# DAS BUILDSYSTEM: MESON

Von Erik Alaverdyan

# Agenda

1. Motivation
2. Was ist ein Buildsystem? (Definition)
3. Meson:
   - *Was ist mit Meson?*
   - *Build-Definition am Beispiel*
   - *Der Build im Buildsystem (a), (b)*
   - *Kompilieren mit Ninja*
   - *Und ohne Buildsystem?*
   - *Buildsprache Meson (allgemein, Dependencies & Include-Directories)*
4. Meson und die Konkurrenz
   - *Vor- und Nachteile*
   - *Geschwindigkeiten im Vergleich*
5. Sollte einer von uns Meson nutzen?
6. Zusammenfassung
7. Literatur/ Quellen

# Motivation

1. *großes Projekt mit langem Build*
2. *Reproduzierbarkeit*
3. *einfacheres Ausführen*

# Was ist ein Buildsystem? (Definition)

- kümmert sich um Kompilierung, Abhängigkeiten (Dependencies), Ausführung des Programms …
- sollte reproduzierbar sein
- ist nicht das selbe wie eine IDE
- im Bestfall unabhängig von IDEs
- ist modifizierbar

# Was ist mit Meson?

- Meson ist ein etwas anderes Buildsystem
- Meson abstrahiert für Ninja -> Backend Default
- ziemlich neu im Geschäft
- Sprachenunterstützung ist klein, aber fein!
- die 90/9/1 Regel
- Wer hat das Licht ausgemacht?! Ich sehe nichts… -> Kein GUI?!
- Du brauchst nur Ninja und Python3

Python3: https://www.python.org/
Ninja: https://github.com/ninja-build/ninja/

Über Python mit pip3:

```
erik$ pip3 install meson
```

# Build-Definition am Beispiel

Der Quellcode aus der Bernd.c Datei:

```c
#include<stdio.h>

int erwiderung() {
  printf("Hi! Na wie gehts?\n");
  return 0;
}
```

Der Quellcode aus der Günther.c Datei:

```c
#include<stdio.h>
#include "Bernd.c"

int main(int argc, char **argv) {
  printf("Hey.\n");
  erwiderung();
  return 0;
}
```

Die Build-Definition bzw. meson.build Datei:

```
project('Begrüßung', 'c')
executable('Hey', sources : 'Günther.c')
```

# Der Build im Buildsystem
## (a)

jetziger Stand


```
Meson Übung 01 erik$ meson setup src builddir
```

Befehl

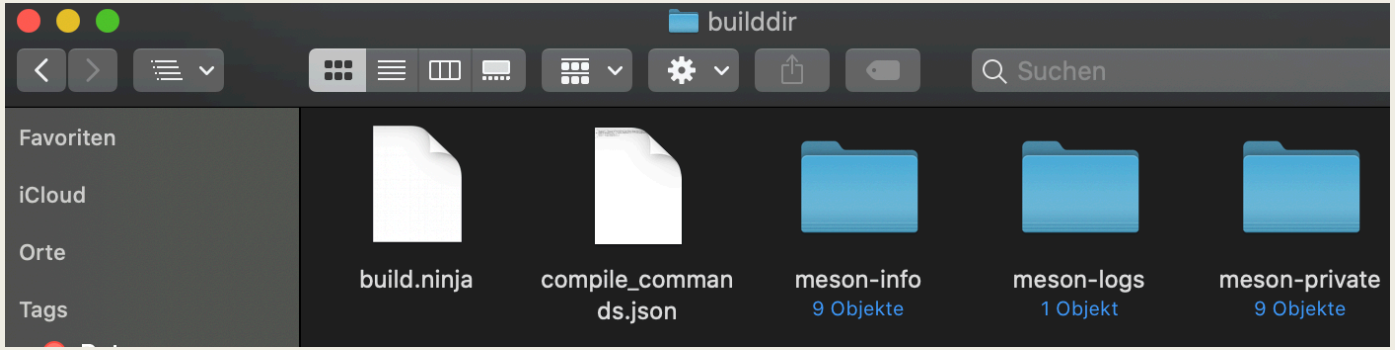
```
The Meson build system
Version: 0.54.1
Source dir: /Users/erik/Desktop/Meson Tests/Meson Übung 01/src
Build dir: /Users/erik/Desktop/Meson Tests/Meson Übung 01/builddir
Build type: native build
Project name: Begrüßung
Project version: undefined
C compiler for the host machine: cc (clang 11.0.0 "Apple clang version 11.0.0 (clan
g-1100.0.33.16)")
C linker for the host machine: cc ld64 530
Host machine cpu family: x86_64
Host machine cpu: x86_64
Build targets in project: 1

Found ninja-1.9.0.git.kitware.dyndep-1.jobserver-1 at /Library/Frameworks/Python.fr
amework/Versions/3.5/bin/ninja
```

Output

# Der Build im Buildsystem (b)

- build.ninja ≈ meson.build
- Berichte in meson-logs
- meson-info für IDE einbindung



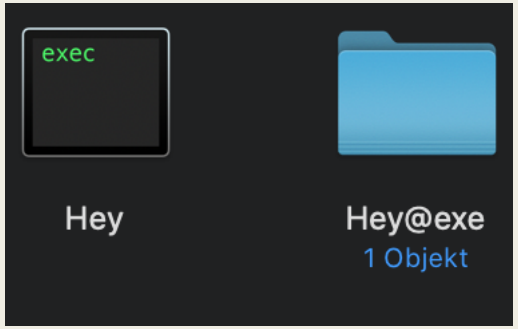

jetziger Stand

# Kompilieren mit Ninja:

- NINJA!

```
Meson Übung 01 erik$ ninja -C builddir
```

---> 




Kompilieren

```
Eriks-MBP:Meson Übung 01 erik$ ./builddir/Hey
Hey.
Hi! Na wie gehts?
```

Ausführen mit Output

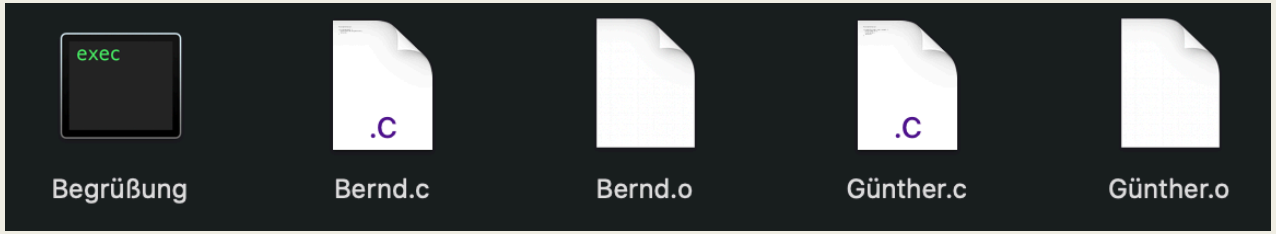
# Und ohne Buildsystem?

- einzeln Kompilieren

```
Eriks-MBP:src erik$ gcc -Wall -g -c Günther.c
Eriks-MBP:src erik$ gcc -Wall -g -c Bernd.c
```

- einzeln Kompilieren

```
Eriks-MBP:src erik$ gcc -o Begrüßung Bernd.o Günther.o -lm
Eriks-MBP:src erik$ ./Begrüßung
Hey.
Hi! Na wie gehts?
```



# Buildsprache Meson
## (Allgemein)

- in der meson.build Datei
- Variablen, Arrays, If-Statements...

a)

```
1  project('Begrüßung', 'c')
2  executable('Hey', sources : 'Günther.c')
```

b)

```
1  project('Begrüßung', 'c')
2  bspArray = ['Günther.c']
3  executable('Hey', sources : bspArray)
```

c)

```
2  bspArray = ['Günther.c']
3  bspArray += ['Bernd.c', 'Berta.c']
4      #bspArrays ist dann ['Günther.c', 'Bernd.c', 'Berta.c']
```

# Buildsprache Meson
## (Allgemein)

d)

```
4  test('TestName', exe, args: ['erstes', 'zweites'], is_parallel : false, )
```

e)

```
4  test('zweiter', exe, priority: 0)
5  test('dritter', exe, priority: -70)
6  test('erster', exe, priority: 2)
```

```
1/3 erster  OK              0.01s
2/3 zweiter OK              0.01s
3/3 dritter OK              0.01s

Ok:                 3
Expected Fail:      0
Fail:               0
Unexpected Pass:    0
Skipped:            0
Timeout:            0
```

# Buildsprache Meson

## (Dependencies & Include-Directories)

- Header Dateien in C/C++:

```
2  incdir = include_directories('Unterverzeichnis')
3  exe = executable('Hey', sources : 'Günther.c', include_directories : incdir)
```

- Bibliotheken & Frameworks:

a)

```
2  zdep = dependency('zlib', version : '>=1.2.8')
3  exe = executable('Hey', sources : 'Günther.c', dependencies : zdep)
```

b)

```
2  zdep = dependency('zlib', required : false)
3  if zdep.found()
4      executable('Hey', sources : 'Günther.c', dependencies : zdep)
5  else
6      executable('Hey', sources : 'Bernd.c')
7  endif
```

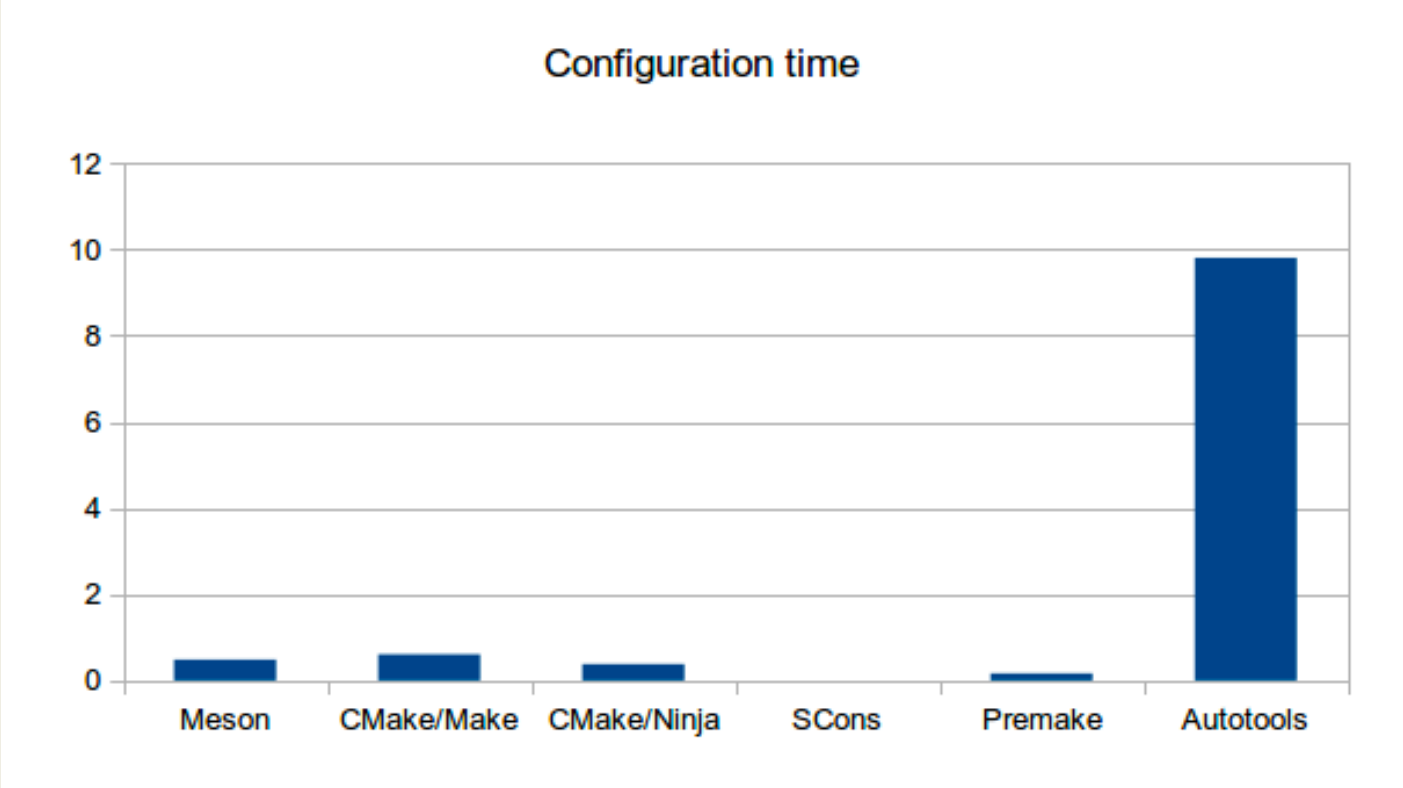
# Meson und die Konkurrenz
## (Vor- und Nachteile)

| | Pro: | Contra: |
|---|---|---|
| **GNU Autotools** | Support für Legacy Unix Plattformen | langsam, kompliziert & schwer zu Debuggen |
| **CMake** | großer Backend Support | hier und da kompliziert zu nutzen |
| **SCons** | Python nutzbar zum schreiben der Build-Scripts/-Definition | langsam und Build-Optionen werden nicht gemerkt |
| **Bazel** | kann mit sehr großen Projekten umgehen | schlechter Support für Windows |
| **Meson** | schnell, benutzerfreundlich und möglichst unsichtbar für den Nutzer | lleine Community, Backendqualität nicht sehr hoch (außer Ninja) und hat noch unbekannte Bugs. |

(Quelle: https://mesonbuild.com/Comparisons.html)

# Meson und die Konkurrenz
## (1. Konfiguration Zeit)

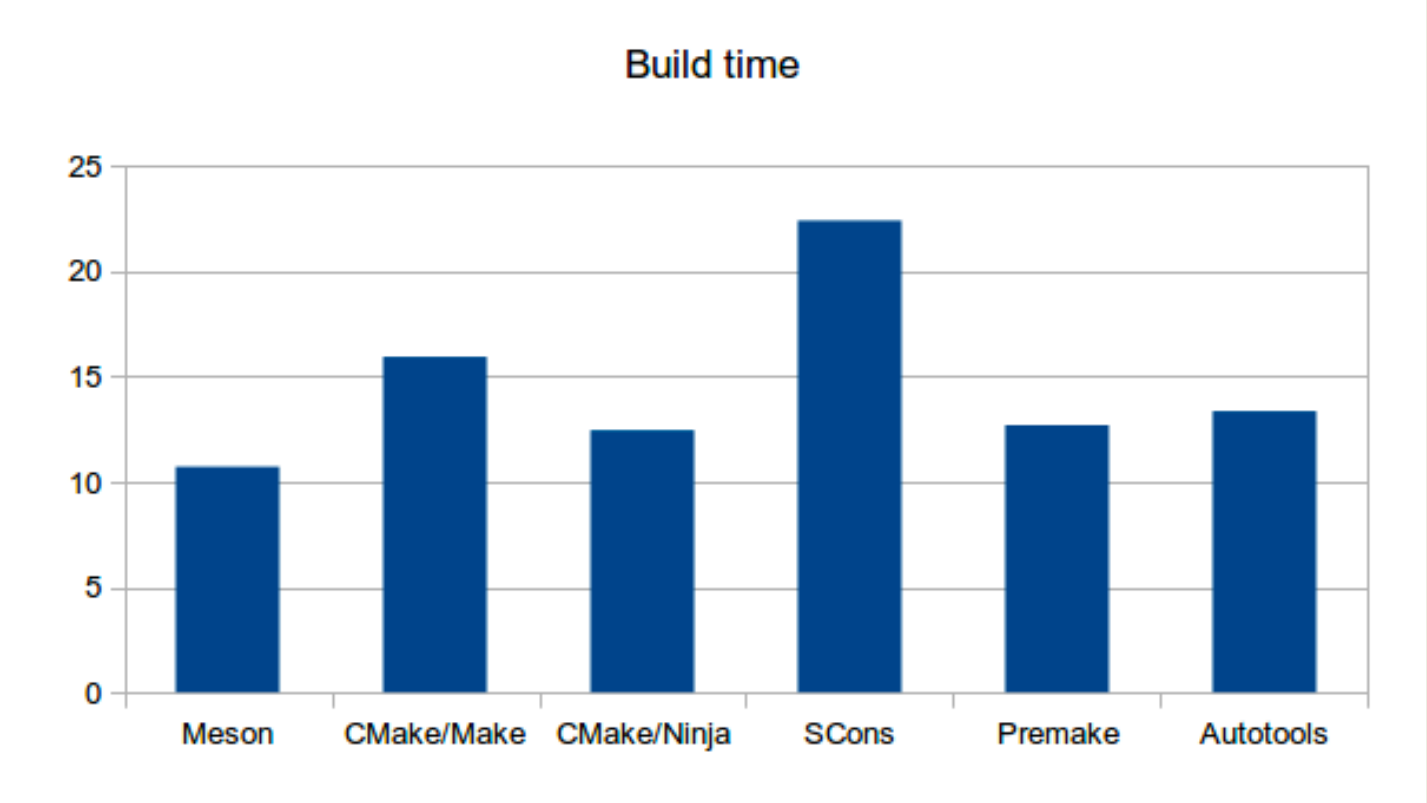
- die Build Zeit
- letzter: Autotools
- erster: Premake
- SCons spezial Fall
- Rest ca. gleich
- 2 mal CMake




(Quelle: https://mesonbuild.com/Comparisons.html)

# Meson und die Konkurrenz
## (2. Kompilierung Zeit)

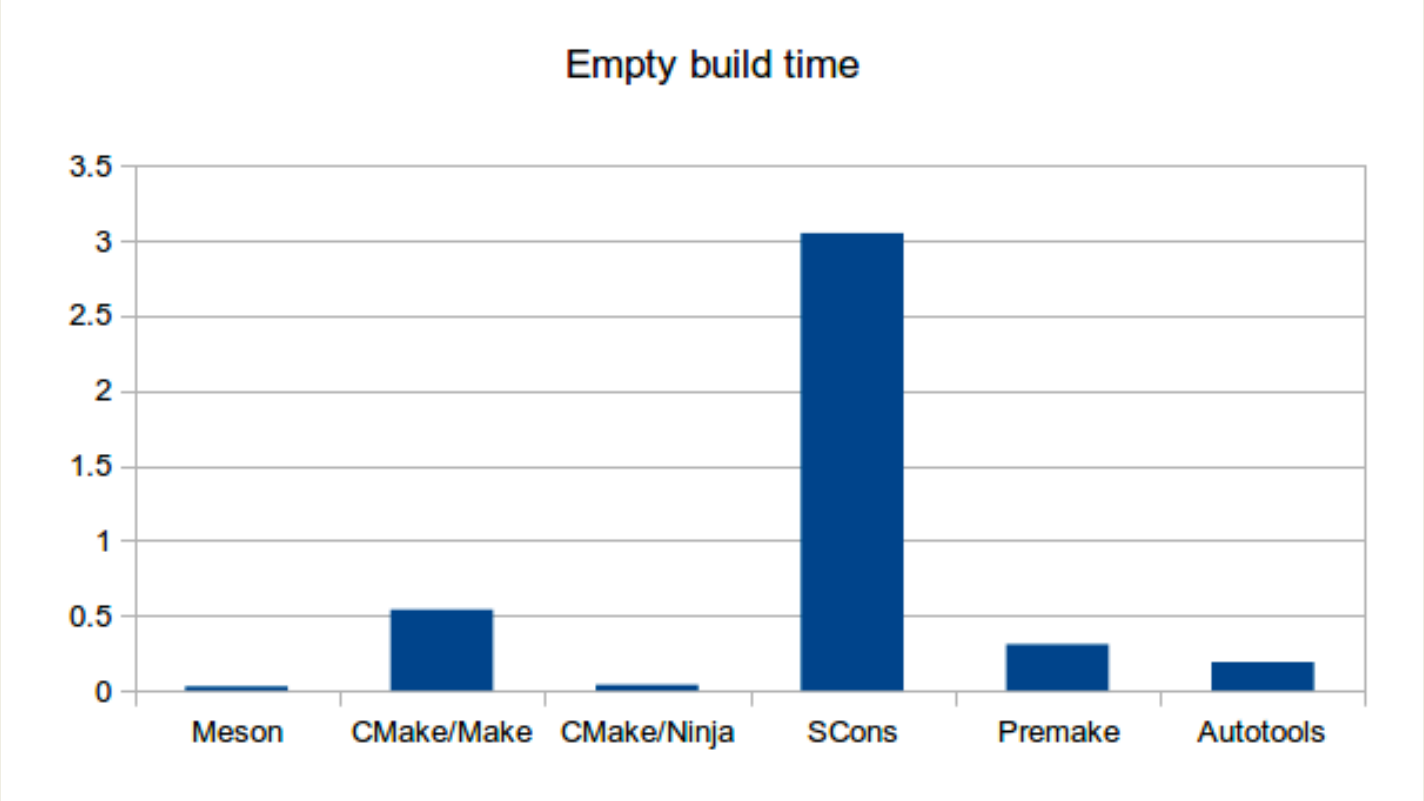
- letzter SCons
- CMake mit Ninja schneller
- Ninja am schnellsten




(Quelle: https://mesonbuild.com/Comparisons.html)

# Meson und die Konkurrenz
## (3. Leere Kompilierung Zeit)

- Ninja ist wieder Platz 1
- alle anderen deutlich langsamer




(Quelle: https://mesonbuild.com/Comparisons.html)

# Sollte einer von uns Meson nutzen?

- Ja, wenn man…
  - *… Erfahrung hat.*
  - *… sich an etwas neuem probieren möchte.*
  - *… größere Projekte in Planung hat.*

- Nein, wenn man…
  - *… schon mit SE1 Probleme hatte.*
  - *… nur kleine Projekte erstellen will.*

# Zusammenfassung

- Wir sehen, wenn es Hauptsächlich um die Schnelligkeit geht, sollte man auf Meson oder zumindest Ninja setzten.
- Zudem kann bei Lust und Laune auch sich selber mal an Meson versuchen, gerade wegen der Benutzerfreundlichkeit, die Meson mit sich bringt.
- Es ist jedoch ziemlich neu und kann wegen seiner kleinen Community auch Bugs haben, die noch nicht entdeckt wurden.
- Wenn man mehr zu Buildsystemen und ihrer Nützlichkeit wissen möchte, kann ich die folgenden Themen empfehlen:
  - *CI (Countinous Integration)*
  - *Unit/ Unity testing*
  - *Linking process*

# Literatur/ Quellen

- empfehlenswerte Literatur:
  - *Die Seite vom Entwickler selbst hat eine sehr schöne und verständliche Dokumentation:* https://mesonbuild.com/Overview.html
- Quellen:
  - *Motivation [Folie 3]*
    - https://mesonbuild.com/Reproducible-builds.html
  - *Was ist ein Buildsystem? (Definition) [Folie 4]*
    - https://www.devopsonwindows.com/what-makes-a-build-system/
    - https://softwareengineering.stackexchange.com/questions/288205/what-are-the-advantages-of-build-scripts
    - https://stackoverflow.com/questions/3209517/why-should-one-use-a-build-system-over-that-which-is-included-as-part-of-an-ide
  - *Was ist mit Meson? [Folie 5]*
    - Jussi Pakkanen - Meson: compiling the world with Python
    - https://mesonbuild.com/Quick-guide.html
  - *Build-Definition am Beispiel [Folie 6]*
    - https://mesonbuild.com/Tutorial.html
    - https://www.youtube.com/watch?v=A3Pq3E1S8ss&t=522s
  - *Der Build im Buildsystem (a) [Folie 7]*
    - https://mesonbuild.com/Running-Meson.html#
  - *Der Build im Buildsystem (b) [Folie 8]*
    - https://mesonbuild.com/IDE-integration.html

# Literatur/ Quellen

- Quellen:
  - *Kompilieren mit Ninja [Folie 9]*
    - https://mesonbuild.com/Tutorial.html
  - *Und ohne Buildsystem? [Folie 10]*
    - https://www.cs.utah.edu/~zachary/isp/tutorials/separate/answer61.html
  - *Buildsprache Meson (Allgemein) [11]*
    - https://mesonbuild.com/Unit-tests.html
    - https://mesonbuild.com/Syntax.html
  - *Buildsprache Meson (Dependencies & include directories) [12]*
    - https://mesonbuild.com/Include-directories.html
    - https://mesonbuild.com/Dependencies.html
  - *Meson und die Konkurrenz (Vor- & Nachteile) [Folie 13]*
    - https://mesonbuild.com/Comparisons.html
  - *Meson und die Konkurrenz (1, 2 & 3) [Folie 14-16]*
    - https://mesonbuild.com/Comparisons.html