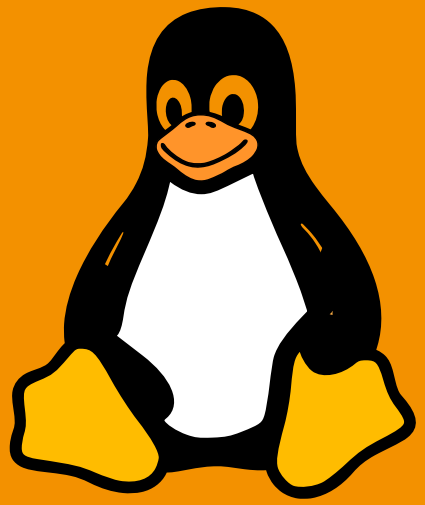
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Prof. Dr. Florian Künzner
Technical University of Applied Sciences Rosenheim, Computer Science

OS 2 – Build



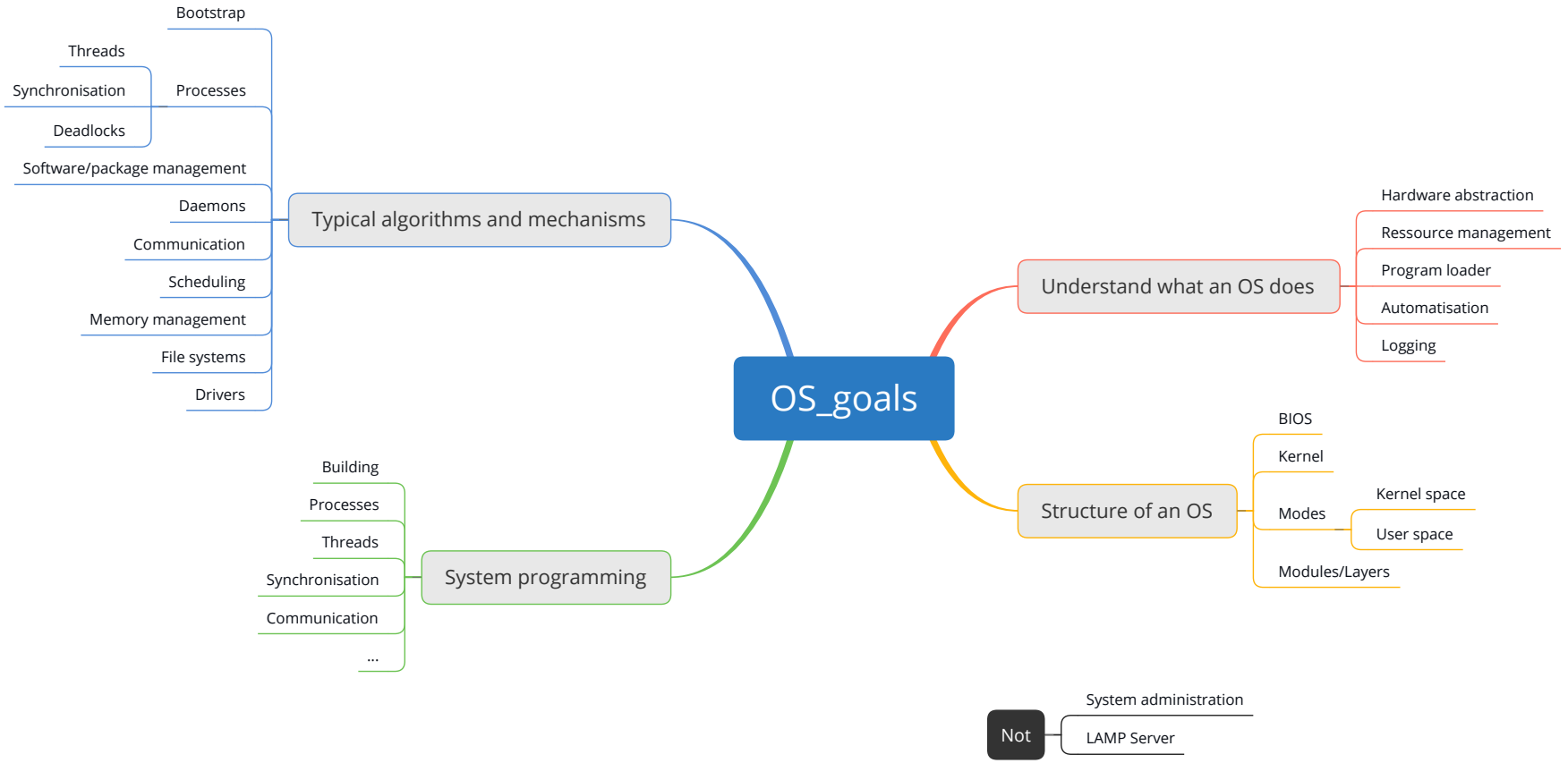
source: iconspng.com

The lecture is based on the work and the documents of Prof. Dr. Ludwig Frank

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal

OS::Build

- Build on command line
- ELF
- Makefile
- Autotools

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Build (one step)

A simple hello world

```
1 #include <stdlib.h> //EXIT_SUCCESS
2 #include <stdio.h>  //printf
3
4 int main(int argc, char const* argv[])
5 {
6     printf("hello world\n");
7     return EXIT_SUCCESS;
8 }
```

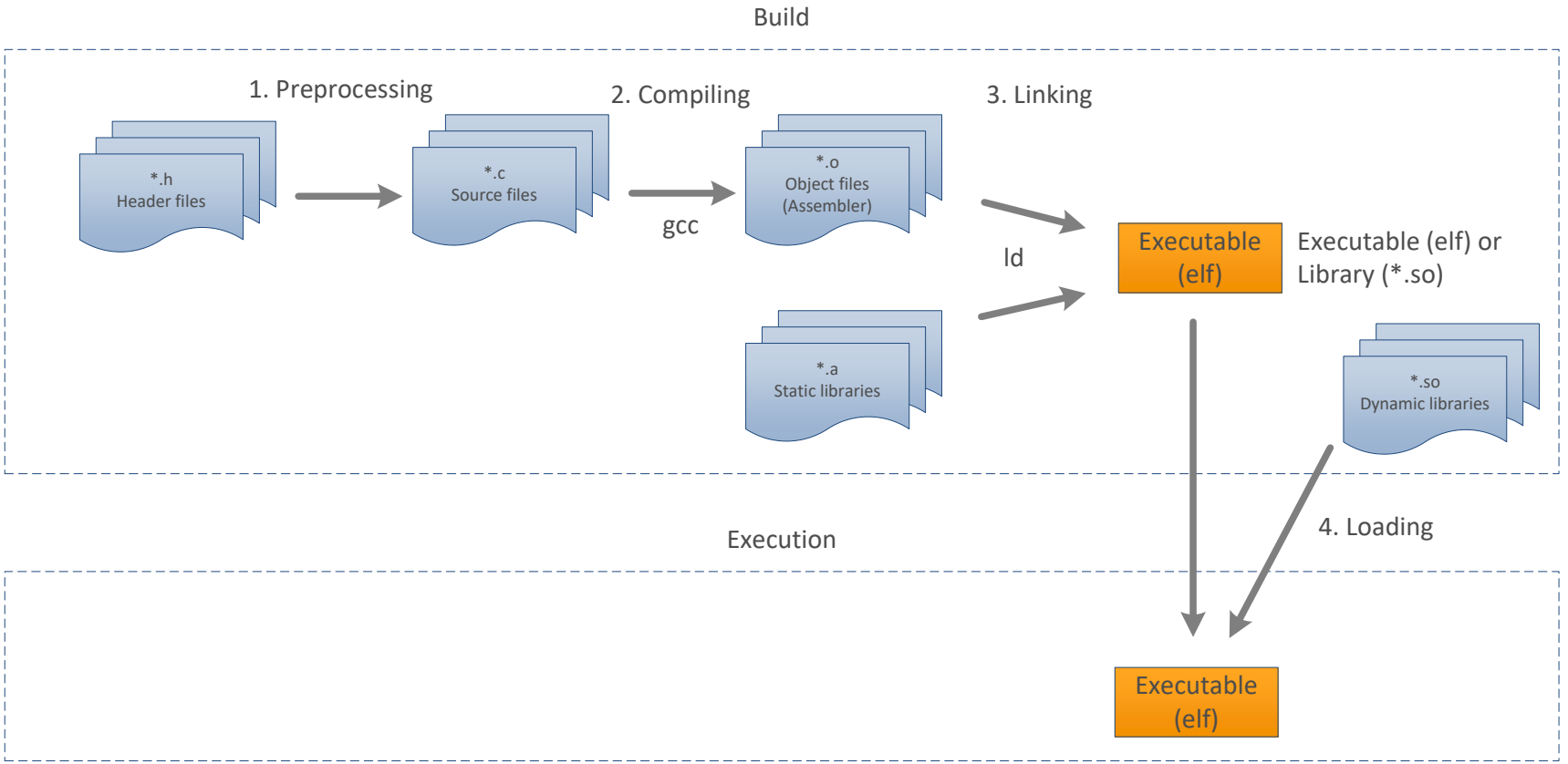
Build and execute on command line

```
gcc -o hello_world main.c #build (compile + link)
./hello_world             #execute
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Build process



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Build (separate steps: compile + link)

A simple hello world

```c
#include <stdlib.h> //EXIT_SUCCESS
#include <stdio.h>  //printf

int main(int argc, char const* argv[])
{
    printf("hello world\n");
    return EXIT_SUCCESS;
}
```

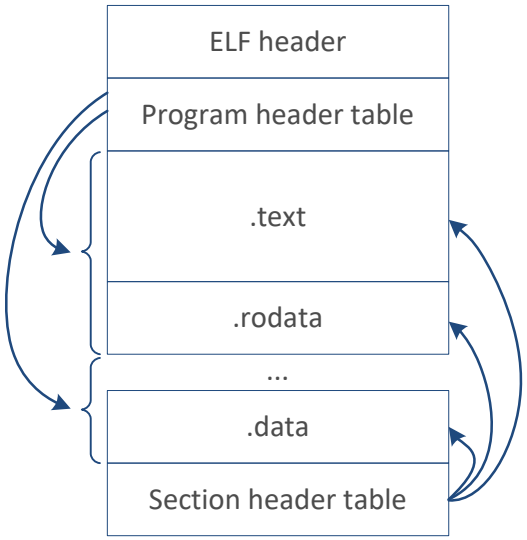
Build and execute on command line

```
gcc -c main.c              #compile main.c into main.o
gcc -o hello_world main.o  #link main.o + deps into hello_world
./hello_world              #execute
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

ELF – Executable and Linking Format



Part	Description
ELF header	Describes the files organization
Program header table	Information to create a process image
Section/Segment	Information like: instructions, data, symbol table, relocation information, ...
.text	- Executable instruction for program
.rodata	- Read-only data
.data	- Global tables, variables, etc.
.bss	- Uninitialized arrays and variables
.comment & .note	- Comments from the compiler/linker
.stab & .stabstr	- Debugging symbols & similar information
.symtab	- Symbol table
...	
Section header table	Locations for the sections

More details:
- `https://manpages.debian.org/stretch/manpages/elf.5.en.html`
- `http://www.skyfree.org/linux/references/ELF_Format.pdf`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Common commands for ELFs

```
1  #!/bin/bash
2
3  strings hello_world     #List all printable strings in a
4                          #binary file.
5
6  ldd hello_world         #List all shared libraries on which
7                          #the object binary depends.
8
9  nm hello_world          #List all symbols from the object file.
10 strip hello_world       #Delete the symbol table information.
11
12 #Display detailed information from object files.
13 objdump -t hello_world #Display symbols
14 objdump -d hello_world #Display disassembly
15
16 readelf -a hello_world #Display information about an ELF
17                         #object file.
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Second example

main.c

```c
#include <stdlib.h>         //EXIT_SUCCESS
#include <stdio.h>          //printf
#include "mathfunctions.h" //int_add

int main(int argc, char const* argv[])
{
    int a = 3, b = 4;

#ifdef USE_SPECIAL_ADD
    printf("use special add\n");
    int result = int_add(a, b);
#else
    int result = a + b;
#endif

    printf("%d + %d = %d\n", a, b, result);
    return EXIT_SUCCESS;
}
```

mathfunctions.h

```c
#ifndef MATH_FUNCTIONS_H
#define MATH_FUNCTIONS_H

/*!
 * Adds two integers a, b.
 */
int int_add(int a, int b);

#endif
```

mathfunctions.c

```c
#include "mathfunctions.h"

int int_add(int a, int b){
    return a + b;
}
```

build: `gcc -D USE_SPECIAL_ADD -o simple_prog mathfunctions.c main.c`

Any problems?

What is the problem with building on the shell with raw `gcc` command?

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Makefile example (1)

Makefile

```
1  #target for the whole program
2  simple_prog: main.o mathfunctions.o
3      gcc -o simple_prog main.o mathfunctions.o
4
5  #target for the main file
6  main.o: main.c
7      gcc -c main.c -D USE_SPECIAL_ADD
8
9  #target for the mathfunctions file
10 mathfunctions.o: mathfunctions.c mathfunctions.h
11     gcc -c mathfunctions.c
12
13 ## syntax:
14 #target: depends_on_file_or_target
15 #    command
16
17 #Behavior: if depends_on has changed, the command is executed
```

Build:

```
make
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Makefile example (2)

Makefile

```
1  CC=gcc
2
3  #target for the whole program
4  simple_prog: main.o mathfunctions.o
5      $(CC) -o simple_prog main.o mathfunctions.o
6
7  #target for the main file
8  main.o: main.c
9      $(CC) -D USE_SPECIAL_ADD -c main.c
10
11 #target for the mathfunctions file
12 mathfunctions.o: mathfunctions.c mathfunctions.h
13     $(CC) -c mathfunctions.c
```

Build:

```
make
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Makefile example (3)

Makefile

```
1  CC=gcc
2  CFLAGS=-DUSE_SPECIAL_ADD
3
4  #target for the whole program
5  simple_prog: main.o mathfunctions.o
6      $(CC) $(CFLAGS) -o simple_prog main.o mathfunctions.o
7
8  #target for the main file
9  main.o: main.c
10     $(CC) $(CFLAGS) -c main.c
11
12 #target for the mathfunctions file
13 mathfunctions.o: mathfunctions.c mathfunctions.h
14     $(CC) $(CFLAGS) -c mathfunctions.c
```

Build:

```
make
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Makefile example (4)

Makefile

```
1  #variables
2  CC=gcc
3  CFLAGS=-I. -D USE_SPECIAL_ADD
4  DEPS = mathfunctions.h
5  OBJ = main.o mathfunctions.o
6
7  #targets
8  %.o: %.c $(DEPS)
9      $(CC) -c -o $@ $< $(CFLAGS)
10
11 simple_prog: $(OBJ)
12     $(CC) -o $@ $^ $(CFLAGS)
13
14 #.PHONY are targets that have no dependencies
15 .PHONY: clean
16 clean:
17     rm -f *.o
```

Build:

```
make
```

Parallel build:

```
make -j
```

Clean:

```
make clean
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

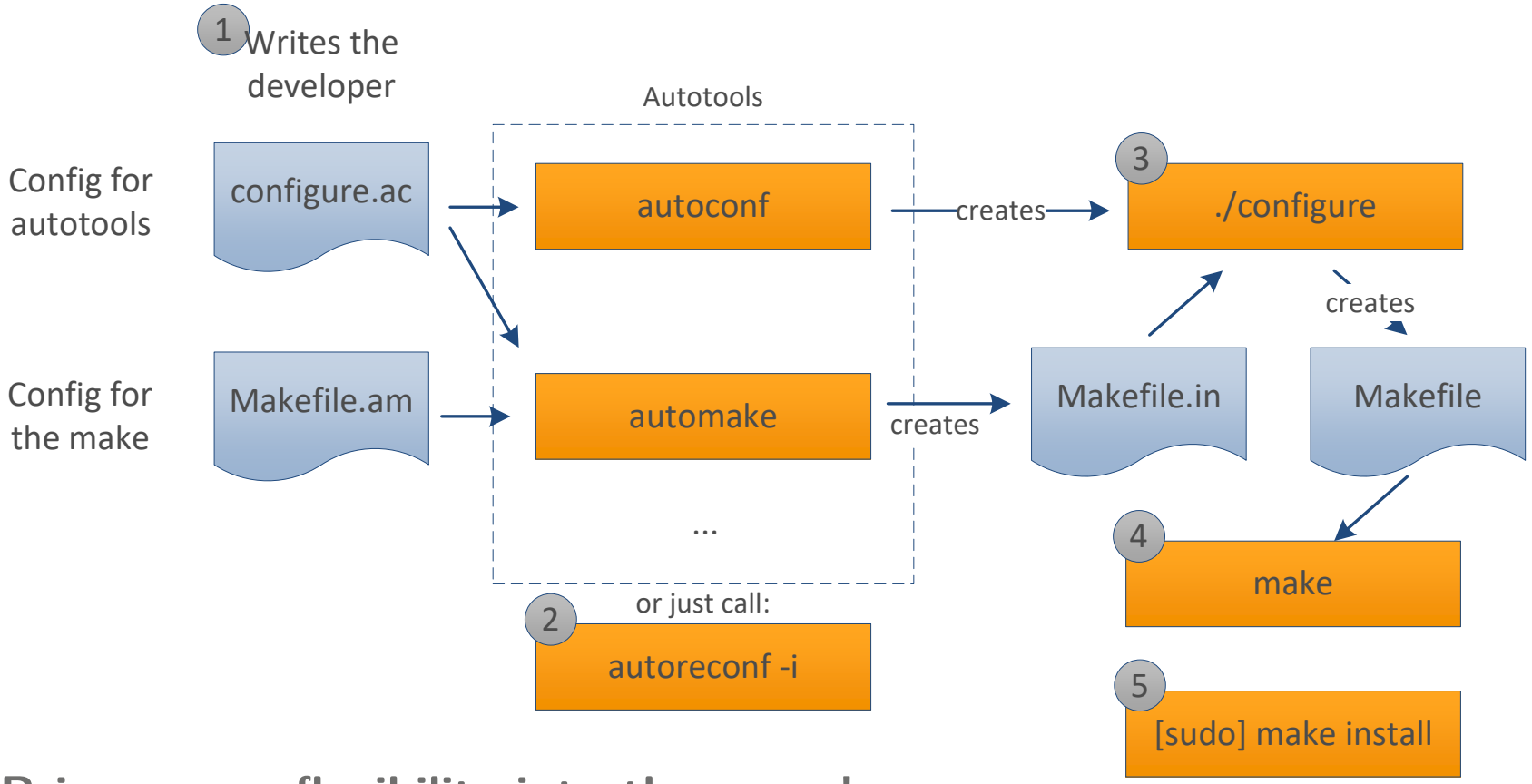
Any problems with Makefiles?

What are the problems with Makefiles?

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Autotools



Brings more flexibility into the game!

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Autotools: configure example

configure.ac

```
1  AC_PREREQ([2.69])
2  AC_INIT([simple_prog], [1.0], [florian.kuenzner@th-rosenheim.de])
3  AC_CONFIG_HEADERS([config.h])
4
5  # Configure to use build-aux for auxilary files
6  AC_CONFIG_AUX_DIR([build-aux])
7
8  # Checks for programs.
9  AC_PROG_CC
10
11 # Checks for header files.
12 AC_CHECK_HEADERS([stdlib.h])
13
14 # Init automake
15 AM_INIT_AUTOMAKE([1.11 -Wall -Werror])
16
17 # Configure creates Makefile
18 AC_CONFIG_FILES([Makefile])
19
20 AC_OUTPUT
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Autotools: make example

Makefile.am

```
1  #target binary
2  bin_PROGRAMS = simple_prog
3
4  #sources
5  simple_prog_SOURCES = main.c mathfunctions.c
6
7  #compiler flags
8  simple_prog_CFLAGS = -DUSE_SPECIAL_ADD
9
10 #manpage
11 man_MANS = simple_prog.1
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Autotools: manpage example

simple_prog.1

```
1  .\" Manpage for simple_prog
2  .\" Contact florian.kuenzner@th-rosenheim.de to correct errors or
3  .TH man 7 "14 Septenber 2018" "1.0" "simple_prog man page"
4  .SH NAME
5  simple_prog \- do something useful
6  .SH SYNOPSIS
7  simple_prog
8  .SH DESCRIPTION
9  simple_prog is a program that does something useful.
10 .SH OPTIONS
11 The simple_prog does not take any options.
12 .SH BUGS
13 No known bugs.
14 .SH AUTHOR
15 Florian Künzner (florian.kuenzner@th-rosenheim.de)
```

View: `man ./simple_prog.1`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Autotools: usage example

```
1  #initialise the build system
2  autoreconf -i
3
4  #create the Makefile
5  ./configure
6
7  #make
8  make -j
9
10 #install
11 sudo make install
12
13 #uninstall
14 sudo make uninstall
15
16 #clean
17 make clean
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Any problems with autotools?

Are there still problems with the build?

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Summary and outlook

Summary

- Build on command line
- ELF
- Makefile
- Autotools

Outlook

- Software management
- Create own packages
- Flatpak