produces larger files. `-9` requires more computing time but produces smaller files. The default setting is `-9`.

`bunzip2` decompresses a file compressed by `bzip2` and removes the suffix `.bz2`. `bunzip2` is equivalent to `bzip2 -d`.

Exercise: Compress a File – Part II

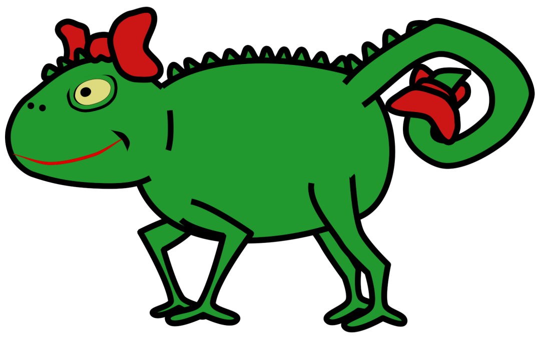
Compress the archive `/tmp/bin.tar` using `bzip2` with three different compression speeds. Write down the file size of the compressed file here:

`-1` ______________________________

`-6` ______________________________

`-9` ______________________________

Copy the `tar` archive into your home directory and unpack the files into a new directory `Training`.



2.1.10 Manage File Permission and Ownership

You can use the command `ls -l` to display the contents of the current directory with the assigned permission for each file or subdirectory.

For example, entering `ls -l` displays the following permission for `MyFile`:

```
geeko@da51:~> ls -l MyFile
-rw-r--r-- 1 geeko users 0 2006-02-07 19:10 MyFile
geeko@da51:~>
```

You can use the command `chmod` to change the permission of a file. Both the owner of a file and root can use this command.

There are options to change the permission for the owner (“`u`”), group (“`g`”), other (“`o`”), or all (“`a`”). To add permission use the option “`+`”. To remove permission, use the option (“`-`”).

The following are examples of using `chmod`:

Example	**Result**
`chmod u+x`	The owner can execute the file.
`chmod g=rw`	All group members can read and write.
`chmod u=rwx`	The owner has all permissions.
`chmod u=rwx,g=rw,o=r`	All permissions for the owner, read and write for the group, read for all other users.
`chmod +x`	All users (owner, group, others) are allowed to execute the file.
`chmod a+x`	All users (owner, group, others) are allowed to execute the file (“`a`” for all).

In the following example, the user geeko allows the other members of the group users to write to the file `hello.txt` by using `chmod`:

```
geeko@da51:~ > ls -l hello.txt
-rw-r--r-- 1 geeko users 0 2006-04-06 12:40 hello.txt
geeko@da51:~ > chmod g+w hello.txt
geeko@da51:~ > ls -l hello.txt
-rw-rw-r-- 1 geeko users 0 2006-04-06 12:40 hello.txt
```

With the option `-R` (recursive) and a specified directory, you can change the access permission of all files and subdirectories under the specified directory.

Exercise: Manage File Permissions – Part I

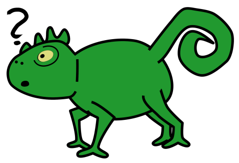
From the directory `Training` and its entire contents, remove the execute permission for "group" and "others".

Besides using letters (`rwx`), you can also use the octal way of representing the permission letters with groups of numbers. Every file and directory in a Linux system has a numerical permission value assigned to it. This value has three digits.

The first digit represents the permission assigned to the file or directory owner. The second digit represents the permission assigned to the group associated with the file or directory. The third digit represents the permission assigned to others.

Each digit is the sum of the following three values assigned to it:

- Read: 4
- Write: 2
- Execute: 1



For example, a file named `MyFile.txt` has the permission of 754.

This means the owner of the file can read, write, and execute the file (4+2+1), the group associated with the file can read and execute the file (4+1), and others can read it (4). If you use numbers, you can add them, as in the following:

Owner	Group	Others
`rwx`	`r-x`	`r--`
`421` (4+2+1=7)	`4-1` (4+1=5)	`4--` (4)

The following are examples of using numbers instead of letters:

Example	Result
`chmod 754 hello.txt`	All permissions for the owner, read and execute for the group, read for all other users (`rwx r-x r--`).
`chmod 777 hello.txt`	All users (user, group, others) can read, write and execute the file (`rwx rwx rwx`).

You also can set the special permission using `chmod`. The following table describes the special file permission for files and directories:

Char.	No.	Name	Files	Directories
`t`	1	Sticky bit	not applicable	Users can only delete files when they are the owner, or when they are root or the owner of the directory. This is usually applied to the directory `/tmp`.
`s`	2	SGID (set GroupID)	When a program is started, this sets the group ID of the process to that of the file's group.	Files created in this directory belong to the directory's group and not to the user's primary group. New directories created in this directory inherit the SGID bit.
`s`	4	SUID (set UserID)	Sets the user ID of the process to that of file's owner when the program is started.	Not applicable

You set the sticky bit with `chmod`, either by using letters (such as `chmod o+t /tmp`) or the numbers (e. g., `chmod 1777 /tmp`).

The sticky bit is listed in the permission for others ("`t`"), as in the following:

```
geeko@da51:~ > ls -ld /tmp
drwxrwxrwt 15 root root 608 2006-04-06 12:45 /tmp
geeko@da51:~ >
```

The following is an example for SUID:

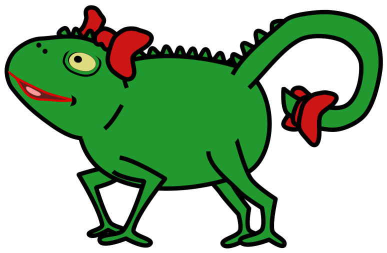
```
geeko@da51:~ > ls -l /usr/bin/passwd
-rwsr-xr-x 1 root shadow 79765 2006-03-24 12:19 /usr/bin/passwd
geeko@da51:~ >
```

Each user is allowed to change his password, but you need root permission to write it into the file `/etc/shadow`. The following is an example for SGID:

```
geeko@da51:~ > ls -l /usr/bin/wall
-rwxr-sr-x 1 root tty 10192 2006-03-22 05:24 /usr/bin/wall
geeko@da51:~ >
```

You can use the command `wall` to send messages to all virtual terminals. It is executed with the permission of the group *tty*.

Attention! These kinds of permissions should be used very carefully. They are only explained here briefly.



Exercise: Manage File Permissions – Part II

What does the following permission values for files mean?

☞ Permission 777

Owner ______________________________

Group ______________________________

Others ______________________________

☞ Permission 755

Owner ______________________________

Group ______________________________

Others ______________________________

☞ Permission 600

Owner ______________________________

Group ______________________________

Others ______________________________

☞ Permission 644

Owner ______________________________

Group ______________________________

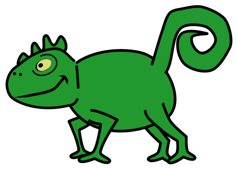
Others ______________________________

☞ Permission 2755

Owner ______________________________

Group ______________________________

Others ______________________________



The user root can use the command `chown` to change the user and group affiliation of a file by using the following syntax:

`chown` *`new_user.new_group file`*

For example:

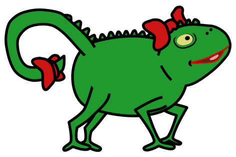
```
da51:~ # chown geeko.users MyFile
da51:~ #
```

To change the owner, but not the group, use the following command syntax:

`chown` *`new_user file`*

For example:

```
da51:~ # chown geeko MyFile
da51:~ #
```



To change the group, but not the user, use the following command syntax:

`chown` *`.new_group file`*

For example:

```
da51:~ # chown .users MyFile
da51:~ #
```

As root, you can also change the file's group affiliation with the command `chgrp`. Use the following syntax:

`chgrp` *`new_group file`*

For example:

```
da51:~ # chgrp users MyFile
da51:~ #
```

A normal user can use the command `chown` to allocate a file of your own to a new group. To do this, use the following syntax:

`chown` *`.new_group file`*

For example:

```
geeko@da51:~> chown .dialout MyFile
geeko@da51:~>
```

You can also use `chgrp`. Use the following syntax:

`chgrp` *`new_group file`*

For example:

```
geeko@da51:~> chgrp dialout MyFile
geeko@da51:~>
```

You can only change your file's group affiliation if you are a member of the new group.

In the following example, root uses `chown` to change the ownership of the file `hello.txt` from geeko to tux:

```
da51:~ # ls -l hello.txt
-rw-r--r-- 1 geeko users 0 2006-04-06 12:43 hello.txt
da51:~ # chown tux.users hello.txt
da51:~ # ls -l hello.txt
-rw-r--r-- 1 tux users 0 2006-04-06 12:43 hello.txt
da51:~ #
```

In the following example, root uses `chown` to limit access to the file `list.txt` for members of the group training:

```
da51:~ # ls -l list.txt
-rw-r----- 1 geeko users 0 2006-04-06 12:43 list.txt
da51:~ # chown .training list.txt
da51:~ # ls -l list.txt
-rw-r----- 1 geeko training 0 2006-04-06 12:43 list.txt
da51:~ #
```

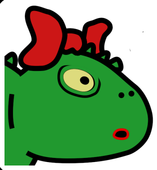
Of course, root and the file owner can still access the file. Although the group has changed, the owner permission remains the same.

The most important option for `chown` and `chgrp` is `-R` to change file owner and owning group recursively. That means that all files in a subdirectory are also changed.

Exercise: Manage File Ownership
Change the owning group of the directory `Training` and its content to *dialout*.

2.2 Understand bash Commands and Features

A modern program has to be graphical with colorful dialogs, buttons, drag & drop functions, etc. The bash shell seems to be unglamorous compared with a Microsoft operating system. Typing on a keyboard seems to be much harder than clicking with a mouse.



Did you ever tryed to change the permission of 120 files using Windows?

Nevertheless, bash provides some functions that simplifys your work. The coolest will be explained in this chapter.

2.2.1 Use Completion of Commands and Filenames

The bash shell supports a function that completes commands and filenames. Just enter the first characters of a command (or a filename) and press Tab. The bash shell completes the name of the command or file. For example, in your home directory press …

☞ M, K, D, Tab → `mkdir`

☞ C, D, ␣, /, H, Tab → `cd /home`

☞ V, I, , ., B, A, Tab → `vi .bash` → _, Tab → `vi .bash_history`

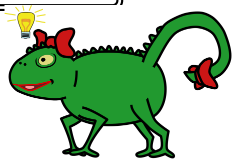
If there is more than one possible match, the bash shell shows all possibilities when you press the Tab key a second time.

For example:

M, K, Tab, Tab →

```
mkallcomposecaches  mkfontdir           mkmanifest
mkcfm               mkfontscale         mknod
mk_cmds             mkhtmlindex         mkpasswd
mkcomposecache      mkhybrid            mktemp
mkdir               mkinfodir           mkzftree
mkdirhier           mkinodedb2          mkzimage_cmdline
mkfifo              mkisofs
```

This feature facilitates entering long filenames.



2.2.2 Use the bash History

bash stores the commands you enter so you have easy access to them. By default, the commands are written into the file `.bash_history` in the user's home directory. The size of this file is set to a maximum of 1,000 entries.

To display the content of the file, use the command `history`.

```
geeko@da51:~> history
    1  ls -i
    2  ls -li
    3  history
geeko@da51:~>
```

To display the commands stored in the history cache (one at a time), use the arrow keys. `Up-arrow` shows the previous command; `Down-arrow` shows the next command. After finding the command you are looking for, edit it as you like; then execute it by pressing `Enter`.

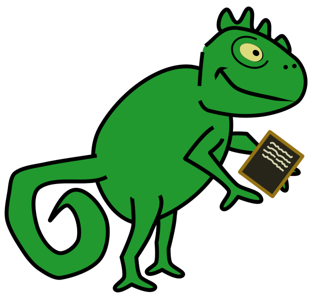
When browsing the entries of the history, you can also select specific commands. Typing one or several letters, or pressing `Page Up` or `Page Down`, displays the preceding or next command beginning with this letter in the history cache.

If you enter a part of the command (not necessarily the beginning of the command), press `Ctrl` + `R` to look for the commands containing the search string in the history list. The commands that match are displayed. The searching process starts with the last command executed.

Exercise: Use the bash History

What are the last commands in your history including the following words:

- ☞ `test`. ____________________
- ☞ `mkdir`. ____________________
- ☞ `mv`. ____________________
- ☞ `cp`. ____________________



2.2.3 Use bash Variables

With shell and environment variables, you are able to configure the behavior of the shell and to customize its environment.

Environment variables control the behavior of a program that is started from a shell. Shell variables control the behavior of shell itself.

The convention is to write environment variables (e. g. PATH) in uppercase letters. If you set your own variables, they should also be written in capitals for the sake of clarity. In shell scripts, use lowercase letters for variable names.

Some important environment variables include the following:

`PATH` – If you start a program, the system looks for it in the directories specified here (the directories are separated by ":"). The order the directories are arranged in is important, since they are searched in turn.

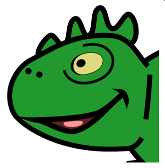
`HOME` – The user's home directory.

`USER` – The user's login name.

To display the value of a shell or environment variable, enter `echo $variable`, as in the following:

```
geeko@da51:~ > echo $HOME
/home/geeko
geeko@da51:~ >
```

For clarity, we will write the "$" in front of the variable's name when we refer to a variable in the following text.



To set the value of a variable or to create a new variable, use the syntax ***variable=value***, as in the following:

```
geeko@da51:~ > MYVAR=myvalue
geeko@da51:~ > echo $MYVAR
myvalue
geeko@da51:~ >
```

The value can be a number, a character, or a string. If the string includes a space, to write the value in full quotes, as in the following:

```
geeko@da51:~ > MYVAR="my value"
geeko@da51:~ > echo $MYVAR
my value
geeko@da51:~ >
```

You can also use `echo` to display text on the screen.

```
geeko@da51:~> echo Hello Geeko
Hallo Geeko
geeko@da51:~>
```

Exercise: Use bash Variables – Part I

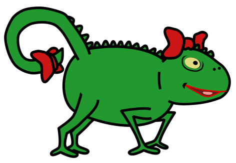
What is the content of the `$PATH` variable on your computer?

____

____

____

____



If you want to print something on the screen, such as
`Variable $HOME is set to` ***`content_of_$HOME`***
the method above does not work.

```
geeko@da51:~> echo Variable $HOME is set to $HOME
Variable /home/geeko is set to /home/geeko
geeko@da51:~>
```

Setting the text in double quotes does not work, either.

```
echo "Variable $HOME is set to $HOME"
Variable /home/geeko is set to /home/geeko
geeko@da51:~>
```

If you use single quotes the result looks like the following.

```
geeko@da51:~> echo 'Variable $HOME is set to $HOME'
Variable $HOME is set to $HOME
geeko@da51:~>
```

There are several solutions, for example:

❶ A combination of quotes

```
geeko@da51:~> echo 'Variable $HOME is set to' "$HOME"
Variable $HOME is set to /home/geeko
geeko@da51:~>
```

❷ A combination of quotes and nonquotes

```
geeko@da51:~> echo 'Variable $HOME is set to' "$HOME"
Variable $HOME is set to /home/geeko
geeko@da51:~>
```

❸ Masking "$" with a backslash

```
geeko@da51:~> echo "Variable \$HOME is set to $HOME"
Variable $HOME is set to /home/geeko
geeko@da51:~>
```

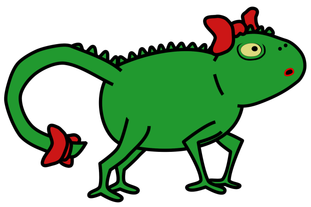
Exercise: Use bash Variables – Part II

Redo all examples in this section by using the variable `$PATH` instead of `$HOME`.

2.2.4 Use bash Aliases

Defining aliases allows you to create shortcuts for commands and their options or to create commands with entirely different names. For example, on a SUSE Linux Enterprise Desktop 10, whenever you enter the commands `dir`, `md`, or `ls`, you are using aliases.

The command `alias` shows you aliases defined on your system. For example, you will see that `dir` is an alias for `ls -l` and `md` is an alias for `mkdir -p`.



The following are examples of aliases that define new commands:

```
geeko@da51:~> alias md
alias md='mkdir -p'
geeko@da51:~> alias dir
alias dir='ls -l'
```

To find out if a command is an alias, use the `type` command. `type` tells you whether the command specified is a built-in shell command, a regular command, a function, or an alias.

For regular commands, the `type` lists the path to the corresponding executable. For aliases, it lists the elements aliased:

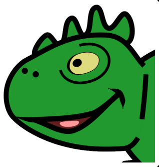
```
geeko@da51:~> type -a ls
ls is aliased to `/bin/ls $LS_OPTIONS'
ls is /bin/ls
```

Explanation: The above example shows that `ls` is an alias used to add some options to the command. The `-a` option shows both the contents of the alias and the path to the original `ls` command. The output shows that `ls` uses the options stored in the variable `LS_OPTIONS`. These options are responsible for `ls` lists different file types in different colors.

Most of the system aliases are defined in the file `/etc/bash.bashrc`. To define aliases, use the **`alias`** command. To delete them, use the **`unalias`** command.

To define an alias, use the following syntax:
`alias` *`aliasname`*`="`*`command options`*`"`

To delete an alias, use the following syntax:
`unalias` *`aliasname`*



An alias defined in this way is only valid for the current shell, as in the following:

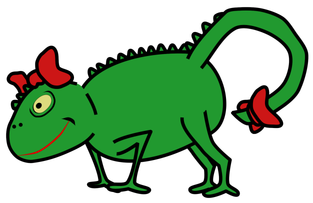
```
geeko@da51:~> alias ls="echo Hello"
geeko@da51:~> ls
Hello
geeko@da51:~> bash
geeko@da51:~> ls
bin   Desktop   Documents   public_html
geeko@da51:~>
```

But if you terminate this shell, the alias is gone. To avoid this, store the alias in one of the shell's configuration files. On the SUSE Linux Enterprise Desktop, the file `~/.alias` is created for aliases defined by each user.

This file is read in by `~/.bashrc`, where a command is included to that effect. Aliases are not relevant to shell scripts at all, but can be a real time saver when using the shell interactively.

Exercise: Use bash Aliases

Create a new alias `spy` that displays the content of a `tar` archive compressed with `gzip`.



2.2.5 Use Search Patterns to Expand Names

Occasionally, you might want to perform operations on a series of files without having to name all the files. In this case, you can use of the following search patterns:

☞ ?

Replaces a single character (except "/").

```
geeko@da51:~> ls /bin/s?
/bin/sh  /bin/su
geeko@da51:~>
```

☞ *

Replaces any string length, including zero characters (except "." at the beginning of a file name and "/").

```
geeko@da51:~> ls /bin/s*
/bin/sash      /bin/setkeycodes   /bin/sh                 /bin/sort
/bin/scsidev   /bin/setleds       /bin/showconsolefont    /bin/stty
/bin/sed       /bin/setmetamode   /bin/showkey            /bin/su
/bin/setfont   /bin/setserial     /bin/sleep              /bin/sync
geeko@da51:~>
```

☞ [0-9]

Replaces any of the characters enclosed (here: numbers from 0 to 9).

```
geeko@da51:~> ls /bin/s[e-h]*
/bin/sed          /bin/setleds      /bin/sh
/bin/setfont      /bin/setmetamode  /bin/showconsolefont
/bin/setkeycodes  /bin/setserial    /bin/showkey
geeko@da51:~>
```

☞ [abcd]

Replaces one of the characters a, b, c and d.

```
geeko@da51:~> ls /bin/s[abcd]*
/bin/sash  /bin/scsidev
```

☞ [a-ew-z]

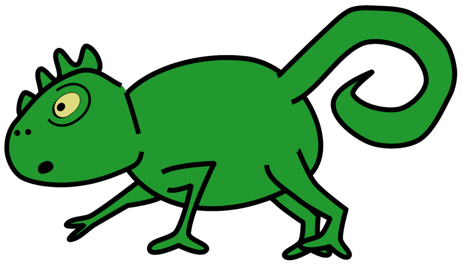
Replaces any character from the ranges a-e and w-z.

```
geeko@da51:~> ls /bin/s[a-cm-t]*
/bin/sash  /bin/scsidev  /bin/sort  /bin/stty
geeko@da51:~>
```

☞ [!abc]

Replaces any character, except a, b, and c.

```
geeko@da51:~> ls /bin/s[!eh]*
/bin/sash  /bin/scsidev  /bin/sleep  /bin/sort  /bin/stty  /bin/su  /bin/sync
geeko@da51:~>
```



Attention! The meaning of some search patterns differs from their meaning as regular expressions (see 2.3.2 on page 89).

If you use search patterns (wild cards), the shell compares them with all filenames and, if they match, replaces the expression with all filenames found.

Attention! The order of characters is different for normal users and for the user root, because both use a different character encoding by default.

For normal users (UTF-8 encoding) the lowercase character follows the uppercase character immediately. For root (POSIX encoding) all the lowercase characters are behind the uppercase characters. This means that between "A" and "C" in POSIX, the only character is "B". But, in UTF-8, there are "a," "B," and "b" in between.

The variable `$LANG` specifies the language. In the following example, the language is set to US English using UTF-8 encoding.

```
geeko@da51:~> echo $LANG
en_US.UTF-8
geeko@da51:~>
```

UTF-8 (Unicode) handles all kind of letters and not only Latin letters, while POSIX (ASCII) only does Latin letters.

Exercise: Use Search Patterns to Expand Names

How many programs in the `/bin` directory have a name...

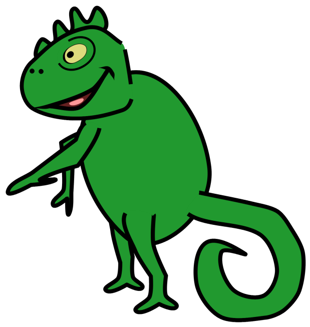
... that begins with "sa"? ______________________

... that begins with "x", "y", or "z"? ______________

... that ends with "s"? ________________________

... with three letters? _________________________

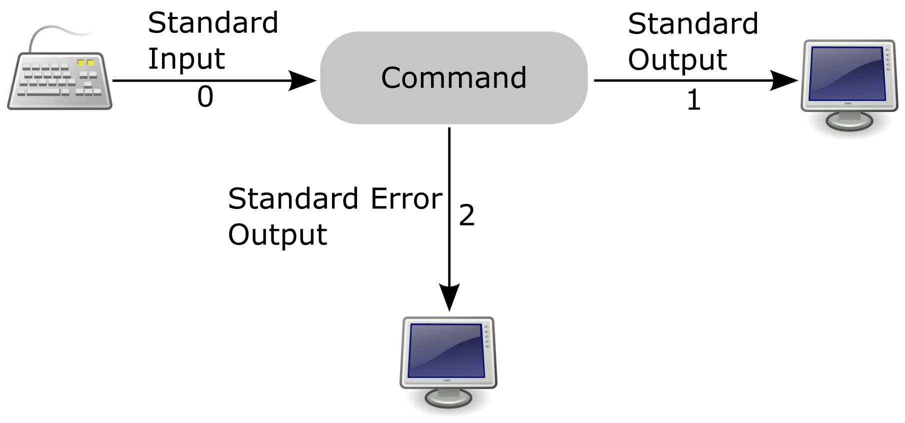
... that include at least one number? _____________

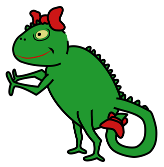


2.3 Understand Advanced Command Execution

2.3.1 Use Piping and Redirection

Linux has three standard data channels:



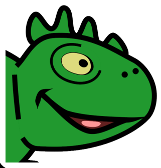


The following describes these channels:

Channel	Number	Description
Standard input (stdin)	0	The currently running program reads the input from this channel (usually the keyboard).
Standard output (stdout)	1	The program sends its output to this channel (usually the monitor).
Standard error output (stderr)	2	Errors are issued through this channel (usually the monitor).

Each channel can be redirected by the shell. For example, standard input can come from a file or standard output and standard error output can be directed to a file. The following are the redirection characters:

`<`	– Redirects standard input.
`>`	– Redirects standard output ("`>`" without a preceding number is just an abbreviation for "`1>`").
`2>`	– Redirects standard error output.



"`>`" overwrites an existing file. If the output should be appended to an existing file, you have to use "`>>`" or "`2>>`".

The following is an example of a standard input, standard output, and standard error output. The default behavior of the shell is shown here:

```
1 geeko@da51:~ > ls /opt /recipe
2 /bin/ls: /recipe: No such file or directory
  /opt:
4 gnome kde3
```

Explanation: The content of the directory `/opt` is shown in line 3 and 4. The directory `/recipe` does not exist. An error message is shown in line 2.

If the standard error output is redirected to `/dev/null`, only the standard output is displayed on the screen:

```
1 geeko@da51:~ > ls /opt /recipe 2> /dev/null
2 /opt:
  gnome kde3
```

To redirect standard output and standard error output to a file (such as "output"), enter the following:

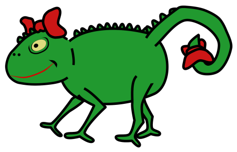
```
1 geeko@da51:~ > ls /opt /recipe > output 2>&1
2 geeko@da51:~ >
```

Explanation: First, the standard output is redirected to the file `output` (`> output`); then the standard error output is directed to the standard output (`2>&1`). The "`&`" refers to the file descriptor that follows ("`1`" for the standard output).

To display the contents of the file list, use the command `cat`, as in the following:

```
1 geeko@da51:~> cat list
2 /bin/ls: /recipe: No such file or directory
  /opt:
4 gnome
  kde3
```

This option of process communication is available not only in the shell, but can also be used in programs directly. All known files in the system can be used as input or output.



Occasionally, you might want to use a file as input for a program that expects input from the keyboard. To do this, the standard input is redirected, as in the following:

```
1 geeko@da51:~ # echo "Hello Tux,
2 >
  > how are you?
4 > Is everything okay?" > greetings
  geeko@da51:~ # mail tux < greetings
```

Explanation: First, the text is redirected to the file `greetings` through the command ">" (lines 1–3). The mail program `mail` receives its input from the file `greetings` (not from the keyboard), and then the email program sends the email to the user geeko (line 4).

Exercise: Use Piping and Redirection – Part I

Create a file labeled `Redirection` in your home directory that includes. . .

❶ . . . the output of the command `ls /`; followed by . . .
❷ . . . the content of the variable `$PATH`; followed by . . .
❸ . . . the error message of the command `cd /root`.

The output of one command can be used as the input for another command by using the pipe ("`|`"):
`command1 | command2`

In a pipe, a maximum of 4 KB of unprocessed data can exist. If the process creating the output tries to write to a full pipe, it is stopped and only allowed to continue if the writing process can be completed. On the other side, the reading process is stopped if it tries to read an empty pipe.

```
geeko@da51:~ > ls -l /etc | less
```

Occasionally, you might want output from a command displayed on the screen and written to a file. To do this, use the command `tee`:

```
geeko@da51:~ > ls -l | tee output
```

In this example, the output of the command is displayed on the screen as well as written to the file `output`.

To redirect the output of several consecutive commands on the command line, the commands must be separated with semi-colons and enclosed in parentheses (*`command1`*`;` *`command2`*`;...)`:

```
geeko@da51:~> (id ; ls ~) > output
geeko@da51:~> cat output
uid=1000(geeko) gid=100(users)
groups=14(uucp),16(dialout),33(video),100(users)
bin
Desktop
Documents
output
public_html
geeko@da51:~>
```

Exercise: Use Piping and Redirection – Part II

The command `wc` (word count) counts words in a text file. With the option `-l` it counts lines.

Pipe the output of the command `ls -l` into the command `wc -l`.

What does the result of this construction mean?

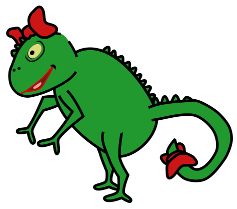
____

____

____

The shell starts a separate subshell for processing the individual commands. To redirect the linked commands, the shell must be forced to execute the command chain in the same subshell by enclosing the expression in parentheses.

Upon completion, every program returns a value that states the success of the execution. If this return value is 0, the command completed successfully. If an error occurred, the return value is greater than 0. (Depending on the program, different return values indicate different errors.)



You can use the command `echo $?` to display a return value.

```
geeko@da51:~> ls
bin  Desktop  Documents  public_html
geeko@da51:~> echo $?
0
geeko@da51:~>
```

The return value can be used to trigger the execution of another command:

Link	Result
command1 && *command2*	*command2* is only executed if *command1* is completed without any errors.
command1 || *command2*	*command2* is only executed if *command1* is completed with an error.

The following illustrates using both "||" and "&&":

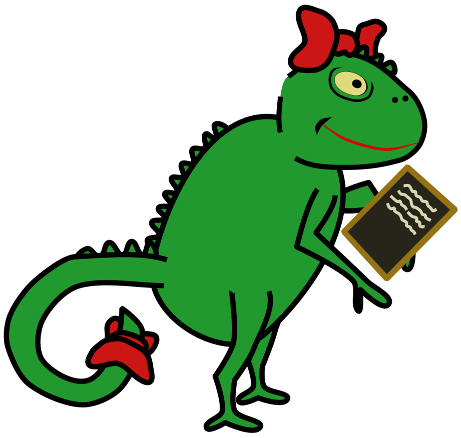
```
geeko@da51:~> ls recipe || ls ~
/bin/ls: recipe: No such file or directory
bin Desktop Documents output public_html test
geeko@da51:~> ls recipe && ls ~
/bin/ls: recipe: No such file or directory
geeko@da51:~>
```

Explanation: The file recipe does not exist and the command `ls recipe` leads to an error (lines 1–2). Because of this, the command `ls ~` in line 1 is executed and in the line 4 is not.

Exercise: Use Piping and Redirection – Part III

What is the return value of the following commands?

- `ls -l | wc -l` ______________________________
- `cd /root` ______________________________
- `less wall` ______________________________
- `echo $LOREM` ______________________________
- `echo $?` ______________________________



2.3.2 Understand Regular Expressions

Regular expressions are strings consisting of metacharacters and literals (regular characters and numerals). In the context of regular expressions, metacharacters are those characters that do not represent themselves but have special meanings.

They can act as placeholders for other characters or can be used to indicate a position in a string.

Attention! It is important to remember that some metacharacters used by the shell for filename expansion have a meaning different from the one discussed here.

Many commands rely on regular expressions for pattern matching, for example, the command `grep` and its variant `egrep`. These commands are used to search files for certain patterns using the syntax
`grep` ***`search_pattern filename`***

The command searches ***`filename`*** for all text that matches ***`search_pattern`*** and prints the lines that contains the pattern. You can also specify several files, in which case the output will not only print the matching line, but also the corresponding filenames.

Several options are available to specify that only the line number should be printed, for instance, or that the matching line should be printed together with leading and trailing context lines.

You can specify search patterns in the form of regular expressions, although the basic `grep` command is limited in this regard. To search for more complex patterns, use the `egrep` command (or `grep -E`) instead, which accepts extended regular expressions.

As a simple way to deal with the difference between the two commands, make sure you use `egrep` in all of your shell scripts. The regular expressions used with `egrep` need to comply with the standard syntax of regular expressions.

You can read details on this topic in the manual page of `grep`.

To avoid having special characters in search patterns interpreted by the shell, enclose the pattern in quotation marks.

The following is an example of using `egrep` and `grep`:

```
geeko@da51:~> egrep (b|B)lurb file*
bash: syntax error near unexpected token ‘|’
geeko@da51:~> grep "(b|B)lurb" file*
geeko@da51:~> egrep "(b|B)lurb" file*
file1:blurb
filei2:Blurb
geeko@da51:~>
```

On the book CD, you can find an archive file "grep-demo.tar.gz" with text files with some meaningless content. We will use these text files for the following explanations.
Uncompress the archive into your home directory.

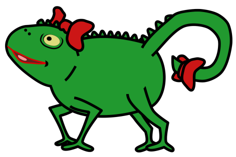
The following are options you can use with the command **grep**:

☞ `-i`

Enables case insensitivity.

```
 1 geeko@da51:~> grep "duis" *.txt
 2 loremipsum.txt:delenit au gue duis dolore te feugat nulla facilisi.
   loremipsum.txt:au gue duis dolore te feugat nulla facilisi.
 4 geeko@da51:~> grep -i "duis" *.txt
   loremipsum.txt:Duis te feugifacilisi.
 6 loremipsum.txt:Duis autem dolor in hendrerit
   loremipsum.txt:delenit au gue duis dolore te feugat nulla facilisi.
 8 loremipsum.txt:Duis te feugifacilisi per suscipit
   loremipsum.txt:Duis te feugifacilisi.
10 loremipsum.txt:Duis autem dolor in hendrerit in
   loremipsum.txt:au gue duis dolore te feugat nulla facilisi.
12 geeko@da51:~>
```

Explanation: The first **grep** command (line 1) finds two lines in the text file (lines 2–3). With the option **-i** (line 4), **grep** ignores upper and lower case and lists the lines with a capital "D" in the word "duis" (lines 5–11).



☞ `-l`

Shows only the names of files that contain the search string.

```
geeko@da51:~> grep "duis" *.txt
loremipsum.txt:delenit au gue duis dolore te feugat nulla facilisi.
loremipsum.txt:au gue duis dolore te feugat nulla facilisi.
geeko@da51:~> grep -l "duis" *.txt
loremipsum.txt
geeko@da51:~>
```

Explanation: If `grep` finds more than one file, it writes the filename at the beginning of each line (lines 2–3). Using the option `-l` (line 4), only the filename is written on the screen (line 5).

☞ `-r`

Searches entire directory trees recursively. This option only works if you specify a directory and not files in the the `grep` command.

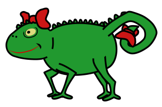
```
1 geeko@da51:~> grep "duis" *
2 loremipsum.txt:delenit au gue duis dolore te feugat nulla facilisi.
  loremipsum.txt:au gue duis dolore te feugat nulla facilisi.
4 geeko@da51:~> grep -r "duis" *
  dirgrep/loremipsum2.txt:delenit au gue duis dolore te feugat nulla facilisi.
6 dirgrep/loremipsum2.txt:au gue duis dolore te feugat nulla facilisi.
  loremipsum.txt:delenit au gue duis dolore te feugat nulla facilisi.
8 loremipsum.txt:au gue duis dolore te feugat nulla facilisi.
  geeko@da51:~>
```

Explanation: The current directory is scanned in line 1. Only the hits of the file `loremipsum.txt` are shown in the output (lines 2–3). With the option `-r` (line 4), the subdirectory `dirgrep` is also scanned and the hits in the file `loremipsum2.txt` are listed (lines 5–8).

☞ `-v`

Shows all lines that do not contain the search string.

```
geeko@da51:~> grep "duis" *.txt
loremipsum.txt:delenit au gue duis dolore te feugat nulla facilisi.
loremipsum.txt:au gue duis dolore te feugat nulla facilisi.
geeko@da51:~> grep -v "duis" *.txt
loremipsum.txt:LOREM IPSUM DOLOR SIT AMET
loremipsum.txt:
loremipsum.txt:Lorem ipsum dolor sit amet,
loremipsum.txt:consectetuer adipiscing elit,
...
numbers.txt:0
numbers.txt:98798
numbers.txt:765765
geeko@da51:~>
```



☞ `-n`

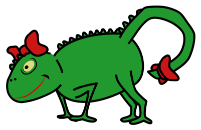
Shows the line numbers.

```
geeko@da51:~> grep -n "duis" *.txt
loremipsum.txt:18:delenit au gue duis dolore te feugat nulla facilisi.
loremipsum.txt:39:au gue duis dolore te feugat nulla facilisi.
geeko@da51:~>
```

☞ `-h`

Shows no filenames.

```
geeko@da51:~> grep -h "duis" *.txt
delenit au gue duis dolore te feugat nulla facilisi.
au gue duis dolore te feugat nulla facilisi.
geeko@da51:~>
```



The following list shows the most important metacharacters and their meanings:

☞ ^

The beginning of the line.

```
geeko@da51:~> grep -n "^et" *.txt
loremipsum.txt:16:et accumsan et iusto odio dignissim
loremipsum.txt:37:et iusto odio dignissim qui blandit
geeko@da51:~>
```

Explanation: With `^et`, "et" is matched if it appears at the beginning of the line.

☞ $

The end of the line.

```
geeko@da51:~> grep -n "at$" *.txt
loremipsum.txt:14:vel illum dolore eu feugiat
geeko@da51:~>
```

Explanation: With `$at`, "at" is matched if it appears at the end of line.

☞ \<

The beginning of the word

```
geeko@da51:~> grep -n "\<se" *.txt
loremipsum.txt:5:sed diem nonummy nibh euismodtincidunt
loremipsum.txt:27:sed diem nonummy nibh euismodtincidunt
geeko@da51:~>
```

Explanation: With `\<se`, all words are matched that start with "se".

☞ \>

The end of the word.

```
geeko@da51:~> grep -n "se\>" *.txt
loremipsum.txt:13:in vulputate velit esse molestie consequat,
loremipsum.txt:34:vulputate velit esse molestie consequat,
geeko@da51:~>
```

Explanation: With `se\>`, all words are matched that end with "se".

☞ `[`*`characters`*`]`

One character from the set.

```
geeko@da51:~> grep -n "[ex]e" *.txt
loremipsum.txt:6:ut lacreet dolore magna aliguam erat volutpat.
loremipsum.txt:8:quis nostrud exerci tution ullamcorper
loremipsum.txt:20:quis nostrud exerci taion
loremipsum.txt:28:ut lacreet dolore magna aliguam erat volutpat.
loremipsum.txt:30:quis nostrud exerci tution ullamcorper
geeko@da51:~>
```

Explanation: `[ex]e` matches all lines containing "ee" or "xe".

Using a hyphen you can specify a range of characters.

```
geeko@da51:~> grep -n "^[a-c]" *.txt
loremipsum.txt:4:consectetuer adipiscing elit,
loremipsum.txt:26:adipiscing elit,
loremipsum.txt:39:au gue duis dolore te feugat nulla facilisi.
geeko@da51:~>
```

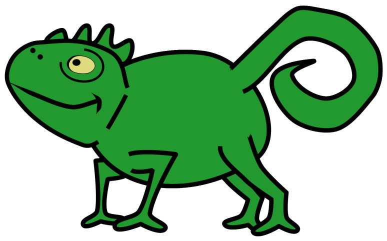
Explanation: `^[a-c]` matches all lines beginning with the characters "a", "b", or "c".

☞ `[^`*`characters`*`]`

None of the specified characters.

```
geeko@da51:~> grep -n "^[^a-c]" *.txt
loremipsum.txt:1:LOREM IPSUM DOLOR SIT AMET
loremipsum.txt:3:Lorem ipsum dolor sit amet,
loremipsum.txt:5:sed diem nonummy nibh euismodtincidunt
...
numbers.txt:11:0
numbers.txt:98798
numbers.txt:765765
geeko@da51:~>
```

Explanation: The `grep` command matches all lines that do not begin with the characters "a", "b", or "c".

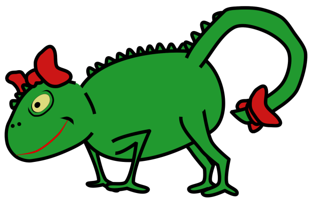


☞ .

Any single character.

```
geeko@da51:~> grep -n "^.$" *.txt
numbers.txt:11:0
geeko@da51:~>
```

Explanation: `^.$` matches all lines with only one character (between the begining and ending of the line).

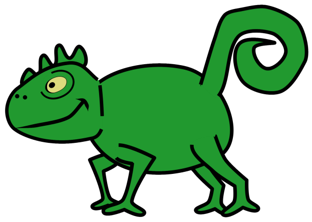


☞ + (Works only with `egrep`)

One or more of the preceding expression.

```
geeko@da51:~> egrep -n "^0+$" *.txt
numbers.txt:1:0000
numbers.txt:11:0
geeko@da51:~>
```

Explanation: `^0+$` matches all lines with one or more “0” (between the begining and ending of the line).



☞ *

Any number (including none) of the preceding single character.

```
geeko@da51:~> grep -n "^0*$" *.txt
loremipsum.txt:2:
numbers.txt:1:0000
numbers.txt:11:0
geeko@da51:~>
```

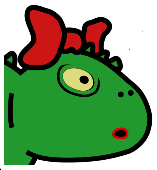
Explanation: `^0*$` matches all lines with zero, one or more “0” (between the begining and ending of the line).

☞ {*min*,*max*} (Works only with `egrep`)

The preceding expression appears ***min*** times at minimum and ***max*** times at maximum.

```
1 geeko@da51:~> egrep -n "[0-9]{4,5}" *.txt
2 numbers.txt:1:0000
  numbers.txt:12:98798
4 numbers.txt:13:765765
  geeko@da51:~>
```

Explanation: `[0-9]{4,5}` matches all lines with four (at minimum) and five (at maximum) numbers.



Are you surprised why there is a six-digit number matched as well in line 4?
The mistake with the previous expression is, that we did not specify what is before or after the fourth or fifth numbers. Have a look the following listing.

```
1 geeko@da51:~> egrep -n "^[0-9]{4,5}$" *.txt
2 numbers.txt:1:0000
  numbers.txt:12:98798
4 geeko@da51:~>
```

☞ `|` (Works only with `egrep`)

The expression before or after the pipe.

```
1 geeko@da51:~> egrep -n "ea|gue" *.txt
2 loremipsum.txt:10:ut aliquip ex ea commodo consequat.
  loremipsum.txt:18:delenit au gue duis dolore te feugat nulla facilisi.
4 loremipsum.txt:31:suscipit lobortis nisl ut aliquip ex ea commodo consequat.
  loremipsum.txt:39:au gue duis dolore te feugat nulla facilisi.
6 geeko@da51:~>
```

Explanation: `ea|gue` matches all lines with the expression “ea” or “gue”.

☞ `(...)` (Works only with **egrep**)

Enclose alternatives for grouping with others.

```
geeko@da51:~> egrep -n "(d|D)uis" *.txt
loremipsum.txt:11:Duis te feugifacilisi.
loremipsum.txt:12:Duis autem dolor in hendrerit
loremipsum.txt:18:delenit au gue duis dolore te feugat nulla facilisi.
loremipsum.txt:23:Duis te feugifacilisi per suscipit
loremipsum.txt:32:Duis te feugifacilisi.
loremipsum.txt:33:Duis autem dolor in hendrerit in
loremipsum.txt:39:au gue duis dolore te feugat nulla facilisi.
geeko@da51:~>
```

Explanation: `(d|D)uis` matches “duis” and “Duis”.

☞ `?` (Works only with **egrep**)

Zero or one of the preceding characters.

```
geeko@da51:~> egrep -n "it,$" *.txt
loremipsum.txt:4:consectetuer adipiscing elit,
loremipsum.txt:26:adipiscing elit,
geeko@da51:~> egrep -n "it,?$" *.txt
loremipsum.txt:4:consectetuer adipiscing elit,
loremipsum.txt:12:Duis autem dolor in hendrerit
loremipsum.txt:23:Duis te feugifacilisi per suscipit
loremipsum.txt:26:adipiscing elit,
loremipsum.txt:37:et iusto odio dignissim qui blandit
loremipsum.txt:38:praesent luptatum zzril delenit
geeko@da51:~>
```

Explanation: `it,$` (line 1) matches all lines that end with “it,”. `it,?$` (line 1) matches all lines that end with “it,” or “it”.

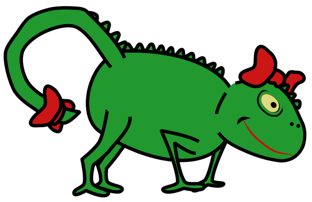
☞ `\`

Masks the following character to remove its special meaning.

```
geeko@da51:~> grep -n "\.$" *.txt
loremipsum.txt:6:ut lacreet dolore magna aliguam erat volutpat.
loremipsum.txt:10:ut aliquip ex ea commodo consequat.
loremipsum.txt:11:Duis te feugifacilisi.
loremipsum.txt:18:delenit au gue duis dolore te feugat nulla facilisi.
loremipsum.txt:22:nisl ut aliquip ex en commodo consequat.
loremipsum.txt:24:lobortis nisl ut aliquip ex en commodo consequat.
loremipsum.txt:28:ut lacreet dolore magna aliguam erat volutpat.
loremipsum.txt:31:suscipit lobortis nisl ut aliquip ex ea commodo consequat.
loremipsum.txt:32:Duis te feugifacilisi.
loremipsum.txt:39:au gue duis dolore te feugat nulla facilisi.
geeko@da51:~>
```

Explanation: `\.$` matches all lines that end with a full stop. Without masking the full stop represents any single character (see page 94).

To learn more about the structure of regular expressions, read the corresponding manual page with `man 7 regex`.



Exercise: Understand Regular Expressions – Part I

What does the following regular expressions mean? How many lines match from the file `loremipsum.txt`?

- `^.t` ______
- `it\>` ______
- `m{2}` ______
- `[^.,]$` ______

Exercise: Understand Regular Expressions – Part II

Create regular expressions that match the following demands. How many hits do you get in the file `loremipsum.txt`?

- All lines with at least one upper-case character.
- All lines with the word “in” or “ad”.
- All lines with a three letter word.

2.3.3 Manage Shell Processes

The Linux shell environment allows processes to run either in the foreground or in the background.

Processes executed in the foreground are started in a terminal window and run until the process completes; the terminal window does not return to a prompt until the program's execution is complete.

Background process execution occurs when a process is started and the terminal window returns to a prompt before the process completes.

Existing processes can be switched from foreground to background execution under the following circumstances:

- ☞ The process must be started in a terminal window or console shell.
- ☞ The process does not require input from the terminal window.

If the process meets this criteria, it can be moved to the background.

Attention! Processes that require input within the terminal can be moved to the background as well, but when input is requested, the process will be suspended until it is brought to the foreground and the requested input is provided.

Commands in a shell can be started in the foreground or in the background. Processes in the foreground can directly receive transmitted signals.

For example, if you enter `xeyes` to start the `xeyes` program, it is running in the foreground. If you press Ctrl + Z, the process stops:

```
[1]+ Stopped xeyes
geeko@da51:~>
```

You can continue running a stopped process in the background by entering `bg`, as in the following:

```
geeko@da51:~> bg
[1]+ xeyes &
geeko@da51:~>
```

The ampersand (line 2) displayed in the output means that the process is now running in the background. Appending an ampersand to a command starts the process in the background (instead of the foreground), as in the following:

```
geeko@da51:~> xeyes &
[2] 4351
geeko@da51:~>
```

With this, the shell from which you started the program is available again for user input immediately.

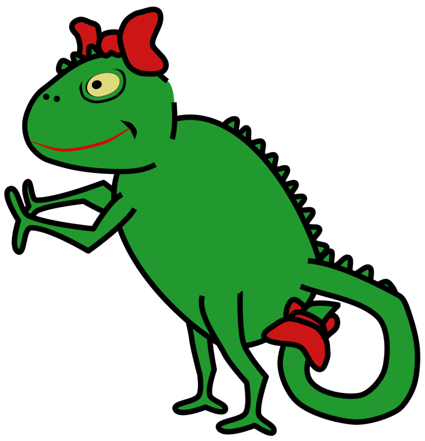
In the above example, both the job ID (`[2]`) and the process ID of the program (`4351`) are returned.

Each process started from the shell is assigned a job ID by the job control of the shell. The command jobs lists the contents of job control, as in the following:

```
geeko@da51:~> jobs
[1]+ Stopped xeyes
[2] Running xeyes &
[4]- Running sleep 99 &
geeko@da51:~>
```

In this example, the process with job ID 3 is already terminated. The processes 2 and 4 are running in the background (notice the ampersand), and process 1 is stopped.

The "+" sign indicates the process that will respond to `fg` without options, and the "-" sign indicates the process that inherits the "+" sign once the process with the "+" sign ends.



The next background process will be assigned the job ID of 5 (highest number+1).

Not only can you continue running a stopped process in the background by using the command `bg`, you can also switch a process to the foreground by entering `fg` *`job_ID`*, as in the following:

```
geeko@da51:~> fg 1
xeyes
```

The shell also informs you about the termination of a process running in the background:

```
[4]- Done sleep 99
```

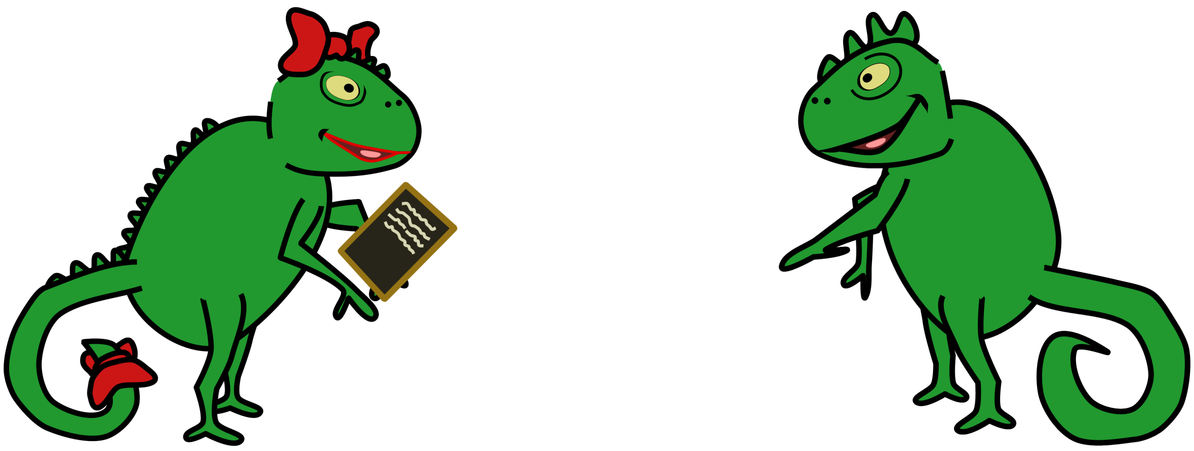
The job ID is displayed in square brackets. `Done` means the process terminated properly. If you see `Terminated` instead, it means that the process was requested to terminate. `Killed` indicates a forceful termination of the process.

Exercise: Manage Shell Processes

Start the Acrobat Reader showing a PDF file from the command line, by entering:
`acroread /usr/share/doc/manual/sled-gnome-user_en/sled-gnome-user_en.pdf`

Stop the process. What happens when you try to use the Acrobat Reader?

Move the acroread process in the backgroud. What happens when you try to use the Acrobat Reader?



2.4 The vi Text Editor

Because most of the services of a Linux computer are configured by editing an ASCII file, you need a text editor. There are a lot of text editors available in Linux, for example:

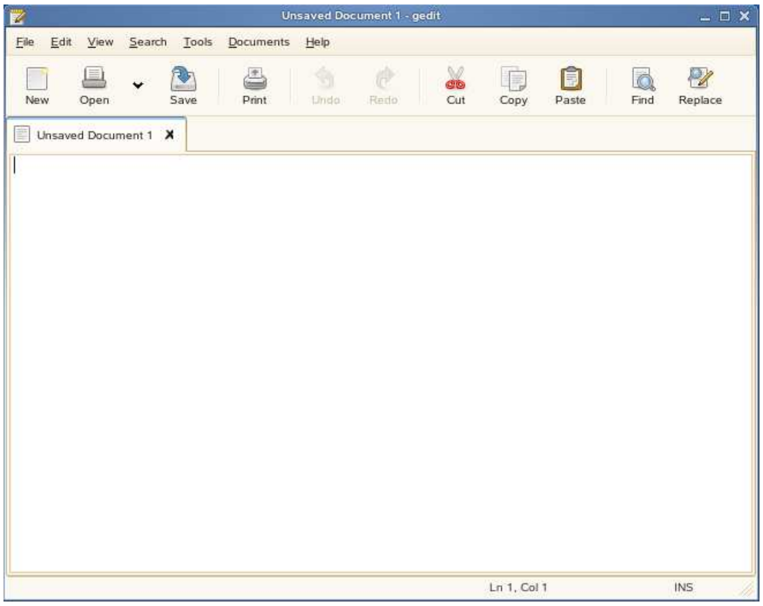
- ☞ vi
- ☞ emacs
- ☞ xemacs
- ☞ xedit
- ☞ gedit
- ☞ kwrite

Every text editor has advantages and disadvantages. There are two kinds of editors:

- ☞ Command line editors
- ☞ Graphical editors

The main advantage of command line editors is that you do not need a graphical user interface to use them. They run in a shell.

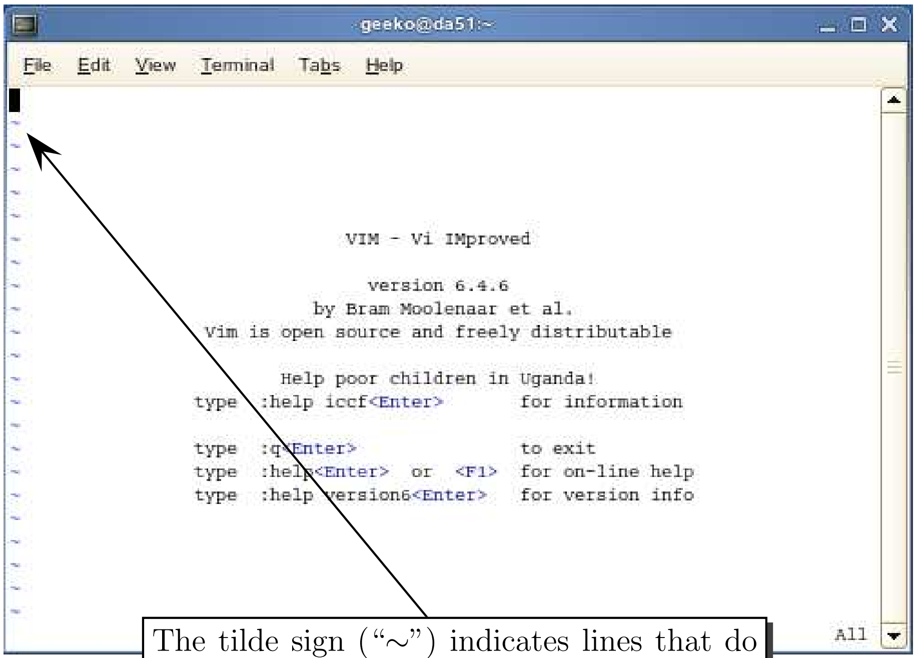
Graphical editors are (normally) easy to use and do not need big explanations. An example is the editor gedit that can be started from the main menu (application group `Tools`).


Unsaved Document 1 - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
Unsaved Document 1
Ln 1, Col 1
INS

Although many factors can be involved when selecting an editor for everyday use, the reason vi is used by most administrators is that it is available on every Linux and UNIX system. Because of this, you should be able to use vi. In SUSE Linux Enterprise Desktop, vim (vi improved) by Bram Moolenaar is the standard vi editor. When you enter `vi`, vim is started via a link to it.

2.4.1 Start vi

You can start vi by entering `vi` or `vim`.

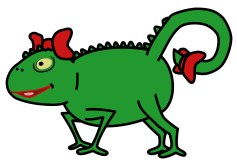


```
~
~
~
~
~
~                    VIM - Vi IMproved
~
~                     version 6.4.6
~                 by Bram Moolenaar et al.
~         Vim is open source and freely distributable
~
~                 Help poor children in Uganda!
~       type  :help iccf<Enter>       for information
~
~       type  :q<Enter>               to exit
~       type  :help<Enter>  or  <F1>  for on-line help
~       type  :help version6<Enter>   for version info
~
~
~
~
~
                                                    All
```

The tilde sign ("~") indicates lines that do not exist yet. The cursor is on the first line.

You can also enter `vi` followed by various options. To edit a file, enter `vi` and the name of a file, as in the following example: `vi exercise`

If a file does not yet exist, it is created. The text of the file appears in an editor at the command line.



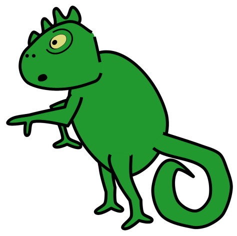
2.4.2 Move the Cursor

You can move the cursor with the K, J, H, and L keys.

Exercise: Move Cursor in vi – Part I

Open a text file with vi. Insert the direction the cursor moves by pressing the following keys:

Key	Direction
K	
J	
H	
L	



If available, you also can use the arrow keys (Up-arrow, Down-arrow, Left-arrow, and Right-arrow) to move the cursor.

With Ctrl + V you can move one screen page down, with Alt + V one screen page up.

Some more interesting keys for moving the cursor:

E	– Moves the cursor to the end of the current word.
W	– Moves the cursor to the beginning of the next word.
B	– Moves the cursor to the beginning of the previous word.
0	– Moves the cursor to the beginning of the current line.
$	– Moves the cursor to the end of the current line.
G + G	– Moves the cursor to the beginning of the first line.
Shift + G	– Moves the cursor to the beginning of the last line.

Exercise: Move Cursor in vi – Part II

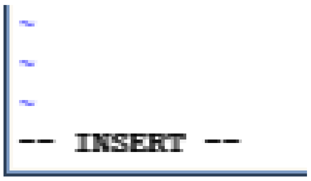
Open a text file with vi and practice the movements mentioned above.

2.4.3 Learn the Working Modes

In contrast to many other editors, vi is mode-oriented. When vi is first started, it is in normal mode. Anything you enter in this mode is considered a command. You must switch to input mode before you can type any text. This can be frustrating to users who are unfamiliar with vi.

In addition to switching modes, you must learn which keys perform which actions, because you cannot use the mouse. However, the number of commands needed for everyday work is fairly small, and you can get used to them quickly.

To enter text, you must first switch the editor to input mode by typing `I` (insert) or pressing the `Ins` key. At the bottom of the screen, you see the message –INSERT–.



```
~
~
~
-- INSERT --
```

Press `Esc` once to take you back to the normal mode. From normal mode you can switch to command-line mode by entering "`:`". The cursor jumps to the last line after ":" and waits for a command entry.

A command will only be carried out in command-line mode after you press `Enter`. Then you are automatically back in normal mode.

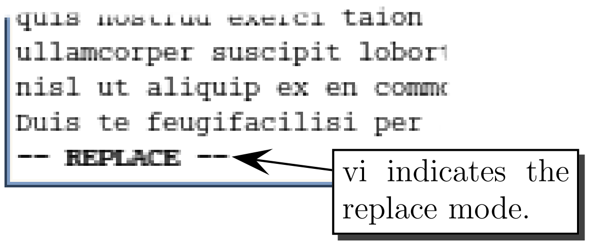
The following is a summary of the available modes (the first three are the most important ones):

Normal mode – When vi starts, it is automatically in this mode. In normal mode, vi can be given commands. E. g. the command "`i`" (entered by pressing `I`) puts it into insert mode and the command "`:`" switches it to command-line mode.

Sometimes the normal mode is also called "Command mode".

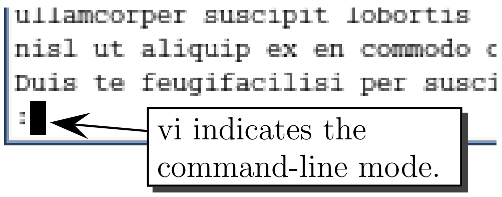
Insert mode – In this mode, vi accepts all input as text. You can enter the insert mode for example by pressing `I` or `A` in the normal mode. Return to normal mode with `Esc`.

A special variant of the insert mode is the "replace mode", where the entered text replaces the previous text. You can switch to replace mode with `Shift`+`R`.

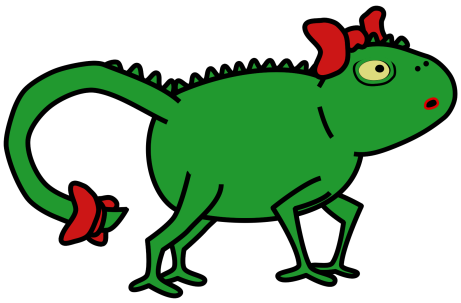


```
quis nostrud exerci taion
ullamcorper suscipit lobor
nisl ut aliquip ex en commo
Duis te feugifacilisi per
-- REPLACE --
```

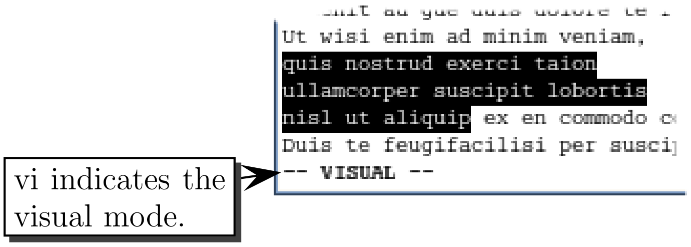
Command-line mode – In this mode, vi accepts commands from the command line. You can enter the command-line mode for example by pressing `:`, `/` or `?`. Pressing `Enter` causes the command to be executed and automatically returns to the normal mode. `Esc` aborts the command-line mode and returns to normal mode.



```
ullamcorper suscipit lobortis
nisl ut aliquip ex en commodo c
Duis te feugifacilisi per susci
:█
```

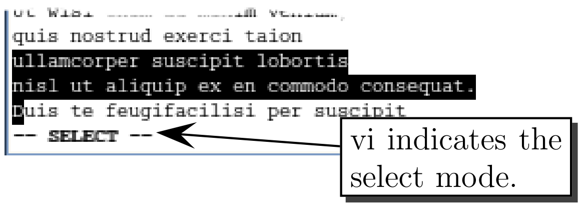


Visual mode – In this mode you can select text for editing, copying and so on. You can enter the visual mode for example by pressing V in the normal mode. You can return to normal mode by pressing Esc.



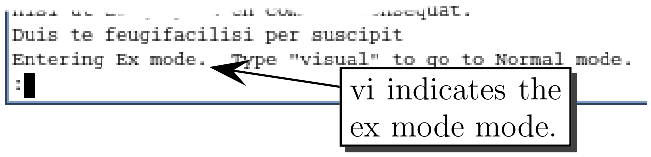
```
Ut wisi enim ad minim veniam,
quis nostrud exerci taion
ullamcorper suscipit lobortis
nisl ut aliquip ex en commodo c
Duis te feugifacilisi per susci
-- VISUAL --
```

Select mode – The select mode is similar to the visual mode, but there are different commands available. You can switch to the select mode by entering `gh` in the normal mode. You can return to normal mode by pressing Esc.



```
quis nostrud exerci taion
ullamcorper suscipit lobortis
nisl ut aliquip ex en commodo consequat.
Duis te feugifacilisi per suscipit
-- SELECT --
```

Ex mode – In the ex mode, vi does not work screen oriented. That means you do not have any cursor for positioning. You have to use the commands of the command-line mode to edit the text. From the normal mode, you can enter the ex mode by pressing Shift+ Q. To exit the ex mode, you have to enter `vi` or `visual`.

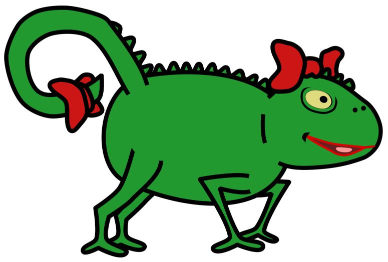


```
Duis te feugifacilisi per suscipit
Entering Ex mode.  Type "visual" to go to Normal mode.
:
```

You can use the following commands in normal mode:

Command	**Result**
`i`	Switches vi to insert mode.
`a`	Switches vi to insert mode (similar to `i`, but the cursor moves one step to the right).
`A`	Switches vi to insert mode (similar to `i`, but the cursor moves to the end of the current line).
`x`	Deletes the character where the cursor is.
`dd`	Deletes the line in which the cursor is located and copies it to the buffer.
`D`	Deletes the rest of the current line from the cursor position.
`yy`	Copies the line in which the cursor is located to the buffer.
`p`, P	Inserts the contents of the buffer after/before current cursor position.
`o`, `O`	Inserts a new blank line below/above the current line.
`r`*`character`*	Replaces the current character by *`character`*.
`J`	Removes the line break at the end of the current line.
~	Changes between upper- and lower-case.
`ZZ`	Saves the current file and ends vi.
`u`	Undoes the last operation.
`.` (Full stop)	Repeats the last operation.
`n`	Repeats the search in the same direction.
`N`	Repeats the search in the opposite direction.

If you want to use a command for several units, place the corresponding number in front of the command. For example, `3x` deletes three characters, `5dd` deletes five lines, and `7yy` copies seven lines to the buffer.



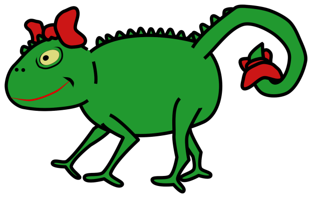
You can use the following commands in command-line mode:

Command	Result
`:q`	Ends vi (if no changes were made).
`:q!`	Ends vi without saving changes in the file.
`:wq` or `:x`	Saves the current file and ends vi.
`:w`	Saves the current file.
`:w` ***`file`***	Saves the current file under the name ***`file`***. (Note: You continue editing the original file, not the new file.)
`:e` ***`file`***	Opens the file ***`file`*** (if no changes were made at the current file).
`:e` ***`file`***	Closes the current file without saving and opens the file ***`file`***.
`:ha`	Prints the current file.
`/`*`pattern`*	Searches forward from the cursor position for *`pattern`*.
`?`*`pattern`*	Searches backward from the cursor position for *`pattern`*.

Exercise: Learn the Working Modes

Use vi to enter the following poetry of William Shakespeare. Save the file as `Poetry.txt` in your home directory.

```
Some glory in their birth, some in their skill,
Some in their wealth, some in their bodies' force,
Some in their garments, though new-fangled ill,
Some in their hawks and hounds, some in their horse;
And every humour hath his adjunct pleasure,
Wherein it finds a joy above the rest:
But these particulars are not my measure;
All these I better in one general best.
Thy love is better than high birth to me,
Richer than wealth, prouder than garments' cost,
Of more delight than hawks or horses be;
And having thee, of all men's pride I boast:
Wretched in this alone, that thou mayst take
All this away and me most wretched make.
```



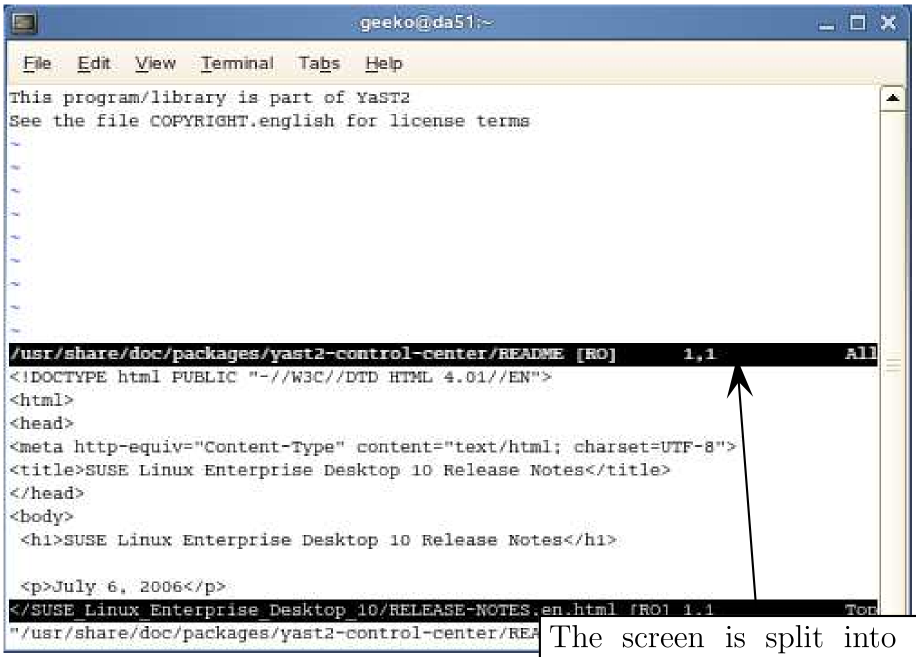
2.4.4 Learn Advanced vi Commands

In the following some more advanced vi features are introduced. They are interesting especially for devolopers.

Split the Screen

Using the command `:sp` ***filename***, the vi screen is split in two frames, and the specified file is shown in upper frame.

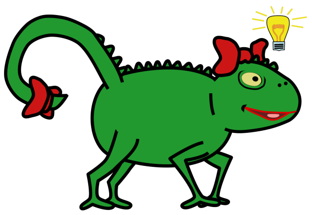
You can switch between the two frames by typing `Ctrl`+`W`, `W` in the command-line mode.



```
This program/library is part of YaST2
See the file COPYRIGHT.english for license terms
~
~
~
~
~
~
~
~
~
~
/usr/share/doc/packages/yast2-control-center/README [RO]      1,1          All
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>SUSE Linux Enterprise Desktop 10 Release Notes</title>
</head>
<body>
 <h1>SUSE Linux Enterprise Desktop 10 Release Notes</h1>

 <p>July 6, 2006</p>
</SUSE_Linux_Enterprise_Desktop_10/RELEASE-NOTES.en.html [RO] 1,1          Top
"/usr/share/doc/packages/yast2-control-center/REA
```

The screen is split into two frames showing different files.

With `Ctrl`+`W`, `_` the current frame is maximized. `Ctrl`+`W`, `=` resizes all frames to the same size.



Auto Completion

In insert mode you can complete a word by pressing Ctrl + P. vi searches for words starting in the same way in the text and offers them to complete the current word. By pressing the key combination several times, you can switch between all alternatives.

Auto completion also works for lines by pressing Ctrl + X, Ctrl + L.

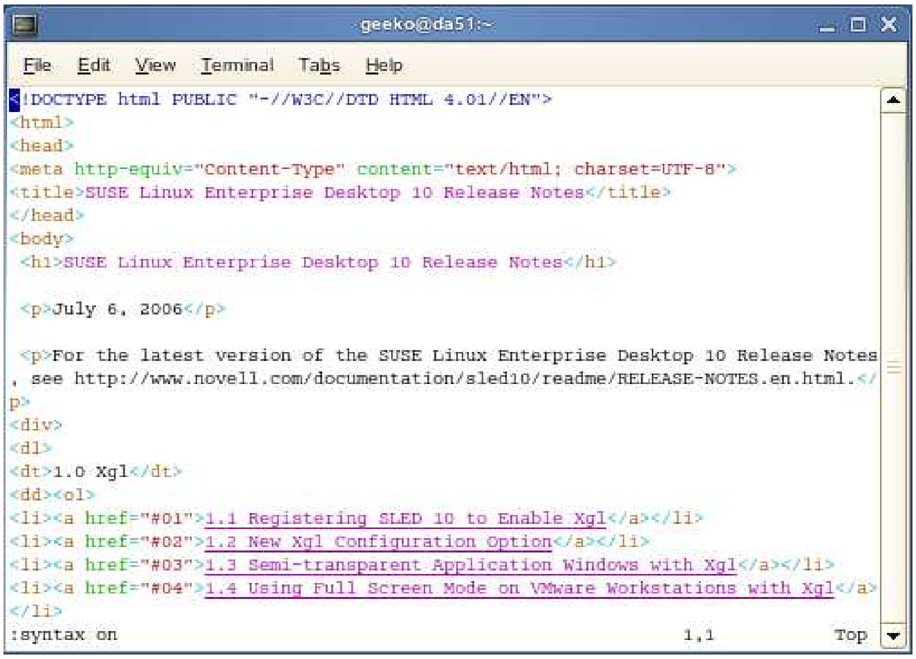
Use Marks

You can set a mark in a text by moving the cursor to the position you want to mark and entering `m`*`character`*. For *`character`* you can use a character of a – z or A – Z.

If you want to jump to the mark later, enter `'`*`character`*.

Enable Syntax Highlighting

When entering `:syntax on` in command-line mode, syntax highlighting is enabled. vi knows a lot of programming languages and writes a computer program in multiple colors. This simplyfies reading the program code.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>SUSE Linux Enterprise Desktop 10 Release Notes</title>
</head>
<body>
 <h1>SUSE Linux Enterprise Desktop 10 Release Notes</h1>

 <p>July 6, 2006</p>

 <p>For the latest version of the SUSE Linux Enterprise Desktop 10 Release Notes
, see http://www.novell.com/documentation/sled10/readme/RELEASE-NOTES.en.html.</
p>
<div>
<dl>
<dt>1.0 Xgl</dt>
<dd><ol>
<li><a href="#01">1.1 Registering SLED 10 to Enable Xgl</a></li>
<li><a href="#02">1.2 New Xgl Configuration Option</a></li>
<li><a href="#03">1.3 Semi-transparent Application Windows with Xgl</a></li>
<li><a href="#04">1.4 Using Full Screen Mode on VMware Workstations with Xgl</a>
</li>
:syntax on                                                    1,1           Top
```

If you want to enable syntax highlighting permanent, add the following to the file ~/.vimrc.

```
syntax on
```

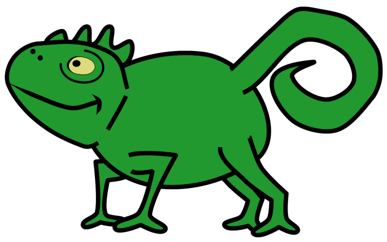
By default, this file does not exist.

To disable syntax highlighting, enter `:syntax off` in command-line mode.

There is a tutorial for vi available on your Linux computer. Enter `vimtutor` in the shell command line to start it.

In the directory `/usr/share/doc/packages/vim/` there are two example configuration files `vimrc_example1` and `vimrc_example2`. Both include a lot of explanations and comments. These lines begin with double quotes.

Exercise: Enable Syntax Highlighting
Enable vi's syntax highlighting permanent on your computer.



2.4.5 Create Macros

Some tasks require a lot of redundant working steps. Suppose there is a file with last names and firstnames:

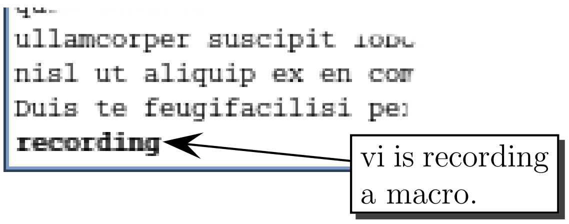
```
George
William
Paul

Smith
Carter
Bailey
```

Now you want to append the first name behind the last name. If you have file with a thousands of names (and not just three), a macro helps you to fulfill this task.

Do the following steps:

❶ Move the cursor to the very beginning of the file (upper left corner).

❷ Enter `Q`, followed by any other characters to start recording a macro. The second character would be the name of the macro.



```
ullamcorper suscipit lobo
nisl ut aliquip ex en com
Duis te feugifacilisi pe
recording
```

❸ Enter the commands that are necessary for your task. Referring to the upper example, the steps could be like the following:

① `D`, `D` (Deletes the first line and copies the name to the clipboard.)

② `3`, `J` (Moves the cursor one line down.)

③ `P` (Inserts the copied name under the current line.)

④ `K` (Moves the cursor one line up.)

⑤ `Shift`+ `A` (Moves the cursor to the end of the line and switches into the insert mode.)

⑥ `,` (Enter a comma.)

⑦ `Esc` (Leave the insert mode.)

⑧ `Shift`+ `J` (Appends the following line to the current one.)

⑨ `G`, `G` (Moves the cursor to the beginning of the file.)

❹ Finish macro recording by entering `Q`.

Now you can execute the macro by entering `@` followed by the "name" of the macro.

For more detailed information about macros, enter `:help recording`.

Exercise: Create Macros

Create a macro, that inserts three spaces at the beginning of a line.

2.4.6 Create Abbreviations

An abbreviation works only in the insert or command-line mode. They have to be defined in command-line mode.

The general syntax to define an abbreviation is:
`abbreviation_command short_form long_form`

The ***`abbreviation_command`*** depends on the mode the abbreviation should be valid for:

`iabbrev` –	Abbreviation is valid only for insert mode.
`cabbrev` –	Abbreviation is valid only for command-line mode.
`abbrev` –	Abbreviation is valid for insert and command-line mode.

The ***`short_form`*** is the text you want to enter. ***`long_form`*** is the text that should be inserted instead.

For example, after defining the following abbreviation,

```
iabbrev asap as soon as possible
```

any time, you enter `asap` in insert mode, `asap` is converted to `as soon as possible`.

It is also possible to use control sequences in the ***`short_form`*** and ***`long_form`***:

Meaning	Enter
Space	`<Space>`
Enter	`<CR>`
Tab	`<Tab>`
Del	`<Del>`
Pos1	`<Home>`
End	`<End>`
Up-arrow	`<Up>`
Down-arrow	`<Down>`
Left-arrow	`<Left>`
Right-arrow	`<Right>`

For example, after defining the following abbreviation

```
iabbrev siny Sincerly yours<CR>Geeko
```

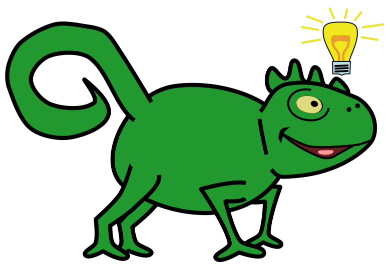
any time, you enter `siny` in insert mode, it is converted to

```
Sincerly yours
Geeko
```

For more detailed information about macros, enter `:help abbreviations`.

Exercise: Create Abbreviations

Create an abbreviation that adds `#!/bin/bash` followed by two line breaks.



2.5 The Basics of bash Programming

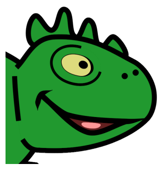
As the Linux shell is programmable, you can use the vi text editor to write a program. Normally a bash script is written in a text file and

begins with the following line (called a *shee-bang line*):
`#!/bin/bash`

Before using a new bash script, you have to make it executable with the command `chmod u+x` *`bash_script`*).

Inside the bash script, you can use all the commands you learned before, such as `echo` to display text on the screen or request the content of a variable (see section 2.2.3 on page 78).

```
#!/bin/bash

echo 'The content of variable $PATH is:' $PATH
```



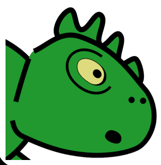
Normally the hash sign ("`#`") is used to mark the following as comment. Comments are not executed. The only exception from this rule is the shee-bang line.

If you execute the script, the output may look like this:

```
geeko@da51:~> ./bash_script
The content of variable $PATH is: /home/geeko/bin:/usr/local/bin:/usr/bin:/usr/X
11R6/bin:/bin:/usr/games:/opt/gnome/bin:/opt/kde3/bin:/usr/lib/jvm/jre/bin:/usr/
lib/mit/bin:/usr/lib/mit/sbin
geeko@da51:~>
```

2.5.1 Variables in bash Scripts

In the example above, we use the environment variable `$PATH`. It is possible to define your own variables that are only valid inside the bash script. The syntax is *`variable_name=value`*.



It is a convention to use lowercase letters for variable names in a program code. Only constants (which do not change their value during the program's run time) are written in uppercase letters.

```
#!/bin/bash

myvar=13

echo 'The content of variable $myvar is:' $myvar
```

To get the content of a variable (e. g., with the `echo` command), you have to add a "`$`" in front of the variable's name (see section 2.1.8 on page 61).

A variable can store numbers, characters, and text strings.

```
#!/bin/bash

a=2
b=x
c=Geeko
d="Geeko and Suzie"

echo '$a is a number:' $a
echo '$b is a character:' $b
echo '$c is a string:' $c
echo '$d is a string with whitespaces:' $d
```

The output looks like the following:

```
$a is a number: 2
$b is a character: x
$c is a string: Geeko
$d is a string with whitespaces: Geeko and Suzie
```

Exercise: Types of Variables

Numbers can be added. Strings can't. Make aware that there are differences between numbers and strings in the kind which operatation is meaningful.

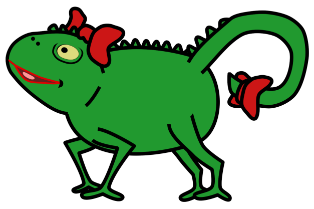
Write down four operations that are typical for numbers and four that are typical for strings.

Operations for numbers:	Operations for strings:
❶ ____________________	❶ ____________________
❷ ____________________	❷ ____________________
❸ ____________________	❸ ____________________
❹ ____________________	❹ ____________________

2.5.2 Compute Numbers

Variables are often combined with mathematical operators. That means that you can compute them. Variables with numbers can be:

- ☞ Added (use operator "+")
- ☞ Subtracted (use operator "-")
- ☞ Multiplied (use operator "*")
- ☞ Devided (use operator "/")
- ☞ Modulated (use operator "%")



Compute Integers

In the following example, we write the sum of the variables `$a` and `$b` into a new variable.

```
#!/bin/bash

a=3
b=2
c=$a+$b

echo '$a+$b='$c
```

The output looks like the following.

```
$a+$b=3+2
```

As you can see, the variable C contains the string "3+2," but not the sum "5". To get the sum, you have the following possibilities:

❶ Allow the variable `$c` only to store numbers.

To do this, use the command `declare` with the option "`-i`" to declare the variable `$c`.

```
#!/bin/bash

a=3
b=2
declare -i c=$a+$b

echo '$a+$b='$c
```

❷ Mark the formula as expression that has to be evaluated.

To calculate at the command line, use the command `expr`.

```
geeko@da51:~ > expr 3 + 2
5
geeko@da51:~ >
```

Attention! In contrast to the method using `declare`, with `expr` there has to be a space before and after the operator(s).

If you use `expr` in a shell script, you have to quote the expression with backticks ("‘") or `$(...)`. Characters that have a special meaning in the bash (such as "*" and brackets) have to be masked by a backslash.

```
#!/bin/bash

a=3
b=2
c=$(expr $a + $b)

echo '$a+$b='$c
```

The expression inside `$(...)` is replaced by the result of the expression.

The output looks like this:

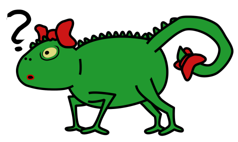
```
$a+$b=5
```

Now let us do the same thing with the division operator.

```
#!/bin/bash

a=3
b=2
c=$(expr $a / $b)

echo '$a/$b='$c
```

The result is not what you might expect:

```
$a/$b=1
```



The results of a division is an integer, and an integer number does not have any decimal places. You can get the rest of a division with the modulo operator:

```
#!/bin/bash

a=3
b=2
c=$(expr $a / $b)
d=$(expr $a % $b)

echo '$a/$b='$c Rest:$d
```

Instead of `expr` you also can use the following syntax `$((...))`. For example:

```
#!/bin/bash

a=3
b=2
c=$(($a + $b))

echo '$a+$b='$c
```

Exercise: Compute Integers
Write a bash program that computes the squares of 6, 9, and 12.

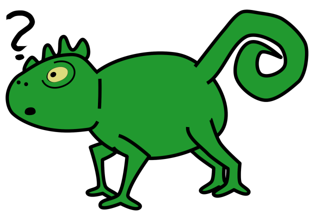
Compute Float Numbers

By default, the bash is not able to calculate with float numbers.

```
#!/bin/bash

a=3
b=2.7
c=$(expr $a + $b)

echo '$a+$b='$c
```

This program leads to the result:

```
expr: non-numeric argument
$a+$b=
```



To solve this problem, use the shell tool called `bc`. `bc` is an arbitrary precision calculator. At the command line, you can use `bc` for calculations.

```
 1 geeko@da51:~> bc
 2 bc 1.06
   Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
 4 This is free software with ABSOLUTELY NO WARRANTY.
   For details type ‘warranty’.
 6 3+4
   7
 8 (10+4)/2
   7
10 2+0.44
   2.44
12 7/3
   2
14 quit
```

Explanation: When entering `bc`, the calculator starts, and you can enter an expression to calculate (lines 6, 8, 10 and 12). To see the result of the calculation, press `Enter` (lines 7, 9, 11, and 13). You can group expressions by using brackets (line 8). You can use `bc` to calculate with float numbers (line 10). By default, `bc` uses the same kind of processing for division (without decimal places) as the shell does (lines 12+13). To exit `bc`, enter `quit` (line 14).

`bc` provides its own programming language, but that would not matter now. To use `bc` within a bash script, use a simple “trick”: piping the formular into `bc`. Use the `echo` command and the “`|`” pipe.

```
geeko@da51:~> echo "2.4+7" | bc
9.4
```

To store the result of an calculation using `bc`, put it into backticks (“‘”).

```
geeko@da51:~> a=‘echo "2.4+7" | bc‘
geeko@da51:~> echo $a
9.4
```

Instead of backticks, you can also use the following syntax with `$(...)`.

```
geeko@da51:~> a=$(echo "2.4+7" | bc)
geeko@da51:~> echo $a
9.4
```

It’s up to you, which syntax you prefer. In this manual, the second one is used.

To see decimal places, enter `scale=`*`number`* in front of the mathematical operation, separated by semicolon.

For example:

```
geeko@da51:~> echo "scale=2; 9/2" | bc
4.50
```

Exercise: Compute Float Numbers

Write a bash program that computes the squares of 1.5, 2.75 and 7.2

2.5.3 Comparing Files, Strings and Numbers

To find out if three is greater than two, you do not need to write a bash program. But in interactive programs a lot of questions appear and the program execution depends on the results. Common questions are, for example:

- ☞ Does the file ***`filename`*** exist?
- ☞ How old is the file?
- ☞ How does the user answer the program's question?
- ☞ Is the value of a variable still inside the valid range?
- ☞ How often has a loop already been completed?

Use the `test` command for testing. The result of a test is either "0" (true) or "1" (false). This result is stored in the variable "$?".

The `test` command knows a lot of parameters. First we want to introduce the parameters concerning files:

- ☞ `-d` *`target`*

  Tests if ***`target`*** is an existing directory.

```
#!/bin/bash

a=/var
b=/etc/profile

test -d $a
echo $a: $?

test -d $b
echo $b: $?
```

The output of this program looks like the following:

```
/var: 0
/etc/profile: 1
```

☞ `-e` ***target***

Tests if ***target*** is an existing file.

```
#!/bin/bash

a=/var
b=2

test -e $a
echo $a: $?

test -e $B
echo $b: $?
```

The output looks like the following:

```
/var: 0
2: 1
```

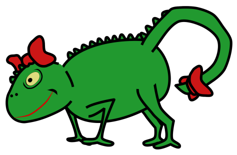
☞ `-f` ***target***

Tests if ***target*** exists and if it is a regular file.

```
#!/bin/bash

a=/var
b=/etc/profile

test -f $a
echo $a: $?

test -f $b
echo $b: $?
```

The output of this program looks like the following. Compare it with the output of parameter `-d`:

```
/var: 1
/etc/profile: 0
```



☞ `-r` ***target***, `-w` ***target***, `-x` ***target***

Tests if the file permission for the current user is set:

- reading (`-r`)
- writing (`-w`)
- executing (`-x`)

```
#!/bin/bash

a=/etc/profile

test -r $a
echo "Read: $?"

test -w $a
echo "Write: $?"

test -x $a
echo "Execute: $?"
```

The file `/etc/profile` has the following permission: `rw-r--r--`. User Geeko is only allowed to read the file. The output of the above script looks like the following:

```
Read: 0
Write: 1
Execute: 1
```

☞ `-s` ***`target`***

Tests if a file is empty.

```
#!/bin/bash

a=/etc/profile

test -s $a
echo $?
```

The output of the script should be "`0`".

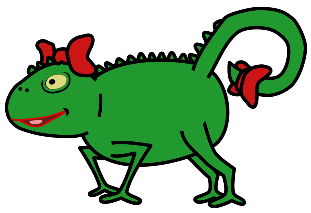
☞ ***`file1`*** `-nt` ***`file2`***

Tests if ***`file1`*** is newer than ***`file2`***. The date of the last modification is used, not the date of the creation.

```
#!/bin/bash

a=/etc/profile
b=/var/log/Xorg.0.log

test $a -nt $b
echo "$a -nt $b:" $?
```

The result should be "`1`" (false), and the output looks like the following:

```
/etc/profile -nt /var/log/Xorg.0.log: 1
```



☞ *`file1`* -ot *`file2`*

Is the counterpart of `-nt` and tests if *`file1`* is older than *`file2`*.

```
#!/bin/bash

a=/etc/profile
b=/var/log/Xorg.0.log

test $a -ot $b
echo "$a -ot $b:" $?
```

The result should be "1" (false), and the output looks like the following:

```
/etc/profile -ot /var/log/Xorg.0.log: 0
```

✍ Exercise: Comparing Files, Strings and Numbers – Part I

Write a bash program that…

… tests if the file `/var/log/messages` exists,
… if it is empty and
… that displays the file permission.

`test` is also able to test and compare strings.

☞ `-z` *`string`*

The result is "`0`" (true), if the string is "0".

```
#!/bin/bash

a=geeko

test -z $a
echo "a: $?"

test -z $b
echo "b: $?"
```

The output looks like the following:

```
a: 1
b: 0
```

Explanation: As the variable `$b` is not defined, it has a length of "0" and the test result is "true".

☞ `-n` *`string`*

This parameter is the counterpart of `-z`. The test result is "true" if the string contains one or more characters.

```
#!/bin/bash

a=geeko

test -n $a
echo "a: $?"

test -n $b
echo "b: $?"
```

The output looks like the following:

```
a: 0
b: 1
```

☞ ***`string1 == string2`***

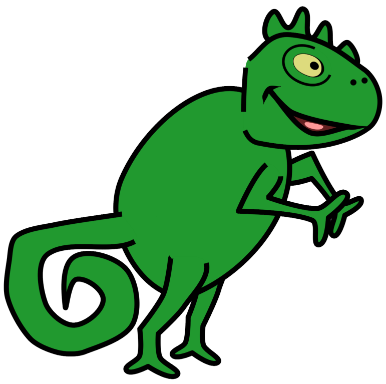
The test result is "true" if ***`string1`*** and ***`string2`*** are the same.

```
1  #!/bin/bash
2
   a=geeko
4  b=suzie
   c=geeko
6
   test $a == $b
8  echo "$a == $b: $?"

10 test $a == $c
   echo "$a == $c: $?"
```

Explanation: Three variables are defined (lines 3–5). The variables `$a` and `$c` are the same. In line 7, the variables `$a` and `$b` are compared. In line 10, the variables `$a` and `$c` are compared.

The output looks like the following:

```
geeko == suzie: 1
geeko == geeko: 0
```



☞ ***`string1 != string2`***

This operator is the counterpart of the "==" operator. The test result is "true" if both strings are *not* the same.

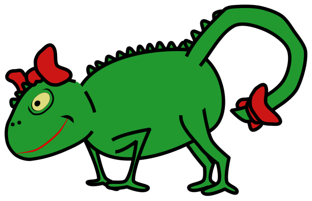
```
#!/bin/bash

a=geeko
b=suzie
c=geeko

test $a != $b
echo "$a != $b: $?"

test $a != $c
echo "$a != $c: $?"
```

The output looks like the following:

```
geeko != suzie: 0
geeko != geeko: 1
```

✍ Exercise: Comparing Files, Strings and Numbers – Part II

Write a bash program that tests whether the string "Linux" and the string "School" are equal.



There are also some operators which compare numbers:

☞ ***integer1*** `-eq` ***integer2***

Tests if ***integer1*** is equal to ***integer2***.

```
#!/bin/bash

a=3
b=2

test $a -eq $b
echo "$a -eq $b: $?"
```

The output looks like the following:

```
3 -eq 2: 1
```

☞ ***integer1*** `-ne` ***integer2***

Is the counterpart of `-eq`. It testes if ***integer1*** is *not* the same as ***integer2***.

```
#!/bin/bash

a=3
b=2

test $a -ne $b
echo "$a -ne $b: $?
```

The output looks like the following:

```
3 -ne 2: 0
```

☞ ***integer1* -gt *integer2***

Tests if ***integer1*** is strictly greater than ***integer2***.

```
#!/bin/bash

a=3
b=2

test $a -gt $b
echo "$a -gt $b: $?"

test $b -gt $a
echo "$b -gt $a: $?"
```

The output looks like the following:

```
3 -gt 2: 0
2 -gt 3: 1
```

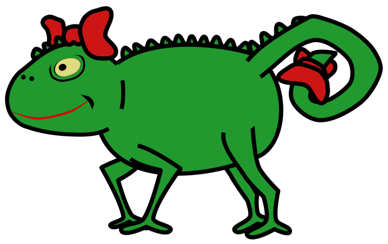
☞ ***integer1* -ge *integer2***

Tests if ***integer1*** is greater than or the same as to ***integer2***.

```
#!/bin/bash

a=3
b=2
c=3

test $a -ge $b
echo "$a -ge $b: $?"

test $a -ge $c
echo "$a -ge $c: $?"
```

The output looks like the following:

```
3 -ge 2: 0
3 -ge 3: 0
```



☞ ***integer1* -lt *integer2***

Tests if ***integer1*** is strictly less than ***integer2***.

```
#!/bin/bash

a=3
b=2

test $a -lt $b
echo "$a -lt $b: $?"

test $b -lt $a
echo "$b -lt $a: $?"
```

The output looks like the following:

```
3 -lt 2: 1
2 -lt 3: 0
```

☞ ***integer1* -le *integer2***

Tests if ***integer1*** is less than or equal to ***integer2***.

```
#!/bin/bash

a=3
b=2
c=3

test $a -le $b
echo "$a -le $b: $?"

test $a -le $c
echo "$a -le $c: $?"
```

The output looks like the following:

```
3 -le 2: 1
3 -le 3: 0
```

Exercise: Comparing Files, Strings and Numbers – Part III

Write a bash program that tests whether the number “13” is greater than the number “11”.

It is also possible to compare float numbers. To do this, you have to use the `bc` command. The syntax differs from the shell syntax:

☞ ***expression1* == *expression2***

Is ***expression1*** the same as ***expression2***?

☞ ***expression1* != *expression2***

Is ***expression1*** not the same as ***expression2***?

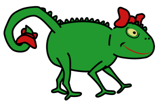
- ☞ *`expression1`* $>$ *`expression2`*

  Is *`expression1`* strictly greater than *`expression2`*?
- ☞ *`expression1`* $>=$ *`expression2`*

  Is *`expression1`* greater than or equal to *`expression2`*?
- ☞ *`expression1`* $<$ *`expression2`*

  Is *`expression1`* strictly less than *`expression2`*?
- ☞ *`expression1`* $<=$ *`expression2`*

  Is *`expression1`* less than or equal to *`expression2`*?



All these operators are shown in the following program:

```
#!/bin/bash

a=3.7

b=$(echo "$a == 3.7" | bc)
echo "$a == 3.7: $b"

b=$(echo "$a != 3.7" | bc)
echo "$a != 3.7: $b"

b=$(echo "$a > 3.7" | bc)
echo "$a > 3.7: $b"

b=$(echo "$a >= 3.7" | bc)
echo "$a >= 3.7: $b"

b=$(echo "$a < 3.7" | bc)
echo "$a < 3.7: $b"

b=$(echo "$a <= 3.7" | bc)
echo "$a <= 3.7: $b"
```

The output looks like the following:

```
3.7 == 3.7: 1
3.7 != 3.7: 0
3.7 > 3.7: 0
3.7 >= 3.7: 1
3.7 < 3.7: 0
3.7 <= 3.7: 1
```

Attention! The return values of `bc` are different to the bash: “0” means the relation is false and “1” means the relation is true.

Exercise: Comparing Files, Strings and Numbers – Part IV

Write a bash program that tests whether the number “13.5” is greater than the number “22”.

Sometimes it is necessary to combine two or more tests, e. g., to test if the value of a variable is greater than a number *but* less than another number. There are two combinations possible:

Logical OR. Test one returns “0” (true) *or* test two returns “0” (true).

		Test1	
		true	false
Test2	true	true	true
	false	true	false

Use the parameter `-o` to create an OR connection between two tests.

```
#!/bin/bash

a=/var
b=/etc/profile

test -f $a -o -f $b
echo "Result: $?"
```

Explanation: This script tests whether the file `/var`, *or* `/etc/profile` is a regular file. For `/var` the result is false, because it is an directory. But `/etc/profile` is a regular file. This is why the whole test returns “0” (true).

Logical AND. Test one returns “0” (true) *and* test two returns “0” (true).

		Test1	
		true	false
Test2	true	true	false
	false	false	false

Use the parameter `-a` to create an AND connection between tests.

```
#!/bin/bash

a=7
b=2

test $a -gt $b -a $(expr $a % 2) -eq 0
echo "Result: $?"
```

Explanation: This script tests if the value of `$a` is greater than the value of `$b` (7 > 2; true) *and* if the value of `$a` is even (false). Because of the last comparison is false, the whole test is false.

The next logical operator we want to introduce is NOT. This operator is called "negation" operator. It is written by an exclamation mark.

```
#!/bin/bash

a=7
b=2

test ! $a -gt $b
echo "Result: $?"
```

Explanation: `$a -gt $b` tests if the value of `$a` is greater than the value of `$b`. In the example above, that is true. But, because of the negation, the "true" is changed to "false".

In `bc` the logical AND is written by "&&" and logical OR by "||". For NOT, use the exclamation mark, too.

Exercise: Comparing Files, Strings and Numbers – Part V

Write a bash program that tests whether the number "13.5" is greater than the number "22" or smaller than "15".

2.5.4 Read from Input

It does not make sense to write a bash program with eight lines just to compare two numbers. And if you need a comparison of two other numbers, you have to write a new program?

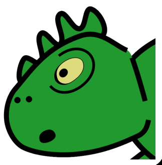
Of course, it is not true, and you can ask the program's user to enter some data. Therefore the command `read` is used.

`read` waits until the user finishes his input by pressing `Enter`. You can specify a variable where the input is stored.

```
1 #!/bin/bash
2
  read a
4 echo "You entered: $aA"
```

Explanation: After this small program starts, it waits for input. The input will be stored in the variable `$a` (line 3). After the user finishes his input by pressing `Enter`, the input is written on the screen (line 4).

If you do not specify a variable after the `read` command, the input is stored in the variable $REPLY.



You can also use `read` to specify more than one variable. The input is separated by white spaces into tokens. The first token is stored in the first variable, the second token in the second variable, and so forth. All remaining text tokens are stored in the last variable.

```
#!/bin/bash

read a b c
echo "First token: $a"
echo "Second token: $b"
echo "Third token: $c"
```

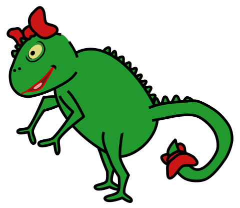
If you enter `This is a short text.`, the output looks like the following:

```
First token: This
Second token: is
Third token: a short text.
```

If there are more variables than text tokens, the remaining variables stay empty.

Exercise: Read from Input

Write a bash program that reads two integer values from input and tests if the first number is larger than the second one.



2.5.5 Examine Cases

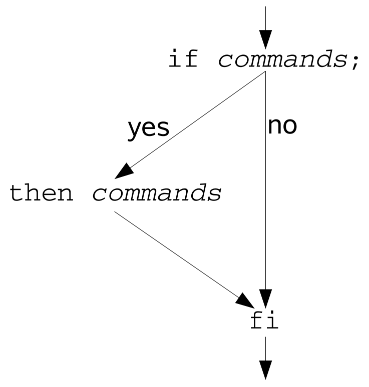
The bash supports two possibilities to control the program flow:

- ☞ `if ... fi`
- ☞ `case ... esac`

If you want to differentiate between a few cases, use the first possibility.

The most simple way to use the `if` statement is the following syntax:

```
if commands; then
    commands
fi
```



First, the command after `if` is evaluated. If the result is “0” (true), the commands after the corresponding `then` are executed. If the result of the `if` command is “1” (false), nothing happens and the program continues after `fi`.

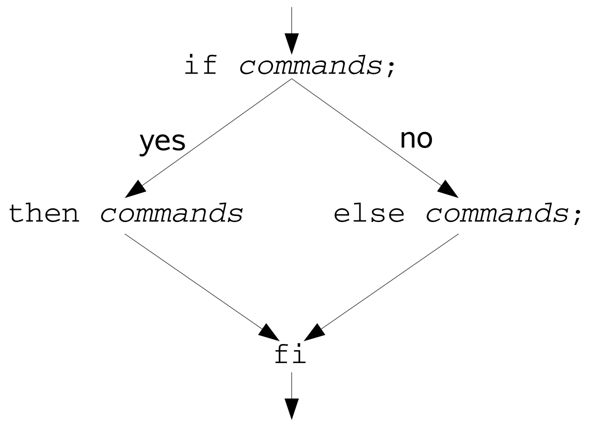
In most cases, the commands after `if` is a test with the `test` command.

```
1 #!/bin/bash
2
  echo "Enter something:"
4 read a

6 if test -e $a; then
      echo "$a is an existing file."
8 fi
```

Explanation: This program asks the user to enter something (lines 3 – 4). Than it tests if the entered string is an existing file (line 6). If so, a message is written on the screen (line 7).

You can add `else` to the construct above:

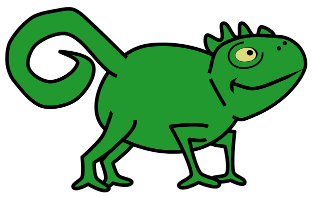
```
if commands; then
    commands
else
    commands
fi
```



The commands after `else` are executed, if the result of the `if` command is false.

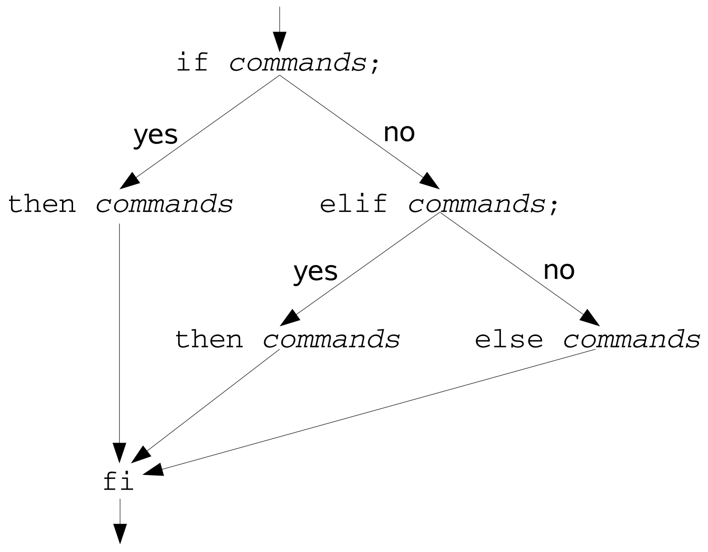
```
1  #!/bin/bash
2
   echo "Enter something:"
4  read a

6  if test -e $a; then
       echo "$a is an existing file."
8  else
       echo "$a is no file."
10 fi
```

Explanation: The program above is extended by lines 8 and 9. Now, if the input is no filename, another message is written.



If you want to check for more than one condition, put `elif` between the `then` command and the `else` command.

```
if commands; then
    commands
elif commands; then
    commands
else commands
    commands
fi
```

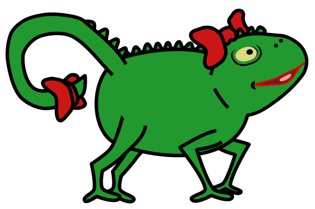


First, the command after `if` is evaluated. If the result is “0” (true), the commands after the corresponding `then` are executed. If the result of the `if` command is false, then the commands after `elif` are evaluated, whether they are true or false. If neither the `if` command nor the `elif` command is true, the `else` commands are executed.

You can use more than one `elif` in a condition to build long chains of conditions.

```
1  #!/bin/bash
2
   echo "Enter something:"
4  read a

6  if test -e $a; then
       echo "$a is an existing file."
8  elif test $a == geeko; then
       echo "Entered geeko."
10 else
       echo "$a is no file and not geeko."
12 fi
```

Explanation: The above program is extended by lines 8 and 9. Now, the entered string is tested whether it is the same as “geeko”. It is tested, if the entered text is not an existing file.



Exercise: Examine Cases – Part I

Write a bash program that reads two integer values from input and tests ...

☞ ... whether the first number is greater than the second one,

☞ ... whether the first number is equal to the second one, or

☞ ... whether the first number is smaller than the second one.

It should display different messages on the screen for each case.

If you have a lot of `if` conditions, it is better to use `case` instead.

The syntax is:

```
case condition in
    pattern|pattern...)
        commands
        ;;
    pattern|pattern...)
        commands
        ;;
esac
```

The patterns are tested if they match the condition. If yes, the commands in the pattern are executed. If not, the next pattern is tested.

```
1  #!/bin/bash
2
   echo "Enter something:"
4  read a

6  case $a in
       [Yy]es|[Tt]rue)
8          echo "You entered Yes/True."
           ;;
10     [Nn]o|[Ff]alse)
           echo "You entered No/False."
12         ;;
   esac
```

Explanation: The entered string is tested whether it is "Yes", "yes", "True", or "true" (line 7). If it does not match, it is tested for "No", "no", "False", or "false" (line 9).

If a pattern matches, and its commands are executed, the program finishes the `case-esac` region and continues after `esac`.

If you want to execute commands in case none of the patterns match, add an asterisk pattern:

```
#!/bin/bash

echo "Enter something:"
read a

case $a in
    [Yy]es|[Tt]rue)
        echo "You entered Yes/True."
        ;;
    [Nn]o|[Ff]alse)
        echo "You entered No/False."
        ;;
    *)
        echo "Wrong input."
esac
```

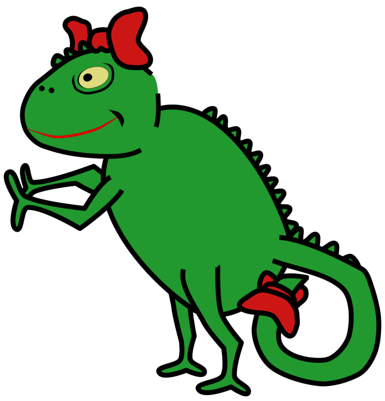
Exercise: Examine Cases – Part II

Write a bash program that reads one one-digit integer number from input and displays the name of the number on the screen (e. g., `One` for “1”).

2.5.6 Use Loops

By using a loop, you can make the interpreter execute a part of the program several times. The bash knows three loops:

1. `while`
2. `until`
3. `for`



The `while` Loop

The syntax of the `while` loop is:

```
while condition; do
    commands
done
```

Normally, the condition is a test and the commands inside the loop are executed as long as the condition is true.

```
1 #!/bin/bash
2
  a=1
4
  while test $a -le 8; do
6     echo "\$a=$a"
      a=$(expr $a + 1)
8 done
```

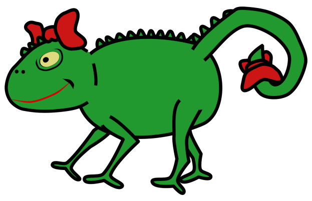
Explanation: In line 3, the variable `$a` is set to "1". In line 5, it is tested if `$a` is less than (or equal to) "8". While the value of `$a` is less than "8", the commands inside the loop are executed. Line 6 creates some output. In line 7 the value of `$a` is increased by "1".

The output of the program above looks like the following:

```
$a=1
$a=2
$a=3
$a=4
$a=5
$a=6
$a=7
$a=8
```

Exercise: Use Loops – Part I

Write a bash program that reads one one-digit integer number from input and counts from one to the entered number. It should display the name of the number on the screen (e. g., `One` for "1") and not the number. Use a `while` loop.



The `until` Loop

The `until` loop is similar to the `while` loop. The syntax is similar, too:

```
until condition; do
    commands
done
```

The difference between `while` and `until` is, that if you use `until`, the *`commands`* are executed *until* the condition is true.

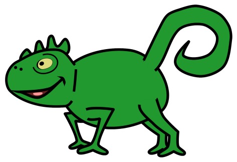
To get the same result as in the program above, the code has to be:

```
#!/bin/bash

a=1

until test $a -gt 8; do
    echo "\$a=$a"
    a=$(expr $a + 1)
done
```

Explanation: `while` is replaced by `until`. "`-le`" is replaced by "`-gt`".

Exercise: Use Loops – Part II

Write a bash program that reads one one-digit integer number from input and counts from one to the entered number. It should display the name of the number on the screen (e. g., `One` for "1") and not the number. Use an `until` loop.



The `for` Loop

The `for` loop differs from `while` and `until`. It is used to repeat code several times. The number of repetition is fixed and defined before.

`for` can be used in two differend ways. The simpliest syntax is:

```
for variable in list; do
    commands
done
```

The variable is set to the first item of the list, and then the commands are executed. The next time, the variable is set to the second item of the list. The commands are executed again, and so on.

```
#!/bin/bash

for a in Geeko Suzie 100 0 Novell; do
    echo "$a"
done
```

The output of this program looks like the following:

```
Geeko
Suzie
100
0
Novell
```

The list can be created while the program is running.

```
#!/bin/bash

for a in $(ls /bin/b*); do
    echo "$a"
done
```

Explanation: All files that are in the `/bin` directory and starts with "b" are used as a list.

The output looks like the following:

```
/bin/basename
/bin/bash
/bin/bluepincat
```

Another syntax of `for` is:

```
for ((variable; condition; change)); do
    commands
done
```

The parts are the following:

`variable`	–	The start value of a variable is set.
`condition`	–	The loop is executed as long as the condition is true.
`change`	–	At the end of the loop, the value of the variable is changed in the defined way.

The following is an example:

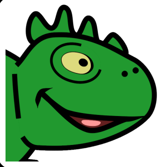
```
1 #!/bin/bash
2
  for ((a=10; a<=15; a++)); do
4     echo "$a"
  done
```

Explanation: The first time line 3 is executed, variable `$a` is set to “10”. It tests, whether the condition (`$a` is less than or the same as “15”) is true. The command(s) of the loop (line 4) are executed once. After `done` (line 5), the value of variable `$a` is increased by “1”. It tests, whether the condition is true again.

If you append a double plus (“`++`”) at a variable in a `for` command, the value is increased by 1. If you append a double minus (“`--`”) at a variable, the value is decreased by 1.

The output of the program looks like the following:

```
10
11
12
13
14
15
```



It is not mandatory that you use the same variable in the three parts of the `for` command.

Exercise: Use Loops – Part III

Write a bash program that reads one one-digit integer number from input and counts from one to the entered number. It should display the name of the number on the screen (e. g., `One` for “1”) and not the number. Use a `for` loop.

Break a Loop

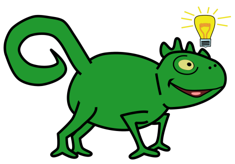
You can exit a loop before it is executed in the specified times. To do this you have to use the command `break`.

```
 1 #!/bin/bash
 2
   a=1
 4
   while true; do
 6     echo "$a"
       a=$(expr $a + 1)
 8     if test $a == 8; then
           break
10     fi
   done
```

Explanation: The variable `$a` is set to “1” in line 3. The `while` command (line 5) returns always “true” and the loop would be executed endless, if there was no `break` (line 9). If the value of `$a` is equal to “8”, the loop is interrupted.

The output looks like the following:

```
1
2
3
4
5
6
7
```



2.5.7 Use Program Options and Parameters

As described in the previous chapters, a command usually knows a lot of options and parameters. You can use them for your program, too.

The number of the passed options is stored in the variable “$#”.

```
#!/bin/bash

echo "$# options passed."
```

Now, let us start this short programm with some parameters:

```
geeko@da51:~> ./script.sh geeko suzie 100 Novell
4 options passed.
```

The command used to start the program is stored in the variable “$0”, the first parameter in variable “$1”, the second in “$2”, and so on.

```
#!/bin/bash

echo "\$0=$0"
echo "\$1=$1"
echo "\$2=$2"
echo "\$3=$3"
```

When starting the program with some parameters, the output, for example, looks like the following:

```
geeko@da51:~> ./script.sh geeko suzie 100 Novell
$0=./script.sh
$1=geeko
$2=suzie
$3=100
```

This method is not very flexible, because you need to know the number of parameters. In the last example, the parameter `Novell` is not computed, because the program expects only three parameters. Additionally, you can only access the command plus nine options.

The following variation is used more often: Test the first option. Then, remove the first item of the list and the second becomes the first item. This process is done by the command `shift`.

Normally, you put this test into a loop. A simple example looks like the following:

```
1 #!/bin/bash
2
  while test $# -gt 0; do
4     echo "Option: $1"
      shift
6 done
```

Explanation: The `test` command tests whether the list of options still has some items (line 3). If so, the first item is printed on the screen (line 4) and removed from the list (line 5). The loop is repeated until all items are removed.

The output of the program above looks like the following:

```
geeko@da51:~> ./script.sh geeko suzie 100 Novell
Option: geeko
Option: suzie
Option: 100
Option: Novell
```

To remove more than one item from the list, use `shift` *`number`*.

Normally a program used to examine the options and parameters looks like the following:

```
1  #!/bin/bash
2
   while test $# -gt 0; do
4      case $1 in
           -a) echo "Option a"
6              shift
               ;;
8          -b) echo "Option b"
               shift
10             ;;
           -f) echo "Option f with argument $2"
12             shift 2
               ;;
14         *)  echo "Wrong option: $1"
               break
16             ;;
       esac
18 done
```

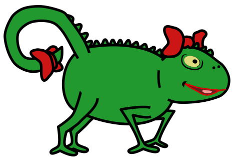
Let us do some tests:

❶
```
geeko@da51:~> ./script.sh -a -f /home/geeko
Option a
Option f with argument /home/geeko
```

❷
```
geeko@da51:~> ./script.sh -f /home/geeko -a
Option f with argument /home/geeko
Option a
```

❸
```
geeko@da51:~> ./script.sh -b /home/geeko
Option b
Wrong option: /home/geeko
```

❹
```
geeko@da51:~> ./script.sh /home/geeko -f
Wrong option: /home/geeko
```



Exercise: Use Program Options and Parameters

Describe what happens in the four examples above. Refer to the line numbers of the previous program code.

❶ ______________________________

____

____

____

❷ ______________________________

____

____

____

❸ ______________________________

____

____

____

❹ ______________________________

____

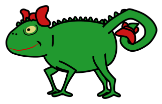
____

____

2.5.8 Create Shell Functions

Functions are parts of a program that are repeated often in a program at different locations. The program code is extracted from the main program into a function. The function is called from the *main program* when needed.

A function has the following has the following syntax:
`name() { commands }`



The function is called from the main program by its name. It is possible to add some parameters.

For example:

```
 1 #!/bin/bash
 2
   sum() {
 4         a=$(expr $1 + $2)
           echo "$1+$2="$a
 6 }

 8 sum 4 5
   sum 2 9
10 sum 12 1
   sum 204 2387
12 sum 5 3
   sum 21 39
```

Explanation: The main program begins in line 8. It only calls the function “sum” six times with different parameters.

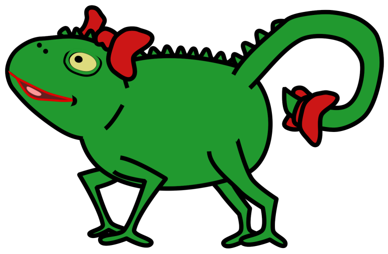
The function (lines 3–6) adds the given parameters and writes the sum on the screen. To refer to the parameters, the variables “$1” and “$2” are used.

Referring to function parameters works the same way as referring to program parameters (see section 2.5.7 on page 143).

The output of the program above is:

```
4+5=9
2+9=11
12+1=13
204+2387=2591
5+3=8
21+39=60
```

With parameters, you can transfer information into the function. To transfer information back into the main program, you can use the command `return` followed by a number.



For example:

```
 1 #!/bin/bash
 2
   validate() {
 4         if [ $1 = yes ]; then
                   return 1
 6         else
                   return 0
 8         fi
   }
10
   a=1
12 while true; do
           echo $a" -> To continue, enter \"yes\""
14         read b
           if validate $b; then
16                 break
           fi
18         a=$(expr $a + 1)
   done
20 echo "Stop!"
```

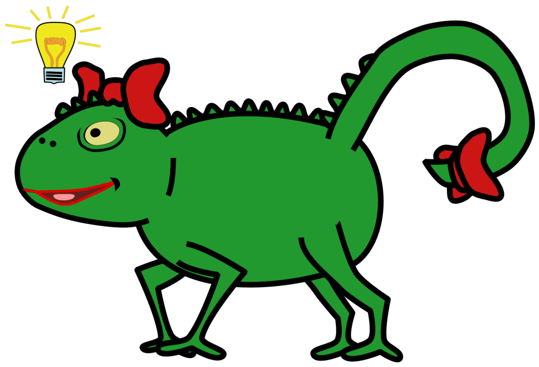
Explanation: The main program starts in line 11. It contains an endless loop (lines 12–19) and asks for some user input (line 14). The loop is only terminated, when the return value of the function `validate()` is "true" (equal to "0").

The function gets the input of the user as command parameter. If the user entered `yes`, the function returns the value "1". In all other cases, it returns "0".

If you enter `yes` four times, the output looks like this:

```
1 -> To continue, enter "yes"
yes
2 -> To continue, enter "yes"
yes
3 -> To continue, enter "yes"
yes
4 -> To continue, enter "yes"
no
Stop!
```

By default, a variable is valid for the whole program. It is called, *global variable*. But it is possible to create variables in a function that are valid only for the function. This type of variable is called a *local variable* and it is created with the command `local`.



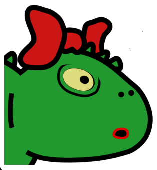
```
#!/bin/bash

localvar() {
        local a=1
        echo "function:" $a
}

a=0
echo "main before:" $a
localvar
echo "main after:" $a
```

Explanation: The main program starts in line 8. A variable `$a` is created and set to “0”. In line 10 a function is called. In this function a local variable is defined, that also is labeled “a” (line 4).

The output of the program is:

```
main before: 0
function: 1
main after: 0
```

If a local variable has the same name as a global variable, the global variable is overlaid by the local variable inside the function. After executing the function, the global variable gets its original value back.

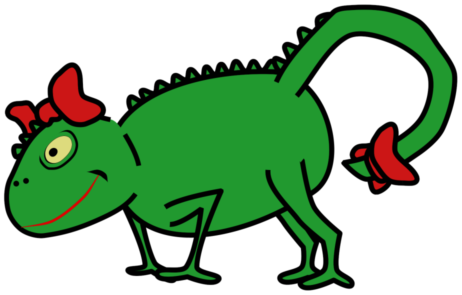


It seems to be easier to just use global variables. But professional developers try to avoid them, because they may get conflicts in programs with thousand of lines and hundreds of variables. It is a good programming style to declare variables only where they are needed.

Exercise: Create Shell Functions

Write a bash program that reads numbers from the input and calculates the mean average of these numbers. If you enter “finish”, the program should end.

Use at least one function to check whether the user entered a number or “finish”.



General Programming Conventions

❶ Write comments in your program.

❷ Use white-spaces to format your program.

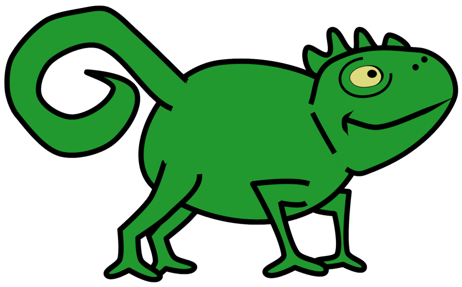
❸ Use self-explaning names for functions and variables.

❹ Use lowercase letters for variable names and function names.

❺ For constants, (they do not change their value during the program's run time) use uppercase letters.

❻ If you need variables for counting loops (e. g., in a `for` statement), use `$i`, `$j`, or `$k`.

❼ If you need a variable for only a short time (e. g., to buffer a value), use `$a`, `$b`, and `$c` for integer numbers and `$x`, `$y`, and `$z` for float numbers.



2.6 Understand the sed Stream Editor

sed is a stream editor that manipulates text files. That means it is different to many other text editors, like vi or gedit.

sed does not work interactively. It is controled by command-line options or a script, like the ex mode of vi.

The stream editor loads the current line of the file into an internal buffer for manipulation. The results are sent to standard out. The original file is never changed. If you want to save the output in a file, you have to redirect the standard output (see section 2.3.1 on page 85).

You can start sed in general by
`sed` *`options 'command' filename`*

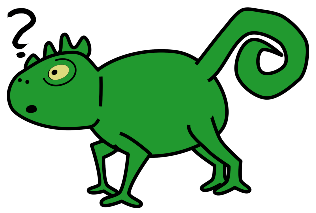
The simplest sed command is "`p`" (print), which prints out a line.

```
geeko@da51:~ > sed 'p' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET
LOREM IPSUM DOLOR SIT AMET


Lorem ipsum dolor sit amet,
Lorem ipsum dolor sit amet,
...
```

As you can see in the previous example, each line is printed out twice. sed sends each line once to standard out by default. To supress this, you have to add the option `-n`.

```
geeko@da51:~ > sed -n 'p' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit,
...
```

Let's look at the *`command`* part of the sed call.



First of all, we can specify an address declaration with the line numbers sed should examine.

Different kinds are possible:

No declaration – All lines

number – Only the line ***number***; e. g.

```
geeko@da51:~ > sed -n '1p' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET
```

start,end – The lines from ***start*** to ***end***; e. g.

```
geeko@da51:~ > sed -n '2,4p' loremipsum.txt

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit,
```

$ – The last line; e. g.

```
geeko@da51:~ > sed -n '36,$p' loremipsum.txt
facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit
praesent luptatum zzril delenit
au gue duis dolore te feugat nulla facilisi.
```

/regexp/ – The lines that match the regular expression ***regexp***; e. g.

```
geeko@da51:~ > sed -n '/^D/p' loremipsum.txt
Duis te feugifacilisi.
Duis autem dolor in hendrerit
Duis te feugifacilisi per suscipit
Duis te feugifacilisi.
Duis autem dolor in hendrerit in
```

Explanation: All lines are printed that begin with an uppercase "D".

number,/regexp/ – The lines from ***number*** to the first line that matches ***regexp***; e. g.

```
geeko@da51:~ > sed -n '9,/^D/p' loremipsum.txt
suscipit lobortis nisl
ut aliquip ex ea commodo consequat.
Duis te feugifacilisi.
```

Explanation: All lines are printed from line 9 to the first line that begins with an uppercase "D".

You can also use a regular expression as starting point in the same way.

The regular expressions of sed are similar to those you learned in section 2.3.2 on page 93. However some more characters need to be escaped with an backslash:

☞ \(...\)

☞ \{...\}

You can combine two or more sed commands using the option "`-e`":

```
geeko@da51:~ > sed -n -e '3p' -e '6p' -e '9p' loremipsum.txt
Lorem ipsum dolor sit amet,
ut lacreet dolore magna aliguam erat volutpat.
suscipit lobortis nisl
```

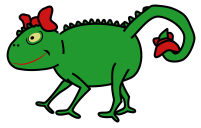
Explanation: The lines 3, 6 and 9 are printed on the screen.

Alternatively, you can separate the single commands by a semicolon.

```
geeko@da51:~ > sed -n '3p;6p;9p' loremipsum.txt
Lorem ipsum dolor sit amet,
ut lacreet dolore magna aliguam erat volutpat.
suscipit lobortis nisl
```

✍ Exercise: Understand sed – Part I

Use sed to view the lines 10 to 15 from the `loremipsum.txt` file.



Some more of the most important sed commands are listed in the following:

☞ "`a`" (append)

Adds one or more lines *after* the specified line(s).

```
geeko@da51:~ > sed '1aHello Geeko' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET
Hello Geeko

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit,
...
```

If you want to add more than one line, you have to mask the line break with a backslash.

```
geeko@da51:~ > sed '1aHello Geeko\
> Hello Suzie' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET
Hello Geeko
Hello Suzie

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit,
...
```

☞ "i" (insert)

Adds one or more lines *before* the specified line(s).

```
geeko@da51:~ > sed '1iHello Geeko' loremipsum.txt
Hello Geeko
LOREM IPSUM DOLOR SIT AMET

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit,
...
```

☞ "r *filename*"

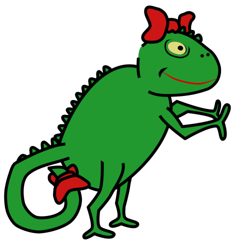
Inserts the content of the specified file behind the specified line(s).

```
geeko@da51:~ > sed '1r numbers.txt' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET
0000
123
...
98798
765765

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit,
sed diem nonummy nibh euismodtincidunt
...
```

☞ "d" (delete)

Deletes the specified line(s).

```
geeko@da51:~ > sed '1,3d' loremipsum.txt
consectetuer adipiscing elit,
sed diem nonummy nibh euismodtincidunt
ut lacreet dolore magna aliguam erat volutpat.
```



☞ "c"

Replaces the specified lines.

```
geeko@da51:~ > sed '1,3cHello Geeko' loremipsum.txt
Hello Geeko
consectetuer adipiscing elit,
sed diem nonummy nibh euismodtincidunt
ut lacreet dolore magna aliguam erat volutpat.
```

Explanation: In this example lines 1 to 3 are replaced by the line “Hello Geeko”.

☞ “w”

Writes the output of sed into the specified file. You can use this command instead of redirecting the output using “>”.

```
geeko@da51:~ > sed -n '1w output.txt' loremipsum.txt
```

Explanation: A new file `output.txt` is created. It includes the first line of the file `loremipsum.txt`.

If you do not use the option “`-n`” like in the example, the content of `loremipsum.txt` is written to the screen (and not to `output.txt`). This option does not affect the file output.txt.

☞ “q” (quit)

Quits the stream editor.

```
geeko@da51:~ > sed '3q' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET

Lorem ipsum dolor sit amet,
```

Exercise: Understand sed – Part II

Use sed to remove the first ten lines of `loremipsum.txt`. The output should be saved in the file `shortlorem.txt`.

☞ *`y/character(s)/character(s)/`*

Changes single characters. The first character from the first set is repaced by the first character of the second set. The second character from the first set is replaced by the second character of the second set, and so on.

```
geeko@da51:~ > sed 'y/oim/XYZ/' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET

LXreZ YpsuZ dXlXr sYt aZet,
cXnsectetuer adYpYscYng elYt,
sed dYeZ nXnuZZy nYbh euYsZXdtYncYdunt
```

Explanation: The lowercase “o” is replaced by “X”. The lowercase “i” is replaced by “Y” and the lowercase “m” is replaced by “Z”.

Instead of using the slash you can use any other character. The last command could also look like `sed 'y?oim?XYZ?' loremipsum.txt` or like `sed 'yBoimBXYZB' loremipsum.txt`
The character next to command “y” defines which character is used to divide the two character sets and finishes the command.

☞ `s/`*`pattern`*`/`*`string`*`/`*`flag(s)`* (substitute)

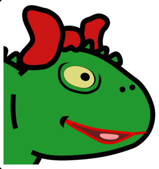
Replaces the first *pattern* (search pattern) with the second *string*. Normally, the search pattern includes regular expressions.

```
geeko@da51:~ > sed 's/dol/XXXXX/' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET

Lorem ipsum XXXXXor sit amet,
consectetuer adipiscing elit,
sed diem nonummy nibh euismodtincidunt
ut lacreet XXXXXore magna aliguam erat volutpat.
...
```

Explanation: The character combination “dol” is replaced by “XXXXX”.

Exercise: Understand sed – Part III

Use sed to replace “quis” by “quod” in the file `loremipsum.txt`. The output should be saved in file `quodlorem.txt`



Instead of using the slash you can use any other character.

Regular expressions are greedy, which means they try to match a string as large as possible. For example,

```
geeko@da51:~ > echo AABBCCAABB | sed s/A.*A/XX/
XXBB
geeko@da51:~ >
```

Explanation: The regular expression matches a string that begins with an “A”, followed by zero or more arbitrary characters and ends with an “A”. Because regular expressions are greedy, the first and last “A” matche and not (for example) the first and the second “A”.

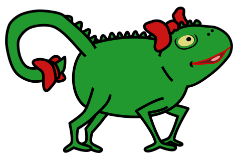
By default the "s" replaces only the first appearance of the search pattern. You can control the behavior of "s" by appending another substitution flag.

`g`	– Any appearance of the search pattern is replaced in a line.
`i`	– The case of the search pattern is ignored.
`p`	– The line where the search pattern matches is put out. In combination with the sed option "`-n`" only the matching lines are put out.

```
geeko@da51:~ > sed -n 's/dol/XXXXX/p' loremipsum.txt
Lorem ipsum XXXXXor sit amet,
ut lacreet XXXXXore magna aliguam erat volutpat.
Duis autem XXXXXor in hendrerit
vel illum XXXXXore eu feugiat
```

`w` ***`filename`***	– The lines that match the search pattern are written to a file such as `sed 's/dol/XXXXX/w output.txt' loremipsum.txt`
`s`	– The string is treated as single line. This makes possible to match a newline character (described by "`\n`")

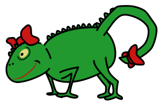
Of course, you can combine the substitution flags. E. g.
`sed 's/dol/XXXXX/gi' loremipsum.txt`



Sometimes you need the pattern that matches the regular expressing in the ***`string`*** part of the sed command again. For this there are nine register where you can store a match. You have to mark an expression you want to save with the grouping brackets `\(...\)` (see section 2.3.2 on page 97). You can refer to nine matches per line by \1 to \9.

```
geeko@da51:~ > sed  's/\([a-c]\)/X\1X/g' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET

Lorem ipsum dolor sit XaXmet,
XcXonseXcXtetuer XaXdipisXcXing elit,
sed diem nonummy niXbXh euismodtinXcXidunt
```

Explanation: The regular expression matches all “a”, “b” and “c” and encloses them with an “X” before and after.



☞ “`n`” (next)

The line that follows the search pattern is examined.

```
geeko@da51:~ >
```

☞ “`h`” (hold), “`g`”, “`G`” and “`x`”

The current line is loaded into a buffer (*pattern space*) from sed. Then, it is manipulated and sent to the output channel. Using the command “`h`”, you can copy the content of the pattern space into the holding buffer. With “`G`” you can insert the content of the holding buffer to the current line.

```
geeko@da51:~ > sed '1h;3G' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET

Lorem ipsum dolor sit amet,
LOREM IPSUM DOLOR SIT AMET
consectetuer adipiscing elit,
```

Explanation: The first line is copied into the holding buffer and is inserted after the third line.

Command “`g`” replaces the current line by the content of the holding buffer.

```
geeko@da51:~ > sed '1h;3g' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET

LOREM IPSUM DOLOR SIT AMET
consectetuer adipiscing elit,
sed diem nonummy nibh euismodtincidunt
```

The command “`x`” exchanges the content of pattern space and holding buffer.

```
geeko@da51:~ > sed '1h;2x;4G' loremipsum.txt
LOREM IPSUM DOLOR SIT AMET
LOREM IPSUM DOLOR SIT AMET
Lorem ipsum dolor sit amet,
consectetuer adipiscing elit,

sed diem nonummy nibh euismodtincidunt
ut lacreet dolore magna aliguam erat volutpat.
```

Explanation: The first command (“`1h`”) copies the line of `loremipsum.txt` into the holding buffer. “2x” exchanges the content of the second line (an empty line) with the content of the holding buffer. After the fourth line the new content of the holding buffer (the empty line) is inserted, because if command "`4G`".

You can write all or part of the sed commands in an external file and call it with sed option "`-f`":
`sed -f commands.txt loremipsum.txt`

The following rules have to be valid for the external command file:

❶ A line that starts with a hash ("`#`") is a comment line and not interpreted.

❷ In front of and after a command, no whitespaces are allowed.

❸ If you write more than one command in a line, you have to divide the commands by semicolons.

The whole command script is executed for each line of the text file.

✍ Exercise: Understand sed – Part IV

Write a sed script, that replaces "dolor" with "XXXXX" and each "o" with "O" in the file `loremipsum.txt`. All lines beginning with "Duis" should be removed. The output should be saved in the file `scriptlorem.txt`.

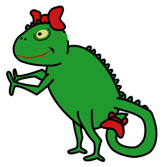
Although the examples here appear a little bit simple, you might use sed later in your bash programs to manipulate files. The introduced functions of adding and removing items of a text file is important, e. g., if you want to evaluate automatically generated log files of physics experiments.

3 Linux Services for Developers

A common misconception of software developers is that they are sitting alone in their dark garages writing programs. But, the reality is different. Professional developers often belong to a team and are responsible for a part of a program. Therefore, they have to work together, mostly in an office.

Writing programs in a team requires tools that help manage the development. In this chapter, we introduce two important services for a developer:

- ☞ A version control system
- ☞ A web server



3.1 The Version Control System Subversion

When two or more people are working at the same document at the same time, you have got a problem: only the version of the person that saves the file last stored. The version of the first person is replaced. It is not possible to restore the first version, even if it was better than the second one.

To solve parts of this problem, you can add a timestamp in your filename. Then, for example, at the end of each day someone collects all new versions of the file, compares them and creates a "final version of the day" with all changes. This might work for small files and small groups, but is not very user friendly, even if you use the command `diff` to find the changes in a file. The syntax is:

```
diff file1 file2
```

The output looks like this:

```
 1 geeko@da51:~> diff loremipsum.txt loremipsum2.txt
 2 11d10
   < Duis te feugifacilisi.
 4 18a18
   > Duis te feugifacilisi.
 6 22c22
   < nisl ut aliquip ex en commodo consequat.
 8 ---
   > ut aliquip ex commodo consequat.
10 34,36d33
   < vulputate velit esse molestie consequat,
12 < vel illum dolore eu feugiat nulla
   < facilisis at vero eros et accumsan
```

Explanation: Each difference starts with a number–character–number combination. The first number is the line number of the first file. The second number is the line number of the second file. The character between describes the kind of change. The most important characters are:

`a` – A line is added (line 4–5).
`d` – A line is deleted (line 2–3).
`c` – The content of the line changed (line 6–9).

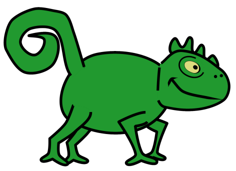
If there are more differences in several consecutive lines, the starting line and the ending line are separated by a colon (line 10). The lines that changed are also listed. A preceeding "<" indicates that the line is not available in the second file anymore. A preceeding ">" indicates that the line new in the second file.

A better choice is to use a versioncontrol software. This has two advantages:

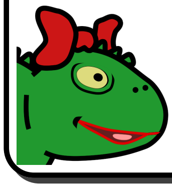
❶ Several people can work on one file simultaneously.

❷ Previous versions of a file are stored and can be restalled.

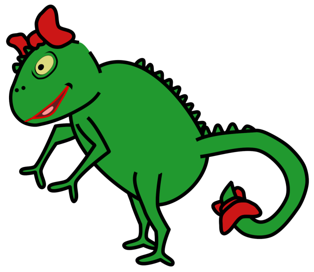
The dinosaur of versioncontrol systems is the open-source software Concurrent Versions System (CVS). But, because of some missing funtionalities (e. g., renaming files), the system Subversion was designed to be the successor to CVS.



There is a free book about Subversion available at `http://svnbook.red-bean.com`.

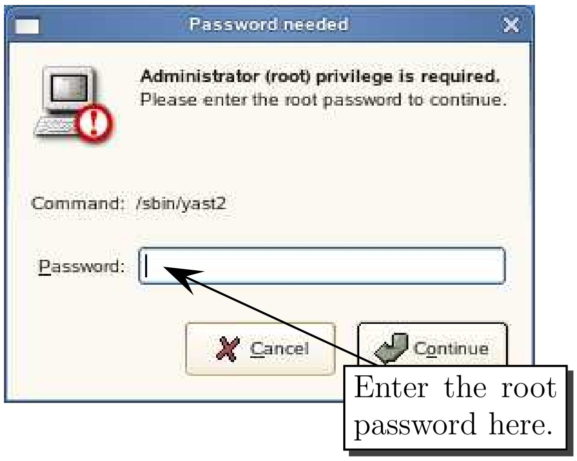


You can use Subversion for all kinds of files, and not only for program code (e. g., OpenOffice documents). We recommend using it when you prepare a presentation with others or you want to be sure that you can access previous document versions.



3.1.1 Install Subversion

Subversion is available on the SUSE Linux Enterprise DVD, but it is not installed by default. To install it, select the YaST icon in the `System` group of the Application Browser.



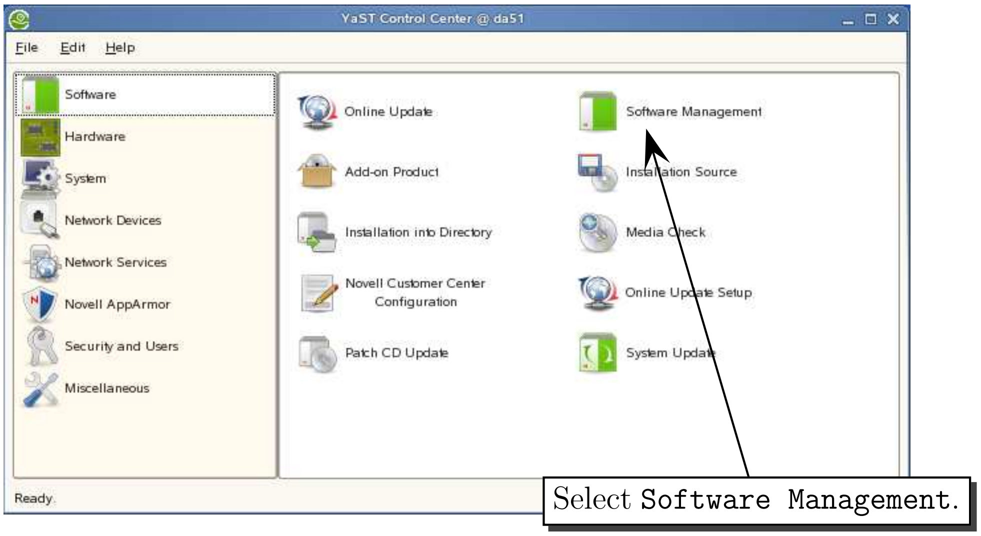
Administrator (root) privilege is required.
Please enter the root password to continue.

Command: /sbin/yast2

Password:

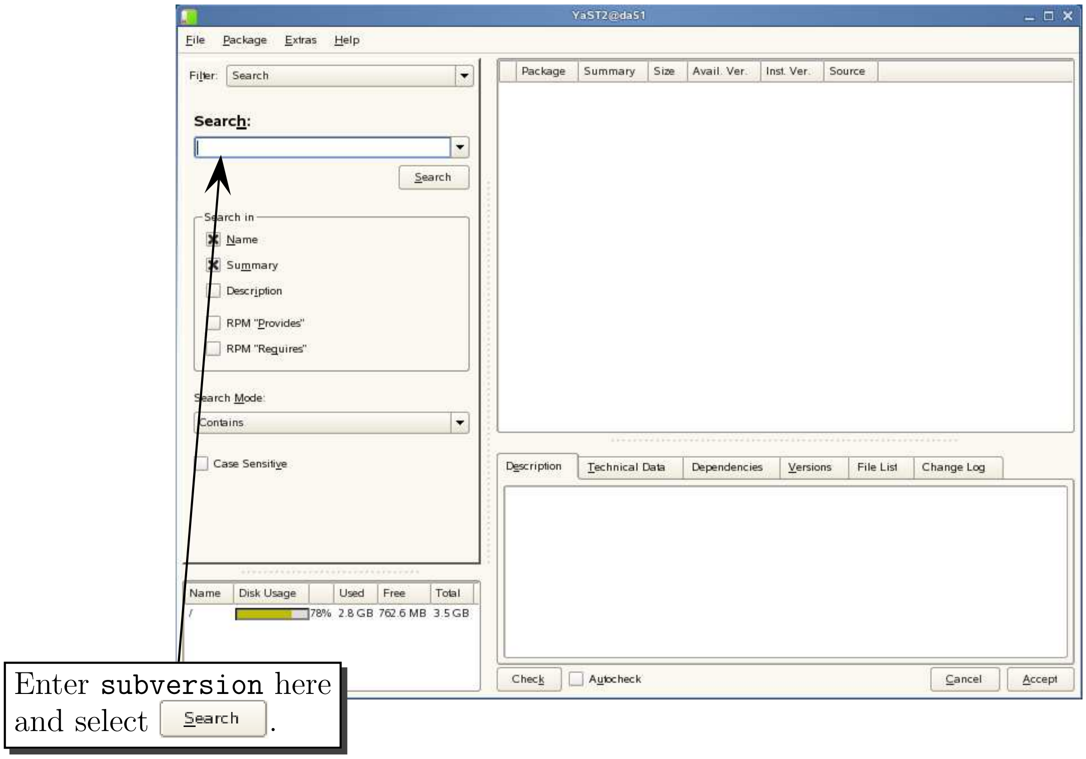
Enter the root password here.

The YaST Control Center appears.

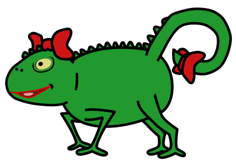


Select `Software Management`.

The Software Management dialog appears.


YaST2@da51
File Package Extras Help
Filter: Search
Search:
Search
Search in
Name
Summary
Description
RPM "Provides"
RPM "Requires"
Search Mode:
Contains
Case Sensitive
Package Summary Size Avail. Ver. Inst. Ver. Source
Description Technical Data Dependencies Versions File List Change Log
Name Disk Usage Used Free Total
/ 78% 2.8 GB 762.6 MB 3.5 GB
Check Autocheck Cancel Accept

Enter `subversion` here and select Search.

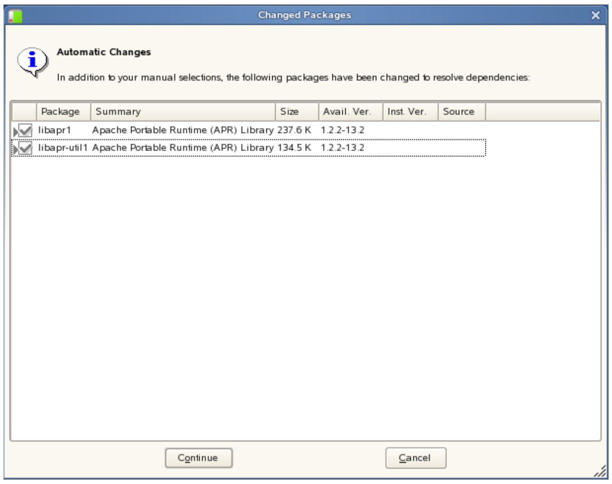


Two packages should be found:

- ☞ `subversion`
- ☞ `subversion-devel`

Select only `subversion` and then Accept.

There should appear a message that two more packages are required to resolve the dependencies.



Changed Packages

Automatic Changes

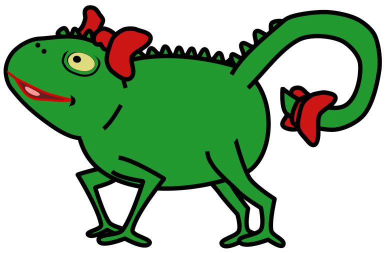
In addition to your manual selections, the following packages have been changed to resolve dependencies:

	Package	Summary	Size	Avail. Ver.	Inst. Ver.	Source
✓	libapr1	Apache Portable Runtime (APR) Library	237.6 K	1.2.2-13.2		
✓	libapr-util1	Apache Portable Runtime (APR) Library	134.5 K	1.2.2-13.2		

Select Continue to go on.

Exercise: Install Subversion

Install Subversion as described above.



3.1.2 Set up a Subversion Repository

In Subversion, you have a central repository for all projects. You do not work in this repository. You always use a local copy, and this is synchonized with the repository.

The repository can be stored on your local computer but also on a computer in the network. The last case is more usual. Subversion supports a lot of network protocols and it does not make any differences where the repository is located. In this course, we will use a local repository.

The first step in working with Subversion is to create a repository. Therefore, you have to use the command `svnadmin`.

`svnadmin` provides some options. The option we want to introduce here is `create`. With this option, a new Subversion repository is created. The syntax is:

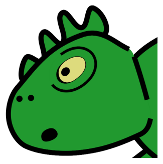
`svnadmin create` *`directory`*

The directory you specify must already exist. The repository is created inside. You can create a repository only in a directory you have write permission for. So, normally you will use the command `svnadmin` as user root.

To switch to user root, you have the following possibilities:

- ☞ Log out as normal user and log in as user root. → Bad solution
- ☞ Switch to one of the six virtual terminals by pressing `Ctrl` + `Alt` + `Fx` and log in as root. → Only a little bit better solution
- ☞ Start a terminal emulation from the main menu and enter the command `su -` (*switch user*). You are then asked to enter the root password. → Best solution

The prompt changes when you are logged in as root. The username is not shown in the prompt anymore and at the end of the prompt there is a hash sign (“`#`”).



Before you create a repository think of where you want to create it. We recommend to use the `/srv` directory.

```
da51:~ # mkdir /srv/repository
da51:~ #
```

Then you can create the repository.

```
da51:~ # svnadmin create /srv/repository/
da51:~ #
```

Some files are created.

```
da51:~ # ls -l /srv/repository/
insgesamt 8
drwxr-xr-x 2 root root 128 2006-10-12 08:20 conf
drwxr-xr-x 2 root root  48 2006-10-12 08:20 dav
drwxr-sr-x 5 root root 256 2006-10-12 08:20 db
-r--r--r-- 1 root root   2 2006-10-12 08:20 format
drwxr-xr-x 2 root root 360 2006-10-12 08:20 hooks
drwxr-xr-x 2 root root 104 2006-10-12 08:20 locks
-rw-r--r-- 1 root root 229 2006-10-12 08:20 README.txt
da51:~ #
```

You also have to make sure that normal users (or better the users of a special group) have the write permission for this directory.

```
da51:~ # chgrp -R users /srv/repository/
da51:~ # chmod -R 775 /srv/repository/
da51:~ #
```

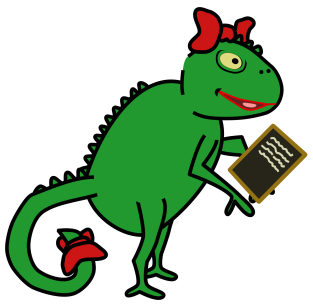
When you have finished this, you can log out of the terminal emulation as user root, by using the `exit` command.

```
da51:~ # exit
exit
geeko@da51:~>
```

You are now logged in as a normal user.

Exercise: Create a Subversion Repository

Create a Subversion repository labeled `repository` in the `/srv` directory. Ensure that all users are allowed to change the content of the repository.



3.1.3 Start a Project

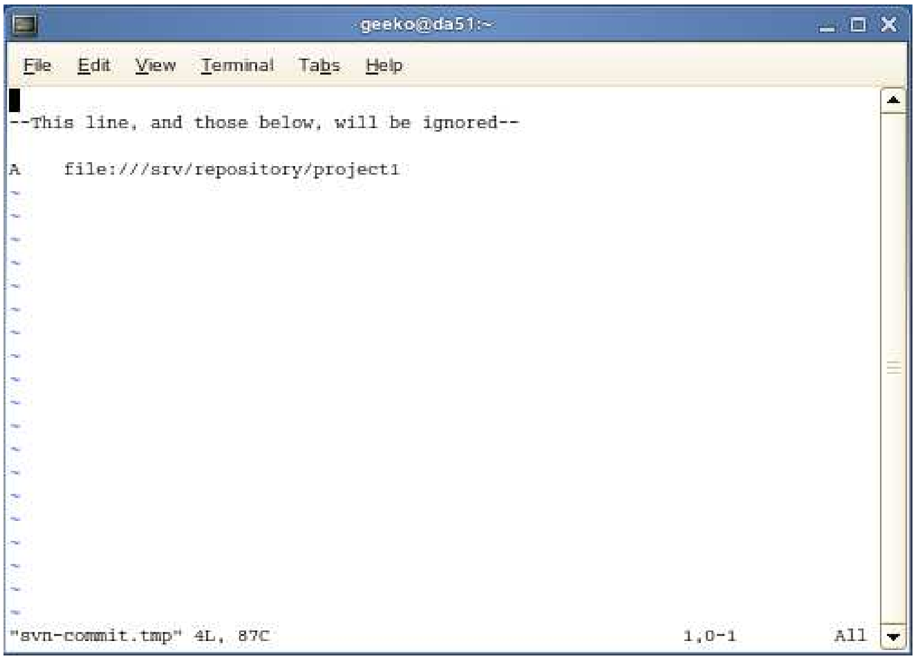
At the moment, you have an empty repository. Now you have to create a directory in Subversion where you can store the files for a project.

This can be done by the command `svn` with the option `mkdir`.

```
geeko@da51:~> svn mkdir file:///srv/repository/project1
```

Explanation: The string `file:///srv/repository/project1` specifies where (`/srv/repository/`) the new project directory should be created and what its name is (`project1`). The prefix `file://` says that it is in the local file system. This notation is similar to the address of web pages (e. g., `http://www.novell.com`).

The vi editor starts, and you can enter a description of your project.



```

--This line, and those below, will be ignored--

A    file:///srv/repository/project1
~
~
~
"svn-commit.tmp" 4L, 87C                                    1,0-1         All
```

After leaving vi, the directory is created, and you should get a message like this:

```
Committed revision 1.
```

Exercise: Create a Subversion Directory

Create a directory `project1` in your Subversion repository.

3.1.4 Prepare for Working with Subversion

In most cases, you already have some project files when you create your Subversion directory. If these files are stored in the same directory in the file system, you can import all files in one step.

The command to do this is `svn import`. You also have to specify the directory you want to import and the location of the subversion directory. For example:

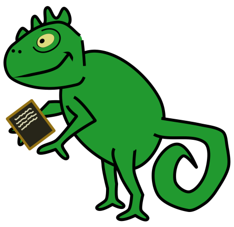
```
geeko@da51:~> svn import my_project file:///srv/repository/project1/
Adding         my_project/numbers.txt
Adding         my_project/loremipsum.txt
Adding         my_project/dirgrep
Adding         my_project/dirgrep/loremipsum2.txt

Committed revision 2.
geeko@da51:~>
```

Explanation: `my_project` is the name of the local directory that should be imported. `file:///srv/repository/project1/` is the location of the Subversion directory where the files are imported.
After entering the command, vi starts again, and you can enter a note. After exiting vi, the files are added.

The notes you have to enter in vi can also be specified after the `svn` option `-m "`*`notes`*`"`.

Exercise: Import Files into Subversion

On the book CD there is an archive file `grep-demo.tar.gz`, which you already know from section 2.3.2 on page 91. Extract the archive into an empty directory `my_project`. Add this directory to `project1`.
Remove the `my_project` directory from your home directory after.



If you want to change the files in the repository, you have to create a local copy in your home directory first. This is done by the command `svn checkout`. You have to specify which Subversion directory you want to check out and where the local copy should be located. For example:

```
geeko@da51:~> svn checkout file:///srv/repository/project1 project1
A    project1/numbers.txt
A    project1/loremipsum.txt
A    project1/dirgrep
A    project1/dirgrep/loremipsum2.txt
Checked out revision 2.
geeko@da51:~>
```

Explanation: `file:///srv/repository/project1` is the location of the Subversion directory that is copied. `project1` is the name of the directory including the local copy. Because `project1` does not include any path description, the directory is created at the current position in the file system.

All files from the repository directory are copied. There is also a hidden directory `.svn` created, which includes meta information of Subversion.

```
geeko@da51:~> ls -la project1/
total 9
drwxr-xr-x  4 geeko users  160 2006-10-12 09:31 .
drwxr-xr-x 25 geeko users 1448 2006-10-12 09:31 ..
drwxr-xr-x  3 geeko users  104 2006-10-12 09:31 dirgrep
-rw-r--r--  1 geeko users 1344 2006-10-12 09:31 loremipsum.txt
-rw-r--r--  1 geeko users   55 2006-10-12 09:31 numbers.txt
drwxr-xr-x  7 geeko users  296 2006-10-12 09:31 .svn
geeko@da51:~>
```

Exercise: Create a Local Copy

Create a local copy of the `project1` Subversion directory. The name of the local copy should also be `project1`.

3.1.5 Use Subversion

During your daily work with Subversion you will do the following tasks frequently:

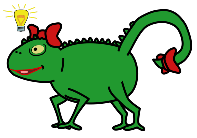
- **Transfer your work into the repository.** When you change the files of your local copy, you have to put your changes frequently (e. g., once a day) into the repository, so that the members of your team can get the latest version.

  This is called *commit.*

- **Get the changes from others.** To be sure that you use the latest version before you start working with your local copy, you have to get all changes from your team members.

  This is called *update.*

- ☞ **Add new files to the repository.** Add a new file to the Subversion directory to ensure that your team members also get it.
- ☞ **Remove files from the repository.** Remove files from the Subversion directory to ensure that your team members do not use it anymore.
- ☞ **Move or rename files inside the repository.** Move or rename a file inside the Subversion directory.
- ☞ **Solve conflicts.** In a large team, you cannot avoid that file is changed by two people in the same time period (that means, between update and commit). Subversion provides functionalities that help solve these conflicts.



Transfer your Work into the Repository

Because you are working with a local copy and not in the Subversion repository directly, you have to syncronize these two directories periodically. On the one hand your changes have to be transfered into the repository and on the other hand the changes from others in the repository have to be transfered to you.

To transfer your changes to the repository, use the command `svn commit`. You can add the name of a file or subdirectory to transfer changes in the file respectively in the subdirectory.

```
geeko@da51:~/project1> svn commit loremipsum.txt
Sending        loremipsum.txt
Transmitting file data .
Committed revision 3.
geeko@da51:~/project1>
```

Explanation: Because the file `loremipsum.txt` is specified in the `svn commit` command, only this file is transfered, although there are also changes in other files in the project.

If you do not add any options the files of the current directory and all subdirectories are transfered.

```
geeko@da51:~/project1> svn commit
Sending        dirgrep/loremipsum2.txt
Sending        numbers.txt
Transmitting file data ...
Committed revision 4.
geeko@da51:~/project1>
```

Explanation: Without any file specification all changes in files of the current directory and all subdirectories are transfered.

If there are no changes, no file will be transfered.

Before the files are transfered, vi starts, and you can enter a note about you changes.

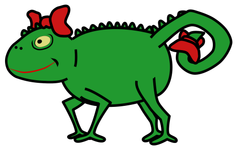
Get the Changes from Others

Similar to `svn commit`, works `svn update`. This command transfers all files that changed into the repository in your working copy.

Without any file specification, all files and subdirectories are checked for updates. But you also can update a single file or subdirectory by appending its filename to the `svn update` command.

```
1 geeko@da51:~/project1> svn update
2 U    loremipsum.txt
  Updated to revision 6.
4 geeko@da51:~/project1>
```

Explanation: There is a newer version of the file `loremipsum.txt` available in the repository. It replaces the version of the file in your working copy.
The "`U`" at the beginning of line 2 indicates that a file is updated.



Exercise: Update Your Working Copy

In section 1.2.1 on page 20, you created a new user account. Open a new terminal emulation and log in as this new user by entering `su - `*`username`*.

This second user should checkout the `project1` Subversion directory into his home directory. Change something in the project files, and commit this changes into the repository.

Now, get these changes into the Subversion working copy of your normal user.

Add New Files to the Repository

To add a new file to the repository, use the command `svn add` *`filename`*.

```
geeko@da51:~/project1> svn add help.txt
A         help.txt
geeko@da51:~/project1>
```

The file is marked to be added. It will be transfered into the repository by the command `svn commit`.

✍ Exercise: Add a File to the Repository

Create a new file as normal user and add it to the repository. Switch to your second user account and get the new file by updating your files.

Remove Files from the Repository

To remove a file from a project in the Subversion repository, use the command `svn delete` *`filename`*.

```
geeko@da51:~/project1> svn delete help.txt
D         help.txt
geeko@da51:~/project1>
```

The file is removed from your working copy immediately. It is removed from the repository after a `svn commit`.

✍ Exercise: Remove a File from the Repository

Remove the file you created in the last exercise as normal user. Switch to your second user account and update your working copy.

Move or Rename Files Inside the Repository

If you want to rename a file or move it in another subdirectory of the project, use the command `svn rename` *`old new`*.

Internally a copy of the old file is added with the new filename. The original file with the old file is deleted.

```
1 geeko@da51:~/project1> svn move dirgrep/loremipsum2.txt dirgrep/README
2 A         dirgrep/README
  D         dirgrep/loremipsum2.txt
4 geeko@da51:~/project1>
```

Explanation: The “`A`” at the beginning of line 2 indicates, that a file is added. The “`D`” at the beginning of line 3 indicates, that a file is deleted.

The changes are made directly in your working copy. In the repository they are made after a `svn commit`.

Exercise: Rename a File

Rename the file `dirgrep/loremipsum2.txt` to `dirgrep/README` as normal user. Switch to your second user account and update your working copy.

Copy Files Inside the Repository

If you want to copy a file or move it into another subdirectory of the project, use the command `svn copy` ***`source destination`***.

```
1 geeko@da51:~/project1> svn copy loremipsum.txt dirgrep/loremipsum3.txt
2 A         dirgrep/loremipsum3.txt
  geeko@da51:~/project1>
```

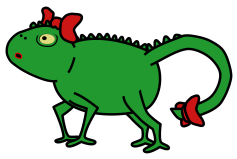
Explanation: The “`A`” at the beginning of line 2 indicates, that a file is added.

The changes are made directly in your working copy. In the repository, they are made after a `svn commit`.

Solve Conflicts

You cannot avoid that file is changed by two people in the same time period (that means, between update and commit).

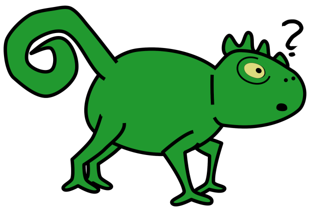
Example: User Tux updates his working copy of the repository project1 at 8:20 a. m. User Geeko updates his working copy of the same repository at 8:30 a. m. Tux finishes his work at the files at 10:00 a. m. and commits his changes into the repository. User Geeko works until 5:00 p. m. on the files and now he wants to commit his changes.



There appears to beno conflict in the upper example, if both do not touch the same files of the project. But, if so, user Geeko gets a conflict message when he tries to commit his files. For example:

```
geeko@da51:~/project1> svn commit
Sending          loremipsum.txt
svn: Commit failed (details follow):
svn: Out of date: '/project1/loremipsum.txt' in transaction '9-1'
svn: Your commit message was left in a temporary file:
svn:    '/home/geeko/project1/svn-commit.tmp'
geeko@da51:~/project1>
```

Explanation: In line 4 you can see that the conflict appears in the file `/project1/loremipsum.txt`.



To see what changed in the file in the meantime, use the command `svn diff` ***`filename`***.

The output looks like this:

```
 1 geeko@da51:~/project1> svn diff loremipsum.txt
 2 Index: loremipsum.txt
   ===================================================================
 4 --- loremipsum.txt      (revision 6)
   +++ loremipsum.txt      (working copy)
 6 @@ -36,4 +36,4 @@
    facilisis at vero eros et accumsan
 8  et iusto odio dignissim qui blandit
    praesent luptatum zzril delenit
10 -au gue duis dolore te feugat nulla facilisi.
   +au duis dolore te feugat nulla facilisi.
12 geeko@da51:~/project1>
```

Explanation: In line 4 and 5 is noted, which symbols are used for the two files. A "-" represents the file `loremipsum.txt` of the repository (line 4), a "+" represents the file `loremipsum.txt` of the working copy (line 5).

In line 6 it is written which lines changed. The first two numbers (separated by comma) are marked with a "-". They represent the line number and column number of the file in the repository. The second two numbers are marked with a "+". They represent the line number and column number of the file in the working copy.

Then some lines of the file are listed (lines 7 – 11). Lines with a prefixed "-" (line 10) are only included in the file version of the repository. Lines with a prefixed "+" (line 11) are only included in the file version of the working copy.

If you see the differences, you can decide now whether you want to change these lines in your working directory.

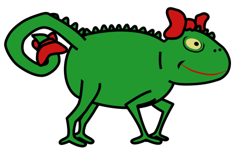
If you want to throw away all your changes, use the command `svn revert` ***`filename`***.

```
geeko@da51:~/project1> svn revert loremipsum.txt
Reverted 'loremipsum.txt'
geeko@da51:~/project1>
```

If you want to resolve the conflict, try `svn update` ***`filename`***. If the changes in the repository version and the working copy do not overlap, both files are merged by Subversion.

```
1 geeko@da51:~/project1> svn update loremipsum.txt
2 G    loremipsum.txt
  Updated to revision 12.
4 geeko@da51:~/project1>
```

Explanation: The "`G`" at the beginning of line 2 stands for merged. After merging the file, the changes can be commited.



If the changes in the repository version and the working copy overlap, you get a conflict when trying `svn update`.

```
1 geeko@da51:~/project1> svn update loremipsum.txt
2 C    loremipsum.txt
  Updated to revision 15.
4 geeko@da51:~/project1>
```

Explanation: The "`C`" at the beginning of line 2 indicates that there is a conflict.

If you open the file with a text editor now, you can see marks in the lines, where the conflict appeared. For example:

```
1 ...
2 facilisis at vero eros et accumsan
  et iusto odio dignissim qui blandit
4 praesent luptatum zzril delenit
  <<<<<<< .mine
6 duis dolore de feugiat nulla facilisi.
  =======
8 au gue duis dolore magna aliguam erat volutpat.
  >>>>>>> .r15
```

Explanation: The text between `<<<<<<< .mine` and `=======` is the version in your working copy (line 6). The text between `=======` and `>>>>>>> .r15` is the version in the repository (line 8). The number `r15` specifies the version number.

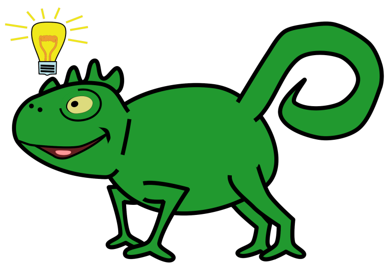
Now you can decide which version you prefer and solve the conflict. After saving the corrected version of the file, tell Subversion that the conflict is solved. This is done by the command `svn resolved` ***`filename`***.

```
1 geeko@da51:~/project1> svn resolved loremipsum.txt
2 Resolved conflicted state of 'loremipsum.txt'
  geeko@da51:~/project1>
```

Now you can commit your changes.

Exercise: Solve Conflicts

Use your two user accounts to create a conflict in the file `loremipsum.txt` that can not be solved by merging files. Resolve the conflict manually.



3.1.6 Manage Subversion

You may ask yourself why you should enter notes in vi all the time? Subversion logs this information with every commit.

```
geeko@da51:~/project1> svn log
------------------------------------------------------------------------
r5 | tux | 2006-10-12 07:54:34 -0400 (Thu, 12 Oct 2006) | 2 lines

Renamed loremipsum2.txt to help.txt

------------------------------------------------------------------------
r4 | tux | 2006-10-12 07:51:24 -0400 (Thu, 12 Oct 2006) | 2 lines

Typo

------------------------------------------------------------------------
r3 | geeko | 2006-10-12 07:51:04 -0400 (Thu, 12 Oct 2006) | 2 lines

Added empty lines after each paragraph.

------------------------------------------------------------------------
r2 | geeko | 2006-10-12 07:49:59 -0400 (Thu, 12 Oct 2006) | 2 lines

Initial import

------------------------------------------------------------------------
r1 | geeko | 2006-10-12 07:49:55 -0400 (Thu, 12 Oct 2006) | 2 lines

Project created

------------------------------------------------------------------------
geeko@da51:~/project1>
```

Explanation: A horizontal line separates the revisions of the project. In the first line of each revision, the following information are listed:

Revision number | User name | Date and time of the commit | Number of comment lines

Then the notes are written.

If you want to see only the logs from revision *X* to *Y* add the option `-r` *X*`:`*Y*.

```
geeko@da51:~/project1> svn log -r 4:5
------------------------------------------------------------------------
r4 | tux | 2006-10-12 07:51:24 -0400 (Thu, 12 Oct 2006) | 2 lines

Typo

------------------------------------------------------------------------
r5 | tux | 2006-10-12 07:54:34 -0400 (Thu, 12 Oct 2006) | 2 lines

Renamed loremipsum2.txt to help.txt

------------------------------------------------------------------------
geeko@da51:~/project1>
```

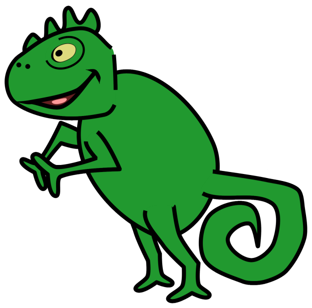
Explanation: The order of log entries differs when you use `-r` *X*`:`*Y* or `-r` *Y*`:`*X*.

If you want to see, which files were changed, use the `--verbose` option.

```
geeko@da51:~/project1> svn log --verbose
------------------------------------------------------------------------
r5 | tux | 2006-10-12 07:54:34 -0400 (Thu, 12 Oct 2006) | 2 lines
Changed paths:
   A /project1/dirgrep/help.txt (from /project1/dirgrep/loremipsum2.txt:3)
   D /project1/dirgrep/loremipsum2.txt

Renamed loremipsum2.txt to help.txt

------------------------------------------------------------------------
r4 | tux | 2006-10-12 07:51:24 -0400 (Thu, 12 Oct 2006) | 2 lines
Changed paths:
   M /project1/loremipsum.txt

Typo

------------------------------------------------------------------------
r3 | geeko | 2006-10-12 07:51:04 -0400 (Thu, 12 Oct 2006) | 2 lines
Changed paths:
   M /project1/loremipsum.txt

Added empty lines after each paragraph.

------------------------------------------------------------------------
r2 | geeko | 2006-10-12 07:49:59 -0400 (Thu, 12 Oct 2006) | 2 lines
Changed paths:
   A /project1/dirgrep
   A /project1/dirgrep/loremipsum2.txt
   A /project1/loremipsum.txt
   A /project1/numbers.txt

Initial import

------------------------------------------------------------------------
r1 | geeko | 2006-10-12 07:49:55 -0400 (Thu, 12 Oct 2006) | 2 lines
Changed paths:
   A /project1

Project created

------------------------------------------------------------------------
geeko@da51:~/project1>
```

Exercise: View Subversion Logs

View your Subversion logs and check whether your notes are self explanatory.



If you want to see a previous version of a text file, you can use the command `svn cat` to view the content. Use the option `-r` to specify the revision number. For example:

```
geeko@da51:~/project1> svn cat -r 2 loremipsum.txt
LOREM IPSUM DOLOR SIT AMET

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit,
sed diem nonummy nibh euismodtincidunt
...
et iusto odio dignissim qui blandit
praesent luptatum zzril delenit
au gue duis dolore te feugat nulla facilisi.
geeko@da51:~/project1>
```

Once you know what the revision number of a file is, you can rebuild it, or parts of it. Use the command `svn merge` to merge a old version with the current one. Therefore you have to know the current revision number (see the log). With the option `-r` ***`current`***`:`***`previous`***, you specify the revisions you want to merge. For example:

```
1 geeko@da51:~/project1> svn merge -r 5:2 file:///srv/repository/project1
2 U    loremipsum.txt
  D    dirgrep/help.txt
4 A    dirgrep/loremipsum2.txt
  geeko@da51:~/project1>
```

Explanation: The current revision is 5. The project should be rolled back to revision 2. The file `loremipsum.txt` is replaced by the old revision (line 2). The file `dirgrep/help.txt` was not available in revision 2. It is deleted (line 3). The file `dirgrep/loremipsum2.txt` is added (line 4), because it was deleted in the meantime.

It is also possible to get conflicts here. They are indicated by a "`C`" at the beginning of a line. Use the methods you learned before (see section 3.1.5 on page 174) to solve them.

If you want to rebuild a file you (or someone else) removed from the repository, use the command `svn copy`. With the option `-r` ***`revision`*** you can specify the revision where the still file exists. You also have to specify where the copy of the deleted file should be located in your working copy and what its name is. For example:

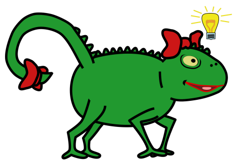
```
geeko@da51:~/project1> svn copy -r 4 file:///srv/repository/project1/dirgrep/lor
emipsum2.txt ./dirgrep/loremipsum2.txt
A         dirgrep/loremipsum2.txt
geeko@da51:~/project1>
```

Explanation: The file `/srv/repository/project1/dirgrep/loremipsum2.txt` was deleted by renaming in revision 5 (see line 5 in the log on page 178). Revision 4 is the last revision that exists. It is copied from this revision into the working copy at `./dirgrep/loremipsum2.txt`.

If the project you manage with Subversion is terminated and you do not need the files in the repository anymore, use `svn rm` to remove the project.

```
geeko@da51:~> svn rm file:///srv/repository/project1

Committed revision 3.
geeko@da51:~>
```

The files of your working copy you can remove manually.



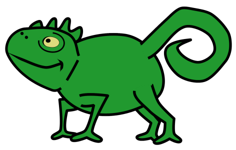
3.1.7 Use Subversion in a Network

If you want to access a Subversion repository that is not located on your local computer but on a server in your network, There are two posibilities:

❶ You can access the server via HTTP. Therefore a web server using WebDAV must run on the server, too.

❷ You can access a Subversion server that runs on the machine.

We want to introduce the second posibility here and what you have to do to set up a Subversion server.



Set Up a Subversion Server

After installing the Subversion software package there is also a programm called `svnserve` installed. This program is the Subversion server. You can start it by entering `svnserv -d` at the command line. There is no further output.

```
geeko@da51:~> svnserve -d
geeko@da51:~>
```

But, now you can use the Subversion commands on another computer to work with the repository. Instead of `file://path` you now have to use `svn://IP-address/path` or `svn://hostname/path`. For example:

☞ Use the IP address of the server.

```
geeko@da53:~> svn checkout svn://10.10.0.51/srv/repository/project1/
A    project1/numbers.txt
A    project1/loremipsum.txt
A    project1/dirgrep
A    project1/dirgrep/help.txt
Checked out revision 5.
geeko@da53:~>
```

☞ Use the hostname of the server.

```
geeko@da54:~> svn checkout svn://da51.digitalairlines.com/srv/repository/projec
t1/
A    project1/numbers.txt
A    project1/loremipsum.txt
A    project1/dirgrep
A    project1/dirgrep/help.txt
Checked out revision 5.
geeko@da54:~>
```

This kind of server connection is not secure, because…

❶ the information that is transfered via the network is not encrypted

❷ every user in the network can access the repository

To solve the first problem, you can connect with the server using the svn+ssh protocol. When you try to connect using the svn+ssh protocol, you have to authenticate.

```
geeko@da54:~> svn checkout svn+ssh://192.168.2.132/srv/repository/project1/
Password:
```

You need a local account on the Subversion server to use this method But, it is still possible to use the simple svn protocol for other users.

To stop the Subversion server, enter `killall svnserve`.

To enforce user authentication, you can start the Subversion server with the option `-t` (tunneled).

```
da51:~ # svnserve -t
( success ( 1 2 ( ANONYMOUS EXTERNAL ) ( edit-pipeline ) ) )
```

It is not possible to connect by using the simple svn protocol now.

```
geeko@da54:~> svn checkout svn://192.168.2.132/srv/repository/project1/
svn: Can't connect to host '192.168.2.132': Connection refused
geeko@da54:~>
```

But svn+ssh works fine.

```
geeko@da54:~> svn checkout svn+ssh://192.168.2.132/srv/repository/project1/
Password:
```

Exercise: Set Up a Subversion Server

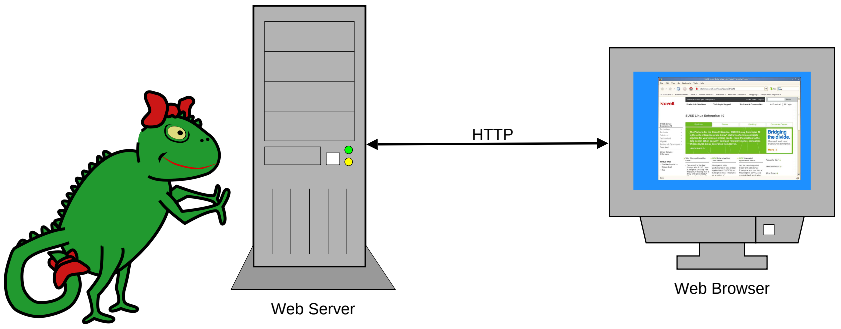
Start a Subversion server on your computer. Check out the files of `project1` of your neighbour.

3.2 The Web Server Apache

Once your program get a stable version, you want to share it with other developers. To do this, you need a web server to provide some documentation and your software packages for download. In this chapter, you will learn how to set up your own web server.

A web server delivers data requested by a web browser. The data have different formats such as HTML files, image files, Flash animations, or sound files.

Web browsers and web servers communicate with each other by using HTTP (Hyper Text Transfer Protocol). The following diagram shows the relationship between browser, server, and HTTP:



Besides delivering data to the web browser, a web server performs tasks such as limiting access to specific web pages, logging access to a file, and encrypting the connection between server and browser.

The commonly used web server is Apache.

3.2.1 Install Apache

The Apache web server is not included in SUSE Linux Enterprise Desktop 10. But you can use the software packages from SUSE Linux Enterprise Server 10, because both are based on the same code. In the directory `packages/` on the book CD, you will find the following files:

- apache2-2.2.0-21.2.i586.rpm
- apache2-example-pages-2.2.0-21.2.i586.rpm
- apache2-prefork-2.2.0-21.2.i586.rpm

There are several ways to install RPM packages. In section 1.3 on page 23 you learned how to install RPM packages from the SUSE Linux Enterprise Desktop DVD using YaST.

In this chapter, you learn how to install software from the command line using the command `rpm`.

Start a terminal emulation. Then switch to user root by entering the command `su -`. You are asked to enter the root password.

To install a software package, use the command `rpm`. In the following, the syntax is described briefly.

All options of `rpm` can be found in the manual page (`man rpm`).

To install all RPM packages of the `/media/cdrecorder/packages/` directory, enter the following:

```
da51:~ # rpm -ihv /media/cdrecorder/packages/*rpm
```

The options are:

`i` – Install package.

`h` – Show the installation process using hash signs.

`v` – Enable verbose output.

After the options, the path is entered where the RPMs can be found. The wildcard ("*") means no or any characters can match the wildcard.

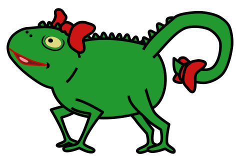
When the installation is finished, the output looks like the following:

```
da51:~ # rpm -ihv /media/cdrecorder/packages/*rpm
Preparing...                ########################################### [100%]
   1:apache2-example-pages  ########################################### [ 33%]
   2:apache2-prefork        ########################################### [ 67%]
   3:apache2                ########################################### [100%]
Starting SuSEconfig, the SuSE Configuration Tool...
Running module permissions only
Reading /etc/sysconfig and updating the system...
Executing /sbin/conf.d/SuSEconfig.permissions...
Checking permissions and ownerships - using the permissions files
        /etc/permissions.d/postfix
        /etc/permissions.d/susehelp
        /etc/permissions
        /etc/permissions.easy
        /etc/permissions.local
setting /usr/sbin/suexec2 to root:root 4755. (wrong permissions 0755)
Finished.
Updating etc/sysconfig/apache2...
looking for old 2.0 modules to be renamed...
Done.
da51:~ #
```

You will find more information on the `rpm` command in Appendix A on page 199.

Exercise: Install Apache

Install the Apache web server as described above.



3.2.2 Start and Test the Web Server

After installing the required software, start the web server. Switch to user root and enter `rcapache2 start`.

```
da51:~ # rcapache2 start
Starting httpd2 (prefork)                                               done
da51:~ #
```

If you want to stop the Apache web server, enter `rcapache2 stop`.

```
da51:~ # rcapache2 stop
Shutting down httpd2 (waiting for all children to terminate)            done
da51:~ #
```

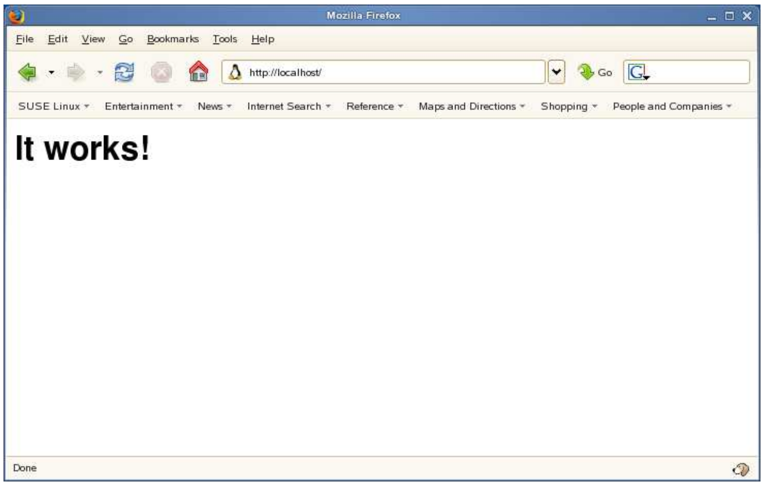
Apache doesn't start automatically when you reboot your computer. If you want the web server to start at boot time, enter `insserv apache2`.

```
da51:~ # insserv apache2
da51:~ #
```

To test the web server, open a web browser and enter the following address:

`http://localhost`

The browser displays the following page:


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/
Go
SUSE Linux Entertainment News Internet Search Reference Maps and Directions Shopping People and Companies
Done

It works!

If your SUSE Linux Enterprise Desktop 10 is connected to a network, you (and other hosts on the network) can remotely access the web server by entering the following:

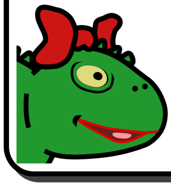
`http://`*`IP-address`*

If your network provides a DNS server, you can use the hostname instead of the IP address.

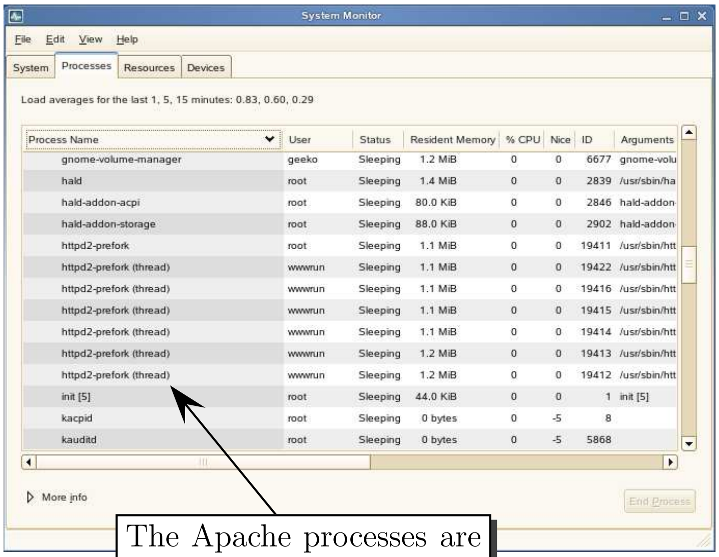
When Apache is running, the system starts some Apache processes, because each web server request needs its own Apache process. Handling more than one requests at the same time is called *pre-forking*.

If you start the GNOME System Monitor from the `System` group in the Application Browser, you can see some more interesting details.

GNOME System Monitor



To get the following window, where you can see all processes (and not only yours), select `View` → `All Processes`. Then, enable viewing threads by selecting `View` → `Threads`. In `Edit` → `Preferences`, activate `User` in the `Process Fields` list in the `Processes` tab.



Load averages for the last 1, 5, 15 minutes: 0.83, 0.60, 0.29

Process Name	User	Status	Resident Memory	% CPU	Nice	ID	Arguments
gnome-volume-manager	geeko	Sleeping	1.2 MiB	0	0	6677	gnome-volu
hald	root	Sleeping	1.4 MiB	0	0	2839	/usr/sbin/ha
hald-addon-acpi	root	Sleeping	80.0 KiB	0	0	2846	hald-addon-
hald-addon-storage	root	Sleeping	88.0 KiB	0	0	2902	hald-addon-
httpd2-prefork	root	Sleeping	1.1 MiB	0	0	19411	/usr/sbin/htt
httpd2-prefork (thread)	wwwrun	Sleeping	1.1 MiB	0	0	19422	/usr/sbin/htt
httpd2-prefork (thread)	wwwrun	Sleeping	1.1 MiB	0	0	19416	/usr/sbin/htt
httpd2-prefork (thread)	wwwrun	Sleeping	1.1 MiB	0	0	19415	/usr/sbin/htt
httpd2-prefork (thread)	wwwrun	Sleeping	1.1 MiB	0	0	19414	/usr/sbin/htt
httpd2-prefork (thread)	wwwrun	Sleeping	1.2 MiB	0	0	19413	/usr/sbin/htt
httpd2-prefork (thread)	wwwrun	Sleeping	1.2 MiB	0	0	19412	/usr/sbin/htt
init [5]	root	Sleeping	44.0 KiB	0	0	1	init [5]
kacpid	root	Sleeping	0 bytes	0	-5	8	
kauditd	root	Sleeping	0 bytes	0	-5	5868	

The Apache processes are called `httpd2-prefork`.

In the `User` column, you can see that only one process runs with root permission. This process started the other Apache processes running with the permission of user wwwrun. These processes will answer to HTTP requests.

Exercise: Start Apache

Start your web server and test whether you can access the web server of your neighbor.

3.2.3 Locate the Document Root of the Web Server

The default directory of the data provided by Apache is /srv/www/htdocs/. This directory is also called *DocumentRoot* of the web server.

After the installation of apache2-example-pages-2.2.0-21.2.i586.rpm, the directory /srv/www/htdocs/ contains some Apache example pages.

```
da51:~ # ls /srv/www/htdocs/
apache_pb.gif   apache_pb2.gif   apache_pb2_ani.gif   gif          robots.txt
apache_pb.png   apache_pb2.png   favicon.ico          index.html
da51:~ #
```

You can replace the data in the DocumentRoot directory to display your own web server content. As the web server runs with the permission of user wwwrun, ensure that this user can read the files in the DocumentRoot directory.

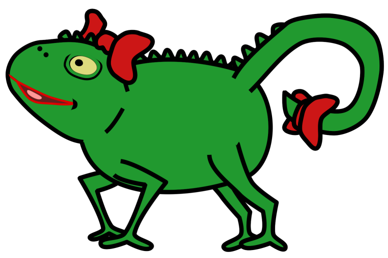
If you create subdirectories in DocumentRoot, you can access these subdirectories with the following web address scheme:

`http://`*`IP-address/subdirectory`*

If no specific file is requested in the address, Apache looks for a file named index.html. You can change this name in the Apache configuration files.

3.2.4 Locate the Apache Configuration Files

The Apache web server has several configuration files located in the directory /etc/apache2/.



The following is a list of the most important Apache configuration files:

httpd.conf. This is the main Apache configuration file.

default-server.conf. This file contains the basic web server setup. However, all options set in this file can be overwritten by other configuration files.

uid.conf. This configuration file sets the user and group ID for Apache. By default, Apache uses the user wwwrun and the group www.

listen.conf. In this configuration file, you can specify the IP addresses and TCP/IP ports Apache is listening to. By default, Apache listens to all assigned interfaces on port 80.

server-tuning.conf. You can use this configuration file to fine tune the performance of Apache. The default values should be fine, unless you need a web server that has to handle a lot of requests at the same time.

error.conf. In this file, you configure what to do if Apache cannot handle a request correctly.

ssl-global.conf. Configure the connection encryption with SSL in this configuration file.

In this manual, we only describe the file default-server.conf.

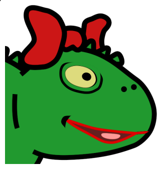
3.2.5 Understand the Basic Rules of the Configuration Files

The options of the Apache configuration files are called *directives*. Directives are case sensitive, e. g., “include” is not the same as “Include.”

Directives can be grouped so that they do not apply to the global server configuration. In the following, the directives only apply to the directory /srv/www/htdocs/:

```
<Directory "/srv/www/htdocs">
        Options None
        AllowOverride None
        Order allow,deny
        Allow from all
</Directory>
```

The directives are grouped by `<Directory "/srv/www/htdocs">` and `</Directory>` which limits their validity to the directory /srv/www/ htdocs/ only.


You can use the “#” character to write comments in the configuration file. All lines starting with a “#” are ignored by the Apache program.

Every time you edit the Apache configuration files, you have to reload the web server by entering `rcapache2 reload`.

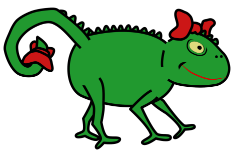
```
da51:~ # rcapache2 reload
Reload httpd2 (graceful restart)                                   done
da51:~ #
```

Sometimes, you have to stop and restart the web server by entering `rcapache2 restart`.

```
da51:~ # rcapache2 restart
Syntax OK
Shutting down httpd2 (waiting for all children to terminate)       done
Starting httpd2 (prefork)                                          done
da51:~ #
```

If you changed something in the configuration files and you want to verify whether the syntax is correct, enter `apache2ctl configtest`. If the syntax is correct, the command displays the following message:

```
da51:~ # apache2ctl configtest
Syntax OK
da51:~ #
```



3.2.6 Understand the Default Apache Configuration

The file /etc/apache2/default-server.conf is the main configuration file of the Apache web server. The follwing is an overview of the most important directives used in that file:

☞ `DocumentRoot`. Specifies the DocumentRoot of the web server.

For example:

```
DocumentRoot "/srv/www/htdocs"
```

☞ `<Directory "`*`directory`*`"> .../<Directory>`. All directives within this block apply only to the specified directory.

For example:

```
1 <Directory "/srv/www/htdocs">
2         Options None
          AllowOverride None
4         Order allow,deny
          Allow from all
6 </Directory>
```

☞ `Options`. With this directive, additional options can be applied to logical blocks, e. g., directories.

For example, in line 2 of the listing above there are no more options.

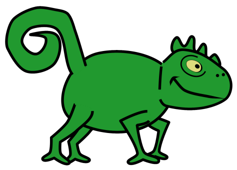
☞ `AllowOverride`. Determines whether directives are allowed to be overwritten by a configuration found in a `.htaccess` file in a directory.

For example, in line 3 of the listing above, it is not allowed to overwrite any directives.

☞ `Alias` *`fakename`* `"`*`realname`*`"`. Allows you to create an alias of a directory.

For example:

```
Alias /icons/ "/usr/share/apache2/icons/"
```



☞ `ScriptAlias`. Allows you to create an alias of a directory containing scripts for dynamic content generation.

For example:

```
ScriptAlias /cgi-bin/ "/srv/www/cgi-bin/"
```

In most cases, the default settings don't need to be changed.

An overview of all Apache directives can be found at `http://httpd.apache.org/docs-2.0/mod/directives.html`.

3.2.7 Limit Access to the Web Server

Normally Apache delivers data to all hosts in the network the web server is connected to. Sometimes it can be useful to limit access to the content delivered by Apache.

The following are the commonly used methods:

- Limit access using IP addresses
- Limit access using user authentication

Limit Access Using IP Addresses

If you want to use IP addresses to limit access to the web server, the following directives are available:

`Order`. This directive defines in which order in which the allow and deny directives are evaluated.

You have the following options:

- `deny,allow`. First, the deny directives are evaluated, then the allow directives. Access is allowed by default. Each client that does not match a deny directive or does match an allow directive, has access to the server.
- `allow,deny`. First, the allow directives are evaluated, then the deny directives. Access is denied by default. Each client that does not match an allow directive or does match a deny directive, does not have access to the server.

- ☞ `Mutual-failure`. Only those hosts listed in the allow list and not in the deny list have access. This has the same effect as `Order allow,deny`.

For example:

```
<Directory "/srv/www/htdocs">
        Options None
        AllowOverride None
        Order allow,deny
        Allow from all
</Directory>
```

`Allow from`. IP addresses or networks listed here are allowed to access the web server.

`Deny from`. IP addresses or networks listed here are not allowed to access the web server.

You can use the following options with the `Deny from` and the `Allow from` directives:

- ☞ `all`. This option applies to all hosts.
- ☞ A (partial) domain-name. This option applies to hosts that names match or end in the given expression (such as novell.com). Only complete domain components are matched. "novell.com" matches `www.novell.com`, but not `foonovell.com`.
- ☞ A full IP address. This option applies to a specific IP address (such as 10.0.0.23).
- ☞ A partial IP address. This option applies to IP addresses starting with the given IP address fragment (such as 10.0.0).
- ☞ A network/netmask pair. This option applies to IP addresses matching to the given network/netmask pair (such as 10.0.0.0/255.255.255.0 or 10.0.0.0/24).

All these directives have to be defined within a `<Directory>` block. They control the access to all data in that directory and all subdirectories.

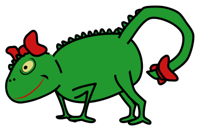
For example:

```
<Directory "/srv/www/htdocs">
Order deny,allow
Deny from all
Allow from 10.0.0.0/24
</Directory>
```

Explanation: In this example, only hosts from the network 10.0.0.0/24 have access to the data in the directory `/srv/www/htdocs/`.

Exercise: Limit Access Using IP Addresses

Ask for your neighbor's IP address. Then configure your Apache web server, that your neighbour's computer is denied access to your web server.



Limit Access Using User Authentication

Using IP addresses, you can control the hosts that accesses the web server, but not the users using these computers.

Apache offers another option of access control called *basic authentication*. If you protect content on your web server with this method, users are required to log in before they can access the data.

To use basic authentication, you first have to create user accounts for the web server. To do this, use the command `htpasswd2`.

The following command creates a password file and an account for the user geeko.

```
da51:~ # htpasswd2 -c /etc/apache2/htpasswd geeko
New password:
Re-type new password:
Adding password for user geeko
da51:~ #
```

Explanation: After entering this command, `htpasswd2` prompts you for a password for the new user. The passwords are stored in the file `/etc/apache2/htpasswd`.

You can specify a different location for the password file, but you have to make sure that it is readable for the user wwwrun and not located within the DocumentRoot of your server.

To use a password file for the first time, use `htpasswd2` with the `-c` option to create the file. If you want to add more users later, use the following command:

`htpasswd2 /etc/apache2/htpasswd` *`username`*

To delete a user from the password file, use the following command:

`htpasswd2 -D /etc/apache2/htpasswd` *`username`*

For example:

```
da51:~ # htpasswd2 -D /etc/apache2/htpasswd geeko
Deleting password for user geeko
da51:~ #
```

After you created user accounts, configure Apache to prompt for a password when accessing restricted data. Add the following lines to the restricted directory's block:

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /etc/apache2/htpasswd
Require user geeko
```

The following describes each line:

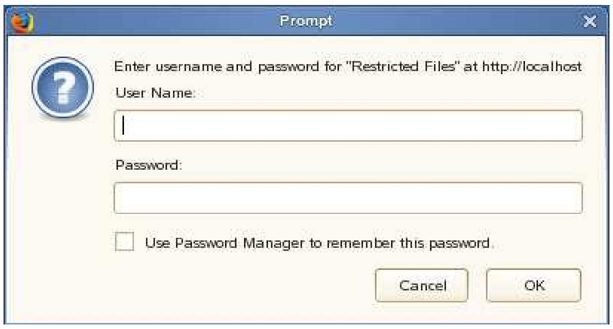
`AuthType Basic`. This directive defines the authentication method. For the type described in this section, the value is `Basic`.

`AuthName "Restricted Files"`. With this directive, you have to choose a label for the restricted directory of your web server. This name is used for the authentication process between browser and web server.

`AuthUserFile /etc/apache2/htpasswd`. This directive defines the password file used for the restricted directory.

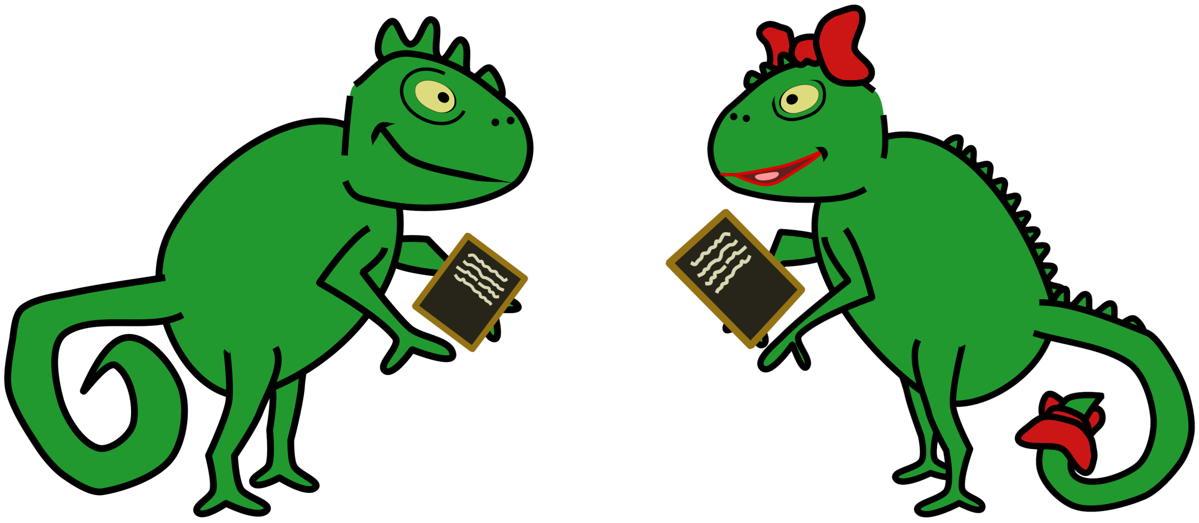
`Require user geeko`. This directive lists the user included in the password file who is allowed to access the directory. You can add more than one user by separating the user names with spaces. You can also use the directive `Require valid-user`. This defines that all valid combinations of user names and passwords are granted access.

If the user wants to access web pages in the protected (sub-)directory, he is prompted to enter his user name and password.


Prompt
Enter username and password for "Restricted Files" at http://localhost
User Name:
Password:
Use Password Manager to remember this password
Cancel
OK

Exercise: Limit Access Using User Authentication

In section 1.2.1 on page 20, you created a new user account. Configure your Apache web server so that this user gains access to your web server by using a user name and password.



Appendix

A The rpm Command

You can install software in the RPM format by using YaST or by using the command `rpm`. YaST ensures the automatic resolution of dependencies, while rpm only controls them. Resolution must be performed by hand.

Using the `rpm` command, you can install software (`rpm -i`), uninstall software (`rpm -e`), and query information from the RPM database (`rpm -q`).

The following are examples of using rpm:

- ☞ `rpm -qa`. Displays a list of all installed packages.
- ☞ `rpm -qi package`. Displays summary information about the specified package, for example:

```
da51:~ # rpm -qi bash
Name        : bash                         Relocations: (not relocatable)
Version     : 3.1                               Vendor: SUSE LINUX Products GmbH
, Nuernberg, Germany
Release     : 24.4                          Build Date: Fri Jun 16 09:14:10 2006
Install Date: Thu Oct 12 05:48:17 2006      Build Host: barbella.suse.de
Group       : System/Shells                 Source RPM: bash-3.1-24.4.src.rpm
Size        : 2627789                          License: GPL
Signature   : DSA/SHA1, Fri Jun 16 09:19:35 2006, Key ID a84edae89c800aca
Packager    : http://bugs.opensuse.org
URL         : http://www.gnu.org/software/bash/bash.html
Summary     : The GNU Bourne-Again Shell
Description :
Bash is an sh-compatible command interpreter that executes commands
read from standard input or from a file.  Bash incorporates useful
features from the Korn and C shells (ksh and csh).  Bash is intended to
be a conformant implementation of the IEEE Posix Shell and Tools
specification (IEEE Working Group 1003.2).



Authors:
--------
    Brian Fox <bfox@gnu.org>
    Chet Ramey <chet@ins.cwru.edu>
Distribution: SUSE Linux Enterprise 10 (i586)
da51:~ #
```

For information about a package that has not been installed, add the option `-p` and specify the complete path to the package in question. Entering `rpm -qpl/`*`software`*`/`*`package`*`.rpm` lists all the files in `package.rpm`, as in the following example:

```
da51:~ # rpm -qpl apache2-example-pages-2.2.0-21.2.i586.rpm
/srv/www/htdocs/apache_pb.gif
/srv/www/htdocs/apache_pb.png
/srv/www/htdocs/apache_pb2.gif
/srv/www/htdocs/apache_pb2.png
/srv/www/htdocs/apache_pb2_ani.gif
/srv/www/htdocs/favicon.ico
/srv/www/htdocs/index.html
/srv/www/htdocs/robots.txt
da51:~ #
```

You can also use the following options with the `rpm` command:

Option	**Description**
`--checksig`	Makes sure a package is complete and correct (PGP signature check).
`-e` or `--erase`	Uninstalls a package.
`-F` or `--freshen`	Updates a package only if the package is already installed in an older version.
`--help`	Lists options for the `rpm` command.
`-i` or `--install`	Installs a package.
`-i --force`	Installs a package even if `rpm` says that the package is already installed.
`-i --nodeps`	Installs a package regardless of unresolved dependencies.
`-qa`	Lists all installed packages.
`-qf`	Displays which package a file belongs to.
`-qi`	Gives brief information about the given package.
`-ql`	Lists all files that belong to a package.
`-qpl`	Lists all files of a package that are not installed.
`-U` or `--upgrade`	Updates a package. (In contrast to `-f`, the package is installed, even if there is no older version of the package installed.)
`-U --oldpackage`	Installs an older version, even if the newer version is already installed (downgrade).

Option	**Description**
`-V` or `--verify`	Compares information about the installed files in the package with information about the files taken from the original package and stored in the `rpm` database. Among other things, verifying compares the size, permissions, type, owner, and group of each file. Any discrepancies are displayed.

A complete summary of all options can be found in the manual page (`man rpm`).

Index