

# Compilation of applications via, make, CMake, and Autotools

Trevor Khwam Tabougua

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen  
Burckhardtweg 4, 37077 Göttingen

November 02, 2022

- Get acquainted with software compilation
- Write Makefiles
- Build a simple library with make
- Generate executables with CMake and Autotools

# Why compiling?



- Compiling means to create an executable – or a library – from the source code
- GWWDG cannot install all software required by users (see modules for what is available)
- Scientific software is often only available as source code
- Compiling on the target system often yields better performance
- Prepackaged software typically requires administrator (root) privileges ...

# Why compiling?



## Example

- Small programs → single file → manual compilation  
`gcc -o hello main.c`

## Problems:


- Harder to manage
- Every change requires long compilation

Solution: **Makefile**

It contains a set of rules that check:

- If certain preconditions are met (files exist or have been updated).
- Runs specific commands as needed if the dependencies changed.

```
1 <target>: {<sources>}  
2   <commands>
```

 Note that the white-space is a tabulator.

Example:

```
1 hello: main.c  
2   gcc main.c -o hello
```

A **clean** rule can also be added

```
1 hello: main.c  
2   gcc main.c -o hello  
3 clean:  
4   rm -f hello
```

In the terminal:

- The command `make` will create the file `hello`
- If there's a cleaning rule, the command `make clean` will remove all the output objects

An **all** rule that depends on all other objects is used.

```
1 all: <target1> <target2>
2 <target1>: {<sources1>}
3   <commands1>
4 <target2>: {<sources2>}
5   <commands2>
```



# Exercise 1



- Clone the git repository  
`https://github.com/KTTrev/Exercises.git`
- cd into `Exercises/test_make`, and write a makefile to compile the following scripts: `main_age.c`, `main_hallo.c`, `main_hello.c`  
Use target names of your choice
- Execute the makefile, and check if the compilation worked properly by running the obtained files
- Include a clean rule, recompile the files again, and clean them afterwards

Time: 10min

Most libraries have complex programs, and dependencies, and managing makefiles can be challenging

Solution: **Makefile Generators**

**CMake** is a cross-platform free and open-source software for managing the build process of software using a compiler-independent method.

It uses a configuration file called `CMakeLists.txt` (placed in the project directory) to generate Makefiles



Assuming you are in the project folder, with the source files and the CMakeFiles.txt

1. `mkdir build`
2. `cd build`
3. `cmake ..`

The makefile is now available, thus the command `make` can be used to generate the executables, as it was previously done

In the previous git repository, if you cd into Exercises/test\_cmake, you will find the three C scripts that were used in the previous exercise, with an additional file: CMakeFiles.txt

- Load cmake/3.21.4 with the following command:  
`module load cmake/3.21.4`
- Use the steps provided in the previous slide to generate the executables

Time: 5 min

Just as CMake, **Autotools** is a quick and easy way to manage and package source code so users can compile and install software

Its primary input files are:

- `configure.ac`
- `Makefile.am`



The basic steps to build Autotools-based software are:

## 1. Configuration

```
./configure --prefix=DIR
```

Will look at the available build environment, verify required dependencies, generate Makefile(s) and a config.h

## 2. Compilation

```
make
```

Actually builds the software component, using the generated Makefiles

## 3. Installation

```
make install
```

Installs what has been built (move binaries into PATH etc.)

# About "--prefix"



- "--prefix" is used to specify the base directory for your software
- use `./configure --prefix=DIR` to install directly in DIR.  
e.g.
  - ▶ `./configure --prefix=$HOME/software/<name-version>`



- Get the source code with the command `wget` :

```
wget https://mirrors.tripadvisor.com/gnu/nano/nano-6.4.tar.xz
```

- Unpack the nano source with `tar -xvf` :

```
tar -xvf nano-6.4.tar.xz
```

- Then:

- ▶ `cd nano-6.4/`

- ▶ `mkdir build`

- ▶ `./configure --prefix=/usr/users/YOUR USERNAME/bin/`

- ▶ `make`

- ▶ `make install`

## Exercise 3



From <https://ftp.gnu.org/gnu/ncurses/>, download the source code of ncurses-6.3, and install it as in the previous example.

Time: 10 min

- **make**
  - ▶ Official manual
  - ▶ Make tutorial
- **CMake**
  - ▶ Official manual
- **Autotools**
  - ▶ Official manual