# Controlling the shell environment in EasyBuild v5.0

Alex Domingo

VRIJE UNIVERSITEIT BRUSSEL

VUB HPC

VLAAMS SUPERCOMPUTER CENTRUM

Vlaanderen is supercomputing

# WHO AM I?

**Hi! I'm Alex** (github: @lexming)

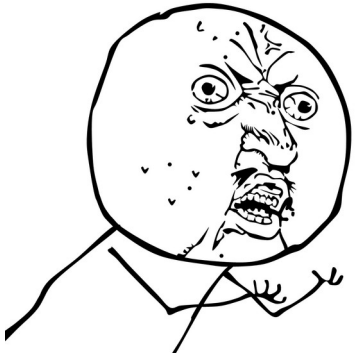
## Maintainer of EasyBuild since 2020

*HPC sysadmin when not doing EB*

- HPC team of Vrije Universiteit Brussel (VUB) since 2019
- PhD in Computational Chemistry

# WHAT IS THIS ABOUT?


```
$ ldd libawesome.so
        libdl.so.2 => /lib/libdl.so.2 (0x00000000)
        libz.so.1 => not found
```



```
/path/to/source.c:42: undefined reference to `GreatSymbol'
error: ld returned 1 exit status
```

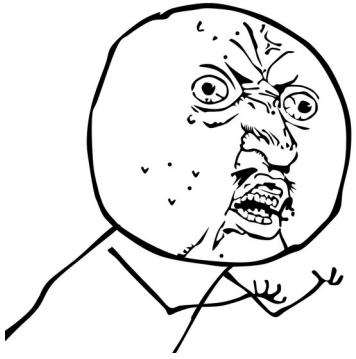
```
error: colors.h: No such file or directory
```

```
$ python -c “import potato”
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named potato
```

# WHAT IS THIS ABOUT?


```
$ ldd libawesome.so
        libdl.so.2 (0x00000000)
        libz.
```

$LD_LIBRARY_PATH



```
/path/to/source.c:42: und
error: ld returned
 to `GreatSymbol'
```

$LIBRARY_PATH

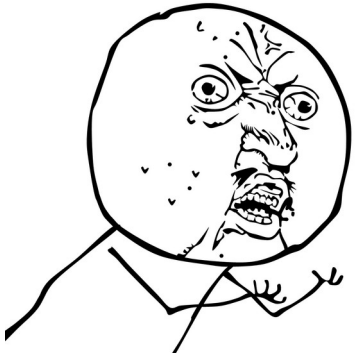
```
error: colo
 or directory
```

$CPATH

```
$ python -c “import
Traceback (mos
  File "
 in <module>
Importl
dule named potato
```

$PYTHONPATH

# WHAT IS THIS ABOUT?

```
$ ldd libawesome.so
        lib                                    .so.2 (0x00000000)
        li                                     ound
```

```
                                               to `GreatSymbol'
```

```
error: colo                                    or directory
```

```
                                               ule named potato
```

**SEARCH PATHS**

Environment variable that defines a list of paths that can potentially contain target files

LIBRARY_PATH

PATH

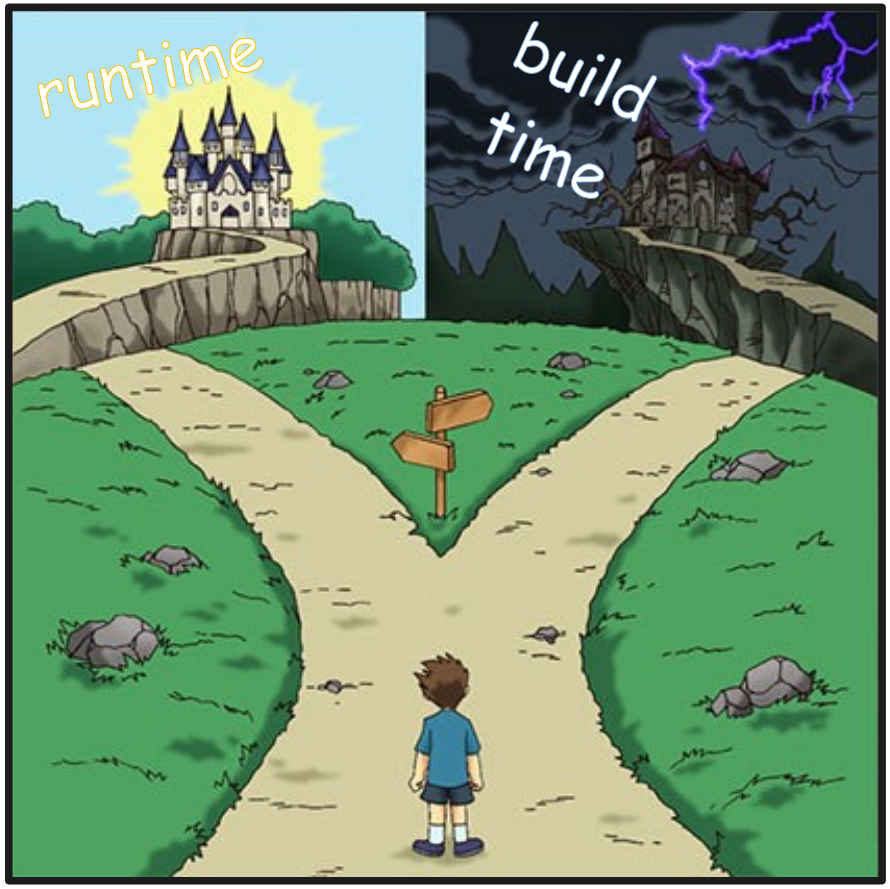
$CPATH

HONPATH

# RUNTIME/BUILD DICHOTOMY

## Runtime

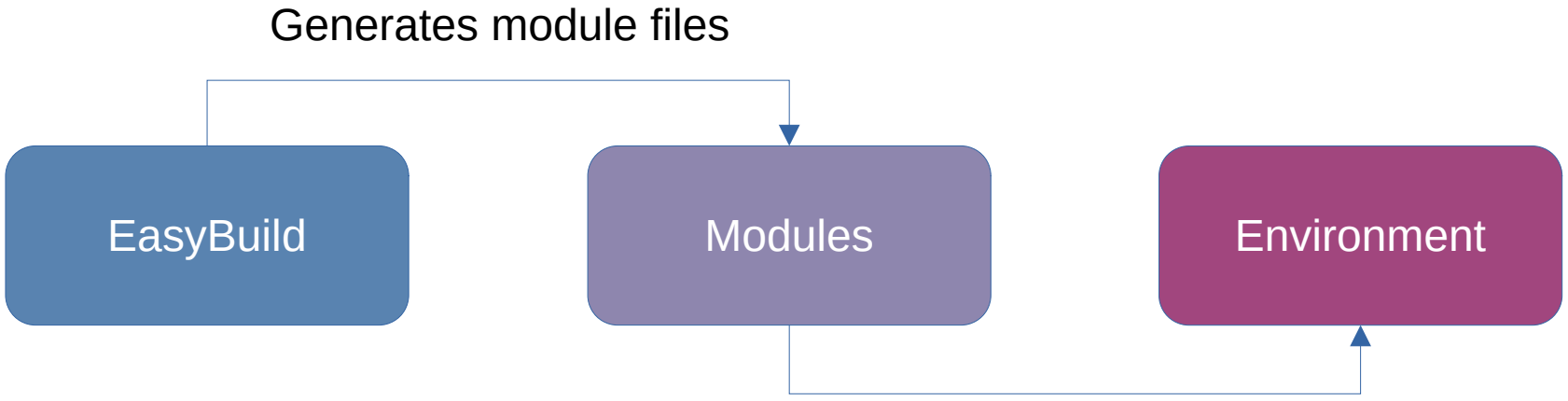
- Modules are king
- EasyBuild is not even running
- Easy to check environment state




## Build time

- Modules are still at play
- EasyBuild does its own tricks
- Tricky to check environment state

# RUNTIME ENVIRONMENT



# RUNTIME ENVIRONMENT IN EB5

## Framework

`[easybuild.framework.easyblock]`

**EasyBlock.make_module_step()**

- `self.modules_header`
- `self.make_module_description()`
- `self.make_module_group_check()`
- `self.make_module_deppaths()`
- `self.make_module_dep()`
- `self.make_module_extend_modpath()`
- **`self.make_module_req()`**
- `self.make_module_extra()`
- `self.make_module_footer()`

# RUNTIME ENVIRONMENT IN EB4

Framework
[easybuild.framework.easyblock]

**EasyBlock.make_module_step()**

- self.modules_header
- self.make_module_description()
- self.make_module_group_check()
- self.make_module_deppaths()
- self.make_module_dep()
- self.make_module_extend_modpath()
- **self.make_module_req()**
- self.make_module_extra()
- self.make_module_footer()

EasyBuild v4.9

**make_module_req()**

- Parse dictionary from make_module_req_guess()
- Do many magic tricks about any possible content in the guess
- ⚠️ No path expansion! ⚠️
- ⚠️ Where is modextrapaths? ⚠️

**make_module_req_guess()**

- Populate dictionary with keys per environment variable
- Contains a default definition

# RUNTIME ENVIRONMENT IN EB5

Framework
[easybuild.framework.easyblock]

**EasyBlock.make_module_step()**

- self.modules_header
- self.make_module_description()
- self.make_module_group_check()
- self.make_module_deppaths()
- self.make_module_dep()
- self.make_module_extend_modpath()
- **self.make_module_req()**
- self.make_module_extra()
- self.make_module_footer()

**EasyBlock.module_load_environment**

- Kind-of-dataclass that contains environment variable definitions
- Plus extra methods to easily manipulate environment variables

EasyBuild v5.0

**make_module_req()**

- Update environment with `modextrapaths`
- Read environment from `module_load_environment`
- Path expansion

# MODULE LOAD ENVIRONMENT IN EB5

## Framework
`[easybuild.tools.modules]`

**`EasyBlock.module_load_environment`**

- Kind-of-dataclass that contains environment variable definitions
- Plus extra methods to easily manipulate environment variables

**`ModuleLoadEnvironment`**

- Contains collection of `ModuleEnvironmentVariable`
- Provides default environment definition
- Special getter, setter and other methods to manipulate the environment
- Capable of defining aliases of environment variables
- **No logic related to exporting environment variables into modules**

**`ModuleEnvironmentVariable`**

- Kind-of-dataclass for a single environment variable
- **Extra attributes**: type of variable, delimiter, placement
- Special getter, setter and other methods to manipulate the environment
- **No logic related to exporting environment variables into modules**

# MODULE LOAD ENVIRONMENT IN EB5

**EasyBuild (framework)**

1) EasyBlock gets from initialization `module_load_environment`
2) Module step calls: make_module_req()
3) Export `module_load_environment` into module file
   - Inject `modextrapaths`
   - Expand paths
   - Generate module file code

**Custom EasyBlocks**

- Module load environment accessible at **all steps**: `self.module_load_environment`
- If you need to check stuff in installation directory, use module step

```
self.module_load_environment.TBBROOT = comp_libs_subdir_paths('tbb')
self.module_load_environment.LD_LIBRARY_PATH.append(os.path.join('lib', 'root'))
self.module_load_environment.remove('LIBRARY_PATH')
```

# MODULE LOAD ENVIRONMENT IN EB5

**EasyBuild (framework)**

1) EasyBlock gets from initialization `module_load_environment`
2) Module step calls: make_module_req()
3) Export `module_load_environment` into module file
   - Inject `modextrapaths`
   - Expand paths
   - Generate module file code

**Custom EasyBlocks**

- Module load environment accessible at **all steps**: `self.module_load_environment`
- If you need to check stuff in installation directory, use module step

**Custom EasyConfigs**

- Parameter `modextrapaths` is the **single interface** to manipulate `module_load_environment`

```
modextrapaths = {
    'ENV_VAR_NAME': 'extra/subdir',
    'WEIRD_ENV_VAR': {
        'paths': ['another/subdir1', 'another/subdir2'],
        'delimiter': '+',
        'prepend': False,
    },
}
```

# BUILD ENVIRONMENT IN EB5

🎉 **EVERYTHING SHOWED UP TO NOW DOES NOT MATTER!** 🎉

Well, modules get loaded in the build environment. But then EasyBuild generates on its own the list of paths to passed to the compiler and linker.

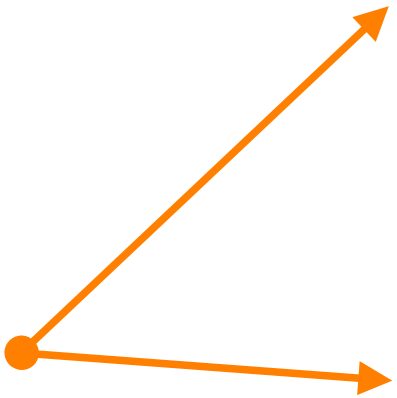
- This is actually good, improves reliability of builds but not entirely relying on modules
- Modules still have a role, but (usually) with less precedence
- Where is that defined? In the **toolchain**!

# BUILD ENVIRONMENT IN EB5

## Framework
`[easybuild.tools.toolchain.toolchain]`

Toolchains are not just a collection of dependencies. They have a lot of logic to run the build:

- Prepare step
  - Handle sysroot
  - Check dependencies
  - Load modules
  - **Add compiler/linker options for each dependency**
  - Prepare RPATH wrappers



**`--search-path-cpp-headers`**

- **`flags`**: `CPPFLAGS` environment variable, which translates into `-I` compiler options
- **`cpath`**: `CPATH` environment variable (less precedence)
- **`include_paths`**: `C_INCLUDE_PATHS` and co. environment variables (even less precedence)

**`--search-path-linker`**

- **`flags`**: `LDFLAGS` environment variable, which translates into `-L` compiler options
- **`library_path`**: `LIBRARY_PATH` environment variable (less precedence)

# BUILD ENVIRONMENT IN EB5

**✨ EasyBuild 5.0 is great! ✨**

- Clear and accessible definition of the environment that will be injected into module files
  - Use `self.module_load_environment` in easyblocks
  - Use `modextrapaths` in easyconfigs
- More control on how build environment is defined:
  - `--search-path-cpp-headers`
  - `--search-path-linker`

# ACKNOWLEDGEMENTS



Thank you for your attention!