Hello World!

To get started with Linux administration, a Ubuntu base Image will be set up in Docker and we will have it print "Hello World!" If you are already working on a Linux System you can skip the Docker part.

Installing Ubuntu Base Docker Image

Open a console (PowerShell or CMD) an enter the following command `docker pull ubuntu`.

Docker will start pulling the Ubuntu image. After pulling the image you can view it with the command `docker image ls`. The Ubuntu image should show up in the resulting list.

```
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    597ce1600cf4   13 days ago   72.8MB
```

Start the Docker Image and Open a Shell in It

Use the command `docker run ubuntu` to start up the container. The command `docker ps -a` will list the running containers.

The container we just sterted should appear here.

```
Insert Container List here
```

Now we have a virtualized Ubuntu image running in Docker. The next step is to start a shell in this image to work with it. To start a shell in this image we use the command `docker exec -it <container name> /bin/bash`.

Now we are working with the shell we just started on our command line.

Execute the 'Hello World!'

To let the shell greet the world the command `echo` will be used. The string to be put out on the shell ('Hello World!'), is supplied as an argument to `echo`.

```
echo 'Hello World!'
```

The file system Outline

Like in Windows files in Linux/Unix are organized in a file tree. The first folder in this tree is called the root folder. While Windows has a separate tree for every device, devices get assigned a folder in the directory tree in Linux. This topic is discussed in chapter 2 of the Linux command line.

The root tree

The branches following the root are the following

- */bin*: contains binaries required to run the system
- */boot*: contains kernel, driver image for startup, boot loader
- */dev*: contains the devices connected to the PC (everything is a file)
- */etc*: contains system-wide configuration files
- */home*: contains the home directories of the users (normal users can only write here)
- */lib*: contains shared libraries
- */lost+found*: used in case of a file system corruption, for recovery purposes
- */media*: contains automatically mounted storage devices
- */mnt*: used for manually mounting storage devices
- */opt*: contains optional software, mainly commercial software
- */proc*: contains peepholes into the kernel
- */root*: home directory of the root user
- */sbin*: contains system binaries
- */tmp*: contains temporary files of programs (emptied on every start)
- */usr*: unix shared resources
    - */usr/bin*: contains programs installed by he distribution
    - */usr/lib*: shared libs for programs in */usr/bin*
    - */usr/local*: software not included in the distribution but installed systemwide
    - */usr/sbin*: contains system administrator programs
    - */usr/share*: contains shared date used by programs in */usr/bin*
    - */usr/share/doc*: contains system documentation
- */var*: contains data which is touched and changed frequently
    - */var/log*: contains logs produces by the system

For administrators the most frequently used directories are */etc*, */var* and */usr*# Navigating the File System This section will describe how to find your way through the file system.

Useful commands

- `cwd`: The command `cwd` will output the path to the directory you are currently in.
- `cd`: change your cwd (use .. to get into the directory above the one you are currently in).
    - Useful Shortcuts:
        - `cd -`: change to previous working directory
        - `cd`/`cd ~`: change to your home dir
- `ls`: To see which files are contained by a directory use the command `ls`. Use `ls -l` for more details.
- `file`: `file` will show you the type of a file.
- `less`: use `less` to get a scrollable view of the contents of a text based file.
    - Controls:
        - q: exit
        - /searchstring: replace searchstring with an actual string you are searching for and hit ENTER to search for it.
            - n: next result
        - g: jump to the beginning of the file
        - G: jump to the end of the file

Exercise

Explore the file system and get to know the navigation commands.

Search For Files

Multiple approaches for searching for files are offered. If you are searching for a file by its file name `locate` is the choice, `find` is used to search for files by its attributes, `grep` allows searching for files via their content.

The `locate` command

Locate performs a database search for paths matching the search term.

```
locate Pictures/Screenshot
```

The above command will search for all files and folders starting with `Screenshot-` laying under a location containing `Pictures`. It simply searches for the specified search term in the full path of a file. An exemplary result of the command can be seen below:

```
/home/user/Desktop/backup/Pictures/Screenshots
/home/user/Desktop/backup/Pictures/Screenshots/35574917.png
/home/user/Desktop/backup/Pictures/Screenshots/Screenshot_20210208_093220.p
ng
/home/user/Pictures/Screenshot_20210921_224523.png
```

Exercise: *Use locate to list all .bak files on the system.*

The `find` command

To search for files by file attributes the `find` command is used. Find requires only the path to search in as an argument. If you give your home-path `~` to it. It lists all files and folders located in your home directory, as we didn't specify what to search for.

```
find ~
```

To specify what to search for flags are used:

```
find ~ -name "Screenshot*"
```

The above command searches for all files and directories beginning with the word `Screenshot` the asterisk is used as a wildcard for a unknown number of unknown chars. Find offers a number of such search flags which help to narrow the search down further.

```
find ~ -name "Screenshot*" -type f
```

This command limits the search to results of the type **f**ile.

File Type	Description
b	Block special device file
c	Character special device file
d	Directory
f	Regular File
l	Symbolic Link

A full list of the flags available can be seen in **The Linux Command Line** in the subchapter *17 - find - Tests* (Table 17-3)

Exercise: *List all files under your home directory (~).*

Logical Operators

The `find` command allows combination of tests via logical operators to search for files more granular.

Operator	Description
-and/-a	True if the tests on both sides are true (Implied as default, when no operator used)
-or/-o	True if one test on the side is true
-not/!	True if the test following is false
( )	Groups operations, modifying the evaluation order

The example below searches for *png* and *pdf* (`-name "*.png" -or -name "*.pdf"`; `-type f`) files, in the home dir of the current user (`~`), owned by the user *dummy* (`-user dummy`)

```
find ~ \( -name "*.png" -or -name "*.pdf" \) -type f -user dummy
```

Exercise: *List all files owned by root or the current logged in user (find out username with `whoami`) in /home.#* Actions

Find can be used to execute actions on the files found.

Predefined Actions

Find offers predefined actions, which can be seen in by calling `find --help`

```
actions: -delete -print0 -printf FORMAT ...
```

The core actions are explained in **The Linux Command Line** in the subchapter *17 - find - Predefined Actions*

The example below will delete all screenshot files of the current user.

```
find ~ -name "Screenshot*" -type f -delete
```

The order of the tests and actions is important as by leaving the logical operator between the tests an `-and` is implied.

```
find ~ -name "Screenshot*" -and -type f -and -delete
```

Order of execution:

- 1. `-name "Screenshot*"` is executed first, without any dependencies
- 2. `-type f` is executed second, in case `-name "Screenshot*"` matched
- 3. `-delete` is executed third in case `-type f` also matched

As the print statement is connected with an `-or` in the example below, it will be executed independent from the statements before, printing all files.

```
find ~ -name "Screenshot*" -and -type f -or print
```

Exercise: *Delete all .bak files in the example directory tree.*

User Defined Actions

It is possible to execute user defined actions on results of the find command by adding the pattern below:

```
-exec command {} ;
```

The `{}` is the symbolic representation of the results pathname. The `;` serves as a delimiter for the command. Instead of `;`, `+` can be used. With the semicolon the action will get executed for every result in once. By using the plus as a delimiter, find will combine pathnames and execute them at once, making the command more efficient.

The above example of deleting screenshots would look like this with a user defined action:

```
find ~ -name "Screenshot*" -type f -exec rm '{}' '+'
```

Exercise: *Create a Bash script, which adds read privileges to all .log files in the exemplary folder structure for the current user. Tip: To add read privileges for the current user use* `chmod u+r <filename>`# Search for file content In this chapter a easy way to search for file content via `grep` will be explained.

`grep`

Grep is a tool to search files via regular expressions it. To see its full usage view its manpage `man grep` or go to the subchapter *3 - 19 - grep* in **The Linux command line**

Grep can be used to search multiple files for a pattern by using the `-R` flag.

```
grep -R "some_pattern" *
```

The asterisk at the end is used to search through all files and folders in the current directory.

Exercise 1: *Search the files in ~/example_tree for the pattern 'example_content'*

Exercise 2: *Use find in combination with grep to search all .txt files in ~/example_tree for the pattern 'example_content'*

Create Directories

This Topic is dealt with in Chapter **4 - mkdir** of The Linux Command Line.

Create Single Directory

To create a new directory use the command `mkdir`.

```
mkdir sampledir
```

Create Directory in Directory

The above example will create the directory sample dir in the current directory. To create a new folder in *sampledir* you can type:

```
mkdir sampledir/otherdir
```

Create Whole Tree

This command assumes sampledir is already existing. To create a whole directory tree if not existing use the Switch `-p`

```
mkdir -p notexisting/sampledir
```

The above example will create the folder *notexisting* and the directory *sampledir* within it.

Create Multiple Directories

To create multiple directories simply write them after each other like that:

```
mkdir sampledir othersampledir
```

Exercice

This exercise will combine the simple Operations listed above. Write a script, which creates the folders *sampledir/withinnerdir* and *singlesampledir*

Deleting Directories

This Topic is dealt with in Chapter **4 - rm** of The Linux Command Line.

Deleting Directories with `rmdir`

To delete a directory the command `rmdir` can be used.

```
rmdir sampledir
```

Delete Directories with `rm`

The `rm` command also can be used to delete directories with its `-r` switch. This will also delete all files and folders in the directory. More on the topic `rm` in the following chapter.

```
rm -r sampledir
```

Exercise

Write a script to create the Directory *sampledir* and delete it.

Deleting Files

This Topic is dealt with in Chapter **4 - rm** of The Linux Command Line.

The `rm` command

The `rm` command is used to delete files and folder on the Linux Command Line. It has 3 important operational flags:

Flag	Long	Usage
-i	--interactive	Prompt for confirmation before deleting
-r	--recursive	Recursively delete -> Delete subfolders and files
-f	--force	do not ask before deleting write protected files, ignore non existent files; **overrides** interactive

The flags are used in the example below:

```
# Deletes the file samplefile after asking for confirmation
rm -i samplefile

# Deletes the content of sampledir and it's content
rm -r sampledir

# Deletes the writeprotected file writeprotectedsample without asking for confirmation
rm -f writeprotectedsample
```

Exercice:

Use the script `020_testenv.sh` to create testfiles in your current folder and write a script to delete the folder *somedir* and its write protected contents without being asked for confirmation.

Copying Files and Directories

This Topic is dealt with in Chapter **4 - cp** of The Linux Command Line.

The `cp` command

The `cp` command is used to copy files and directories (`-r` switch has to be set) Its usage is `cp <src> <dest>`, multiple source files can be given like `cp <src1> <src2> ... <dest>` To copy file *a* to location *b* use:

```
cp a b
```

As a result the file *a* will be copied to file *b*

To copy a whole folder use the `-r` switch just like in the `rm` command.

```
cp -r somefolder someotherfolder
```

Exercise

Execute the *020_testenv.sh* script. Write a script to copy the file *somedir/writeprotected* to be next to *somedir* (same folder level) with the filename *writeprotected.bak*. And create a copy of *somedir* (*somedir.bak*)

Useful switches

- `-i` To avoid overwriting ask for confirmation
- `-u` Update, does not copy if destination file is same or newer

Moving Files and Directories

This Topic is dealt with in Chapter **4 - cp** of The Linux Command Line.

The `mv` command

The `mv` command is used to move or rename files and directories. Its Usage is very similar to the `cp` commands. It shares the useful switches mentioned in **Copying Files and Directories**. Its also possible to define multiple files and dirs to be copied `mv <src1> <src2> ... <dest>` To rename a file simply move it on the same 'level':

```
mv a b
```

This changes the filename from *a* to *b*

To move a file or directory to an other place give the new place as last parameter

```
mv a somedir/
```

This moves a to the subfolder *somedir*

Exercise

Execute the *020_testenv.sh* script. Make use of one of the useful switches and move the file *somedir/writeprotected* to *someotherdir*, but only if it is newer than the file *someotherdir/writeprotected*

Linking Basics

Linux offers two ways of linking files. *The Linux Command Line* treats this topic in *Chapter 4 - ln - Create Links*

Hardlinks

Hardlinks are the original way of linking files in the Unix world. In fact each file you see in a file explorer is a hardlink to a file, which gives the file its name. A file can have multiple hardlinks. A will get deleted/dereferenced when all hardlinks referencing it get deleted. Hardlinks can only point to files in the same filesystem.

Softlinks/Symbolic Links

Soflinks are the modern way of linking files in Unix Systems, they overcome the limits of hardlinks. Softlinks are special files containing text pointers to the file they are referencing.

Direct Comparison

The following code shows a directory with the file `test.txt`, a hardlink and a softlink to it. It can be seen, that the original file and the hardlink share the same inode number - they are quintessentially the same. The softlink has a different inode number - it is an independent file. The softlink additionally is pointing to the original test.txt file.

```
9181703 -rw-rw-r--. 2 dummy dummy 0 Dec 27 14:58 test.txt
9181703 -rw-rw-r--. 2 dummy dummy 0 Dec 27 14:58 hardlink
9181859 lrwxrwxrwx. 1 dummy dummy 8 Dec 27 14:59 softlink -> test.txt
```

Excursion: Inode Number

The inode number is an index in the so called inode table. In this table information about the file is stored, like its owner, size, permissions, or location. A files inode basically is its identity.# Creating Hardlinks

The command `ln` is used for creating links.

To create hardlinks use it like this:

```
ln <original-file> <link-location>
# For example
ln test.txt hardlink
```

Creating Softlinks

To create softlinks use the `ln` command with the `-s` flag:

```
ln -s <original-file> <link>
# For example
ln -s test.txt softlink
```

Exercise

Use the command `touch test` to create the empty file test in your current location. Write a script to create a hardlink called `test_hardlink` to the file test and create a softlink, ``test_softlink`to the hardlink. **Extra:** Use the command`ls -li test*` to view all three files and compare their inode numbers. # Read, and Use System Documentation

Most commands/programs used on Linux offer manpages. Manpages are extensive manuals/documentations of the programs. To access a Programs man page type:

```
man ls
```

The man page can be searched for patterns by typing `/` and the pattern you want to search for, hitting `ENTER` takes you to the first result. You can jump between the results by hitting `n` (next) and `p` (previous). To exit the man page hit `q`.

Exercise

- View the manpage for the command `ls` and list the files in the directory in *long format recursively* sorted *by time*.
- Lookup the usage of the commands `whereis` and `apropos` which will help you finding commands and exploring the system.

Introduction Input/Output Redirection

In this section input/output (I/O) redirection will be explained. The according chapter in the Linux command line is 1.6

Standard Input, Output, Error

Standard output (stdout) and standard error (stderr) follow the Unix philosophy of everything is a file. If a program on the command line needs to output its result it writes them to the *stdout* file, status messages like errors are written into the *stderr* file. Programs often grab their inputs from the standard input file (stdin), this is by default connected to the keyboard. Which files are used for standard input, output, error can be altered via Input/Output Redirection.# Output Redirection In this chapter the functionality of output redirection will be discussed

Ouput Redirection Overwriting

To redirect the output into a file the operator `>` is used. Using `>` will overwrite the contents of the destination file given. In the example the output of the command `ls -la` will be written into the file *ls-output*

```
ls -la > output
```

Now we use the command `cat` to view the content of the file. Cat takes a file as an argument and displays its contents to the command line. We won't see an output on the command line, but the file now looks something like this:

```
drwxr-xr-x 4 dummy dummy 4096 Jan 25 10:38 .
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:37 ..
drwxr-xr-x 8 dummy dummy 4096 Jan 25 10:54 .git
-rw-r--r-- 1 dummy dummy   17 Jan 25 10:38 .gitignore
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:42 01_basic_commands
-rw-r--r-- 1 dummy dummy 1070 Jan 25 10:37 LICENSE
-rw-r--r-- 1 dummy dummy 3605 Jan 25 10:38 README.md
```

Output Redirection Appending

By using the `>>` operator output will be appended to the specified file if we repeat the above example our file should look like this:

```
drwxr-xr-x 4 dummy dummy 4096 Jan 25 10:38 .
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:37 ..
drwxr-xr-x 8 dummy dummy 4096 Jan 25 10:54 .git
-rw-r--r-- 1 dummy dummy   17 Jan 25 10:38 .gitignore
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:42 01_basic_commands
-rw-r--r-- 1 dummy dummy 1070 Jan 25 10:37 LICENSE
-rw-r--r-- 1 dummy dummy 3605 Jan 25 10:38 README.md
drwxr-xr-x 4 dummy dummy 4096 Jan 25 10:38 .
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:37 ..
drwxr-xr-x 8 dummy dummy 4096 Jan 25 10:54 .git
-rw-r--r-- 1 dummy dummy   17 Jan 25 10:38 .gitignore
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:42 01_basic_commands
-rw-r--r-- 1 dummy dummy 1070 Jan 25 10:37 LICENSE
-rw-r--r-- 1 dummy dummy 3605 Jan 25 10:38 README.md
```

Error Redirection

The standard error can be redirected with `2>`/`2>>`. In the example it is tried to list the content of a restricted folder resulting in an error.

```
ls /root > output
```

Again we won't see an output, but our file will look like this:

```
ls: cannot open directory '/root': Permission denied
```

Redirect Both Outputs

To redirect bot outputs the `&>`/`&>>` operators can be used.

Exercise

Experiment with output redirection to see how it works. For example use `echo foo > output` to write foo to the file output and overwrite or append content.

Piping

Piping is a special variant of I/O redirection. It is used to use the output of one command as the input of an other.

To pipe the `|` operator is used.

```
ls | wc -w
```

The above example will give the output of `ls` to the command `wc -w`. `wc -w` counts the words in a given file. So by piping the output of `ls` into it sums up the count of words used as filenames in the current folder.

Filters

Filters take the content as input, alter it and then output it. Following are some common filters:

- `sort`: will sort input alphanumerically
- `uniq`: accepts sorted list and deleted duplicates
- `wc`: as seen above counts words in input
- `grep`: as seen in Chapter Search for Files can be used to filter for expressions

The above commands/filters offer more functionality than the basic one described, see their man pages for more details.

The `tee` Utility

`tee` reads the standard input and copies it to standard output and one or more specified files.

```
ls | tee output | grep output
```

The above example will write the outputs of `ls` to the file *output* and then search the output of `ls` (forwarded by `tee`) for the word output. If there is no file named *output* in the folder in which the command is executed it will have no output on the first run as the file *output* doesn't exist at first.

Exercise

Count the occurances of the word foo in the file *054_exercise.txt* by using the commands `grep` and `wc` as well as piping. If you are unsure about the usage of the commands look up their man pages.

Persmission Basics

The topic permissions is discussed in chapter 9 of The Linux Command Line. If you list the contents of a directory with `ls -l` you can see the owner, group and permissions set for a file.

```
-rwxr--r-- 1 dummy dummy   43 Jan 26 09:34 output
```

Ownerships

The first name that can be seen (dummy) is the *user* owning the file, the second one ist the *group* owning the file A user can be member of multiple groups, but can only have one primary group (in this case dummy). The group owning a file can be different from the owners primary group. You can view the current user and its infos with the command `id`. To see the groups you are part of use `groups`.

Permissions

The permissons are split into three flag triplets:

- *r*: being able to **read** the contents of the file
- *w*: being able to **write** to the file
- *x*: being able to **eXecute** the file

The first *rwx* triplet defines the permissions of the owner over the file, the second one the permissions of the group and the third one the permissions of other.

Sudo

In case you don't have the needed permissions to perform a operation you can use `sudo` - super user do. To use sudo your user has to be part of the group `sudoers` on Debain based machines and `wheel` on RedHat based machines.

Exercise

Use `id`to get some infos about your user.

Changing the Owner and Group of a File

In this chapter it will be explained how to change the owner and the group owning a file.

Changing the Owner of a file

To change the owner of a file the command `chown` is used:

```
chown dummy output
```

The above example will change the owner of the file *output* to the user *dummy*.

Changing the Owning Group of a File

To change the group owing a file, the commands `chgrp`

```
chgrp dummygrp output
```

```
chown dummy:dummygrp output
```

The above examples both will change the owning group of *output* to dummy group. By using `chgrp` you don't have to know the owner of the file to change the group.

Exercise

- Create a file *examplefile* with the command `touch examplefile` and change its owning group to *nobody*.
- Lookup the man pages of `chown` and `chgrp` to learn more about the commands

Changing Permissions of a File

To change the permissions of a file the command `chmod` is used.

```
chmod u+w output
```

In the above example the *u*ser (owner) of the file output gets granted write permissions on it.

Additive/Subtractive Model

As shown above the entity you want to grant/revoke permissions is selected via its initial:

- u: user/owner
- g: group
- o: other

- a: all

You can use the following operators to assign/revoke permissions

- +: add permission
- -: revoke permission
- =: set permissions

Some examples:

- `g-r+w` the same as `g-r,g+w` -> revokes read permission, grants write permission
- `ou=g` -> grants other and user same rights as group
- see the man page line 220 for more

Number-Based Model

Imagine each flag of a rights triplet gets assigned a number:

- r: 4
- w: 2
- x: 1

Now select the permissions you want to assign and add up their numbers. This way `rw-` would result in a *6*. Now do this for each entity and you get a number like 644 for `rw-r--r--`. This way permissions can be assigned with `chmod`:

```
chmod 0644 output
```

The leading zero can be left out, it is used to express an octal value (0644 (octal) -> 0b 110 100 100 (binary)).

Exercise

- Create the file `examplefile` and revoke all permissions of group and other.
- Read the man page if `chmod` to learn more about it.

Comparing File Content

This topic is based on chapter 20 - Comparing Text of the Linux Command Line. As learned in the previous chapter file content can be viewed using the commands `less` (scrollable) or `cat` (print whole file to terminal). The contents can be compared manually by viewing them with the above commands, or the command `diff` can be used. The tool gets called as follows:

```
diff file1 file2
```

The output will highlight the differences between the files:

```
3c3
< an exemplary line
---
> a difference
```

The differing line is displayed above.

Exercise

Compare the files *testfile1* und *testfile2*. In which line is the difference? What is the differing text?# Manipulating File Content Using Vim The command line editor Vim can be used to manipulate the content of a file. After opening the editor using `vim file` it is in command mode. The editor has multiple modes and commands:

- INSERT mode - Manipulate the file content - Press *i* in command mode to get there
- VISUAL mode - Select text - Press *v* to get there
  - press *y* to yank/copy the selected text to a vim register
  - press *p* to paste yanked text
  - press *x* or *Del* to cut text
- VISUAL LINE mode - select lines of text - Press *Shift-v*
- VISUAL BLOCK mode - select blocks of text - Press *Strg-v*
  - select a multiline one cursor wide text and press Shift-I get into 'multicursor mode'. After pressing ESC the changes you made to the first line of the selection will be applied to all lines

The most important commands are:

- :w - write the file
- :q - close the editor
- :q! - force close the editor
- :wq - write and close
- :w !sudo tee % - write to a witeprotected file (can also be done by opening vim with sudo)

Press ESC form any mode to get back to command mode

Vimdiff

Call vim with the flag `-d` and two files as arguments to get the differences between the files highlighted.

Exercise

Remove the difference between the files testfile1 and testfile2 in testfile2 using vim.# Regular Expression Basics

Regular Expressions can be used to define patterns matching specific parts of text. This part is based on chapter 19 of The Linux Command Line.

Wildcards

To match text either literal characters or wildcards can be used. Some exemplary wildcards:

- \w - matches any word character (\W everything else)
- \d - matches digits (\D everything else)
- \s - matches spaces (\S everything else)
- [a-z1-9#] - matches the characters a, b, c ... z, 1, ... 9 and #. Any characters can be defined, some will have to be escaped.
- [^a-z] - matches any character not in a-z
- . - matches any character
- ^ - matches beginning of a line
- $ - matches end of a line
- \n - newline

Quantifiers

Besides wildcards quantifiers are also used in regexes:

- * - any count of characters 0-x
- + - at least one character
- ? - zero or one character
- {3} - three characters
- {1,3} - one to three characters

Grouping

Character sequences can be grouped, this can be used to further quantify them or to use them in find and replace actions. To group sequences but them in brackets ().

Use | between two sequences or characters to search for the one OR the other.

Example

The following regex matches the pattern of an IP Address. The dot has to be escaped to serve as literal dot, not as wildcard.

```
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
```

The website regex101.com can be used to learn more about regexes and to experiment with them

Exercise

Go to regex101.com and write a regex to match the following date pattern "Monday 07.02.2022"[a-zA-Z]{6,9} \d{2}\.\d{2}\.\d{4}# Use Regexes to Analyze texts

Vim, Less, Man and Co.

In Vim, Less or Man interfaces type /, enter a regex pattern and hit enter to search for it.

Find and Replace in Vim

To use find and replace in vim enter the following pattern in command mode:

```
# Search and replace in current line
:s/pattern/newtext/g
# Search and replace in whole file
:%s/pattern/newtext/g
```

The `g` behind the last slash is a switch, the following other switches can be used:

- g - global, go on after first match
- i - case insensitive
- I - case sensitive
- c - confirm before replacing

Grep

As mentioned before in the Search Files section Grep can be used to search for file content. Use `grep` with the `-E` flag to search in extended regex mode. Grep uses a different notation for character classes:

- \d -> [[:digit:]]
- \s -> [[:space:]]
- \w -> [[:alnum:]]

Check the grep man page for more.

Some useful flags when searching with grep:

- -i - ignore case
- -v - invert match
- -c - count
- -l - list files with matches
- -L - list files without matches
- -n - print line number together with match

Exercise

Find all occurrences of a date matching the date pattern defined earlier in the file 102_exercise using `grep`.

Compressing Files

This topic is based on chapter 18 of the Linux Command Line. While under Windows `zip` is the default compression format on Linux `gzip` and `bzip2` are used.

The Tool Gzip

Usage of `gzip`:

```
# Compressing files
gzip foo
# Decompressing files
gzip -d foo.gz
## or
gunzip foo.gz
```

Bzip is used in the same way but uses an other compression algorithm internally.

Exercise

- read the man page of `gzip` to learn more about it
- look at the size of the file *111_exercise* using `ls -lh`, compress it using either `gzip` or `bzip2` and look at the new files size. Decompress the file again.# Archiving Files The default utility to archive, compress, unpack and decompress files on Linux is `tar`.

The Tool Tar

Tar allows the user to bundle many files into one. It has built in support for `gzip` and `bzip2`

Tar has the following modes with which it can be called:

- c - create an archive
- x - extract an archive
- r - append files to end of archive
- t - list contents of the archive
- d - compare archives
- u - update files in archive with older timestamp than supplied files

The modes can be combined with other options like:

- f - specify the archives name on command line
- z - use gzip with tar
- a - auto compression, choose compression program

```
tar -cf file.tar file
```

The Tar mode always has to lead before the other options.

Examples

Creating an compressed archive:

```
tar -czf archive.tar.gz file0 file1 file2 file3
```

Unpack compressed archive:

```
tar -xf archive.tar.gz
```

Exercise

- Read tars man page to learn more about it.
- Pack the file *111_exercise* into an archive, append the file *112_exercise*. Gzip the archive.

Literature Overview

Book	**Author**	**Description**	**Link**
Shell-Programmierung	Jürgen Wolf	How to create Shell-Scripts properly	https://openbook.rheinwerk-verlag.de/shell_programmierung/
Linux	Johannes Plötner, Steffen Wendzel	Intdroduction to Linux as a Daily Driver, Shell Basics Chapter 6 ff.	https://openbook.rheinwerk-verlag.de/linux/
Mastering Unix Shell Scripting	Randal K. Michael	Provides exemplary shell scripts for common tasks	https://ebookcentral.proquest.com/lib/mosbach-dhbw/detail.action?docID=152713&query=Shell+administration
Wicked Cool Shell Scripts : 101 Scripts for Linux, Mac OS X, and UNIX Systems	Dave Taylor and Dave Taylor	Provides exemplary shell scripts for common tasks	https://ebookcentral.proquest.com/lib/mosbach-dhbw/detail.action?docID=273515&query=Shell+administration
Linux All-In-One for Dummies	Emmett Dulaney	Book 5 is dedicated to Linux Administration Tasks, General Basics can be read in the other books	https://ebookcentral.proquest.com/lib/mosbach-dhbw/detail.action?docID=5431318&query=Shell+administration

On a normal PC the BIOS searches all attached media for a Master Boor Record (MBR). The MBR contains Information like the bootloader and the partition table. The partition table keeps track over the partitions of the attached media. The bootloader is responsible for loading an operating system.

During the boot process the hardware is booted and the OS kernel is loaded, the started kernel then starts the *init* process which in turn reads configuration files to start corresponding services. During startup *init* goes through multiple runlevels, starting the system stepwise:

0. Halt - closing network connections, writing file buffers, unmounting files
1. Minimal System - single user, only basic system resources loaded, maintenance mode
2. Local Multiuser - no network
3. Full Multiuser - on console, network operational
4. Undefined - Can be defined by user
5. Full Multiuser GUI - full multiuser mode with graphical user interface
6. Reboot

Most systems either boot to level 3 or 5, depending on the requirement for a graphical use interface. When shutting down the runlevels are taken down till level 0 is reached.

Systemd

Modern systems use systemd rather than initd as an init deamon. Systemd uses unit dependencies to determine if a service needs an other service to run successfully before it is started. This ensures the integrity of the startup.

Exercise

Read the `systemd` manpage if you want to learn more. The topic services will be dealt with in a later chapter.# Shutting Down Systems

Linux systems can be safely shutdown with the command `shutdown` it offers a variety of flags to customize the shutdown the mod important are:

- -c - cancel a planned shutdown
- -H - the system will be halted after the shutdown
- -P - the system will be powered off after the shutdown
- -r - the system will be rebooted after shutdown

A new shutdown is planned by typing:

```
shutdown -h 10
```

The above example will schedule a shutdown of the system 10 minutes form now. To shutdown a system instantly the time `now` can be specified.

Some Frontends

The commands `halt` and `poweroff` are frontends for the command shutdown. They use `shutdown` to execute their purpose. Halt should leave the system powered on after the shutdown. Poweroff should power off the machine.

Exercise

- Read the `shutdown man page to learn more about it.

- Write a script to instantly shut down and halt your system.

Rebooting Systems

To reboot a system the commands `reboot` or `shutdown` with an additional `-r` flag can be used. Reboot will call `shutdown` to reboot the machine# Different Operating Modes of a Linux System A Linux System can be run in **Text Console Mode** and **GUI Mode** if a GUI is installed. For the pure Terminal Mode the virtual tty (teletype) terminals are used. Linux systems have 7 virtual terminals: tty1 to tty7. When booting into a system with a GUI installed some ttys are used to display the GUI, on Ubuntu tty1 shows the display manager is shown, tty2 shows the GUI for the user fist logged in. To change into terminal mode the remaining ttys can be accessed by pressing `CTRL + ALT + F[3-7]`# Boot Loader Basics This chapter is based on Chapter 27.4 of Linux - Rheinwerk and the Ubuntu Documentation

The boot loader is placed in the master boot record of a hard drive. Typically a boot loader offers multiple selectable operating systems to start. The default boot loader on modern Linux Systems is GRUB2 (GRand Unified Boot Loader2).

There are other formerly common legacy boot loaders like the Linux Loader (LiLo) or GRUB1 which are not used anymore.# Installing GRUB2 A boot loader will be installed per default when installing a new Linux system.

It can be necessary to reinstall a boot loader for troubleshooting reasons.

Chroot environment

To be able to access a system with a broken boot loader chrooting can be used. Chrooting enables to change the root directory a Linux system is running from to the root directory of an other system. The chrooting can be done from a Live system:

1. Boot the live system
2. Mount the original root partition (i.e. `sudo mount /dev/sd.. /mnt`)
3. Additionally necessary system folders need to be bound
   1. `sudo mount --bind /dev /mnt/dev`
   2. `sudo mount --bind /dev/pts /mnt/dev/pts`
   3. `sudo mount --bind /sys /mnt/sys`
   4. `sudo mount --bind /proc /mnt/proc`
   5. `sudo mount --bind /run /mnt/run`
4. Change into the origional system: `sudo chroot /mnt /bin/bash`

For more information see the Ubuntu Documentation

Simple Reinstallation

To reinstall grub for example in case of a damaged MBR do the following:

```
sudo grub-install /dev/sda # where to place the boot loader
```

By doing this the configuration files */etc/default/grub* and */etc/grub.d/** are untouched.

Complete Reinstallation

To do a complete reinstall with new configuration, etc. do the following. Before the actual installation it has to be ensured that the grub packages are installed, on Ubuntu the packages

- grub-common
- grub-pc/grub-efi
- os-prober (for detecting other operating systems on pc)

are needed. They can be installed with the `apt` utility.

To ensure a clean new installation the existing configurations can be purged with:

```
sudo apt-get purge grub-* os-prober grub-gfxpayload-lists
```

To install teh new boot loader, first install the packages listed above:

For Bios PCs:

```
sudo apt-get install grub-pc os-prober grub-gfxpayload-lists
```

For EFI PCs:

```
sudo apt update
sudo apt-get install grub-efi-amd64-signed os-prober #  grub-efi-amd64 or
grub-efi-ia32 (for 32Bit systems) also possible
```

In the masks following the installation destination of grub has to be determined.

Configuring GRUB2

There are two places GRUB2 can be configured form:

- */etc/default/grub* - Actual configuration file
- */etc/grub.d/* - Folder containing GRUB configuration files.

After altering the configuration in one of this places the program `grub-makeconfig` has to be run, to build the file */boot/grub/grub.cfg*. The program has a wrapper, `grub-update`, which calls `grub-mkconfig -o /boot/grub/grub.cfg`.

Some common configuration options are:

- *GRUB_DEFAULT=0* - default entry grub starts after a timeout

- *GRUB_TIMEOUT=0* - timeout in seconds before the default entry is started# Process Properties This chapter is based of the chapter 26 of Linux - Rheinwerk.

Basics

Processes are programs being executed. When a program is started its process in entered into the process control block of the kernel. This data structure keeps information about the process like its Process ID (PID) or the User ID (UID) of the user that started it.

Types of Processes

There are different types of processes:

- **Background Processes**: Started from the shell by putting a `&` behind a program call. The `&` sends the process to the background. Background processes can be listed with `jobs` (`jobs -l` for more details) and brought to the foreground with `fg %<job-id>`. `Ctrl-Z` stops a running job and keeps it listed in the jobs table, the stopped job can be continued by typing `bg %<JID>` for background or `fg %<JID>` for continuing it in the foreground.
- **Daemons**: Daemons work in the Background too, they are used to fulfill tasks, that need no direct control, i.e. an Email- or Webserver. Daemons are normally started during the boot process.

Properties of Tasks

Using the tool `top` a taskmanager-like interface will appear in your bash.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
COMMAND
  297 friedri+  20   0   10044   5068   3360 S   0.0   0.0   0:00.18 bash
```

In the above example an exemplary entry taken from the top command can be seen. Top shows the main properties of an process:

- **PID**: The Process ID of the shown process
- **USER**: The User who started the process
- **PR**: Priority of the Process, Processes with higher priority get more CPU time.
- **NI**: Nice value of the process, affects priority of the process
- **VIRT**: Total amount of virtual memory consumed by the process (memory in RAM and memory swapped to disk)
- **RES**: Actual RAM memory usage
- **SHR**: Amount of memory chared with other processes
- **%MEM**: Memory used in relation to total RAM memory

Exercise

Start a background process with the command `sleep 300`, list the background jobs and bring the created job to the foreground.

Manage Processes

Processes can be controlled by signals sent via the terminal.

List Processes

To get an interactive view over the current processes `top` can be used. The tool `ps` gives a static list of the processes of the current session. To see every process on the system the flag `-e` can be used: `ps -e` The tool `pstree` shows processes and their parents in a tree-like view. The tool `pidof` can be used to find PIDs of a program if it has been started as a process.

Send signals to processes

The tool `kill` is used to send signals to processes, some useful signals are listed below:

- **Kernel Signals** - The receiver of this signals is the kernel
  - 9 - SIGKILL/KILL - The process is killed by the kernel
  - 19 - SIGSTOP/STOP - The process will be stopped/paused
  - 18 - SIGCONT/CONT - A stopped/paused process will continued
- **Interceptable Signals** - This signals can be intercepted by the process
  - 1 - SIGHUP/HUP - The process is ordered to halt and restart
  - 14 - SIGALRM/ALARM - Notifies about an alarm set with the syscall alarm()
  - 15 - SIGTERM/TERM - Notifies the program to gracefully terminate itself

Signals can be sent like the following:

```
kill -<Signal ID> <PID>
kill -<Signal Name> <PID>
kill <PID> # Kill alone sent SIGTERM -> kill -SIGTERM <PID>
```

To let the process 500 be killed by the kernel we would send `kill -9 500`.

Exercise

Create a process by sending the program `sleep 300` to the background. Kill the PID displayed when sending it to the background via the kernel.# Logfiles This chapter is based on chapter 14.4 of Linux - Rheinwerk and the Ubuntu Documentation Per default the system logs are logged to the directory */var/log* on Linux systems. The folder contains logs created by the system and installed services. Some of the most important logs are listed below:

- **auth.log** *(Debian)* or **secure** *(RedHat)* - Log of login attempts
- **dpkg** and **apt/history.log** - Log of the package management in Ubuntu
- **dmsg** and **kern.log** - Kernel messages (*dmsg* shows the last while *kern.log* shows older messages too)
- **daemon.log** - Installed services may create their own log files or log into *daemon.log*
- **syslog** *(Debian)* or **messages** *(RedHat)* - Global system activities are logged here
- **cron** - Log of scheduled jobs (more on this topic later)

View Log Files

Most log files can be viewed with the standard text viewer tools like:

- `cat` - print whole file to terminal
- `less` - Scroll through a file - `Shift-G` -> End of file; `gg` -> Start of file; arrow keys or mouse wheel to move cursor; `/` -> Search for Pattern
- `tail` - View only last few lines of a file

The tool `watch` can be used to execute a program periodically, thus by calling `tail <logfile>` with watch the view will be updated as new logs come in. A similar behavior can be achieved with the command `less -f` which will try to keep scrolling when reaching the end of a file. New logs will be appended to an existing log file, so if you are looking for the freshest logs to investigate something they will be at or near the end of a log file.

Journal

On newer systems *syslog* is replaced by *journal*. The syslogs can be viewed with the command `sudo journalctl` rather than via the actual log file. The journal will be displayed with `less`.

```
Mar 08 16:46:46 server sudo[843154]: pam_unix(sudo:session): session closed
for user root
Mar 08 16:46:46 server sudo[843154]:  user : TTY=pts/8 ; PWD=/var/log ;
USER=root ; COMMAND=/usr/bin/journalctl -f
Mar 08 16:46:46 server sudo[843154]: pam_unix(sudo:session): session opened
for user root by (uid=0)
```

Above the journal log entry for opening the journal with sudo can be seen. Leading is the timestamp, followed by the hostname, then the process name with pid and the processes log message as a tail.

Exercise

Execute `echo Find Me | systemd-cat -p warning`, `systemd-cat` can be used to send log messages to the journal. Search the journal for the message you just sent.

Schedule Tasks

This chapter is based on chapter 14.3 of Linux- Rheinwerk.

Cron

To schedule tasks on Linux the utility cron is used. Cron jobs are usually scripts or commands, which get executed by the cron daemon. The jobs can be defined in various locations:

- */etc/crontab*: Global crontab for the administrator
- */var/spool/cron/crontab*: Crontabs of local users named after the users
- */etc/cron.[daily|weekly|monthly]*: Scripts can be placed here and will be executed daily, hourly or monthly

- */etc/cron.d*: Crontab files can be placed here to be executed

The Crontab Format

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
  17 *  *  *  * root    cd / && run-parts --report /etc/cron.hourly
```

In the above example it can be seen how cronjobs are timed. There are 5 fields for defining when to run the task

- Minute
- Hour
- Day of Month
- Month
- Day of Week

An asterisk `*` means, that the task will be run every unit of time (minute, hour, day, ...). A task scheduled with `17 5 * * *` will run every month, every day, on hour *5* and minute *17*. The time frame is followed by the user which the task belongs to. The entry is ended with the actual command to be run.

The Crontab Tool

The crontab file is not to be edited manually, instead the tool `crontab` is used. The following flags are useful when working with the tool:

- `-u <username>`: Edit the crontab of the specified user
- `-l`: Print the content of the corresponding crontab
- `-r`: Delete the crontab file
- `-e`: Edit the crontab file

The `at` command

The `at` command can be used to schedule a task to run once. At can be given a task with the following syntax:

```
at <time to run the task at> -f <script to run>
```

By calling `at` without the `-f` flag a task to run can be defined in an interactive mode, which can be exited with `Crtl-D.`

The time can be defined in the following patterns

- `at hh:MM`
- `at <time> + <time frame>` e.g. + 2 hours
- `at hh:mm dd.MM`

Exercise

Create a crontab entry which will write 'Coming from Crontab' into the journal (see chapter before) every 5 minutes.Run `crontab -e` enter

```
15  *  *  *  * echo 'Coming from Crontab' | systemd-cat
```

save and close the editor. For user crontabs no username is needed.# Check if a Task was Executed The cron daemon logs the execution of tasks to syslog. As a consequence the execution of the tasks can be seen in the system journal (systemd logs).

The journal can be viewed with the following command:

```
sudo journalctl
# To get to the end of the file press Shift+G
```

An example taken from here below:

~~~
 Nov 02 17:10:01 testbox CRON[2210]: (testuser) WRONG INODE INFO
(crontabs/testuser)
```# Updating Software

This chapter is based on the chapter 14.2 of Linux - Rheinwerk Publishing.

## Package Management

The default package manager on Debian based distributions like Ubuntu is
`apt`.
Apt can be used to install, update and remove software packages.
Besides installing the actual software package the package manager also
installs packages which the actual package needs as dependencies.

### Software packages

Software packages contain the binaries used to execute said software,
documentation like man pages, HTML documentation, configuration files and
in some cases scripts to start/quit the software.
Per default these packages are downloaded from remote repositories by `apt`
but these packages can also be downloaded from other sources, like for
example the website of a specific software.
~~~

~~~
### Repositories

When installing a not local package with `apt` it gets the packages from
remote repositories listed in the file */etc/apt/sources.list*.
The command `apt update` indexes the packages and versions located in the
repositories listed in */etc/apt/sources.list*.

### APT Operations

The apt default operations are:

- install: Install a new package.
- update: Update the package index.
- upgrade: Upgrade packages but do not install new dependencies or remove
old ones.
- dist-upgrade: Upgrade with resolved dependencies.
- remove: Remove an installed package and keep its data.
- purge: Remove an installed package together with its data, like
configuration files

### Examples

As `apt update` updates the index od packages and their versions located in
the remote repositories, it has to be run before each operation.
To install the `vim` editor run:

``` bash
sudo apt update
sudo apt install vim
~~~

To upgrade a specific package if installed:

```
sudo apt update
sudo apt install --only-upgrade vim
```

To update all packages on the system:

```
sudo apt update
sudo apt upgrade
```

If a software vendors offer their software directly from their website. On Debian based systems, *deb* packages need to be downloaded these can be installed by the following commands:

```
sudo apt update
sudo apt install ./Downloads/your_downloaded_package.deb
```

Exercice

- Read the man page of apt/apt-get if you want to learn more.
- Install the web server nginx and remove it completely using `apt`. # Check Resources

To check on available system resources the tool `top` can be used. Below is an exemplary output of `top`:

```
top - 15:34:28 up  1:26,  3 users,  load average: 0.50, 0.54, 0.61
Tasks: 398 total,   1 running, 397 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  0.6 sy,  0.0 ni, 98.4 id,  0.0 wa,  0.0 hi,  0.0 si,
0.0 st
MiB Mem :  15413.3 total,   9340.8 free,   3648.2 used,   2424.3 buff/cache
MiB Swap:  26913.0 total,  26913.0 free,      0.0 used.  11246.0 avail Mem
```

At the end of the top line the average load of the CPU is displayed in the last minute, the last five minutes and the last fifteen minutes. In the next line the CPU loads on each core are shown in percent, followed by the usage of the physical memory, followed by the usage of the usage of the swap memory.

The tool `htop` offers a human readable way of displaying the same statistics, it can be installed from the package repositories.

Exercise

Install the tool `htop` with `apt install` and display the available resources.# Check on Processes

The tool `htop` can also be used to view running processes and their resource usage, press `F4` to filter for a process you are searching.

Exercise

Open `htop` and search for the process *systemd*.

Changing Kernel Runtime Parameters

This chapter is based of the RedHat documentation

Kernel Runtime Parameters

Kernel Runtime Parameters can be changed at runtime, they can be addressed via the command `sysctl` and the configuration files and folders */etc/sysctl.confl/etc/sysctl.d/*.

The configurable variables are divided into classes, on Ubuntu systems the following Classes are present:

- abi - Execution domains and personalities
- debug - Kernel debugging interfaces
- dev - Device-specific information
- fs - Global and specific file system tunables
- kernel - Global kernel tunables

- net - Network tunables
- user - User Namespace limits
- vm - Tuning and management of memory, buffers, and cache
- *List taken from RedHat documentation*

A full list of variables can be viewed with the `sysctl` command and the `-a` flag: `sysctl -a`.

Configure Parameters Temporally

To alter the value of a kernel runtime parameter temporally use sysctl like the following:

```
sysctl <class>.<parameter>=<value>
```

Configure Parameters Permanently

To alter a runtime parameter permanently redirect the output to of the above command to be appended to the configuration file */etc/sysctl.conf*:

```
sysctl <class>.<parameter>=<value> >> /etc/sysctl.conf
```

Alternatively the configuration files in */etc/sysctl.d/* can be modified manually. Either create a new file or modify an existing one in */etc/sysctl.d/*, the files have to contain one value assignment per line:

```
<class0>.<parameter0>=<value0>
<class1>.<parameter1>=<value1>
...
<classN>.<parameterN>=<valueN>
```

Example

The hostname of your computer can be changed with the variable `kernel.hostname`. A documentation of runtime parameters can be found here

Exercice

- Find out the current hostname of your host.
- Change the hostname to 'dhbw-testhost'

Shell Scripts

This chapter is based on the chapter 11 of Linux - Rheinwerk Verlag. Shell scripts make it possible to write down complex tasks into scripts. These scripts can be used to automate management and orchestration.

Hello World!

Shell scripts typically start with a shebang or hashbang:

```
#!/bin/bash
...
```

The shebang tells the calling shell which binary to execute the script with. A script can contain all commands that can be executed by the shell by default. An exemplary script could look like this:

```
#!/bin/bash

echo HelloWorld!
```

Comments

Comments can be started with a `#`:

```
# this is a comment
echo foo # this is a comment too
```

Exercise

Write a script which echoes your name and put in a comment describing what it does.

Variables

Variables are created and set by defining its name and assigning it a value: Note that no spaces are allowed before and after the equal sign.

```
variablename="variablevalue"
```

Variables types can be:

- Strings - as shown above
- Characters - 'c'
- Numbers - 42

To access the value of a set variable put the `$` sign in front of it:

```
echo $variablename
```

The result of a function call can be assigned to a variable by putting the call in braces preceded by a `$` sign:

```
var=$(echo foo)
echo $var
```

Arrays

Arrays can be created by putting values in braces:

```
int_array = (1 2 3 4)
```

Specific values can be set by referencing them by their index:

```
int_array[1] = 10
```

The values can be accessed by putting the variable and the whished index into curly braces:

```
echo ${int_array[1]} # will output 10
```

Exercise

Write a script, which saves the count of files in the current directory into the variable `fileCount` and print the variable to the terminal.# Conditionals Conditionals can be used to execute tasks if a condition is met. A conditional block is started with `if` and closed with `fi`.

```
var=5
if [ var = 5 ]
then
    echo var is 5
else
    echo var is not 5
fi
```

An overview of bash test and conditional operators, which can be used for conditionals can be found on this cheatsheet and on the bash man page.

Examples for other binary operators that can be used in comparisons are:

- `-eq` - equal
- `-ne` - not equal
- `-le` - less or equal

- `-lt` - less than
- `-ge` - greater or equal
- `-gt` - greater than

Exercise

Write a script, which checks the count of files in the folder it is executed and echoes 'Too much files!' if there are 5 or more files in the folder.# Loops

for

The for loop in bash can be used to iterate over arrays:

```
for var in (1 2 3 4 5)
do
    echo var
done
```

With for it is also possible to iterate over the files in directory:

```
# This will print the filenames of all files in the current folder
for file in *.txt
    echo $datei
done
```

while

The while loop can be used to execute code while a condition is satisfied:

```
var=5
while [ var != 0 ]
do
    echo var
    var=var-1
done
# will print 5 to 1 to the terminal
```

An endless while loop can be created by taking `1` (true) as a condition or by replacing the condition with `:`, see below:

```
while [ 1 ]
    echo foo
done
## is the same as
while :
    echo foo
done
```

until

Until loops till a value isn't satisfied anymore. It can be compared to a negative while loop `while [ !0 ]` is the same as `until[ 0 ]`.

Exercise

To put the learned together, write a script, which endlessly loops and echoes 'Too much files in here!' in case there are 5 or more files in the directory it is executed in.# Manage Startup Services The automatic start of services on system startup can be managed with the tool `systemctl`.

Systemctl allows to start, stop, check, enable and disable services. An enabled service will start on startup, a disabled one will not. Systemctl is used like this:

```
systemctl <operation> <service>
```

With some of the most common operations being the following:

- start
- stop
- status - *for checking the status of the service*
- enable
- disable

All operations except the `status` will require root privileges.

Checking a Service

Checking the status of a service will bring up the following view:

```
crond.service - Command Scheduler
     Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor
preset: enabled)
     Active: active (running) since Thu 2022-04-07 12:31:44 CEST; 1h 20min
ago
   Main PID: 1399 (crond)
      Tasks: 1 (limit: 18355)
     Memory: 1.4M
        CPU: 94ms
     CGroup: /system.slice/crond.service
             └─1399 /usr/sbin/crond -n

Apr 07 12:31:44 fedora crond[1399]: (CRON) INFO (running with inotify
support)
```

The second value (`enabled`) in the loaded field indicates whether the service is enabled to start on startup. Below the services statistics its log entries are displayed.

Example

To disable the cron daemon, which is responsible for executing the following command:

```
sudo systemctl disable cron
```

Exercise

- Read the `systemctl` manpage to learn more.
- Restart the cron service with `systemctl` and check its status to ensure it restarted properly. *Info:* Using this feature can be useful to restart a service after altering its configuration.# Configuring Services Resources for this section are the Suse Documentation, a Medium Article, a StackOverflow Contribution and a Python Documentation.

Following the everything is a file directive of Linux/Unix services are defined in files. These files are located in */etc/systemd/system*.

Example Service

To demonstrate the configuration of systemd services in this section a python server service will be created.

Python Server

Python offers a online way of starting a webserver, by loading the module *http.server*:

```
python -m http.server
```

To display something create a folder */home/your-user/www* and create a file *index.html* with the following content in it.

```
<h1>I am a Simple Python Server!</h1>
```

The Service File

Lets create a simple web server service file:

```
[Unit]
Description=Simple Python Service
After=network.target
[Service]
```

```
WorkingDirectory=/home/liam/Documents/www
ExecStart=/usr/bin/env python3 -m http.server

[Install]
WantedBy=multi-user.target
```

In the first section *[Default]* the service is described (its name is based on the service filename). Systemd maintains an order when starting services, to start services other services depend on before those depending on them. This dependencies are defined via the *After* field, which will let the server service start after the *network.target* service.

In the next section *[Service]* details for executing the service are defined. *WorkingDirectory* sets the directory the service will be executed in. *ExecStart* defines the actual executable executed.

In the field *WantedBy* in the *[Install]* section defines the system state systemd tries to start the service in (see 011_booting_systems). The value `multi-user.target` closely matches the system state 3 (Full Multiuser - on console, network operational).# SELinux This chapter is based on the Fedora documentation [1], [2], [3], [4]

Mandatory Access Control

SELinux acts as an additional security layer (Mandatory Access Control (MAC)), added on top of the standard access policy (users, groups, permissions) called Discretionary Access Control (DAC), with DAC being evaluated before SELinux rules. It allows a more granular way of defining access controls. In SELinux every process and resource has an label called SELinux context.

These contexts are used to define how processes are allowed to interact with each other and their environment. SELinux provides a whitelist based model, meaning things have to be explicitly allowed and are blocked by default.

SELinux is the default MAC tool for **RedHat** based systems.

SELinux Contexts

SELinux Contexts have the following fields:

- user
- role
- type
- security level

The SELinux context type is the important field when concerned with interactions between processes.

Examples

Best practice for context types is for them to end on `_t` Examples of SELinux types are:

- httpd_t - webserver context
- tmp_t - context for files in */tmp*

As typically the webserver Apache is configured to be allowed access to files tagged with `httpd_t` it can access contents necessary for serving a web page or service. Per default Apache has no access rights to the `tmp_t` type preventing it from access there. For more examples see the Fedora documentation.

View File Contexts

To view contexts assigned to files the command `ls` can be used with the flag `-Z` producing the following exemplary output:

```
unconfined_u:object_r:user_home_t:s0 01_basic_commands
```

Change File Contexts

File contexts can be altered with the command `chcon` is used:

```
chcon -t <new_context_type> <file>
```

When used with the `-t` switch, like above, the context type of a file is changed. The `-R` flag can be used to alter the contexts of a directory and its contents recursively.

View Process Contexts

To view the context of a process `ps` can be used with the flags `-eZ`:

```
system_u:system_r:crond_t:s0-s0:c0.c1023 1302 ?  00:00:00 crond
```

This reveals that the process crond has the context type crond_t giving it access to cron relevant files.

Exercise

Read the linked documentations if you want to learn more.

Apparmor

Apparmor is an other MAC tool, it is commonly used on **Debain** based systems. This section is based on the SUSE Documentation

AppArmor Profiles

AppArmor policies are defined in sol called profiles. The profiles are typically found in */etc/apparmor.d* An exemplary AppArmor Profile can be seen below:

```
## App Armor Profiles offer the possibility to include extra files.
## In this case a file containing variable definitions is included.
#include <tunables/global>

## naming the application to confine and starting the block defining it
/usr/bin/foo {
   ## Inclusion of an Abstraction
   ## Abstractions offer predefined templates for different use cases.
   #include <abstractions/base>

   ## Capability definition:
   capability setgid,
   ## Definition of allowed network access:
   network inet tcp,

   ## Allowing linking from /etc/sysconfig/foo to /etc/foo.conf:
   link /etc/sysconfig/foo -> /etc/foo.conf,

   ## Block of path access definitions:
   ## Unconfined execution mode:
   /bin/mount            ux,
   ## Use {} to define multiple possible options - in this case either an
empty string or 'u':
   /dev/{,u}random     r,
   ## Use * as a wildcard for any combination of characters :
   /etc/foo/*            r,
   /lib/ld-*.so*         mr,
   ## Use [] for defining ranges:
   /proc/[0-9]**         r,
   ## Usage of a variable:
   /@{HOME}/.foo_file   rw,
   ## Owner conditional -> applying this rule only to the owner of the
file.
   ## 'other' can be used to apply the rule to everybody but the user.
   owner /shared/foo/** rw,
   ## Execute local profile for this file
   ## See below for local profile definition
   /usr/bin/foobar       Cx,
   ## Use the profile bin_generic for executing this role
   /bin/**               Px -> bin_generic

   ## Definition of local profile
   profile /usr/bin/foobar {
      /bin/bash          rmix,
      /bin/cat           rmix,
      /bin/more          rmix,
      /var/log/foobar*   rwl,
      /etc/foobar        r,
   }
}
```

Taken from here.

The following access controls can be used:

- r - read mode - a program can read a resource
- w - write mode (cannot be specified together with a) - a program has full write access to a resource
- a - append mode (cannot be specified together with w) - a program can append to file, cannot alter the file in an other way
- k - file locking mode - a program has the ability to take file locks
- l - link mode - a program has access to hard links
- link -> - link pair rule (cannot be combined with any other)
- m - memory map as executable
- Ux - unconfined execution
- Px - use external profile for this
- Cx - use local profile for this

Rules can be preceded by `allow` (default -> has not to be written) or `deny` to apply the corresponding action.

Exercise

- Write an AppArmor Profile for the imaginary application *foobar* granting it write access to */tmp*. Include the `base` abstraction.
- Read into the */etc/apparmor.d/abstractions* file to understand what is does.

Identify Package Belongings

This section is based on the Ubuntu documentation and a LinuxAudit Article. On systems using `apt` as a package manager the tool `apt-file` can be used to determine which package a file belongs to.

Commands

The tool knows the following commands:

- update - updates the index
- search - searches all known packages for a file matching *pattern*
- list - list all files of the package matching *pattern*
- purge - purges the cache

Example

The below example lists all files belonging to vim:

```
apt-file list vim
```

On RedHat

On RedHat the package manager `rpm` has the capabilities to do the same operations with the following flags:

- `-qf <file>` - list packages using file
- `-ql <package>` - list files in package

Example

The below example does the same as the one above:

```
rpm -ql vim
```

Exercise

- Install the tool `apt-file`
- Update the `apt-file` index
- Search for the package containing */bin/ls*# User Creation and Modification This topic is based on the chapter 13.2 of the book Linux from Rheinwerk-Verlag

Creating Users

There are two available options to create users on the command line: `useradd` and `adduser`. Useradd takes the information for creation as parameters, while adduser asks for them. Adduser is a frontend for useradd, which feeds useradd the information entered.

Exemplary `useradd` call:

```
useradd dummy
```

This will add the user dummy to the system. As no parameters were specified it will have no home directory and password.

Change or Create a Password

The command `passwd` can be used to change the password of an user or create one if not existing.

```
passwd dummy
```

You will be asked for a new password for the user dummy

Exercise

- Look at the man pages of `useradd`, `adduser` and `passwd` to learn more
- Create a user *dummy* with the password *secret* (hint: adding users requires root privileges)# Working with Users In this section it will be explained how to modify and delete users.

Modify Users

Users can be modified with the command `usermod`. Usermod supplies the following basic flags:

- -l: change username
- -d: change home dir
    - -dm: move current home dir
- -L: lock user
- -U: unlock user

```
usermod -d /home/dummy dummy
```

This will assign the user dummy the homedir */home/dummy*

Delete Users

To delete users the commands `userdel` or `deluser` are used:

```
userdel dummy
deluser dummy
```

Deluser offers the possibility to remove the home directory of the user (`--remove-home`), all files of the user (`--remove-all-files`), and backup the users files as an archive (`--backup`) Userdel offers the possibility to remove the users home directory (`-r`).s

Exercise

- Read the manpages of `usermod` to learn more
- Change the username from *dummy* to *buster*
- Delete the user *buster*# Group Creation and Modification

Create Groups

Use `groupadd` or `addgroup` to create a new group. Analogous to the user creation, `addgroup` is a interactive frontend for `addgroup`.

```
addgroup new_group
```

Modify Groups

To modify groups use the `groupmod` command:

```
groupmod -n exemplary_group new_group
```

The example above changes the groupname from *exemplary_group* to *new_group*

Delete Groups

To delete groups use the `groupdel` command:

```
groupdel exemplary_group
```

Exercise

- Read the manpages for `groupadd`/`addgroup`, `groupmod` and `groupdel` if you want to learn more
- Create a group `new_group` and rename it to `dummy_group`# Manage group Memberships To add an user to a group use `usermod`. Usermod offers three options of modifying a Users groups.
- -g: modifies users main group
- -G: define list of Groups the user is part of
- -Ga: append a Group to the list

Exercise

- Create a user *dummy* and add him to the group *dummy_group*# User Creation and Modification This topic is based on the chapter 13.2 of the book Linux from Rheinwerk-Verlag

Creating Users

There are two available options to create users on the command line: `useradd` and `adduser`. Useradd takes the information for creation as parameters, while adduser asks for them. Adduser is a frontend for useradd, which feeds useradd the information entered.

Exemplary `useradd` call:

```
useradd dummy
```

This will add the user dummy to the system. As no parameters were specified it will have no home directory and password.

Change or Create a Password

The command `passwd` can be used to change the password of an user or create one if not existing.

```
passwd dummy
```

You will be asked for a new password for the user dummy

Exercise

- Look at the man pages of `useradd`, `adduser` and `passwd` to learn more
- Create a user *dummy* with the password *secret* (hint: adding users requires root privileges)# Working with Users In this section it will be explained how to modify and delete users.

Modify Users

Users can be modified with the command `usermod`. Usermod supplies the following basic flags:

- -l: change username
- -d: change home dir
  - -dm: move current home dir
- -L: lock user
- -U: unlock user

```
usermod -d /home/dummy dummy
```

This will assign the user dummy the homedir */home/dummy*

Delete Users

To delete users the commands `userdel` or `deluser` are used:

```
userdel dummy
deluser dummy
```

Deluser offers the possibility to remove the home directory of the user (`--remove-home`), all files of the user (`--remove-all-files`), and backup the users files as an archive (`--backup`) Userdel offers the possibility to remove the users home directory (`-r`).s

Exercise

- Read the manpages of `usermod` to learn more
- Change the username from *dummy* to *buster*
- Delete the user *buster*# Group Creation and Modification

Create Groups

Use `groupadd` or `addgroup` to create a new group. Analogous to the user creation, `addgroup` is a interactive frontend for `addgroup`.

```
addgroup new_group
```

Modify Groups

To modify groups use the `groupmod` command:

```
groupmod -n exemplary_group new_group
```

The example above changes the groupname from *exemplary_group* to *new_group*

Delete Groups

To delete groups use the `groupdel` command:

```
groupdel exemplary_group
```

Exercise

- Read the manpages for `groupadd`/`addgroup`, `groupmod` and `groupdel` if you want to learn more
- Create a group `new_group` and rename it to `dummy_group`# Manage group Memberships To add an user to a group use `usermod`. Usermod offers three options of modifying a Users groups.
- -g: modifies users main group
- -G: define list of Groups the user is part of
- -Ga: append a Group to the list

Exercise

- Create a user *dummy* and add him to the group *dummy_group*insert the following into ~/.bash_profile

```
echo Hello Liam
```

Replace Liam with your name.# Configuring the Local User Environment This Section is based of an article by RedHat and Chapter 07 of Linux - Rheinwerk Verlag.

The user environment is configured via files, in a default configuration they are:

- *.bash_logout*
- *.bash_profile* (sometimes *.profile*)
- *.bashrc*

Templates for these files can be found in */etc/skel*. Everything contained in */etc/skel* will be copied into a newly created users home directory. The binary path is defined in this file. To alter the template for the user environment, alter the files in */etc/skel*.

.bash_profile

This file gets executed for login shells. In a default configuration it calls *.bashrc*. Define your additional environment variables here.

.bashrc

This file gets executed each time you open a new bash while already being logged in.

Useful Environment Variables

Listed below are some commonly used environment variables and their usage:

- PATH - String that contains all directories bash looks for executables (Add dir with `export PATH=$PATH:<new dir>`)
- SHELL - path to the executable of your preferred shell (usually */bin/bash*)
- EDITOR - default editor
- PS1 - your default bash prompt (new ones can be created in online editors)

.bash_logout

Gets executed when a bash is closed if it exists.

Exercise

Make the bash say Hello <your name here> when you login.

Configuring the Global Environment

This Section is based of an article by RedHat.

There are three files to configure your environment globally:

- /etc/profile
- /etc/bashrc
- /etc/environment

/etc/profile

This file gets executed each time a user logs in. It calls all scripts residing in */etc/profile.d*

/etc/bashrc

This file (on Debian /etc/bash.bashrc) gets executed for each user when opening a new bash while being logged in.

/etc/environment

This file is read on each login. The difference between the other two is, that it is no script but rather contains variable definitions directly. Variables like *http_proxy* for Proxy configuration can be placed here:

```
http_proxy = "https://proxyurl.tld:port"
```

Exercise

Place a script in */etc/profile.d* that greets each new login with "Hi there from <your name>".Place a file *greeting.sh* with the following content in */etc/profile.d*.

```
#!/bin/bash
echo "Hi there from Liam"
```

Replacing "Liam" with your name.# Defining User Resource Limits This section is based of the Ubuntu Server Cookbook.

The command `ulimit` controls the system resources a user is allowed to use. A process started by a user has to obey the resource limits set for this user. The limits are defined in the file */etc/security/limits.conf*.

Resources can be defined for the *domains*: user (syntax: username) and group (syntax: @groupname). Limits are differentiated into two *types* hard and soft limits. While soft limits can be altered by the user, hard limits are defined by the administrator and are a cap for the soft limits.-

The following resource *items* can be limited (taken from */etc/security/limits.conf*):

- core - limits the core file size (KB)
- data - max data size (KB)
- fsize - maximum filesize (KB)
- memlock - max locked-in-memory address space (KB)
- nofile - max number of open file descriptors
- rss - max resident set size (KB)
- stack - max stack size (KB)
- cpu - max CPU time (MIN)
- nproc - max number of processes
- as - address space limit (KB)
- maxlogins - max number of concurrent logins for this user
- maxsyslogins - max number of logins on the system
- priority - the priority to run user process with
- locks - max number of file locks the user can hold
- sigpending - max number of pending signals
- msgqueue - max memory used by POSIX message queues (bytes)
- nice - max nice priority allowed to raise to values: [-20, 19]
- rtprio - max realtime priority

The syntax for defining new limits is: `<domain> <type> <item> <value>` For example:

```
@student        hard    nproc          40
```

The above example grants a member of the group student a maximum count of 40 processes.

Exercise

Write a hard limit for members of the group student, to limit their logins to 2

~~~
@student        hard    maxlogins       2
```# User Privileges
This chapter is based on chapter 13.4 of the book Linux - Rheinwerk Verlag
User Types on Linux are differentiated into **root** and **standard**
users.
Principally the root user has unlimited access to the system, while a
standard user only can access files which he has permissions to.

## Super User Do
The command `sudo` allows standard users to execute commands as root:

``` bash
sudo vim /etc/sudoers
~~~

In the above command the file */etc/sudoers* gets opened with vim. Per default only the root user has access to this file. This file regulates the access to the sudo command. Per default users who are in the group *sudo* (Debian)/*wheel* (RedHat) have access to run sudo.

Exercise

Create a file in */root*# Access to the Root User The root account is meant to be used for single tasks, which need elevated rights, **not** as a daily user for the system administrator.

During the Setup Process of a new system the password for the root user either gets set or disabled.

To change to the root user from a standard user either use `su root` if a password is set and known or `sudo su` to use the sudoer privileges of the standard user if present.

Exercise

Switch to the root user and insert the text: 'Written As Root' to the file.# Pluggable Autentication Modules This chapter is based on the Red Hat Documentation. Pluggable Authentication Modules (PAM) central way to authenticate. PAMs offer different types of authentication (Kerberos, NIS, local filesystem) and are therefor called pluggable.

Because of its pluggable nature PAM gives administrators great flexibility on configuring authentication on systems. PAM can be used by a wide range of applications.

On RedHat Systems the tools `authconfig` or `authselect` are recommended to configure PAM, on Ubuntu the configuration files are edited directly.

The configurations files are `/etc/pam.conf` or any file in `/etc/pam.d` (if `/etc/pam.d` exists `/etc/pam.conf` will be ignored). Each of these files should consist of multiple rule lines. Each rule-line has a syntax as follows:

```
service type control module-path module-arguments
```

(For files in the `/etc/pam.d` the service field is omitted, instead the name of the file references the service)

```
auth        substack     system-auth
auth        include      postlogin
account     required     pam_nologin.so
...
```

Above is an excerpt of the content of the */etc/pam.d/login* file. In the first line the **auth** content of the file */etc/pam.d/system-auth* is evaluated as an substack, meaning the statements will be checked, but if one fails only the substack fails. In the second line the **auth** content of the file */etc/pam.d/postlogin* gets included, meaning the auth lines in the file are evaluated, if one fails, the original stack fails too. In the third line the pam module *pam_nologin.so* gets included as required, meaning it succeeding is necessary to continue the stack, **pam_nologin.so** fails if the file */etc/nologin* exists.

The auth type calls are used to identify the user. The account type is used to grant/restrict access based on defined criteria, in the case of the *pam_noloing.so* module the presence of the *nologin* file.

Exercise

- Read the manpage of *pam.conf* to learn more about types, controls and the general definition of the rules.# Basic Network Configuration This section is based on an LinuxConfig article, a Stackoverflow post and the Ubuntu Wiki On Debian network interfaces are defined in the */etc/network/interfaces* file. This file includes the files in */etc/network/interfaces.d*.

WLAN

To configure a wlan interface a file in */etc/network/interfaces.d* has to be created with the following contents:

```
auto wlan0
iface wlan0 inet dhcp
wpa-essid <wlan-ssid>
wpa-psk <wpa-key>
```

The interface `wlan0` has to be replaced with the actual interface, which can be determined with the command `ip a`:

```
2: enp2s0f0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 ...
...
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
...
```

The above example shows an output of `ip a`, the wlan interface `wlp3s0` and the ethernet interface `enp2s0f0` can be identified.

Ethernet

To configure a ethernet interface use the following content:

```
auto eth0
iface eth0 inet dhcp
```

WLAN as well as ethernet are configured to retrieve their IP address via dhcp. The system has to be restarted to apply the settings.

The `auto` tag ensures the interfaces are brought up on startup.

The Networking Service

The default networking service on Ubuntu is `systemd-networkd`. Systemctl can be used to start, stop, enable and disable it and to check its status as seen in the file **111_manage_startup_services**.

Exercise

- View the network devices on your machine.# Hostname Resolution This chapter is based on the freedesktop documentation and the Ionos Guide.

Hostnames

Hostnames or domain names are a human readable identifier to identify network devices. A hostname points to the IP of a device. To be able to connect to a device its hostname has to be resolved to its IP address. This is done by hostname resolution. Host machines which are in a network can be addressed by the resolved IP address.

Hostnames offer the advantage that they stay the same, while the IP address can change. Which means they can serve as a reliable identifier for a device or service.

Static Resolution

On Linux systems static domain name resolution can be defined in the file */etc/hosts*:

```
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
```

The file points the localhost domains to the localhost ip addresses. An entry for *example.com* would look like this:

```
93.184.216.34 example.com
```

Dynamic Resolution

As the IP of a network target might change, dynamic hostname resolution offers a way to connect to the device anyway, without knowing its exact IP. To do this a query is sent to a DNS server, the Server resolves the hostname to an IP and sends it back to the host. To check the currently used nameservers `resolvectl status` can be used:

```
...
DNS Servers: 192.168.178.1
...
```

To alter the resolve configuration written in */etc/resolve.conf* the tool `resolvconf` has to be used, as the file is managed. After installing the tool the file */etc/resolvconf/resolv.conf.d/head* can be altered. An additional name server can be added to this file. The format of the entry would be:

```
nameserver <IP>
```

After editing the file the `resolvconf` has to be run with the flag `-u` to update the */etc/resolv.conf* file.

Exercise

- Install `resolvconf`
- Add the Google DNS Server *8.8.8.8* to the systems configuration

Packet Filtering

This section is based on chapter 32.6 of Linux - Rheinwerk. On Linux packet filtering mechanisms are built into the Kernel. They can be configured with the `iptables` tool.

Iptable Tables

Iptables has various functionalities, besides package filtering it also offers NAT possibilities. These functionalities are configured in different tables. The table to edit is specified by the `-t` flag. The default table is the `filter` table responsible for the package filtering configuration.

Iptable Chains

Iptables sends packages down different chains based on rules. These chains determine what happens with the package.

There are several predefined chains but it is also possible to create custom ones with the `iptables -N` command.

The following filtering chains are predefined:

- `INPUT` - Contains rules for packages designated for the local client
- `OUTPUT` - Contains rules applied to all outgoing packages
- `FORWARD` - Contains riles for all to be forwarded packages

Iptables offers the following operations on chains:

- `-N <chain>` - Create new chain
- `-X <chain>` - Delete Chain
- `-E <old> <new>` - Rename a chain
- `-L <chain>` - List rules of specified chain
- `-F <chain>` - Flushes chain - Deletes all rules in chain

Iptable rules

The iptable chains are made up of a collection of rules. Rules are evaluated from top to bottom, of one rule matches it will be applied to the package and the processing will be halted. These are the most common rule operations:

- `-A <chain> <rule>` - add rule to an existing chain
- `-I <chain> <position> <rule>` - insert rule into chain on specified index; when no position specified, the rule will be inserted at index 0
- `-R <chain> <position> <rule>` - Remove rule at specific index
- `-D <chain> <rule-number/rule-name>` - Deletes rule matching rule-number or name

Rules are made up of filtering expressions and destinations. The destinations can be seen above in the listing below:

The following chains are possible:

- ACCEPT - The package will be allowed
- DROP - The package will discarded
- QUEUE - Forward the package to userspace (if supported by kernel)
- RETURN - Return to calling chain (if in user defined chain)
- LOG - Log into the kernel log
- REJECT - Same behavior as drop, sends back *Port Unreachable* error message.
- user defined chain - Jump into a user defined chain

The most common filtering expressions can be seen below, they can be negated with a `!`:

- `-s <source>` - Filters for the source of a package
- `-d <destination>` - Filters for the destination of a package
- `-i <interface>` - Filters for the interface that received the package
- `-o <interface>` - Filters for the interface to output the package
- `-p` - Filters for the protocol used to transmit the package (most common are TCP or UDP)

Based on the used protocol filtering more filter modules will be loaded. Modules can also be loaded via the `-m <module>` flag. See the iptables manpage for more information.

When creating a rule the flag `-j` is used to *jump* to destination.

Network destinations can be applied in the following forms:

- Network Name
- Hostname
- Network Address + Subnet Mask
- IP-Address

Persistance

Updates made via iptables will be temporary and reset after a reboot. To make the changes persistent the command `iptables-save` and redirecting the output to */etc/iptables/rules.v4* (for Debian) will make the changes persistent between reboots:

```
# Debian
iptables-save > /etc/iptables/rules.v4
# RedHat
iptables-save > /etc/sysconfig/iptables
```

Taken from here.

Example

An exemplary rule can look like this:

```
iptables -A INPUT -p tcp -j TCP
```

This will add a call to the custom chain `TCP` for all incoming TCP packages.

Exercise

Create a rule for all Incoming packages that drops the packages coming from the IP-Address `192.168.178.150`

Static IP Routing

This section is based on the RedHat documentation and a Linux Config Article.

Show Current Routing Table

To show the currently routing table the command `ip route show` is used.

Temporal IP Routing

To configure a temporal IP route the command `ip route` with the following opeations is used:

- add: `ip route add <destination>/<netmask> <gateway>`

- del: `ip route del <destination>/<netmask>`
- change: `ip route change <destination>/<netmask> <gateway>`
- append: `ip route append <destination>/<netmask> <gateway>`
- replace: `ip route replace <destination>/<netmask> <gateway>`

```
# add a route to statically rout the ip address 192.0.2.1 over the address
10.0.0.1 as a hop/gateway
ip route add 192.0.2.1 via 10.0.0.1
```

The address to route also can be specified as a combination of IP address and netmask.

Permanent IP Routing

Routes created by the ip command will only be applied temporally. To make the route permanent it has to be added to the file */etc/netplan/50-cloud-init.yaml* with the following syntax:

```
network:
    ethernets:
        enp0s3:
            dhcp4: false
            addresses: [192.168.1.202/24]
            gateway4: 192.168.1.1
            nameservers:
              addresses: [8.8.8.8,8.8.4.4,192.168.1.1]
            routes:
            - to: 172.16.0.0/24
              via: 192.168.1.100
    version: 2
```

Taken from here

In the example the addresses `172.16.0.0/24` will be routed over the gateway `192.168.1.100` this applies to the interface `enp0s3`.

To apply this configuration the command `sudo netplan apply` has to be run.

Exercise

Take the above netplan example an add a route to `172.18.52.31` via `192.168.1.101`.# Time Synchronization This section os based on the German Ubuntu Documentation. To synchronize the time via network the network time protocol can be used under Linux.

Install NTP

To install the NTP package the command `sudo apt install ntp` has to be used.

Configure NTP

To configure the NTP the file `/etc/ntp.conf` has to be edited. In this file the time servers to use can be specified:

```
server 0.de.pool.ntp.org
server 1.de.pool.ntp.org
server 2.de.pool.ntp.org
server 3.de.pool.ntp.org
```

Above the German time pool servers are configured. NTP servers can be searched here. After a restart the service will use the newly configured servers. It is recommended to use multiple time servers. The timeserver with the lowest offset to **Stratum 0** (original time sources) will be used. The command `ntpq -p` can be used to check the currently configured time servers.

Exercise

- Install the NTP package
- Configure NTP to use the time server pools recommended for Croatia here.

Configure a Caching DNS Server

This section is based on a cloudflare article and a LinuxTeck guide.

Bind

The package *BIND* provides the service *named* which can serve as a caching DNS server. The caching server is able to cache queries for a certain time - TTL (Time to Live). This allows for faster response times.

Installation

On Debian Systems the package *bind9* has to be installed (`apt`). To ensure the name server starting on each startup it has to be enabled in systemd (`systemctl`).

Per default the installed service will cache DNS queries.

Configuration

To configure the *named* service the file */etc/named.conf* has to be edited. In the default configuration the DNS server accepts from localhost. To allow traffic from other systems add the statement any to the following lines:

```
listen-on port 53 { 127.0.0.1; any; };
allow-query { localhost; any; };
allow-query-cache { localhost; any; };
```

Iterative and Recursive DNS

When querying iterative the DNS request is forwarded by DNS server to the next responsible DNS server. This results in the client communicating directly to the involved DNS servers. When querying recursive the first DNS server communicates with all other DNS servers in the chain to answer the query himself.

Exercise

- Install the package *bind9*
- Configure view the configuration file */etc/named.conf*# DNS Zones Based on this Ubuntu Wiki article, the examples are taken from there.

DNS Zones are

Forward vs. Forward Lookup

In Reverse Lookup the DNS server is responsible for resolving the IP address to the hostname. In Forward Lookup the DNS server is responsible for resolving the hostname to the IP address.

Defining Zones

When configuring a new zone two files have to be created, one for the forward lookup and one for the reverse lookup. The naming convention of the files is the following:

```
# Forward Lookup
db.domainname
# Reverse Lookup
db.z.y.x
# z.y.x being parts of the IP address x.y.z (192.168.0.* -> db.0.168.192)
# note the reverse order!
```

An exemplary forward lookup file looks like this:

```
;; db.domainname
;; Forwardlookupzone für domainname
;;
$TTL 2D
@       IN      SOA     rechnername.domainname. mail.domainname. (
                        2006032201      ; Serial
                                8H      ; Refresh
                                2H      ; Retry
                                4W      ; Expire
                                3H )    ; NX (TTL Negativ Cache)

@                               IN      NS      rechnername.domainname.
                                IN      MX      10 mailserver.domainname.
                                IN      A       192.168.0.10

rechnername                     IN      A       192.168.0.10
localhost                       IN      A       127.0.0.1
rechner1                        IN      A       192.168.0.200
```

```
mailserver                      IN      A       192.168.0.201
rechner2                        IN      CNAME   mailserver
```

Forward Lookup

The time to life (`TTL`) specifies the time frame information is cached. After the `TTL` entry the `SOA` (Start of Authority resource records) entry is defined. It specifies the following values:

- *zone-origin* - `rechnername.domainname` - Fully Qualified Domain Name (FQDN) of the primary DNS server.
- *zone-contact* - `mail.domainname` - the email address of the responsible person, the `@` is substituted for a `.` (mail.domainname would be mail@domainname)
- *serial* - `2006032201` - important for versioning, typically in the format *YYYYMMDDSS* (current time) to allow secondary DNS servers to notice changes.
- The following entries should be on good default values and don't have to be changed.
- *refresh* - Refresh interval for secondary DNS servers
- *retry* - Time a secondary DNS server waits to send another query if the previous one failed.
- *expire* - Timeout time for a connection attempt of a secondary DNS server to the primary server.
- *nx* - Time frame for negative caching (caching of failed lookup attempts).

The entry types being the following:

- `NS` - **value:** FQDN of a nameserver - *primary and secondary DNS servers of domain should have an entry*
- `A` - **value:** IP address - *resolves name -> IP address*
- `CNAME` - **value:** hostname of the specified client
- `MX` - **value:** priority, hostname - *mail-server for the domain + priority -> servers with low priority will be tried first.*
- `PTR` - **value:** FQDN - *reverse lookup IP -> name (e.g. foo.example.com)*

Reverse Lookup

An exemplary reverse lookup file looks like this:

```
;; db.0.168.192
;; Reverselookupzone für domainname
;;
$TTL 2D
@       IN      SOA     rechnername.domainname. mail.domainname. (
                                2006032201      ; Serial
                                        8H      ; Refresh
                                        2H      ; Retry
                                        4W      ; Expire
                                        3H )    ; TTL Negative Cache

@       IN      NS      rechnername.domainname.

10      IN      PTR     rechnername.domainname.
```

```
200     IN      PTR     rechner1.domainname.
201     IN      PTR     rechner2.domainname.
```

Here the PTR entry type comes to use. The numbers in the first row represent the last byte of the IP address.

Exercise:

Read the Wikipedia article on DNS if you want to learn more.

students-guide-to-linux

- [x] Essential Commands 25%
  - [x] Log into local & remote graphical and text mode consoles
  - [x] Search for files
  - [x] Evaluate and compare the basic file system features and options
  - [x] Compare and manipulate file content
  - [x] Use input-output redirection (e.g. >, >>, |, 2>)
  - [x] Analyze text using basic regular expressions
  - [x] Archive, backup, compress, unpack, and uncompress files
  - [x] Create, delete, copy, and move files and directories
  - [x] Create and manage hard and soft links
  - [x] List, set, and change standard file permissions
  - [x] Read, and use system documentation
- [x] Operation of Running Systems 20%
  - [x] Boot, reboot, and shut down a system safely
  - [x] Boot or change system into different operating modes
  - [x] Install, configure and troubleshoot bootloaders
  - [x] Diagnose and manage processes
  - [x] Locate and analyze system log files
  - [x] Schedule tasks to run at a set date and time
  - [x] Verify completion of scheduled jobs
  - [x] Update software to provide required functionality and security
  - [x] Verify the integrity and availability of resources
  - [x] Verify the integrity and availability of key processes
  - [x] Change kernel runtime parameters, persistent and non-persistent
  - [x] Use scripting to automate system maintenance tasks
  - [x] Manage the startup process and services (In Services Configuration)
  - [x] List and identify SELinux/AppArmor file and process contexts
  - [x] Manage Software
  - [x] Identify the component of a Linux distribution that a file belongs to
- [x] User and Group Management 10%

  - [x] Create, delete, and modify local user accounts
  - [x] Create, delete, and modify local groups and group memberships
  - [x] Manage system-wide environment profiles
  - [x] Manage template user environment
  - [x] Configure user resource limits
  - [x] Manage user privileges
    - [x] Manage access to the root account
  - [x] Configure PAM

- [x] Networking 12%

  - [x] Configure networking and hostname resolution statically or dynamically
  - [x] Configure network services to start automatically at boot
  - [x] Implement packet filtering
  - [x] Start, stop, and check the status of network services
  - [x] Statically route IP traffic
  - [x] Synchronize time using other network peers

- [ ] Service Configuration20%

  - [ ] Configure a caching DNS server
  - [ ] Maintain a DNS zone
  - [ ] Configure email aliases
  - [ ] Configure SSH servers and clients
  - [ ] Restrict access to the HTTP proxy server
  - [ ] Configure an IMAP and IMAPS service
  - [ ] Query and modify the behavior of system services at various operating modes
  - [ ] Configure an HTTP server
  - [ ] Configure HTTP server log files
  - [ ] Configure a database server
  - [ ] Restrict access to a web page
  - [ ] Manage and configure containers
  - [ ] Manage and configure Virtual Machines

- [ ] Storage Management 13%

  - [ ] List, create, delete, and modify physical storage partitions
  - [ ] Manage and configure LVM storage
  - [ ] Create and configure encrypted storage
  - [ ] Configure systems to mount file systems at or during boot
  - [ ] Configure and manage swap space
  - [ ] Create and manage RAID devices
  - [ ] Configure systems to mount file systems on demand
  - [ ] Create, manage and diagnose advanced file system permissions
  - [ ] Setup user and group disk quotas for filesystems
  - [ ] Create and configure file systems

Hello World!

To get started with Linux administration, a Ubuntu base Image will be set up in Docker and we will have it print "Hello World!" If you are already working on a Linux System you can skip the Docker part.

Installing Ubuntu Base Docker Image

Open a console (PowerShell or CMD) an enter the following command `docker pull ubuntu`.

Docker will start pulling the Ubuntu image. After pulling the image you can view it with the command `docker image ls`. The Ubuntu image should show up in the resulting list.

```
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    597ce1600cf4   13 days ago    72.8MB
```

Start the Docker Image and Open a Shell in It

Use the command `docker run ubuntu` to start up the container. The command `docker ps -a` will list the running containers.

The container we just sterted should appear here.

```
Insert Container List here
```

Now we have a virtualized Ubuntu image running in Docker. The next step is to start a shell in this image to work with it. To start a shell in this image we use the command `docker exec -it <container name> /bin/bash`.

Now we are working with the shell we just started on our command line.

Execute the 'Hello World!'

To let the shell greet the world the command `echo` will be used. The string to be put out on the shell ('Hello World!'), is supplied as an argument to `echo`.

~~~~
echo 'Hello World!'
```# The file system Outline

Like in Windows files in Linux/Unix are organized in a file tree. The first
folder in this tree is called the root folder. While Windows has a separate
tree for every device, devices get assigned a folder in the directory tree
in Linux.
This topic is discussed in chapter 2 of the Linux command line.

## The root tree
The branches following the root are the following

- */bin*: contains binaries required to run the system
- */boot*: contains kernel, driver image for startup, boot loader
~~~~

```
- */dev*: contains the devices connected to the PC (everything is a file)
- */etc*: contains system-wide configuration files
- */home*: contains the home directories of the users (normal users can
only write here)
- */lib*: contains shared libraries
- */lost+found*: used in case of a file system corruption, for recovery
purposes
- */media*: contains automatically mounted storage devices
- */mnt*: used for manually mounting storage devices
- */opt*: contains optional software, mainly commercial software
- */proc*: contains peepholes into the kernel
- */root*: home directory of the root user
- */sbin*: contains system binaries
- */tmp*: contains temporary files of programs (emptied on every start)
- */usr*: unix shared resources
  + */usr/bin*: contains programs installed by he distribution
  + */usr/lib*: shared libs for programs in */usr/bin*
  + */usr/local*: software not included in the distribution but installed
systemwide
  + */usr/sbin*: contains system administrator programs
  + */usr/share*: contains shared date used by programs in */usr/bin*
  + */usr/share/doc*: contains system documentation
- */var*: contains data which is touched and changed frequently
  + */var/log*: contains logs produces by the system

For administrators the most frequently used directories are */etc*, */var*
and */usr*# Navigating the File System
This section will describe how to find your way through the file system.

## Useful commands
- `cwd`: The command `cwd` will output the path to the directory you are
currently in.
- `cd`: change your cwd (use .. to get into the directory above the one you
are currently in).
  - Useful Shortcuts:
      - `cd -`: change to previous working directory
      - `cd`/`cd ~`: change to your home dir
- `ls`: To see which files are contained by a directory use the command
`ls`. Use `ls -l` for more details.
- `file`: `file` will show you the type of a file.
- `less`: use `less` to get a scrollable view of the contents of a text
based file.
  - Controls:
    - q: exit
    - /searchstring: replace searchstring with an actual string you are
searching for and hit ENTER to search for it.
      - n: next result
    - g: jump to the beginning of the file
    - G: jump to the end of the file

## Exercise
Explore the file system and get to know the navigation commands.
```

~~~
# Search For Files

Multiple approaches for searching for files are offered. If you are
searching for a file by its file name `locate` is the choice, `find` is
used to search for files by its attributes, `grep` allows searching for
files via their content.

## The `locate` command

Locate performs a database search for paths matching the search term.

``` bash
locate Pictures/Screenshot
~~~

The above command will search for all files and folders starting with `Screenshot-` laying under a location containing `Pictures`. It simply searches for the specified search term in the full path of a file. An exemplary result of the command can be seen below:

```
/home/user/Desktop/backup/Pictures/Screenshots
/home/user/Desktop/backup/Pictures/Screenshots/35574917.png
/home/user/Desktop/backup/Pictures/Screenshots/Screenshot_20210208_093220.p
ng
/home/user/Pictures/Screenshot_20210921_224523.png
```

Exercise: *Use locate to list all .bak files on the system.*

The `find` command

To search for files by file attributes the `find` command is used. Find requires only the path to search in as an argument. If you give your home-path `~` to it. It lists all files and folders located in your home directory, as we didn't specify what to search for.

```
find ~
```

To specify what to search for flags are used:

```
find ~ -name "Screenshot*"
```

The above command searches for all files and directories beginning with the word `Screenshot` the asterisk is used as a wildcard for a unknown number of unknown chars. Find offers a number of such search flags which help to narrow the search down further.

```
find ~ -name "Screenshot*" -type f
```

This command limits the search to results of the type **f**ile.

File Type	Description
b	Block special device file
c	Character special device file
d	Directory
f	Regular File
l	Symbolic Link

A full list of the flags available can be seen in **The Linux Command Line** in the subchapter *17 - find - Tests* (Table 17-3)

Exercise: *List all files under your home directory (~).*

Logical Operators

The `find` command allows combination of tests via logical operators to search for files more granular.

Operator	Description
-and/-a	True if the tests on both sides are true (Implied as default, when no operator used)
-or/-o	True if one test on the side is true
-not/!	True if the test following is false
( )	Groups operations, modifying the evaluation order

The example below searches for *png* and *pdf* (`-name "*.png" -or -name "*.pdf"`; `-type f`) files, in the home dir of the current user (`~`), owned by the user *dummy* (`-user dummy`)

```
find ~ \( -name "*.png" -or -name "*.pdf" \) -type f -user dummy
```

Exercise: *List all files owned by root or the current logged in user (find out username with `whoami`) in /home.*# Actions

Find can be used to execute actions on the files found.

Predefined Actions

Find offers predefined actions, which can be seen in by calling `find --help`

```
actions: -delete -print0 -printf FORMAT ...
```

The core actions are explained in **The Linux Command Line** in the subchapter *17 - find - Predefined Actions*

The example below will delete all screenshot files of the current user.

```
find ~ -name "Screenshot*" -type f -delete
```

The order of the tests and actions is important as by leaving the logical operator between the tests an `-and` is implied.

```
find ~ -name "Screenshot*" -and -type f -and -delete
```

Order of execution:

- 1. `-name "Screenshot*"` is executed first, without any dependencies
- 2. `-type f` is executed second, in case `-name "Screenshot*"` matched
- 3. `-delete` is executed third in case `-type f` also matched

As the print statement is connected with an `-or` in the example below, it will be executed independent from the statements before, printing all files.

```
find ~ -name "Screenshot*" -and -type f -or print
```

Exercise: *Delete all .bak files in the example directory tree.*

User Defined Actions

It is possible to execute user defined actions on results of the find command by adding the pattern below:

```
-exec command {} ;
```

The `{}` is the symbolic representation of the results pathname. The `;` serves as a delimiter for the command. Instead of `;`, `+` can be used. With the semicolon the action will get executed for every result in once. By using the plus as a delimiter, find will combine pathnames and execute them at once, making the command more efficient.

The above example of deleting screenshots would look like this with a user defined action:

```
find ~ -name "Screenshot*" -type f -exec rm '{}' '+'
```

Exercise: *Create a Bash script, which adds read privileges to all .log files in the exemplary folder structure for the current user. Tip: To add read privileges for the current user use* `chmod u+r <filename>`# Search for file content In this chapter a easy way to search for file content via `grep` will be explained.

`grep`

Grep is a tool to search files via regular expressions it. To see its full usage view its manpage `man grep` or go to the subchapter *3 - 19 - grep* in **The Linux command line**

Grep can be used to search multiple files for a pattern by using the `-R` flag.

```
grep -R "some_pattern" *
```

The asterisk at the end is used to search through all files and folders in the current directory.

Exercise 1: *Search the files in ~/example_tree for the pattern 'example_content'*

Exercise 2: *Use find in combination with grep to search all .txt files in ~/example_tree for the pattern 'example_content'*

Create Directories

This Topic is dealt with in Chapter **4 - mkdir** of The Linux Command Line.

Create Single Directory

To create a new directory use the command `mkdir`.

```
mkdir sampledir
```

Create Directory in Directory

The above example will create the directory sample dir in the current directory. To create a new folder in *sampledir* you can type:

```
mkdir sampledir/otherdir
```

Create Whole Tree

This command assumes sampledir is already existing. To create a whole directory tree if not existing use the Switch `-p`

```
mkdir -p notexisting/sampledir
```

The above example will create the folder *notexisting* and the directory *sampledir* within it.

Create Multiple Directories

To create multiple directories simply write them after each other like that:

```
mkdir sampledir othersampledir
```

Exercice

This exercise will combine the simple Operations listed above. Write a script, which creates the folders *sampledir/withinnerdir* and *singlesampledir*

Deleting Directories

This Topic is dealt with in Chapter **4 - rm** of The Linux Command Line.

Deleting Directories with `rmdir`

To delete a directory the command `rmdir` can be used.

```
rmdir sampledir
```

Delete Directories with `rm`

The `rm` command also can be used to delete directories with its `-r` switch. This will also delete all files and folders in the directory. More on the topic `rm` in the following chapter.

```
rm -r sampledir
```

Exercise

Write a script to create the Directory *sampledir* and delete it.

Deleting Files

This Topic is dealt with in Chapter **4 - rm** of The Linux Command Line.

The `rm` command

The `rm` command is used to delete files and folder on the Linux Command Line. It has 3 important operational flags:

Flag	Long	Usage
-i	--interactive	Prompt for confirmation before deleting
-r	--recursive	Recursively delete -> Delete subfolders and files
-f	--force	do not ask before deleting write protected files, ignore non existent files; **overrides** interactive

The flags are used in the example below:

```
# Deletes the file samplefile after asking for confirmation
rm -i samplefile

# Deletes the content of sampledir and it's content
rm -r sampledir

# Deletes the writeprotected file writeprotectedsample without asking for
confirmation
rm -f writeprotectedsample
```

Exercice:

Use the script `020_testenv.sh` to create testfiles in your current folder and write a script to delete the folder *somedir* and its write protected contents without being asked for confirmation.

Copying Files and Directories

This Topic is dealt with in Chapter **4 - cp** of The Linux Command Line.

The `cp` command

The `cp` command is used to copy files and directories (`-r` switch has to be set) Its usage is `cp <src> <dest>`, multiple source files can be given like `cp <src1> <src2> ... <dest>` To copy file *a* to location *b* use:

```
cp a b
```

As a result the file *a* will be copied to file *b*

To copy a whole folder use the `-r` switch just like in the `rm` command.

```
cp -r somefolder someotherfolder
```

Exercise

Execute the *020_testenv.sh* script. Write a script to copy the file *somedir/writeprotected* to be next to *somedir* (same folder level) with the filename *writeprotected.bak*. And create a copy of *somedir* (*somedir.bak*)

Useful switches

- `-i` To avoid overwriting ask for confirmation
- `-u` Update, does not copy if destination file is same or newer

Moving Files and Directories

This Topic is dealt with in Chapter **4 - cp** of The Linux Command Line.

The `mv` command

The `mv` command is used to move or rename files and directories. Its Usage is very similar to the `cp` commands. It shares the useful switches mentioned in **Copying Files and Directories**. Its also possible to define multiple files and dirs to be copied `mv <src1> <src2> ... <dest>` To rename a file simply move it on the same 'level':

```
mv a b
```

This changes the filename from *a* to *b*

To move a file or directory to an other place give the new place as last parameter

```
mv a somedir/
```

This moves a to the subfolder *somedir*

Exercise

Execute the *020_testenv.sh* script. Make use of one of the useful switches and move the file *somedir/writeprotected* to *someotherdir*, but only if it is newer than the file *someotherdir/writeprotected*

Linking Basics

Linux offers two ways of linking files. *The Linux Command Line* treats this topic in *Chapter 4 - ln - Create Links*

Hardlinks

Hardlinks are the original way of linking files in the Unix world. In fact each file you see in a file explorer is a hardlink to a file, which gives the file its name. A file can have multiple hardlinks. A will get deleted/dereferenced when all hardlinks referencing it get deleted. Hardlinks can only point to files in the same filesystem.

Softlinks/Symbolic Links

Soflinks are the modern way of linking files in Unix Systems, they overcome the limits of hardlinks. Softlinks are special files containing text pointers to the file they are referencing.

Direct Comparison

The following code shows a directory with the file `test.txt`, a hardlink and a softlink to it. It can be seen, that the original file and the hardlink share the same inode number - they are quintessentially the same. The softlink has a different inode number - it is an independent file. The softlink additionally is pointing to the original test.txt file.

```
9181703 -rw-rw-r--. 2 dummy dummy 0 Dec 27 14:58 test.txt
9181703 -rw-rw-r--. 2 dummy dummy 0 Dec 27 14:58 hardlink
9181859 lrwxrwxrwx. 1 dummy dummy 8 Dec 27 14:59 softlink -> test.txt
```

Excursion: Inode Number

The inode number is an index in the so called inode table. In this table information about the file is stored, like its owner, size, permissions, or location. A files inode basically is its identity.# Creating Hardlinks

The command `ln` is used for creating links.

To create hardlinks use it like this:

```
ln <original-file> <link-location>
# For example
ln test.txt hardlink
```

Creating Softlinks

To create softlinks use the `ln` command with the `-s` flag:

```
ln -s <original-file> <link>
# For example
```

```
ln -s test.txt softlink
```

Exercise

Use the command `touch test` to create the empty file test in your current location. Write a script to create a hardlink called `test_hardlink` to the file test and create a softlink, ``test_softlink`to the hardlink. **Extra:** Use the command`ls -li test*` to view all three files and compare their inode numbers. # Read, and Use System Documentation

Most commands/programs used on Linux offer manpages. Manpages are extensive manuals/documentations of the programs. To access a Programs man page type:

```
man ls
```

The man page can be searched for patterns by typing `/` and the pattern you want to search for, hitting `ENTER` takes you to the first result. You can jump between the results by hitting `n` (next) and `p` (previous). To exit the man page hit `q`.

Exercise

- View the manpage for the command `ls` and list the files in the directory in *long format recursively* sorted *by time*.
- Lookup the usage of the commands `whereis` and `apropos` which will help you finding commands and exploring the system.

Introduction Input/Output Redirection

In this section input/output (I/O) redirection will be explained. The according chapter in the Linux command line is 1.6

Standard Input, Output, Error

Standard output (stdout) and standard error (stderr) follow the Unix philosophy of everything is a file. If a program on the command line needs to output its result it writes them to the *stdout* file, status messages like errors are written into the *stderr* file. Programs often grab their inputs from the standard input file (stdin), this is by default connected to the keyboard. Which files are used for standard input, output, error can be altered via Input/Output Redirection.# Output Redirection In this chapter the functionality of output redirection will be discussed

Ouput Redirection Overwriting

To redirect the output into a file the operator `>` is used. Using `>` will overwrite the contents of the destination file given. In the example the output of the command `ls -la` will be written into the file *ls-output*

```
ls -la > output
```

Now we use the command `cat` to view the content of the file. Cat takes a file as an argument and displays its contents to the command line. We won't see an output on the command line, but the file now looks something like this:

```
drwxr-xr-x 4 dummy dummy 4096 Jan 25 10:38 .
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:37 ..
drwxr-xr-x 8 dummy dummy 4096 Jan 25 10:54 .git
-rw-r--r-- 1 dummy dummy   17 Jan 25 10:38 .gitignore
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:42 01_basic_commands
-rw-r--r-- 1 dummy dummy 1070 Jan 25 10:37 LICENSE
-rw-r--r-- 1 dummy dummy 3605 Jan 25 10:38 README.md
```

Output Redirection Appending

By using the `>>` operator output will be appended to the specified file if we repeat the above example our file should look like this:

```
drwxr-xr-x 4 dummy dummy 4096 Jan 25 10:38 .
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:37 ..
drwxr-xr-x 8 dummy dummy 4096 Jan 25 10:54 .git
-rw-r--r-- 1 dummy dummy   17 Jan 25 10:38 .gitignore
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:42 01_basic_commands
-rw-r--r-- 1 dummy dummy 1070 Jan 25 10:37 LICENSE
-rw-r--r-- 1 dummy dummy 3605 Jan 25 10:38 README.md
drwxr-xr-x 4 dummy dummy 4096 Jan 25 10:38 .
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:37 ..
drwxr-xr-x 8 dummy dummy 4096 Jan 25 10:54 .git
-rw-r--r-- 1 dummy dummy   17 Jan 25 10:38 .gitignore
drwxr-xr-x 5 dummy dummy 4096 Jan 25 10:42 01_basic_commands
-rw-r--r-- 1 dummy dummy 1070 Jan 25 10:37 LICENSE
-rw-r--r-- 1 dummy dummy 3605 Jan 25 10:38 README.md
```

Error Redirection

The standard error can be redirected with `2>`/`2>>`. In the example it is tried to list the content of a restricted folder resulting in an error.

```
ls /root > output
```

Again we won't see an output, but our file will look like this:

```
ls: cannot open directory '/root': Permission denied
```

Redirect Both Outputs

To redirect bot outputs the `&>`/`&>>` operators can be used.

Exercise

Experiment with output redirection to see how it works. For example use `echo foo > output` to write foo to the file output and overwrite or append content.

Piping

Piping is a special variant of I/O redirection. It is used to use the output of one command as the input of an other.

To pipe the `|` operator is used.

```
ls | wc -w
```

The above example will give the output of `ls` to the command `wc -w`. `wc -w` counts the words in a given file. So by piping the output of `ls` into it sums up the count of words used as filenames in the current folder.

Filters

Filters take the content as input, alter it and then output it. Following are some common filters:

- `sort`: will sort input alphanumerically
- `uniq`: accepts sorted list and deleted duplicates
- `wc`: as seen above counts words in input
- `grep`: as seen in Chapter Search for Files can be used to filter for expressions

The above commands/filters offer more functionality than the basic one described, see their man pages for more details.

The `tee` Utility

`tee` reads the standard input and copies it to standard output and one or more specified files.

```
ls | tee output | grep output
```

The above example will write the outputs of `ls` to the file *output* and then search the output of `ls` (forwarded by `tee`) for the word output. If there is no file named *output* in the folder in which the command is executed it will have no output on the first run as the file *output* doesn't exist at first.

Exercise

Count the occurances of the word foo in the file *054_exercise.txt* by using the commands `grep` and `wc` as well as piping. If you are unsure about the usage of the commands look up their man pages.

Persmission Basics

The topic permissions is discussed in chapter 9 of The Linux Command Line. If you list the contents of a directory with `ls -l` you can see the owner, group and permissions set for a file.

```
-rwxr--r-- 1 dummy dummy   43 Jan 26 09:34 output
```

Ownerships

The first name that can be seen (dummy) is the *user* owning the file, the second one ist the *group* owning the file A user can be member of multiple groups, but can only have one primary group (in this case dummy). The group owning a file can be different from the owners primary group. You can view the current user and its infos with the command `id`. To see the groups you are part of use `groups`.

Permissions

The permissons are split into three flag triplets:

- *r*: being able to **read** the contents of the file
- *w*: being able to **write** to the file
- *x*: being able to **eXecute** the file

The first *rwx* triplet defines the permissions of the owner over the file, the second one the permissions of the group and the third one the permissions of other.

Sudo

In case you don't have the needed permissions to perform a operation you can use `sudo` - super user do. To use sudo your user has to be part of the group `sudoers` on Debain based machines and `wheel` on RedHat based machines.

Exercise

Use `id`to get some infos about your user.

Changing the Owner and Group of a File

In this chapter it will be explained how to change the owner and the group owning a file.

Changing the Owner of a file

To change the owner of a file the command `chown` is used:

```
chown dummy output
```

The above example will change the owner of the file *output* to the user *dummy*.

Changing the Owning Group of a File

To change the group owing a file, the commands `chgrp`

```
chgrp dummygrp output
```

```
chown dummy:dummygrp output
```

The above examples both will change the owning group of *output* to dummy group. By using `chgrp` you don't have to know the owner of the file to change the group.

Exercise

- Create a file *examplefile* with the command `touch examplefile` and change its owning group to *nobody*.
- Lookup the man pages of `chown` and `chgrp` to learn more about the commands

Changing Permissions of a File

To change the permissions of a file the command `chmod` is used.

```
chmod u+w output
```

In the above example the *u*ser (owner) of the file output gets granted write permissions on it.

Additive/Subtractive Model

As shown above the entity you want to grant/revoke permissions is selected via its initial:

- u: user/owner
- g: group
- o: other
- a: all

You can use the following operators to assign/revoke permissions

- +: add permission
- -: revoke permission
- =: set permissions

Some examples:

- `g-r+w` the same as `g-r,g+w` -> revokes read permission, grants write permission
- `ou=g` -> grants other and user same rights as group
- see the man page line 220 for more

Number-Based Model

Imagine each flag of a rights triplet gets assigned a number:

- r: 4
- w: 2
- x: 1

Now select the permissions you want to assign and add up their numbers. This way `rw-` would result in a *6*. Now do this for each entity and you get a number like 644 for `rw-r--r--`. This way permissions can be assigned with `chmod`:

```
chmod 0644 output
```

The leading zero can be left out, it is used to express an octal value (0644 (octal) -> 0b 110 100 100 (binary)).

Exercise

- Create the file `examplefile` and revoke all permissions of group and other.
- Read the man page if `chmod` to learn more about it.

Comparing File Content

This topic is based on chapter 20 - Comparing Text of the Linux Command Line. As learned in the previous chapter file content can be viewed using the commands `less` (scrollable) or `cat` (print whole file to terminal). The contents can be compared manually by viewing them with the above commands, or the command `diff` can be used. The tool gets called as follows:

```
diff file1 file2
```

The output will highlight the differences between the files:

```
3c3
< an exemplary line
---
> a difference
```

The differing line is displayed above.

Exercise

Compare the files *testfile1* und *testfile2*. In which line is the difference? What is the differing text?# Manipulating File Content Using Vim The command line editor Vim can be used to manipulate the content of a file. After opening the editor using `vim file` it is in command mode. The editor has multiple modes and commands:

- INSERT mode - Manipulate the file content - Press *i* in command mode to get there
- VISUAL mode - Select text - Press *v* to get there
  - press *y* to yank/copy the selected text to a vim register
  - press *p* to paste yanked text
  - press *x* or *Del* to cut text
- VISUAL LINE mode - select lines of text - Press *Shift-v*
- VISUAL BLOCK mode - select blocks of text - Press *Strg-v*
  - select a multiline one cursor wide text and press Shift-I get into 'multicursor mode'. After pressing ESC the changes you made to the first line of the selection will be applied to all lines

The most important commands are:

- :w - write the file
- :q - close the editor
- :q! - force close the editor
- :wq - write and close
- :w !sudo tee % - write to a witeprotected file (can also be done by opening vim with sudo)

Press ESC form any mode to get back to command mode

Vimdiff

Call vim with the flag `-d` and two files as arguments to get the differences between the files highlighted.

Exercise

Remove the difference between the files testfile1 and testfile2 in testfile2 using vim.# Regular Expression Basics

Regular Expressions can be used to define patterns matching specific parts of text. This part is based on chapter 19 of The Linux Command Line.

Wildcards

To match text either literal characters or wildcards can be used. Some exemplary wildcards:

- \w - matches any word character (\W everything else)
- \d - matches digits (\D everything else)
- \s - matches spaces (\S everything else)
- [a-z1-9#] - matches the characters a, b, c ... z, 1, ... 9 and #. Any characters can be defined, some will have to be escaped.
- [^a-z] - matches any character not in a-z
- . - matches any character
- ^ - matches beginning of a line
- $ - matches end of a line
- \n - newline

Quantifiers

Besides wildcards quantifiers are also used in regexes:

- * - any count of characters 0-x
- + - at least one character
- ? - zero or one character
- {3} - three characters
- {1,3} - one to three characters

Grouping

Character sequences can be grouped, this can be used to further quantify them or to use them in find and replace actions. To group sequences but them in brackets ().

Use | between two sequences or characters to search for the one OR the other.

Example

The following regex matches the pattern of an IP Address. The dot has to be escaped to serve as literal dot, not as wildcard.

```
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
```

The website regex101.com can be used to learn more about regexes and to experiment with them

Exercise

Go to regex101.com and write a regex to match the following date pattern "Monday 07.02.2022"[a-zA-Z]{6,9} \d{2}\.\d{2}\.\d{4}# Use Regexes to Analyze texts

Vim, Less, Man and Co.

In Vim, Less or Man interfaces type /, enter a regex pattern and hit enter to search for it.

Find and Replace in Vim

To use find and replace in vim enter the following pattern in command mode:

```
# Search and replace in current line
:s/pattern/newtext/g
# Search and replace in whole file
:%s/pattern/newtext/g
```

The `g` behind the last slash is a switch, the following other switches can be used:

- g - global, go on after first match
- i - case insensitive
- I - case sensitive
- c - confirm before replacing

Grep

As mentioned before in the Search Files section Grep can be used to search for file content. Use `grep` with the `-E` flag to search in extended regex mode. Grep uses a different notation for character classes:

- \d -> [[:digit:]]
- \s -> [[:space:]]
- \w -> [[:alnum:]]

Check the grep man page for more.

Some useful flags when searching with grep:

- -i - ignore case
- -v - invert match
- -c - count
- -l - list files with matches
- -L - list files without matches
- -n - print line number together with match

Exercise

Find all occurrences of a date matching the date pattern defined earlier in the file 102_exercise using `grep`.

Compressing Files

This topic is based on chapter 18 of the Linux Command Line. While under Windows `zip` is the default compression format on Linux `gzip` and `bzip2` are used.

The Tool Gzip

Usage of `gzip`:

```
# Compressing files
gzip foo
# Decompressing files
gzip -d foo.gz
## or
gunzip foo.gz
```

Bzip is used in the same way but uses an other compression algorithm internally.

Exercise

- read the man page of `gzip` to learn more about it
- look at the size of the file *111_exercise* using `ls -lh`, compress it using either `gzip` or `bzip2` and look at the new files size. Decompress the file again.# Archiving Files The default utility to archive, compress, unpack and decompress files on Linux is `tar`.

The Tool Tar

Tar allows the user to bundle many files into one. It has built in support for `gzip` and `bzip2`

Tar has the following modes with which it can be called:

- c - create an archive
- x - extract an archive
- r - append files to end of archive
- t - list contents of the archive
- d - compare archives
- u - update files in archive with older timestamp than supplied files

The modes can be combined with other options like:

- f - specify the archives name on command line
- z - use gzip with tar
- a - auto compression, choose compression program

```
tar -cf file.tar file
```

The Tar mode always has to lead before the other options.

Examples

Creating an compressed archive:

```
tar -czf archive.tar.gz file0 file1 file2 file3
```

Unpack compressed archive:

```
tar -xf archive.tar.gz
```

Exercise

- Read tars man page to learn more about it.
- Pack the file *111_exercise* into an archive, append the file *112_exercise*. Gzip the archive.

Literature Overview

Book	Author	Description	Link
Shell-Programmierung	Jürgen Wolf	How to create Shell-Scripts properly	https://openbook.rheinwerk-verlag.de/shell_programmierung/
Linux	Johannes Plötner, Steffen Wendzel	Intdroduction to Linux as a Daily Driver, Shell Basics Chapter 6 ff.	https://openbook.rheinwerk-verlag.de/linux/
Mastering Unix Shell Scripting	Randal K. Michael	Provides exemplary shell scripts for common tasks	https://ebookcentral.proquest.com/lib/mosbach-dhbw/detail.action?docID=152713&query=Shell+administration
Wicked Cool Shell Scripts : 101 Scripts for Linux, Mac OS X, and UNIX Systems	Dave Taylor and Dave Taylor	Provides exemplary shell scripts for common tasks	https://ebookcentral.proquest.com/lib/mosbach-dhbw/detail.action?docID=273515&query=Shell+administration
Linux All-In-One for Dummies	Emmett Dulaney	Book 5 is dedicated to Linux Administration Tasks, General Basics can be read in the other books	https://ebookcentral.proquest.com/lib/mosbach-dhbw/detail.action?docID=5431318&query=Shell+administration

On a normal PC the BIOS searches all attached media for a Master Boor Record (MBR). The MBR contains Information like the bootloader and the partition table. The partition table keeps track over the partitions of the attached media. The bootloader is responsible for loading an operating system.

During the boot process the hardware is booted and the OS kernel is loaded, the started kernel then starts the *init* process which in turn reads configuration files to start corresponding services. During startup *init* goes through multiple runlevels, starting the system stepwise:

0. Halt - closing network connections, writing file buffers, unmounting files
1. Minimal System - single user, only basic system resources loaded, maintenance mode
2. Local Multiuser - no network
3. Full Multiuser - on console, network operational
4. Undefined - Can be defined by user
5. Full Multiuser GUI - full multiuser mode with graphical user interface
6. Reboot

Most systems either boot to level 3 or 5, depending on the requirement for a graphical use interface. When shutting down the runlevels are taken down till level 0 is reached.

Systemd

Modern systems use systemd rather than initd as an init deamon. Systemd uses unit dependencies to determine if a service needs an other service to run successfully before it is started. This ensures the integrity of the startup.

Exercise

Read the `systemd` manpage if you want to learn more. The topic services will be dealt with in a later chapter.# Shutting Down Systems

Linux systems can be safely shutdown with the command `shutdown` it offers a variety of flags to customize the shutdown the mod important are:

- -c - cancel a planned shutdown
- -H - the system will be halted after the shutdown
- -P - the system will be powered off after the shutdown
- -r - the system will be rebooted after shutdown

A new shutdown is planned by typing:

```
shutdown -h 10
```

The above example will schedule a shutdown of the system 10 minutes form now. To shutdown a system instantly the time `now` can be specified.

Some Frontends

The commands `halt` and `poweroff` are frontends for the command shutdown. They use `shutdown` to execute their purpose. Halt should leave the system powered on after the shutdown. Poweroff should power off the machine.

Exercise

- Read the `shutdown man page to learn more about it.

- Write a script to instantly shut down and halt your system.

Rebooting Systems

To reboot a system the commands `reboot` or `shutdown` with an additional `-r` flag can be used. Reboot will call `shutdown` to reboot the machine# Different Operating Modes of a Linux System A Linux System can be run in **Text Console Mode** and **GUI Mode** if a GUI is installed. For the pure Terminal Mode the virtual tty (teletype) terminals are used. Linux systems have 7 virtual terminals: tty1 to tty7. When booting into a system with a GUI installed some ttys are used to display the GUI, on Ubuntu tty1 shows the display manager is shown, tty2 shows the GUI for the user fist logged in. To change into terminal mode the remaining ttys can be accessed by pressing `CTRL + ALT + F[3-7]`# Boot Loader Basics This chapter is based on Chapter 27.4 of Linux - Rheinwerk and the Ubuntu Documentation

The boot loader is placed in the master boot record of a hard drive. Typically a boot loader offers multiple selectable operating systems to start. The default boot loader on modern Linux Systems is GRUB2 (GRand Unified Boot Loader2).

There are other formerly common legacy boot loaders like the Linux Loader (LiLo) or GRUB1 which are not used anymore.# Installing GRUB2 A boot loader will be installed per default when installing a new Linux system.

It can be necessary to reinstall a boot loader for troubleshooting reasons.

Chroot environment

To be able to access a system with a broken boot loader chrooting can be used. Chrooting enables to change the root directory a Linux system is running from to the root directory of an other system. The chrooting can be done from a Live system:

1. Boot the live system
2. Mount the original root partition (i.e. `sudo mount /dev/sd.. /mnt`)
3. Additionally necessary system folders need to be bound
   1. `sudo mount --bind /dev /mnt/dev`
   2. `sudo mount --bind /dev/pts /mnt/dev/pts`
   3. `sudo mount --bind /sys /mnt/sys`
   4. `sudo mount --bind /proc /mnt/proc`
   5. `sudo mount --bind /run /mnt/run`
4. Change into the origional system: `sudo chroot /mnt /bin/bash`

For more information see the Ubuntu Documentation

Simple Reinstallation

To reinstall grub for example in case of a damaged MBR do the following:

```
sudo grub-install /dev/sda # where to place the boot loader
```

By doing this the configuration files */etc/default/grub* and */etc/grub.d/** are untouched.

Complete Reinstallation

To do a complete reinstall with new configuration, etc. do the following. Before the actual installation it has to be ensured that the grub packages are installed, on Ubuntu the packages

- grub-common
- grub-pc/grub-efi
- os-prober (for detecting other operating systems on pc)

are needed. They can be installed with the `apt` utility.

To ensure a clean new installation the existing configurations can be purged with:

```
sudo apt-get purge grub-* os-prober grub-gfxpayload-lists
```

To install teh new boot loader, first install the packages listed above:

For Bios PCs:

```
sudo apt-get install grub-pc os-prober grub-gfxpayload-lists
```

For EFI PCs:

```
sudo apt update
sudo apt-get install grub-efi-amd64-signed os-prober #  grub-efi-amd64 or
grub-efi-ia32 (for 32Bit systems) also possible
```

In the masks following the installation destination of grub has to be determined.

Configuring GRUB2

There are two places GRUB2 can be configured form:

- */etc/default/grub* - Actual configuration file
- */etc/grub.d/* - Folder containing GRUB configuration files.

After altering the configuration in one of this places the program `grub-makeconfig` has to be run, to build the file */boot/grub/grub.cfg*. The program has a wrapper, `grub-update`, which calls `grub-mkconfig -o /boot/grub/grub.cfg`.

Some common configuration options are:

- *GRUB_DEFAULT=0* - default entry grub starts after a timeout

- *GRUB_TIMEOUT=0* - timeout in seconds before the default entry is started# Process Properties This chapter is based of the chapter 26 of Linux - Rheinwerk.

Basics

Processes are programs being executed. When a program is started its process in entered into the process control block of the kernel. This data structure keeps information about the process like its Process ID (PID) or the User ID (UID) of the user that started it.

Types of Processes

There are different types of processes:

- **Background Processes**: Started from the shell by putting a `&` behind a program call. The `&` sends the process to the background. Background processes can be listed with `jobs` (`jobs -l` for more details) and brought to the foreground with `fg %<job-id>`. `Ctrl-Z` stops a running job and keeps it listed in the jobs table, the stopped job can be continued by typing `bg %<JID>` for background or `fg %<JID>` for continuing it in the foreground.
- **Daemons**: Daemons work in the Background too, they are used to fulfill tasks, that need no direct control, i.e. an Email- or Webserver. Daemons are normally started during the boot process.

Properties of Tasks

Using the tool `top` a taskmanager-like interface will appear in your bash.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
COMMAND
  297 friedri+  20   0   10044   5068   3360 S   0.0   0.0   0:00.18 bash
```

In the above example an exemplary entry taken from the top command can be seen. Top shows the main properties of an process:

- **PID**: The Process ID of the shown process
- **USER**: The User who started the process
- **PR**: Priority of the Process, Processes with higher priority get more CPU time.
- **NI**: Nice value of the process, affects priority of the process
- **VIRT**: Total amount of virtual memory consumed by the process (memory in RAM and memory swapped to disk)
- **RES**: Actual RAM memory usage
- **SHR**: Amount of memory chared with other processes
- **%MEM**: Memory used in relation to total RAM memory

Exercise

Start a background process with the command `sleep 300`, list the background jobs and bring the created job to the foreground.

Manage Processes

Processes can be controlled by signals sent via the terminal.

List Processes

To get an interactive view over the current processes `top` can be used. The tool `ps` gives a static list of the processes of the current session. To see every process on the system the flag `-e` can be used: `ps -e` The tool `pstree` shows processes and their parents in a tree-like view. The tool `pidof` can be used to find PIDs of a program if it has been started as a process.

Send signals to processes

The tool `kill` is used to send signals to processes, some useful signals are listed below:

- **Kernel Signals** - The receiver of this signals is the kernel
  - 9 - SIGKILL/KILL - The process is killed by the kernel
  - 19 - SIGSTOP/STOP - The process will be stopped/paused
  - 18 - SIGCONT/CONT - A stopped/paused process will continued
- **Interceptable Signals** - This signals can be intercepted by the process
  - 1 - SIGHUP/HUP - The process is ordered to halt and restart
  - 14 - SIGALRM/ALARM - Notifies about an alarm set with the syscall alarm()
  - 15 - SIGTERM/TERM - Notifies the program to gracefully terminate itself

Signals can be sent like the following:

```
kill -<Signal ID> <PID>
kill -<Signal Name> <PID>
kill <PID> # Kill alone sent SIGTERM -> kill -SIGTERM <PID>
```

To let the process 500 be killed by the kernel we would send `kill -9 500`.

Exercise

Create a process by sending the program `sleep 300` to the background. Kill the PID displayed when sending it to the background via the kernel.# Logfiles This chapter is based on chapter 14.4 of Linux - Rheinwerk and the Ubuntu Documentation Per default the system logs are logged to the directory */var/log* on Linux systems. The folder contains logs created by the system and installed services. Some of the most important logs are listed below:

- **auth.log** *(Debian)* or **secure** *(RedHat)* - Log of login attempts
- **dpkg** and **apt/history.log** - Log of the package management in Ubuntu
- **dmsg** and **kern.log** - Kernel messages (*dmsg* shows the last while *kern.log* shows older messages too)
- **daemon.log** - Installed services may create their own log files or log into *daemon.log*
- **syslog** *(Debian)* or **messages** *(RedHat)* - Global system activities are logged here
- **cron** - Log of scheduled jobs (more on this topic later)

View Log Files

Most log files can be viewed with the standard text viewer tools like:

- `cat` - print whole file to terminal
- `less` - Scroll through a file - `Shift-G` -> End of file; `gg` -> Start of file; arrow keys or mouse wheel to move cursor; `/` -> Search for Pattern
- `tail` - View only last few lines of a file

The tool `watch` can be used to execute a program periodically, thus by calling `tail <logfile>` with watch the view will be updated as new logs come in. A similar behavior can be achieved with the command `less -f` which will try to keep scrolling when reaching the end of a file. New logs will be appended to an existing log file, so if you are looking for the freshest logs to investigate something they will be at or near the end of a log file.

Journal

On newer systems *syslog* is replaced by *journal*. The syslogs can be viewed with the command `sudo journalctl` rather than via the actual log file. The journal will be displayed with `less`.

```
Mar 08 16:46:46 server sudo[843154]: pam_unix(sudo:session): session closed
for user root
Mar 08 16:46:46 server sudo[843154]:  user : TTY=pts/8 ; PWD=/var/log ;
USER=root ; COMMAND=/usr/bin/journalctl -f
Mar 08 16:46:46 server sudo[843154]: pam_unix(sudo:session): session opened
for user root by (uid=0)
```

Above the journal log entry for opening the journal with sudo can be seen. Leading is the timestamp, followed by the hostname, then the process name with pid and the processes log message as a tail.

Exercise

Execute `echo Find Me | systemd-cat -p warning`, `systemd-cat` can be used to send log messages to the journal. Search the journal for the message you just sent.

Schedule Tasks

This chapter is based on chapter 14.3 of Linux- Rheinwerk.

Cron

To schedule tasks on Linux the utility cron is used. Cron jobs are usually scripts or commands, which get executed by the cron daemon. The jobs can be defined in various locations:

- */etc/crontab*: Global crontab for the administrator
- */var/spool/cron/crontab*: Crontabs of local users named after the users
- */etc/cron.[daily|weekly|monthly]*: Scripts can be placed here and will be executed daily, hourly or monthly

- */etc/cron.d*: Crontab files can be placed here to be executed

The Crontab Format

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
  17 *  *  *  * root    cd / && run-parts --report /etc/cron.hourly
```

In the above example it can be seen how cronjobs are timed. There are 5 fields for defining when to run the task

- Minute
- Hour
- Day of Month
- Month
- Day of Week

An asterisk `*` means, that the task will be run every unit of time (minute, hour, day, ...). A task scheduled with `17 5 * * *` will run every month, every day, on hour *5* and minute *17*. The time frame is followed by the user which the task belongs to. The entry is ended with the actual command to be run.

The Crontab Tool

The crontab file is not to be edited manually, instead the tool `crontab` is used. The following flags are useful when working with the tool:

- `-u <username>`: Edit the crontab of the specified user
- `-l`: Print the content of the corresponding crontab
- `-r`: Delete the crontab file
- `-e`: Edit the crontab file

The `at` command

The `at` command can be used to schedule a task to run once. At can be given a task with the following syntax:

```
at <time to run the task at> -f <script to run>
```

By calling `at` without the `-f` flag a task to run can be defined in an interactive mode, which can be exited with `Crtl-D.`

The time can be defined in the following patterns

- `at hh:MM`
- `at <time> + <time frame>` e.g. + 2 hours
- `at hh:mm dd.MM`

Exercise

Create a crontab entry which will write 'Coming from Crontab' into the journal (see chapter before) every 5 minutes.Run `crontab -e` enter

```
15  *  *  *  * echo 'Coming from Crontab' | systemd-cat
```

save and close the editor. For user crontabs no username is needed.# Check if a Task was Executed The cron daemon logs the execution of tasks to syslog. As a consequence the execution of the tasks can be seen in the system journal (systemd logs).

The journal can be viewed with the following command:

```
sudo journalctl
# To get to the end of the file press Shift+G
```

An example taken from here below:

~~~
 Nov 02 17:10:01 testbox CRON[2210]: (testuser) WRONG INODE INFO
(crontabs/testuser)
```# Updating Software

This chapter is based on the chapter 14.2 of Linux - Rheinwerk Publishing.

## Package Management

The default package manager on Debian based distributions like Ubuntu is
`apt`.
Apt can be used to install, update and remove software packages.
Besides installing the actual software package the package manager also
installs packages which the actual package needs as dependencies.

### Software packages

Software packages contain the binaries used to execute said software,
documentation like man pages, HTML documentation, configuration files and
in some cases scripts to start/quit the software.
Per default these packages are downloaded from remote repositories by `apt`
but these packages can also be downloaded from other sources, like for
example the website of a specific software.
~~~

~~~
### Repositories

When installing a not local package with `apt` it gets the packages from
remote repositories listed in the file */etc/apt/sources.list*.
The command `apt update` indexes the packages and versions located in the
repositories listed in */etc/apt/sources.list*.

### APT Operations

The apt default operations are:

- install: Install a new package.
- update: Update the package index.
- upgrade: Upgrade packages but do not install new dependencies or remove
old ones.
- dist-upgrade: Upgrade with resolved dependencies.
- remove: Remove an installed package and keep its data.
- purge: Remove an installed package together with its data, like
configuration files

### Examples

As `apt update` updates the index od packages and their versions located in
the remote repositories, it has to be run before each operation.
To install the `vim` editor run:

``` bash
sudo apt update
sudo apt install vim
~~~

To upgrade a specific package if installed:

```
sudo apt update
sudo apt install --only-upgrade vim
```

To update all packages on the system:

```
sudo apt update
sudo apt upgrade
```

If a software vendors offer their software directly from their website. On Debian based systems, *deb* packages need to be downloaded these can be installed by the following commands:

```
sudo apt update
sudo apt install ./Downloads/your_downloaded_package.deb
```

Exercice

- Read the man page of apt/apt-get if you want to learn more.
- Install the web server nginx and remove it completely using `apt`. # Check Resources

To check on available system resources the tool `top` can be used. Below is an exemplary output of `top`:

```
top - 15:34:28 up  1:26,  3 users,  load average: 0.50, 0.54, 0.61
Tasks: 398 total,   1 running, 397 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  0.6 sy,  0.0 ni, 98.4 id,  0.0 wa,  0.0 hi,  0.0 si,
0.0 st
MiB Mem :  15413.3 total,   9340.8 free,   3648.2 used,   2424.3 buff/cache
MiB Swap:  26913.0 total,  26913.0 free,      0.0 used.  11246.0 avail Mem
```

At the end of the top line the average load of the CPU is displayed in the last minute, the last five minutes and the last fifteen minutes. In the next line the CPU loads on each core are shown in percent, followed by the usage of the physical memory, followed by the usage of the usage of the swap memory.

The tool `htop` offers a human readable way of displaying the same statistics, it can be installed from the package repositories.

Exercise

Install the tool `htop` with `apt install` and display the available resources.# Check on Processes

The tool `htop` can also be used to view running processes and their resource usage, press `F4` to filter for a process you are searching.

Exercise

Open `htop` and search for the process *systemd*.

Changing Kernel Runtime Parameters

This chapter is based of the RedHat documentation

Kernel Runtime Parameters

Kernel Runtime Parameters can be changed at runtime, they can be addressed via the command `sysctl` and the configuration files and folders */etc/sysctl.conf/ /etc/sysctl.d/*.

The configurable variables are divided into classes, on Ubuntu systems the following Classes are present:

- abi - Execution domains and personalities
- debug - Kernel debugging interfaces
- dev - Device-specific information
- fs - Global and specific file system tunables
- kernel - Global kernel tunables

- net - Network tunables
- user - User Namespace limits
- vm - Tuning and management of memory, buffers, and cache
- *List taken from RedHat documentation*

A full list of variables can be viewed with the `sysctl` command and the `-a` flag: `sysctl -a`.

Configure Parameters Temporally

To alter the value of a kernel runtime parameter temporally use sysctl like the following:

```
sysctl <class>.<parameter>=<value>
```

Configure Parameters Permanently

To alter a runtime parameter permanently redirect the output to of the above command to be appended to the configuration file */etc/sysctl.conf*:

```
sysctl <class>.<parameter>=<value> >> /etc/sysctl.conf
```

Alternatively the configuration files in */etc/sysctl.d/* can be modified manually. Either create a new file or modify an existing one in */etc/sysctl.d/*, the files have to contain one value assignment per line:

```
<class0>.<parameter0>=<value0>
<class1>.<parameter1>=<value1>
...
<classN>.<parameterN>=<valueN>
```

Example

The hostname of your computer can be changed with the variable `kernel.hostname`. A documentation of runtime parameters can be found here

Exercice

- Find out the current hostname of your host.
- Change the hostname to 'dhbw-testhost'

Shell Scripts

This chapter is based on the chapter 11 of Linux - Rheinwerk Verlag. Shell scripts make it possible to write down complex tasks into scripts. These scripts can be used to automate management and orchestration.

Hello World!

Shell scripts typically start with a shebang or hashbang:

```
#!/bin/bash
...
```

The shebang tells the calling shell which binary to execute the script with. A script can contain all commands that can be executed by the shell by default. An exemplary script could look like this:

```
#!/bin/bash

echo HelloWorld!
```

Comments

Comments can be started with a `#`:

```
# this is a comment
echo foo # this is a comment too
```

Exercise

Write a script which echoes your name and put in a comment describing what it does.

Variables

Variables are created and set by defining its name and assigning it a value: Note that no spaces are allowed before and after the equal sign.

```
variablename="variablevalue"
```

Variables types can be:

- Strings - as shown above
- Characters - 'c'
- Numbers - 42

To access the value of a set variable put the `$` sign in front of it:

```
echo $variablename
```

The result of a function call can be assigned to a variable by putting the call in braces preceded by a `$` sign:

```
var=$(echo foo)
echo $var
```

Arrays

Arrays can be created by putting values in braces:

```
int_array = (1 2 3 4)
```

Specific values can be set by referencing them by their index:

```
int_array[1] = 10
```

The values can be accessed by putting the variable and the whished index into curly braces:

```
echo ${int_array[1]} # will output 10
```

Exercise

Write a script, which saves the count of files in the current directory into the variable `fileCount` and print the variable to the terminal.# Conditionals Conditionals can be used to execute tasks if a condition is met. A conditional block is started with `if` and closed with `fi`.

```
var=5
if [ var = 5 ]
then
    echo var is 5
else
    echo var is not 5
fi
```

An overview of bash test and conditional operators, which can be used for conditionals can be found on this cheatsheet and on the bash man page.

Examples for other binary operators that can be used in comparisons are:

- `-eq` - equal
- `-ne` - not equal
- `-le` - less or equal

- `-lt` - less than
- `-ge` - greater or equal
- `-gt` - greater than

Exercise

Write a script, which checks the count of files in the folder it is executed and echoes 'Too much files!' if there are 5 or more files in the folder.# Loops

for

The for loop in bash can be used to iterate over arrays:

```
for var in (1 2 3 4 5)
do
    echo var
done
```

With for it is also possible to iterate over the files in directory:

```
# This will print the filenames of all files in the current folder
for file in *.txt
    echo $datei
done
```

while

The while loop can be used to execute code while a condition is satisfied:

```
var=5
while [ var != 0 ]
do
    echo var
    var=var-1
done
# will print 5 to 1 to the terminal
```

An endless while loop can be created by taking `1` (true) as a condition or by replacing the condition with `:`, see below:

```
while [ 1 ]
    echo foo
done
## is the same as
while :
    echo foo
done
```

until

Until loops till a value isn't satisfied anymore. It can be compared to a negative while loop `while [ !0 ]` is the same as `until[ 0 ]`.

Exercise

To put the learned together, write a script, which endlessly loops and echoes 'Too much files in here!' in case there are 5 or more files in the directory it is executed in.# Manage Startup Services The automatic start of services on system startup can be managed with the tool `systemctl`.

Systemctl allows to start, stop, check, enable and disable services. An enabled service will start on startup, a disabled one will not. Systemctl is used like this:

```
systemctl <operation> <service>
```

With some of the most common operations being the following:

- start
- stop
- status - *for checking the status of the service*
- enable
- disable

All operations except the `status` will require root privileges.

Checking a Service

Checking the status of a service will bring up the following view:

```
crond.service - Command Scheduler
     Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor
preset: enabled)
     Active: active (running) since Thu 2022-04-07 12:31:44 CEST; 1h 20min
ago
   Main PID: 1399 (crond)
      Tasks: 1 (limit: 18355)
     Memory: 1.4M
        CPU: 94ms
     CGroup: /system.slice/crond.service
             └─1399 /usr/sbin/crond -n

Apr 07 12:31:44 fedora crond[1399]: (CRON) INFO (running with inotify
support)
```

The second value (`enabled`) in the loaded field indicates whether the service is enabled to start on startup. Below the services statistics its log entries are displayed.

Example

To disable the cron daemon, which is responsible for executing the following command:

```
sudo systemctl disable cron
```

Exercise

- Read the `systemctl` manpage to learn more.
- Restart the cron service with `systemctl` and check its status to ensure it restarted properly. *Info:* Using this feature can be useful to restart a service after altering its configuration.# Configuring Services Resources for this section are the Suse Documentation, a Medium Article, a StackOverflow Contribution and a Python Documentation.

Following the everything is a file directive of Linux/Unix services are defined in files. These files are located in */etc/systemd/system*.

Example Service

To demonstrate the configuration of systemd services in this section a python server service will be created.

Python Server

Python offers a online way of starting a webserver, by loading the module *http.server*:

```
python -m http.server
```

To display something create a folder */home/your-user/www* and create a file *index.html* with the following content in it.

```
<h1>I am a Simple Python Server!</h1>
```

The Service File

Lets create a simple web server service file:

```
[Unit]
Description=Simple Python Service
After=network.target
[Service]
```

```
WorkingDirectory=/home/liam/Documents/www
ExecStart=/usr/bin/env python3 -m http.server

[Install]
WantedBy=multi-user.target
```

In the first section *[Default]* the service is described (its name is based on the service filename). Systemd maintains an order when starting services, to start services other services depend on before those depending on them. This dependencies are defined via the *After* field, which will let the server service start after the *network.target* service.

In the next section *[Service]* details for executing the service are defined. *WorkingDirectory* sets the directory the service will be executed in. *ExecStart* defines the actual executable executed.

In the field *WantedBy* in the *[Install]* section defines the system state systemd tries to start the service in (see 011_booting_systems). The value `multi-user.target` closely matches the system state 3 (Full Multiuser - on console, network operational).# SELinux This chapter is based on the Fedora documentation [1], [2], [3], [4]

Mandatory Access Control

SELinux acts as an additional security layer (Mandatory Access Control (MAC)), added on top of the standard access policy (users, groups, permissions) called Discretionary Access Control (DAC), with DAC being evaluated before SELinux rules. It allows a more granular way of defining access controls. In SELinux every process and resource has an label called SELinux context.

These contexts are used to define how processes are allowed to interact with each other and their environment. SELinux provides a whitelist based model, meaning things have to be explicitly allowed and are blocked by default.

SELinux is the default MAC tool for **RedHat** based systems.

SELinux Contexts

SELinux Contexts have the following fields:

- user
- role
- type
- security level

The SELinux context type is the important field when concerned with interactions between processes.

Examples

Best practice for context types is for them to end on `_t` Examples of SELinux types are:

- httpd_t - webserver context
- tmp_t - context for files in */tmp*

As typically the webserver Apache is configured to be allowed access to files tagged with `httpd_t` it can access contents necessary for serving a web page or service. Per default Apache has no access rights to the `tmp_t` type preventing it from access there. For more examples see the Fedora documentation.

View File Contexts

To view contexts assigned to files the command `ls` can be used with the flag `-Z` producing the following exemplary output:

```
unconfined_u:object_r:user_home_t:s0 01_basic_commands
```

Change File Contexts

File contexts can be altered with the command `chcon` is used:

```
chcon -t <new_context_type> <file>
```

When used with the `-t` switch, like above, the context type of a file is changed. The `-R` flag can be used to alter the contexts of a directory and its contents recursively.

View Process Contexts

To view the context of a process `ps` can be used with the flags `-eZ`:

```
system_u:system_r:crond_t:s0-s0:c0.c1023 1302 ?  00:00:00 crond
```

This reveals that the process crond has the context type crond_t giving it access to cron relevant files.

Exercise

Read the linked documentations if you want to learn more.

Apparmor

Apparmor is an other MAC tool, it is commonly used on **Debain** based systems. This section is based on the SUSE Documentation

AppArmor Profiles

AppArmor policies are defined in sol called profiles. The profiles are typically found in */etc/apparmor.d* An exemplary AppArmor Profile can be seen below:

```
## App Armor Profiles offer the possibility to include extra files.
## In this case a file containing variable definitions is included.
#include <tunables/global>

## naming the application to confine and starting the block defining it
/usr/bin/foo {
   ## Inclusion of an Abstraction
   ## Abstractions offer predefined templates for different use cases.
   #include <abstractions/base>

   ## Capability definition:
   capability setgid,
   ## Definition of allowed network access:
   network inet tcp,

   ## Allowing linking from /etc/sysconfig/foo to /etc/foo.conf:
   link /etc/sysconfig/foo -> /etc/foo.conf,

   ## Block of path access definitions:
   ## Unconfined execution mode:
   /bin/mount            ux,
   ## Use {} to define multiple possible options - in this case either an
empty string or 'u':
   /dev/{,u}random     r,
   ## Use * as a wildcard for any combination of characters :
   /etc/foo/*            r,
   /lib/ld-*.so*         mr,
   ## Use [] for defining ranges:
   /proc/[0-9]**         r,
   ## Usage of a variable:
   /@{HOME}/.foo_file   rw,
   ## Owner conditional -> applying this rule only to the owner of the
file.
   ## 'other' can be used to apply the rule to everybody but the user.
   owner /shared/foo/** rw,
   ## Execute local profile for this file
   ## See below for local profile definition
   /usr/bin/foobar       Cx,
   ## Use the profile bin_generic for executing this role
   /bin/**               Px -> bin_generic

   ## Definition of local profile
   profile /usr/bin/foobar {
      /bin/bash          rmix,
      /bin/cat           rmix,
      /bin/more          rmix,
      /var/log/foobar*   rwl,
      /etc/foobar        r,
   }
}
```

Taken from here.

The following access controls can be used:

- r - read mode - a program can read a resource
- w - write mode (cannot be specified together with a) - a program has full write access to a resource
- a - append mode (cannot be specified together with w) - a program can append to file, cannot alter the file in an other way
- k - file locking mode - a program has the ability to take file locks
- l - link mode - a program has access to hard links
- link -> - link pair rule (cannot be combined with any other)
- m - memory map as executable
- Ux - unconfined execution
- Px - use external profile for this
- Cx - use local profile for this

Rules can be preceded by `allow` (default -> has not to be written) or `deny` to apply the corresponding action.

Exercise

- Write an AppArmor Profile for the imaginary application *foobar* granting it write access to */tmp*. Include the `base` abstraction.
- Read into the */etc/apparmor.d/abstractions* file to understand what is does.

Identify Package Belongings

This section is based on the Ubuntu documentation and a LinuxAudit Article. On systems using `apt` as a package manager the tool `apt-file` can be used to determine which package a file belongs to.

Commands

The tool knows the following commands:

- update - updates the index
- search - searches all known packages for a file matching *pattern*
- list - list all files of the package matching *pattern*
- purge - purges the cache

Example

The below example lists all files belonging to vim:

```
apt-file list vim
```

On RedHat

On RedHat the package manager `rpm` has the capabilities to do the same operations with the following flags:

- `-qf <file>` - list packages using file
- `-ql <package>` - list files in package

Example

The below example does the same as the one above:

```
rpm -ql vim
```

Exercise

- Install the tool `apt-file`
- Update the `apt-file` index
- Search for the package containing */bin/ls*# User Creation and Modification This topic is based on the chapter 13.2 of the book Linux from Rheinwerk-Verlag

Creating Users

There are two available options to create users on the command line: `useradd` and `adduser`. Useradd takes the information for creation as parameters, while adduser asks for them. Adduser is a frontend for useradd, which feeds useradd the information entered.

Exemplary `useradd` call:

```
useradd dummy
```

This will add the user dummy to the system. As no parameters were specified it will have no home directory and password.

Change or Create a Password

The command `passwd` can be used to change the password of an user or create one if not existing.

```
passwd dummy
```

You will be asked for a new password for the user dummy

Exercise

- Look at the man pages of `useradd`, `adduser` and `passwd` to learn more
- Create a user *dummy* with the password *secret* (hint: adding users requires root privileges)# Working with Users In this section it will be explained how to modify and delete users.

Modify Users

Users can be modified with the command `usermod`. Usermod supplies the following basic flags:

- -l: change username
- -d: change home dir
    - -dm: move current home dir
- -L: lock user
- -U: unlock user

```
usermod -d /home/dummy dummy
```

This will assign the user dummy the homedir */home/dummy*

Delete Users

To delete users the commands `userdel` or `deluser` are used:

```
userdel dummy
deluser dummy
```

Deluser offers the possibility to remove the home directory of the user (`--remove-home`), all files of the user (`--remove-all-files`), and backup the users files as an archive (`--backup`) Userdel offers the possibility to remove the users home directory (`-r`).s

Exercise

- Read the manpages of `usermod` to learn more
- Change the username from *dummy* to *buster*
- Delete the user *buster*# Group Creation and Modification

Create Groups

Use `groupadd` or `addgroup` to create a new group. Analogous to the user creation, `addgroup` is a interactive frontend for `addgroup`.

```
addgroup new_group
```

Modify Groups

To modify groups use the `groupmod` command:

```
groupmod -n exemplary_group new_group
```

The example above changes the groupname from *exemplary_group* to *new_group*

Delete Groups

To delete groups use the `groupdel` command:

```
groupdel exemplary_group
```

Exercise

- Read the manpages for `groupadd`/`addgroup`, `groupmod` and `groupdel` if you want to learn more
- Create a group `new_group` and rename it to `dummy_group`# Manage group Memberships To add an user to a group use `usermod`. Usermod offers three options of modifying a Users groups.
- -g: modifies users main group
- -G: define list of Groups the user is part of
- -Ga: append a Group to the list

Exercise

- Create a user *dummy* and add him to the group *dummy_group*# User Creation and Modification This topic is based on the chapter 13.2 of the book Linux from Rheinwerk-Verlag

Creating Users

There are two available options to create users on the command line: `useradd` and `adduser`. Useradd takes the information for creation as parameters, while adduser asks for them. Adduser is a frontend for useradd, which feeds useradd the information entered.

Exemplary `useradd` call:

```
useradd dummy
```

This will add the user dummy to the system. As no parameters were specified it will have no home directory and password.

Change or Create a Password

The command `passwd` can be used to change the password of an user or create one if not existing.

```
passwd dummy
```

You will be asked for a new password for the user dummy

Exercise

- Look at the man pages of `useradd`, `adduser` and `passwd` to learn more
- Create a user *dummy* with the password *secret* (hint: adding users requires root privileges)# Working with Users In this section it will be explained how to modify and delete users.

Modify Users

Users can be modified with the command `usermod`. Usermod supplies the following basic flags:

- -l: change username
- -d: change home dir
    - -dm: move current home dir
- -L: lock user
- -U: unlock user

```
usermod -d /home/dummy dummy
```

This will assign the user dummy the homedir */home/dummy*

Delete Users

To delete users the commands `userdel` or `deluser` are used:

```
userdel dummy
deluser dummy
```

Deluser offers the possibility to remove the home directory of the user (`--remove-home`), all files of the user (`--remove-all-files`), and backup the users files as an archive (`--backup`) Userdel offers the possibility to remove the users home directory (`-r`).s

Exercise

- Read the manpages of `usermod` to learn more
- Change the username from *dummy* to *buster*
- Delete the user *buster*# Group Creation and Modification

Create Groups

Use `groupadd` or `addgroup` to create a new group. Analogous to the user creation, `addgroup` is a interactive frontend for `addgroup`.

```
addgroup new_group
```

Modify Groups

To modify groups use the `groupmod` command:

```
groupmod -n exemplary_group new_group
```

The example above changes the groupname from *exemplary_group* to *new_group*

Delete Groups

To delete groups use the `groupdel` command:

```
groupdel exemplary_group
```

Exercise

- Read the manpages for `groupadd`/`addgroup`, `groupmod` and `groupdel` if you want to learn more
- Create a group `new_group` and rename it to `dummy_group`# Manage group Memberships To add an user to a group use `usermod`. Usermod offers three options of modifying a Users groups.
- -g: modifies users main group
- -G: define list of Groups the user is part of
- -Ga: append a Group to the list

Exercise

- Create a user *dummy* and add him to the group *dummy_group*insert the following into ~/.bash_profile

```
echo Hello Liam
```

Replace Liam with your name.# Configuring the Local User Environment This Section is based of an article by RedHat and Chapter 07 of Linux - Rheinwerk Verlag.

The user environment is configured via files, in a default configuration they are:

- *.bash_logout*
- *.bash_profile* (sometimes *.profile*)
- *.bashrc*

Templates for these files can be found in */etc/skel*. Everything contained in */etc/skel* will be copied into a newly created users home directory. The binary path is defined in this file. To alter the template for the user environment, alter the files in */etc/skel*.

.bash_profile

This file gets executed for login shells. In a default configuration it calls *.bashrc*. Define your additional environment variables here.

.bashrc

This file gets executed each time you open a new bash while already being logged in.

Useful Environment Variables

Listed below are some commonly used environment variables and their usage:

- PATH - String that contains all directories bash looks for executables (Add dir with `export PATH=$PATH:<new dir>`)
- SHELL - path to the executable of your preferred shell (usually */bin/bash*)
- EDITOR - default editor
- PS1 - your default bash prompt (new ones can be created in online editors)

.bash_logout

Gets executed when a bash is closed if it exists.

Exercise

Make the bash say Hello <your name here> when you login.

Configuring the Global Environment

This Section is based of an article by RedHat.

There are three files to configure your environment globally:

- /etc/profile
- /etc/bashrc
- /etc/environment

/etc/profile

This file gets executed each time a user logs in. It calls all scripts residing in */etc/profile.d*

/etc/bashrc

This file (on Debian /etc/bash.bashrc) gets executed for each user when opening a new bash while being logged in.

/etc/environment

This file is read on each login. The difference between the other two is, that it is no script but rather contains variable definitions directly. Variables like *http_proxy* for Proxy configuration can be placed here:

```
http_proxy = "https://proxyurl.tld:port"
```

Exercise

Place a script in */etc/profile.d* that greets each new login with "Hi there from <your name>".Place a file *greeting.sh* with the following content in */etc/profile.d*.

```
#!/bin/bash
echo "Hi there from Liam"
```

Replacing "Liam" with your name.# Defining User Resource Limits This section is based of the Ubuntu Server Cookbook.

The command `ulimit` controls the system resources a user is allowed to use. A process started by a user has to obey the resource limits set for this user. The limits are defined in the file */etc/security/limits.conf*.

Resources can be defined for the *domains*: user (syntax: username) and group (syntax: @groupname). Limits are differentiated into two *types* hard and soft limits. While soft limits can be altered by the user, hard limits are defined by the administrator and are a cap for the soft limits.-

The following resource *items* can be limited (taken from */etc/security/limits.conf*):

- core - limits the core file size (KB)
- data - max data size (KB)
- fsize - maximum filesize (KB)
- memlock - max locked-in-memory address space (KB)
- nofile - max number of open file descriptors
- rss - max resident set size (KB)
- stack - max stack size (KB)
- cpu - max CPU time (MIN)
- nproc - max number of processes
- as - address space limit (KB)
- maxlogins - max number of concurrent logins for this user
- maxsyslogins - max number of logins on the system
- priority - the priority to run user process with
- locks - max number of file locks the user can hold
- sigpending - max number of pending signals
- msgqueue - max memory used by POSIX message queues (bytes)
- nice - max nice priority allowed to raise to values: [-20, 19]
- rtprio - max realtime priority

The syntax for defining new limits is: `<domain> <type> <item> <value>` For example:

```
@student        hard    nproc           40
```

The above example grants a member of the group student a maximum count of 40 processes.

Exercise

Write a hard limit for members of the group student, to limit their logins to 2

~~~
@student        hard    maxlogins       2
```# User Privileges
This chapter is based on chapter 13.4 of the book Linux - Rheinwerk Verlag
User Types on Linux are differentiated into **root** and **standard**
users.
Principally the root user has unlimited access to the system, while a
standard user only can access files which he has permissions to.

## Super User Do
The command `sudo` allows standard users to execute commands as root:

``` bash
sudo vim /etc/sudoers
~~~

In the above command the file */etc/sudoers* gets opened with vim. Per default only the root user has access to this file. This file regulates the access to the sudo command. Per default users who are in the group *sudo* (Debian)/*wheel* (RedHat) have access to run sudo.

Exercise

Create a file in */root*# Access to the Root User The root account is meant to be used for single tasks, which need elevated rights, **not** as a daily user for the system administrator.

During the Setup Process of a new system the password for the root user either gets set or disabled.

To change to the root user from a standard user either use `su root` if a password is set and known or `sudo su` to use the sudoer privileges of the standard user if present.

Exercise

Switch to the root user and insert the text: 'Written As Root' to the file.# Pluggable Autentication Modules This chapter is based on the Red Hat Documentation. Pluggable Authentication Modules (PAM) central way to authenticate. PAMs offer different types of authentication (Kerberos, NIS, local filesystem) and are therefor called pluggable.

Because of its pluggable nature PAM gives administrators great flexibility on configuring authentication on systems. PAM can be used by a wide range of applications.

On RedHat Systems the tools `authconfig` or `authselect` are recommended to configure PAM, on Ubuntu the configuration files are edited directly.

The configurations files are `/etc/pam.conf` or any file in `/etc/pam.d` (if `/etc/pam.d` exists `/etc/pam.conf` will be ignored). Each of these files should consist of multiple rule lines. Each rule-line has a syntax as follows:

```
service type control module-path module-arguments
```

(For files in the `/etc/pam.d` the service field is omitted, instead the name of the file references the service)

```
auth        substack     system-auth
auth        include      postlogin
account     required     pam_nologin.so
...
```

Above is an excerpt of the content of the */etc/pam.d/login* file. In the first line the **auth** content of the file */etc/pam.d/system-auth* is evalueted as an substack, meaning the statements will be checked, but if one fails only the substack fails. In the second line the **auth** content of the file */etc/pam.d/postlogin* gets included, meaning the auth lines in the file are evaluated, if one fails, the original stack fails too. In the third line the pam module *pam_nologin.so* gets included as required, meaning it succeeding is necessary to continue the stack, **pam_nologin.so** fails if the file */etc/nologin* exists.

The auth type calls are used to identify the user. The account type is used to grant/restrict access based on defined criteria, in the case of the *pam_noloing.so* module the presence of the *nologin* file.

Exercise

- Read the manpage of *pam.conf* to learn more about types, controls and the general definition of the rules.# Basic Network Configuration This section is based on an LinuxConfig article, a Stackoverflow post and the Ubuntu Wiki On Debian network interfaces are defined in the */etc/network/interfaces* file. This file includes the files in */etc/network/interfaces.d*.

WLAN

To configure a wlan interface a file in */etc/network/interfaces.d* has to be created with the following contents:

```
auto wlan0
iface wlan0 inet dhcp
wpa-essid <wlan-ssid>
wpa-psk <wpa-key>
```

The interface `wlan0` has to be replaced with the actual interface, which can be determined with the command `ip a`:

```
2: enp2s0f0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 ...
...
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
...
```

The above example shows an output of `ip a`, the wlan interface `wlp3s0` and the ethernet interface `enp2s0f0` can be identified.

Ethernet

To configure a ethernet interface use the following content:

```
auto eth0
iface eth0 inet dhcp
```

WLAN as well as ethernet are configured to retrieve their IP address via dhcp. The system has to be restarted to apply the settings.

The `auto` tag ensures the interfaces are brought up on startup.

The Networking Service

The default networking service on Ubuntu is `systemd-networkd`. Systemctl can be used to start, stop, enable and disable it and to check its status as seen in the file **111_manage_startup_services**.

Exercise

- View the network devices on your machine.# Hostname Resolution This chapter is based on the freedesktop documentation and the Ionos Guide.

Hostnames

Hostnames or domain names are a human readable identifier to identify network devices. A hostname points to the IP of a device. To be able to connect to a device its hostname has to be resolved to its IP address. This is done by hostname resolution. Host machines which are in a network can be addressed by the resolved IP address.

Hostnames offer the advantage that they stay the same, while the IP address can change. Which means they can serve as a reliable identifier for a device or service.

Static Resolution

On Linux systems static domain name resolution can be defined in the file */etc/hosts*:

```
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
```

The file points the localhost domains to the localhost ip addresses. An entry for *example.com* would look like this:

```
93.184.216.34 example.com
```

Dynamic Resolution

As the IP of a network target might change, dynamic hostname resolution offers a way to connect to the device anyway, without knowing its exact IP. To do this a query is sent to a DNS server, the Server resolves the hostname to an IP and sends it back to the host. To check the currently used nameservers `resolvectl status` can be used:

```
...
DNS Servers: 192.168.178.1
...
```

To alter the resolve configuration written in */etc/resolve.conf* the tool `resolvconf` has to be used, as the file is managed. After installing the tool the file */etc/resolvconf/resolv.conf.d/head* can be altered. An additional name server can be added to this file. The format of the entry would be:

```
nameserver <IP>
```

After editing the file the `resolvconf` has to be run with the flag `-u` to update the */etc/resolv.conf* file.

Exercise

- Install `resolvconf`
- Add the Google DNS Server *8.8.8.8* to the systems configuration

Packet Filtering

This section is based on chapter 32.6 of Linux - Rheinwerk. On Linux packet filtering mechanisms are built into the Kernel. They can be configured with the `iptables` tool.

Iptable Tables

Iptables has various functionalities, besides package filtering it also offers NAT possibilities. These functionalities are configured in different tables. The table to edit is specified by the `-t` flag. The default table is the `filter` table responsible for the package filtering configuration.

Iptable Chains

Iptables sends packages down different chains based on rules. These chains determine what happens with the package.

There are several predefined chains but it is also possible to create custom ones with the `iptables -N` command.

The following filtering chains are predefined:

- `INPUT` - Contains rules for packages designated for the local client
- `OUTPUT` - Contains rules applied to all outgoing packages
- `FORWARD` - Contains riles for all to be forwarded packages

Iptables offers the following operations on chains:

- `-N <chain>` - Create new chain
- `-X <chain>` - Delete Chain
- `-E <old> <new>` - Rename a chain
- `-L <chain>` - List rules of specified chain
- `-F <chain>` - Flushes chain - Deletes all rules in chain

Iptable rules

The iptable chains are made up of a collection of rules. Rules are evaluated from top to bottom, of one rule matches it will be applied to the package and the processing will be halted. These are the most common rule operations:

- `-A <chain> <rule>` - add rule to an existing chain
- `-I <chain> <position> <rule>` - insert rule into chain on specified index; when no position specified, the rule will be inserted at index 0
- `-R <chain> <position> <rule>` - Remove rule at specific index
- `-D <chain> <rule-number/rule-name>` - Deletes rule matching rule-number or name

Rules are made up of filtering expressions and destinations. The destinations can be seen above in the listing below:

The following chains are possible:

- ACCEPT - The package will be allowed
- DROP - The package will discarded
- QUEUE - Forward the package to userspace (if supported by kernel)
- RETURN - Return to calling chain (if in user defined chain)
- LOG - Log into the kernel log
- REJECT - Same behavior as drop, sends back *Port Unreachable* error message.
- user defined chain - Jump into a user defined chain

The most common filtering expressions can be seen below, they can be negated with a `!`:

- `-s <source>` - Filters for the source of a package
- `-d <destination>` - Filters for the destination of a package
- `-i <interface>` - Filters for the interface that received the package
- `-o <interface>` - Filters for the interface to output the package
- `-p` - Filters for the protocol used to transmit the package (most common are TCP or UDP)

Based on the used protocol filtering more filter modules will be loaded. Modules can also be loaded via the `-m <module>` flag. See the iptables manpage for more information.

When creating a rule the flag `-j` is used to *jump* to destination.

Network destinations can be applied in the following forms:

- Network Name
- Hostname
- Network Address + Subnet Mask
- IP-Address

Persistance

Updates made via iptables will be temporary and reset after a reboot. To make the changes persistent the command `iptables-save` and redirecting the output to */etc/iptables/rules.v4* (for Debian) will make the changes persistent between reboots:

```
# Debian
iptables-save > /etc/iptables/rules.v4
# RedHat
iptables-save > /etc/sysconfig/iptables
```

Taken from here.

Example

An exemplary rule can look like this:

```
iptables -A INPUT -p tcp -j TCP
```

This will add a call to the custom chain `TCP` for all incoming TCP packages.

Exercise

Create a rule for all Incoming packages that drops the packages coming from the IP-Address `192.168.178.150`

Static IP Routing

This section is based on the RedHat documentation and a Linux Config Article.

Show Current Routing Table

To show the currently routing table the command `ip route show` is used.

Temporal IP Routing

To configure a temporal IP route the command `ip route` with the following opeations is used:

- add: `ip route add <destination>/<netmask> <gateway>`

- del: `ip route del <destination>/<netmask>`
- change: `ip route change <destination>/<netmask> <gateway>`
- append: `ip route append <destination>/<netmask> <gateway>`
- replace: `ip route replace <destination>/<netmask> <gateway>`

```
# add a route to statically rout the ip address 192.0.2.1 over the address
10.0.0.1 as a hop/gateway
ip route add 192.0.2.1 via 10.0.0.1
```

The address to route also can be specified as a combination of IP address and netmask.

Permanent IP Routing

Routes created by the ip command will only be applied temporally. To make the route permanent it has to be added to the file */etc/netplan/50-cloud-init.yaml* with the following syntax:

```
network:
    ethernets:
        enp0s3:
            dhcp4: false
            addresses: [192.168.1.202/24]
            gateway4: 192.168.1.1
            nameservers:
              addresses: [8.8.8.8,8.8.4.4,192.168.1.1]
            routes:
            - to: 172.16.0.0/24
              via: 192.168.1.100
    version: 2
```

Taken from here

In the example the addresses `172.16.0.0/24` will be routed over the gateway `192.168.1.100` this applies to the interface `enp0s3`.

To apply this configuration the command `sudo netplan apply` has to be run.

Exercise

Take the above netplan example an add a route to `172.18.52.31` via `192.168.1.101`.# Time Synchronization This section os based on the German Ubuntu Documentation. To synchronize the time via network the network time protocol can be used under Linux.

Install NTP

To install the NTP package the command `sudo apt install ntp` has to be used.

Configure NTP

To configure the NTP the file `/etc/ntp.conf` has to be edited. In this file the time servers to use can be specified:

```
server 0.de.pool.ntp.org
server 1.de.pool.ntp.org
server 2.de.pool.ntp.org
server 3.de.pool.ntp.org
```

Above the German time pool servers are configured. NTP servers can be searched here. After a restart the service will use the newly configured servers. It is recommended to use multiple time servers. The timeserver with the lowest offset to **Stratum 0** (original time sources) will be used. The command `ntpq -p` can be used to check the currently configured time servers.

Exercise

- Install the NTP package
- Configure NTP to use the time server pools recommended for Croatia here.

Configure a Caching DNS Server

This section is based on a cloudflare article and a LinuxTeck guide.

Bind

The package *BIND* provides the service *named* which can serve as a caching DNS server. The caching server is able to cache queries for a certain time - TTL (Time to Live). This allows for faster response times.

Installation

On Debian Systems the package *bind9* has to be installed (`apt`). To ensure the name server starting on each startup it has to be enabled in systemd (`systemctl`).

Per default the installed service will cache DNS queries.

Configuration

To configure the *named* service the file */etc/named.conf* has to be edited. In the default configuration the DNS server accepts from localhost. To allow traffic from other systems add the statement any to the following lines:

```
listen-on port 53 { 127.0.0.1; any; };
allow-query { localhost; any; };
allow-query-cache { localhost; any; };
```

Iterative and Recursive DNS

When querying iterative the DNS request is forwarded by DNS server to the next responsible DNS server. This results in the client communicating directly to the involved DNS servers. When querying recursive the first DNS server communicates with all other DNS servers in the chain to answer the query himself.

Exercise

- Install the package *bind9*
- Configure view the configuration file */etc/named.conf*# DNS Zones Based on this Ubuntu Wiki article, the examples are taken from there.

DNS Zones are

Forward vs. Forward Lookup

In Reverse Lookup the DNS server is responsible for resolving the IP address to the hostname. In Forward Lookup the DNS server is responsible for resolving the hostname to the IP address.

Defining Zones

When configuring a new zone two files have to be created, one for the forward lookup and one for the reverse lookup. The naming convention of the files is the following:

```
# Forward Lookup
db.domainname
# Reverse Lookup
db.z.y.x
# z.y.x being parts of the IP address x.y.z (192.168.0.* -> db.0.168.192)
# note the reverse order!
```

An exemplary forward lookup file looks like this:

```
;; db.domainname
;; Forwardlookupzone für domainname
;;
$TTL 2D
@       IN      SOA     rechnername.domainname. mail.domainname. (
                        2006032201      ; Serial
                                8H      ; Refresh
                                2H      ; Retry
                                4W      ; Expire
                                3H )    ; NX (TTL Negativ Cache)

@                               IN      NS      rechnername.domainname.
                                IN      MX      10 mailserver.domainname.
                                IN      A       192.168.0.10

rechnername                     IN      A       192.168.0.10
localhost                       IN      A       127.0.0.1
rechner1                        IN      A       192.168.0.200
```

```
mailserver                      IN      A       192.168.0.201
rechner2                        IN      CNAME   mailserver
```

Forward Lookup

The time to life (`TTL`) specifies the time frame information is cached. After the `TTL` entry the `SOA` (Start of Authority resource records) entry is defined. It specifies the following values:

- *zone-origin* - `rechnername.domainname` - Fully Qualified Domain Name (FQDN) of the primary DNS server.
- *zone-contact* - `mail.domainname` - the email address of the responsible person, the `@` is substituted for a `.` (mail.domainname would be mail@domainname)
- *serial* - `2006032201` - important for versioning, typically in the format *YYYYMMDDSS* (current time) to allow secondary DNS servers to notice changes.
- The following entries should be on good default values and don't have to be changed.
- *refresh* - Refresh interval for secondary DNS servers
- *retry* - Time a secondary DNS server waits to send another query if the previous one failed.
- *expire* - Timeout time for a connection attempt of a secondary DNS server to the primary server.
- *nx* - Time frame for negative caching (caching of failed lookup attempts).

The entry types being the following:

- `NS` - **value:** FQDN of a nameserver - *primary and secondary DNS servers of domain should have an entry*
- `A` - **value:** IP address - *resolves name -> IP address*
- `CNAME` - **value:** hostname of the specified client
- `MX` - **value:** priority, hostname - *mail-server for the domain + priority -> servers with low priority will be tried first.*
- `PTR` - **value:** FQDN - *reverse lookup IP -> name (e.g. foo.example.com)*

Reverse Lookup

An exemplary reverse lookup file looks like this:

```
;; db.0.168.192
;; Reverselookupzone für domainname
;;
$TTL 2D
@       IN      SOA     rechnername.domainname. mail.domainname. (
                                2006032201      ; Serial
                                        8H      ; Refresh
                                        2H      ; Retry
                                        4W      ; Expire
                                        3H )    ; TTL Negative Cache

@       IN      NS      rechnername.domainname.

10      IN      PTR     rechnername.domainname.
```

```
200     IN      PTR     rechner1.domainname.
201     IN      PTR     rechner2.domainname.
```

Here the PTR entry type comes to use. The numbers in the first row represent the last byte of the IP address.

Exercise:

Read the Wikipedia article on DNS if you want to learn more.

students-guide-to-linux

- [x] Essential Commands 25%
    - [x] Log into local & remote graphical and text mode consoles
    - [x] Search for files
    - [x] Evaluate and compare the basic file system features and options
    - [x] Compare and manipulate file content
    - [x] Use input-output redirection (e.g. >, >>, |, 2>)
    - [x] Analyze text using basic regular expressions
    - [x] Archive, backup, compress, unpack, and uncompress files
    - [x] Create, delete, copy, and move files and directories
    - [x] Create and manage hard and soft links
    - [x] List, set, and change standard file permissions
    - [x] Read, and use system documentation
- [x] Operation of Running Systems 20%
    - [x] Boot, reboot, and shut down a system safely
    - [x] Boot or change system into different operating modes
    - [x] Install, configure and troubleshoot bootloaders
    - [x] Diagnose and manage processes
    - [x] Locate and analyze system log files
    - [x] Schedule tasks to run at a set date and time
    - [x] Verify completion of scheduled jobs
    - [x] Update software to provide required functionality and security
    - [x] Verify the integrity and availability of resources
    - [x] Verify the integrity and availability of key processes
    - [x] Change kernel runtime parameters, persistent and non-persistent
    - [x] Use scripting to automate system maintenance tasks
    - [x] Manage the startup process and services (In Services Configuration)
    - [x] List and identify SELinux/AppArmor file and process contexts
    - [x] Manage Software
    - [x] Identify the component of a Linux distribution that a file belongs to
- [x] User and Group Management 10%

    - [x] Create, delete, and modify local user accounts
    - [x] Create, delete, and modify local groups and group memberships
    - [x] Manage system-wide environment profiles
    - [x] Manage template user environment
    - [x] Configure user resource limits
    - [x] Manage user privileges
        - [x] Manage access to the root account
    - [x] Configure PAM

- [x] Networking 12%

    - [x] Configure networking and hostname resolution statically or dynamically
    - [x] Configure network services to start automatically at boot
    - [x] Implement packet filtering
    - [x] Start, stop, and check the status of network services
    - [x] Statically route IP traffic
    - [x] Synchronize time using other network peers

- [ ] Service Configuration20%

    - [ ] Configure a caching DNS server
    - [ ] Maintain a DNS zone
    - [ ] Configure email aliases
    - [ ] Configure SSH servers and clients
    - [ ] Restrict access to the HTTP proxy server
    - [ ] Configure an IMAP and IMAPS service
    - [ ] Query and modify the behavior of system services at various operating modes
    - [ ] Configure an HTTP server
    - [ ] Configure HTTP server log files
    - [ ] Configure a database server
    - [ ] Restrict access to a web page
    - [ ] Manage and configure containers
    - [ ] Manage and configure Virtual Machines

- [ ] Storage Management 13%

    - [ ] List, create, delete, and modify physical storage partitions
    - [ ] Manage and configure LVM storage
    - [ ] Create and configure encrypted storage
    - [ ] Configure systems to mount file systems at or during boot
    - [ ] Configure and manage swap space
    - [ ] Create and manage RAID devices
    - [ ] Configure systems to mount file systems on demand
    - [ ] Create, manage and diagnose advanced file system permissions
    - [ ] Setup user and group disk quotas for filesystems
    - [ ] Create and configure file systems