UNIX Introduction

Cluster Computing in Frankfurt

Anja Gerbes

Goethe University in Frankfurt/Main
Center for Scientific Computing

August 27, 2018

Center for Scientific Computing Frankfurt

Hessisches Kompetenzzentrum für Hochleistungsrechnen

Motivation

The Terminal
The Command Line
Unix
Linux
Shell Scripting
Bash

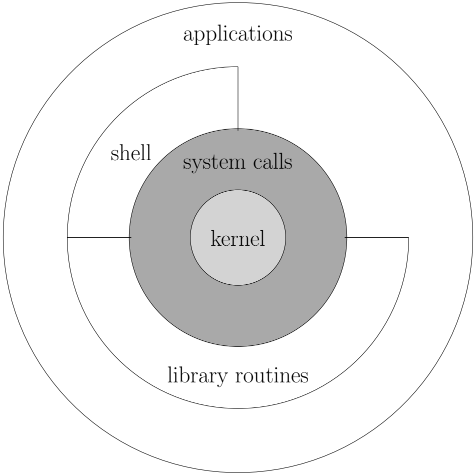
Operating System
UNIX Structure

- UNIX
  - is an operating system specification.
  - has many implementations (Linux, Mac OS X, Free BSD, etc.)
  - it controls the hardware, run programs, manage resources & communicate with other computers.
- The kernel
  - is the heart of the operating system.
  - handles memory management, input & output requests, & program scheduling.
  - interacts with hardware & most of the tasks like memory management, task scheduling & file management.
  - Users communicate with the kernel through system calls with the help of shell, libraries & other applications (e.g. Graphical User Interface)
- The shell is a command line interpreter.

Operating System
UNIX Structure

- Hardware provides basic computing resources.
  - CPU
  - RAM
  - I/O
  - PMC
- Utility Programs assists in system management & software development.
- Application Programs defines the ways in which the system resources are used to solve the computing problems of the users.
  - compilers
  - database systems
  - business programs

Operating System
UNIX Structure



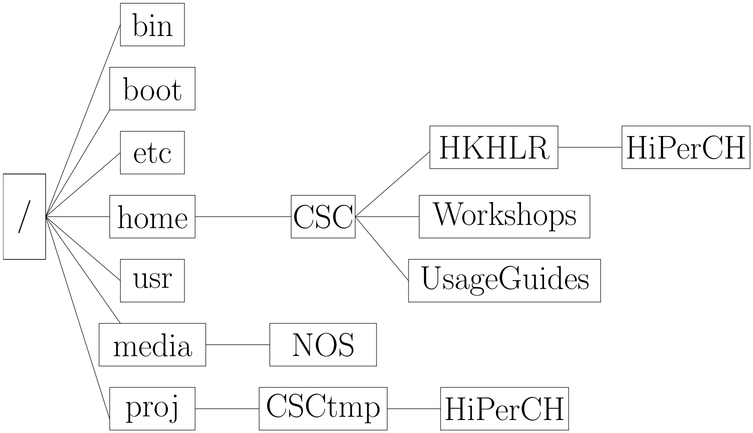
Operating System
UNIX Structure

- UNIX has files & processes.

processes	executing program identified by a unique PID. PID - process identifier
files	are a collection of data & organized into directories. everything is a file, e.g.

- modem
- keyboard
- hard-drive
- gpu

File System Basics
Directory Structure



File System Basics
Directory Structure

`/`	root directory of the file system
`/boot`	boot mechanism for programs & configuration files
`/bin`	system executables
`/lib`	essential libraries
`/etc`	system configuration files & scripts
`/var`	files that may change often, e.g. log files
`/media`	default mount point for removable devices

File System Basics
Directory Structure

`/home`	contains user home directories
`/root`	is the home directory for the `root` account
`/dev`	has device nodes
`/proc`	system state and kernel options
`/opt`	contains locally installed software
`/usr`	additional programs & resources
`/tmp`	contains temporary files, may be automatically cleared

UNIX Terminal
Terminal Usage

copy & paste in terminals

copy:	Highlight the text you want, make a copy …
paste:	…and paste at the desired location with a click of the mouse wheel (or with `Shift-Insert`).

filename completion

By typing part of the name of a command, filename or a directory & pressing the `Tab` key, the shell will complete as much as possible.

Getting Help

man

`man <command>` views the man pages

man cp

```
NAME
      cp - copy files and directories
SYNOPSIS
      cp [OPTION]... [-T] SOURCE DEST
      cp [OPTION]... SOURCE... DIRECTORY
      cp [OPTION]... -t DIRECTORY SOURCE...
DESCRIPTION
Copy SOURCE to DEST, or multiple SOURCE(s)to DIRECTORY.
      <options>
```

Documentation

- https://ubuntudanmark.dk/filer/fwunixref.pdf
- http://cheatsheetworld.com/programming/unix-linux-cheat-sheet/
- http://www.mathcs.emory.edu/~valerie/courses/fall10/155/resources/unix_cheatsheet.html
- http://www.cyberciti.biz/tips/linux-unix-commands-cheat-sheets.html
- http://www.tutorialspoint.com/unix/unix-using-variables.htm
- www.ee.surrey.ac.uk/Teaching/Unix/books-uk.html
- www.tldp.org

Filename Conventions

- all characters allowed **but** `/` and `\0` (the *null character*) should be avoided in file names
- for portability the official recommendation is use only `a-z` `A-Z` `0-9` `.` `-` `_`

Wildcards

wildcards

the * wildcard `list*` will list all files in the current directory starting with `list...`

`*list` will list all files in the current directory ending with `...list`

the ? wildcard the character `?` will match exactly one character

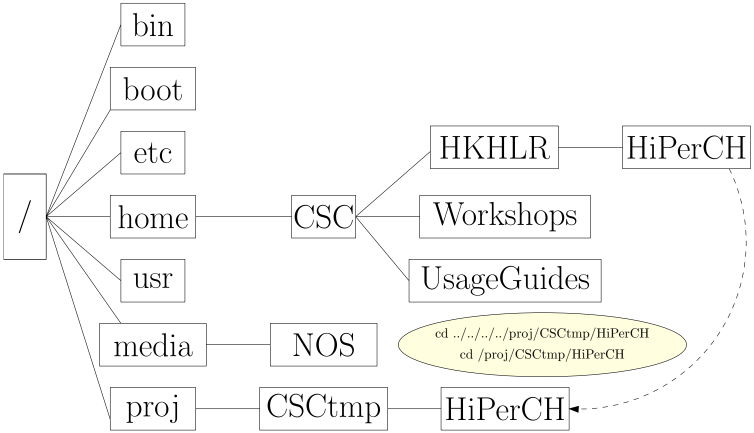
example for `?ist`:

```
List
list
gist
.ist
```

Paths

- any file can be found following a path
- **.** current directory
- **..** parent directory
- **~** home directory of current user
- case sensitive file names
- absolute paths always start with /
- relative paths refers to the current working directory

Elementary Commands
Absolute vs. Relative Paths



```
cd ../../../../proj/CSCtmp/HiPerCH
cd /proj/CSCtmp/HiPerCH
```

Elementary Commands
Listing Files and Directory

`ls`

`ls` lists the contents of a directory

Syntax: `ls <options>`

`-a`	view hidden files, **with** `.` and `..`
`-A`	view hidden files, **without** `.` and `..`
`-d`	list directories with `*/`
`-F`	list files & directories with special characters at the end
`-h`	human readable units
`-l`	list directory information
`-g`	list directory information, but do not list owner
`-1`	display one file per line

Elementary Commands

Listing Files and Directory

ls

`ls` lists the contents of a directory

Syntax: `ls <options>`

`-r`	list files in reverse order
`-R`	list subdirectories recursively
`-s`	print the allocated size of each file, in blocks
`-S`	sort by file size
`-t`	open last edited file
`-lag`	list access rights for all files
`-ltr`	reverse output order
`-lSrh`	sort files by file size

Elementary Commands
Making Directories

`mkdir`

`mkdir <directory>` creates an new `directory` in the current directory

Example for `mkdir`

`mkdir foo/bar` $\longrightarrow$ Error!

`mkdir -p foo/bar`

Syntax: `mkdir <options> <directory>`

`-p`	creating a whole path

Elementary Commands

Print Working Directory & Changing to a different Directory

`pwd`

`pwd` displays the current directory

`cd`

`cd` changes the current directory

Elementary Commands
Print Working Directory & Changing to a different Directory

Syntax: `cd <path>`

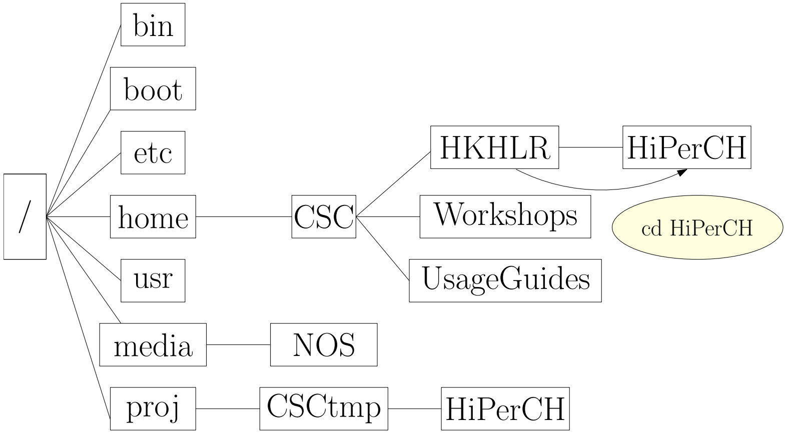
Example for `cd` & `pwd`

```
cd foo
pwd

cd bar
pwd

cd
pwd

cd foo/bar
pwd
```

Example for `cd` & `pwd`

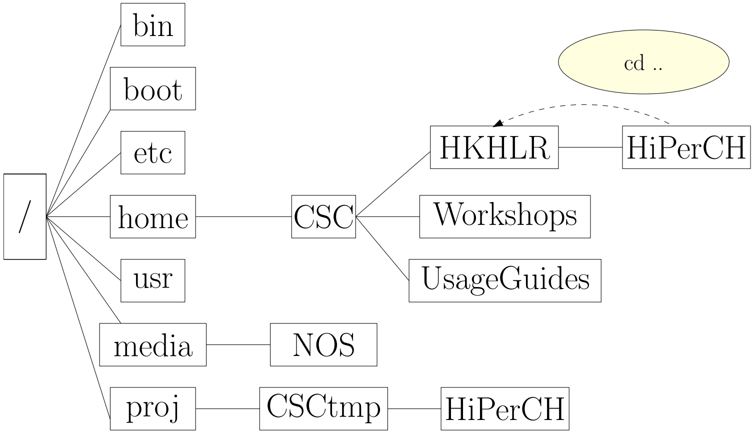
```
cd ..
pwd

cd .
pwd

cd ~
pwd
```

Elementary Commands
Changing to a different Directory

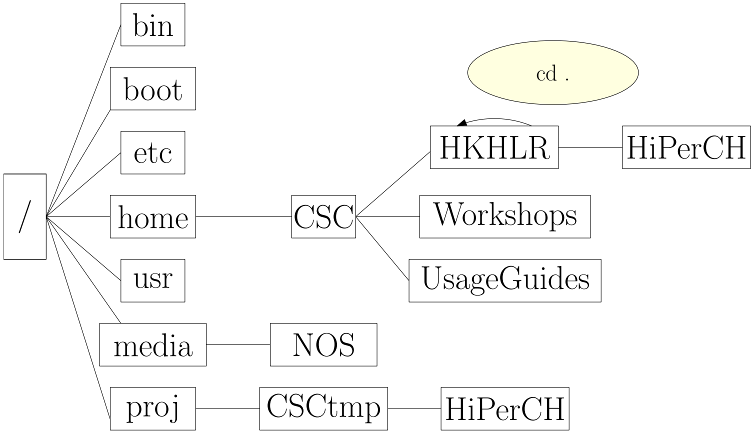


Elementary Commands

Changing to parent Directory



Elementary Commands
Current Directory



Elementary Commands
Copying Files

cp

`cp <source> <destination>` copies one or more files or directories from `source` to `destination`

Syntax: `cp <source> <destination>`

`-i`	confirm before overwriting (interactive mode)
`-r` | `-R`	copy directories recursively
`-p`	preserve attributes of file or directory while copying
`-l`	create hard link to a file or directory
`-s`	create soft link to a file or directory
`-v`	explain what is being done

Elementary Commands

Copying Files

Example for `cp`

```
cd foo
cp ~/some_file.text bar
ls -l
ls bar
cp -p ~/some_file.text .
ls -l
cp bar baz
cp -r bar baz
ls bar baz
```

Elementary Commands

Moving Files

`mv`

`mv <source> <destination>` moves & rename files & directories

Example for `mv`

```
mv oldname newname
mv filename /dest/dir
```

Syntax: `mv <source> <target>`

`-f`	do not prompt before overwriting existing files
`-i`	prompt before overwriting each existing destination file
`-n`	do not overwrite any existing file
`-u`	update

Elementary Commands

Removing Files and Directories

rm

`rm` deletes files & directories

Solution

backup your data

Warning!

`rm` deletes **without** further inquiry! There is no such thing as the trash!

`rm -rf <file/directory>`

removes everything!

Syntax: `rm <options> <file|directory>`

`-f`	delete file without prompting
`-r` | `-R`	remove directories and their contents recursively
`-i`	prompt before every removal
`-p`	delete nested directories

Elementary Commands

Removing Directories

rmdir

`rmdir <directory>` deletes the `directory` in the current directory

Warning!

`rmdir` deletes **only** empty directories!

Syntax: `rmdir <options> <directory>`

`-r`	delete directory recursively

Redirections

- $>$ symbol is used to redirect the output of a command.
  `<command> > <file>`
- $<$ symbol is used to redirect the input of a command.
  `<command> < <file>`
- $< \quad >$ get input from `file1` & write to `file2`
  `<command> < <file1> > <file2>`
  `sort < old.txt > new.txt` sorts old & saves as new.
- $>>$ appends the output to a file.
  `<command> >> <file>`
- $2>\&$ combine `stderr` and `stdout` into the `stdout` stream for further manipulation
- $|$ pipe symbol connects the output of `command1` directly to the input of `command2`

Redirections

Exercise

What is the difference between:

```
./a.out > outfile 2>& 1
./a.out 2>& 1 > outfile
```

Hint!

The shells process their command lines from left to right.

Displaying the Contents of a File on the Screen

`echo`

`echo <string>` prints the `<string>` passed to it as an argument

`cat`

`cat <file>` prints the contents of files passed to it as arguments & concatenate files together

Example for `cat`

```
cat file1.txt file2.txt > new.txt
```

Displaying the Contents of a File on the Screen

`clear`

`clear` clears the terminal screen

`head`

`head <file>` displays the first few lines of a file

`tail`

`tail <file>` displays the last few lines of a file

Syntax: `head` & `tail`

`-n num`	print the first `num` lines

Displaying the Contents of a File on the Screen

`more`

`more <file>` open files & display contents on the screen

`less`

`less <file>` open files & display contents on the screen

Warning!

If there are only strange characters are displayed after the output of a binary, type the command `reset`.

Searching the Contents of a File

grep

`grep 'keyword' <file>` search a file for keywords

Syntax: `grep`

`-A` | `-B` | `-C`	displaying lines before/after/around the match
`-c`	counting the number of matches
`-i`	case insensitive search
`-n`	precede each matching line with line number
`-o`	show only the matched string
`-v`	invert match
`-w`	search words
`-ivc`	print the number of lines without a particular words

Searching the Contents of a File

Example

```
grep Quark example.log > Quark.text
cat Quark.text
echo "These are the occurrences of Quark" >> Quark.
text
cat Quark.text
echo " in the log file" > Quark.text
cat Quark.text
```

Searching the Contents of a File

wc

`wc` count number of lines|words|characters in file

Syntax: `wc`

`-c` | `-m` | `-l` | `-w`	print the byte | character | newline | word counts
`-L`	print the length of the longest line

Example for `wc`

```
wc example.log wc -l example.log grep Quark example.
log grep Quark example.log | wc -l
```

Editing the Contents of a File

- Unix Editors are `vi`, `emacs` and `nano`.
- Midnight Commander `mc` is a directory browser/file manager for Unix-like operating systems.

File Access Permissions

There are 9 permission bits for each file divided in 3 categories.

	owner	group	other
	u	g	o
no rights	– – –	– – –	– – –
read	r – –	r – –	r – –
read & write	r w –	r w –	r w –
read & execute	r – x	r – x	r – x
read, write & execute	r w x	r w x	r w x

add permission	+
take away permission	–
execute & access directory	x

File Access Permissions

Warning!

A file should only be readable, writable and executable only for yourself in most cases.

```
- r w x - - - - - -
```

File Access Permissions
File Type

```
- rw - - - - - - -  1  gerbes  hkhlr  567  Okt 18 22:00  refsheet
```

`-`	normal file
`d`	directory
`l`	link
`Others`	various special files

File Access Permissions

Permissions

```
- rw - - - - - - -  1  gerbes  hkhlr  567  Okt 18 22:00  refsheet
```

`r`	read
`w`	write
`x`	execute
`others`	various special settings

File Access Permissions
Links

```
- rw - - - - - - - 1 gerbes hkhlr 567 Okt 18 22:00 refsheet
```

We will ignore links for now.

File Access Permissions
Users

```
- rw - - - - - - - 1 gerbes hkhlr 567 Okt 18 22:00 refsheet
```

The user that owns this file.

File Access Permissions
Groups

```
- rw - - - - - - -  1  gerbes  hkhlr  567  Okt 18 22:00  refsheet
```

The group that owns this file.

File Access Permissions
Size

```
- rw - - - - - - - 1 gerbes hkhlr 567 Okt 18 22:00 refsheet
```

The size of this file, listed in bytes.

File Access Permissions

Last Change Date

```
- rw - - - - - - - 1 gerbes hkhlr 567 Okt 18 22:00 refsheet
```

The last time the file was changed.

File Access Permissions
Name

```
- rw - - - - - - -  1  gerbes  hkhlr  567  Okt 18 22:00  refsheet
```

The file name.

File Access Permissions

Changing Permissions

`chmod`

`chmod` changes a file mode. Only the owner of a file can use `chmod` to change the permissions of a file.

Syntax:	`chmod <options>`
`u+x <file>`	making the `<file>` executable
`go-w <file>`	`<file>` is no longer writeable
`u+rw <file>`	providing `r` & `w` access to a user
`u-w <file>`	removing execute permissions to a user
`a+rx <directory>`	adding `r` & `w` permissions to all directories
`a+r <directory>/*`	everybody can read the content of the `<directory>`

Process Management

Syntax	**Description**
`ps`	displays information about the process status of all processes
`top`	lists running processes
`bg | &`	moves the current process to the background
`fg`	moves the current process to the foreground
	typing `fg` with no job number foregrounds the last suspended process
`jobs`	lists background & suspended processes

Process Management

Syntax	Description
`kill | ctrl-C`	stop the process
	it is not possible to kill of others users' processes!
`kill -9`	non-catchable, non-ignorable kill
`ctrl-D`	ending terminal line input
`ctrl-Z`	suspend the current process

File System Basics

Command	**Description**
`file <filename>`	identifies the file type
`find <filename|dir>`	finds a file/directory
`apropos <command>`	searching for commands
`touch <filename>`	creates a blank file or modifies an existing files attributes
`whereis <filename>`	shows the location of a file
`which <filename>`	shows the location of a file if it is in your `PATH`

File System Basics
Disc Usage

`du`

`du` estimate file space usage

Syntax: `du <options>`

`<file>`	shows how much space has the `<file>`
`-h`	human readable units
`-s`	show occupied space as a sum

File System Basics

File System Disk Space Usage

`df`

`df` reports the amount of available disk space being used by file systems

Syntax:	`df <options>`
`-h`	human readable units

System Info
Environment Variables

printenv

`printenv` prints the values of the specified environment

System Variable	**Description**
`HOSTNAME`	your computers name
`HOME`	home directory of the current user
`PATH`	the search path for commands
`USER`	the ID of the current user
`PWD`	the current working directory

System Info
History of Commands

`history`

`history` shows command history list

System Variable	**Description**
`HISTFILE`	the files name which command history is saved
`HISTFILESIZE`	max number of lines contained in the history file
`HISTSIZE`	the number of commands to remember in the command history

System Info

Command	Description
`date`	to display & set system date time
`reset`	resets the terminal screen if is not displaying correctly
`uname`	print information about the current system

Tape Archive

`tar`

`tar` is an archiving file format

Syntax: `tar <options>`

`-A`	appends tar files to an archive
`-c`	creates a new archive
`-d`	shows the differences between archive & file system
`--delete`	deletes from the archive
`-f`	use archive file

Tape Archive

`tar`

`tar` is an archiving file format

Syntax: `tar <options>`

`-j`	`r` or `w` archives using the bzip2 compressor
`-r`	appends files to the end of a tar archive
`-t`	lists the contents of an archive
`-u`	updates the tar archive
`-x`	extracts files from a tar archive
`-z`	`r` or `w` archives through gzip

Tape Archive

Example for `tar`

```
tar -cf <archivename>.tar <directory>
```
will pack all files in a directory

```
tar -czf <archivename>.tar.gz <directory>
```
will pack & compress all files in a directory using gzip

```
tar -cjf <archivename>.tar.bz2 <directory>
```
will pack & compress all files in a directory using bzip2

```
tar -xfz <archivename>.tar.gz
```
to unpack (and uncompress) use `x` instead of `c`

Tape Archive

Example for `tar`

```
tar cvf stuff.tar example.log some_file.text Quark.
text
ls -ltrh
tar tf stuff.tar
rm stuff.tar
tar cvzf stuff.tar.gz example.log some_file.text
Quark.text
ls -ltrh
tar tf stuff.tar.gz
rm example.log some_file.text Quark.text
ls
tar xvzf stuff.tar.gz
ls
```

Secure Connections with `ssh`

Password Authentication

- $\substack{\text{user}\\\text{system}}$ is authenticated to the $\substack{\text{system}\\\text{user}}$ using password only
- all transmitted data is encrypted
- host authentication is performed via fingerprint comparison (user responsibility)

`ssh`

```
ssh <username>@<remote-host> <command>
```

Syntax: `ssh <options>`

`-f` sends `ssh` to background

Example for `ssh`

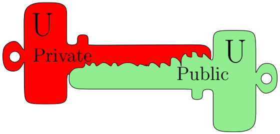
```
ssh loewe
ssh loewe ls
```

Generating Keys

`ssh-keygen`

`ssh-keygen -l` calculates the fingerprint of a public key

`ssh-keygen -t dsa | ssh-keygen -t rsa` generates a key pair

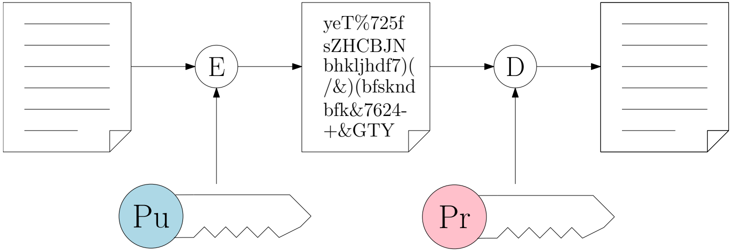


Authentication using Public Key

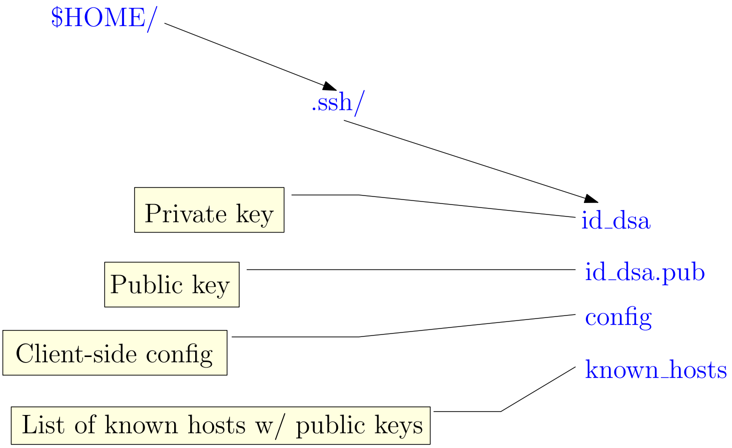
- every user owns a pair of keys, one private & one public
- public key can be known to everybody & allows to commmunicate with the user
- private key must be secret
- when the user’s public key is deposited at the remote host, the user can be authenticated without a password
- only the private key fits the corresponding public key

Authentication using Public Key

- a message will be encrypted with one of the keys & can only be decrypted with the corresponding other key
- with ssh, the session key is negotiated & authentication is performed using this mechanism



User Configuration, Client Side

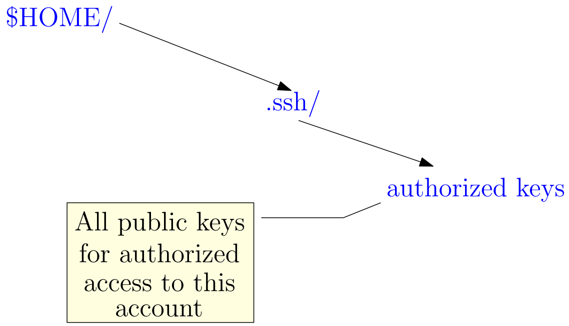


User Configuration, Client Side

Example for User Configuration

```
Host        loewe
Hostname    loewe-csc.hhlr-gu.de

Host        other_host
Hostname    other.host.name.org
User        anja

Host        yah
Hostname    yet.another.host.org
Port        45667
```

User Configuration, Server Side



Data Transfer

Secure Copy

`scp`

`scp <source> <destination>` copies files over a secure, encrypted network connection

- remote source or destination:

  `<username>@<remote-host>:<path>`
- wildcards are allowed

Syntax: `scp <options>`

`-r`	allows recursive copying into subfolders

Data Transfer

Remote Synchronization

rsync

`rsync <source> <destination>` is a tool for data transfer & synchronization of data between remote systems

- `rsync` uses `ssh` by default

Syntax: `rsync <options>`

`-a`	archive mode
`-e`	specify the remote shell to use
`-v`	increase verbosity
`-z`	compress file data during the transfer
`--stats`	give some file-transfer statistics