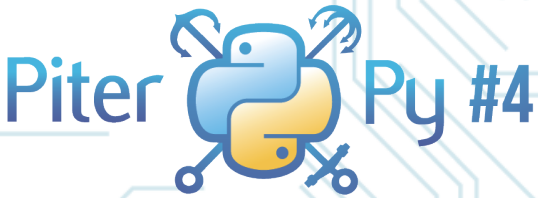


Saint Petersburg
3-4 November 2017

Power through simplicity

Use of Python in the Meson build system

Contact information

Jussi Pakkanen
ЮССИ ПАККАНЕН

@jpakkane
Rakettitiede oy
Finland

Let’s talk about
building software.

Meson build system in a nutshell

- Build system for many languages
- Supports Linux, OSX, Windows, iOS, Android
- Must scales to tens of thousands of files
- Optimized for programmer productivity
- Pure Python 3
- Use of any dependency outside Python standard library is forbidden

Who is using it

- systemd
- GStreamer multimedia framework
- Many Gnome projects such as Gnome Builder, Gnome Software, GTK, Glib
- Pitivi video editor
- X server, Wayland, Mesa graphics stack
- See website for a full list

Why are they using it?

Why did people transition from Perl to Python?

Readability
Clarity
Understandability

Also speed

Debian Package Auto-Building

Build logs for systemd on mips

Package: systemd Go

PTS – Tracker – Changelog – Bugs – packages.d.o – Source

Build logs for systemd on mips [X]

Version	Architecture	Result	Build date	Builder	Build time	Disk space
234-1 (sid)	mips	Maybe-Successful	2017-07-13 17:58:17	mips-sil-01	49m	652.59 MB
233-10 (sid)		Maybe-Successful	2017-07-03 19:46:01	mips-sil-01	1h 49m	659.1 MB
233-9 (sid)		Maybe-Successful	2017-06-19 23:19:19	mips-manda-01	1h 49m	655.71 MB
233-8 (experimental)		Maybe-Successful	2017-05-29 15:42:03	mips-aql-06	2h 14m	655.68 MB
233-7 (experimental)		Maybe-Successful	2017-05-24 13:20:14	mips-aql-06	2h 14m	655.55 MB
233-6 (experimental)		Maybe-Successful	2017-04-29 13:52:18	mips-aql-02	4h 8m	654.75 MB
233-5 (experimental)		Maybe-Successful	2017-03-22 01:27:05	mips-aql-05	3h 57m	654.23 MB
233-4 (experimental)		Maybe-Successful	2017-03-16 22:50:22	mips-aql-02	3h 56m	654.18 MB
233-3 (experimental)		Maybe-Failed	2017-03-03 19:32:22	mips-aql-06	18m	52.19 MB
233-2 (experimental)		Maybe-Successful	2017-03-03 13:18:10	mips-manda-01	1h 52m	653.92 MB
233-1 (experimental)		Maybe-Failed	2017-03-02 18:34:25	mips-manda-01	1h 46m	442.4 MB
232-25 (sid)		Maybe-Successful	2017-06-05 00:39:33	mips-sil-01	1h 43m	578.21 MB
232-24 (sid)		Maybe-Successful	2017-05-30 00:08:19	mips-manda-01	1h 42m	578.55 MB
232-23 (sid)		Maybe-Successful	2017-05-06 21:15:33	mips-manda-01	1h 41m	578.34 MB
232-22 (sid)		Maybe-Successful	2017-03-28 23:15:42	mips-sil-01	1h 41m	577.93 MB
232-21 (sid)		Maybe-Successful	2017-03-22 01:00:44	mips-aql-04	3h 36m	577.71 MB
232-20 (sid)		Maybe-Successful	2017-03-16 19:56:37	mips-sil-01	1h 41m	577.7 MB
232-19 (sid)		Maybe-Successful	2017-03-02 11:41:31	mips-manda-01	1h 41m	577.33 MB
232-18 (sid)		Maybe-Successful	2017-02-14 00:51:45	mips-aql-04	3h 36m	576.98 MB
232-17 (sid)		Maybe-Successful	2017-02-10 13:58:25	mips-aql-06	1h 42m	576.54 MB
232-16 (sid)		Maybe-Successful	2017-02-09 19:25:37	mips-manda-01	1h 41m	544.37 MB
232-15 (sid)		Maybe-Successful	2017-02-02 04:26:32	mips-aql-06	1h 41m	542.88 MB
232-14 (sid)		Maybe-Successful	2017-01-23 19:12:42	mips-aql-05	3h 31m	542.8 MB
232-13 (sid)		Maybe-Successful	2017-01-22 09:30:29	mips-sil-01	1h 42m	542.32 MB
232-12 (sid)		Maybe-Successful	2017-01-18 20:57:18	mips-sil-01	1h 41m	542.33 MB
232-11 (sid)		Maybe-Successful	2017-01-18 15:13:42	mips-aql-06	1h 41m	542.33 MB
232-10 (sid)		Maybe-Failed	2017-01-14 22:37:15	mips-aql-05	3h 23m	430.39 MB

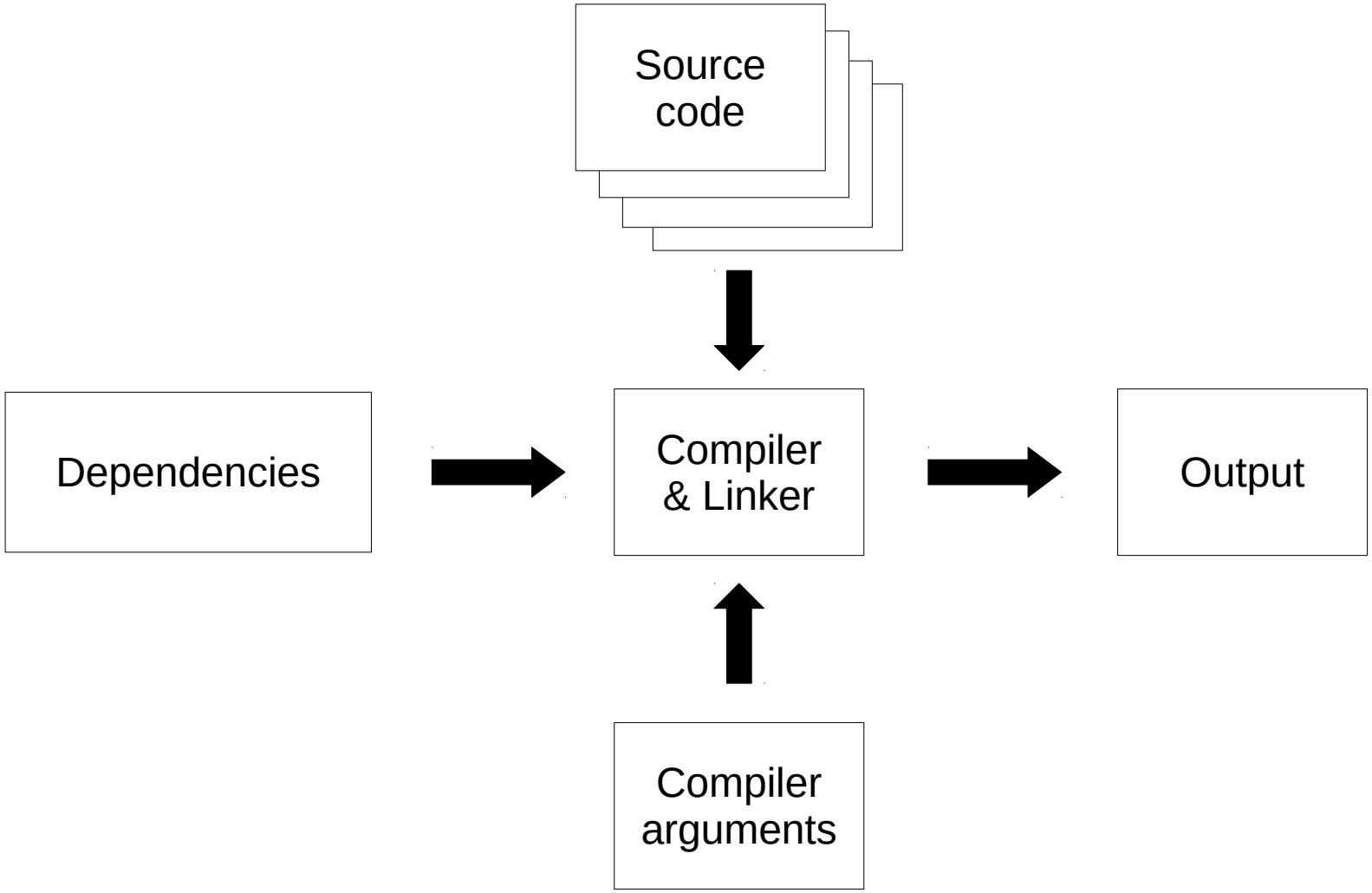
Meson’s defining feature is its domain specific language for defining builds.

A simple example

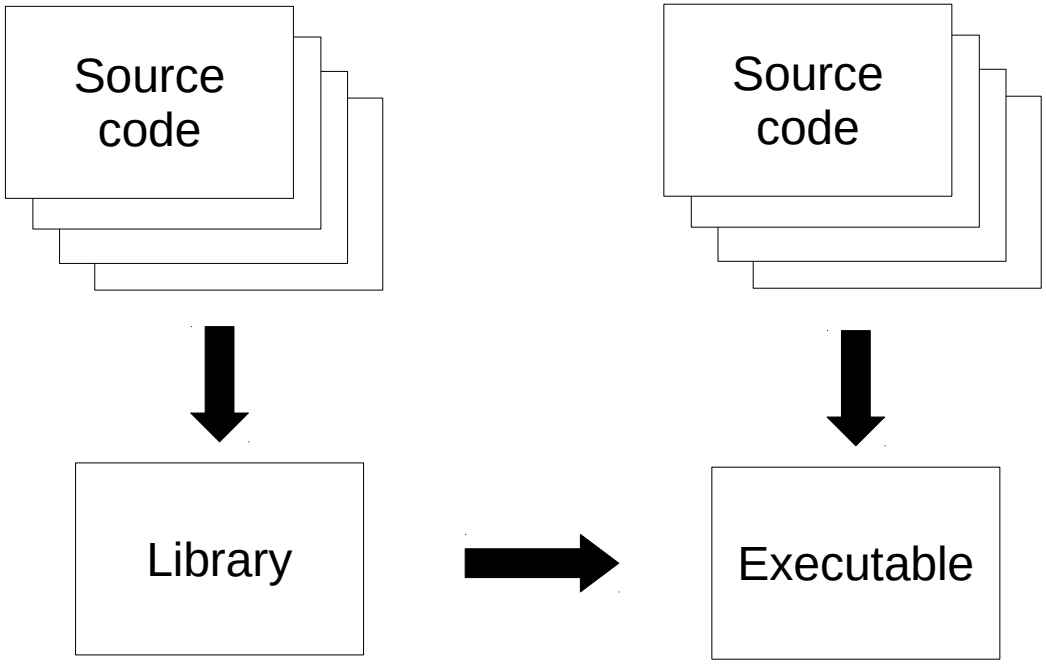
```
project('sample', 'c')
dep = dependency('glib-2.0')
exe = executable('myprog', 'prog.c',
  dependencies : dep)
test('mytest', exe)
```

What does a build system actually do?

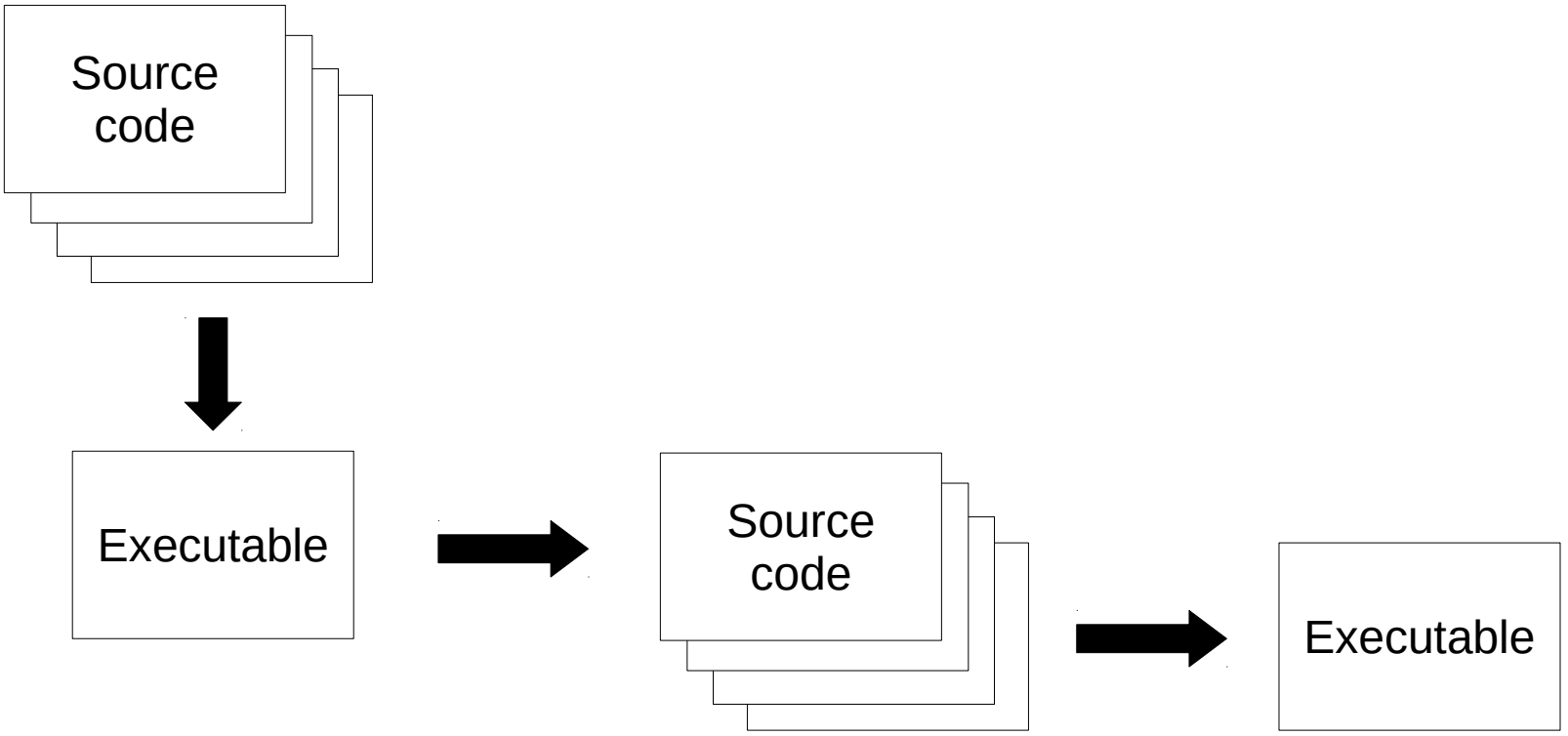
The simple level



The advanced level



The expert level



Those are all the primitives
you need to create a build system.

No, really.

Peculiarities of national code

Every language has its own quirks but the overall workflow is mostly the same.

A rough outline of workflow

- Read build definition files
- Run checks on the system, such as determining the sizes of primitives
- Evaluate target dependency graph
- Serialise graph to build steps
- Write serialisation in backend format
- Reinit self on reconfiguration

More specifically

- String parsing and manipulation
- Running external programs and parsing their output
- Graphs of moderate size (~ 1-1000 targets)
- Cross platform file manipulation
- Cross platform process management
- For dependencies, cross platform network operations

These are the things
Python excels at.

Let’s compare sizes
of different build systems.

Data from openhub.net.

GNU Autotools

23 000 lines of autoconf
87 000 lines of shell
14 000 lines of Perl

CMake

900 000 lines of C/C++
210 000 lines of CMakescript

Google Bazel

630 000 lines of Java
370 000 lines of C and C++

Meson

25 000 lines of Python
5000 lines of C and C++

An order of magnitude in size
is a big difference in understanding.

Easier to hack on and contribute.

This is the power of Python.

Laborious things become simple.

Good tools and the correct level of abstraction allow you to do great things.

Would it make sense to
rewrite Meson in a different language?

Maybe.

But not due to performance.

Duck typing makes large Python codebases hard to understand and refactor

Putting it all together.

Let’s create a
Python extension module.

Using

C
C++
Rust
Fortran

In a single module

The Meson build definition

```
project(‘polysnake’, ‘c’, ‘cpp’, ‘rust’, ‘fortran’)

py3_mod = import(‘python3’)
py3_dep = dependency(‘python3’)

rustlib = static_library(‘func’, ‘func.rs’)

py3_mod.extension_module(‘polysnake’,
  ‘polysnake.c’, ,’func.cpp’, ‘ffunc.f90’,
  link_with : rustlib,
  dependencies : py3_dep)
```

Simple things are simple.

Hard things are possible.

Life is too valuable to spend babysitting compiler flags.

In conclusion

- Python will become a core build dependency for the userland of a modern Linux system
- Efficient use of Python’s strengths can yield roughly the same functionality in 1/10th of lines of code
- Building source code has gotten a lot easier and less aggravating
- Simplicity is power!