



*building software with ease*

## Writing Easyconfig Files: The Basics

Kenneth Hoste

`kenneth.hoste@ugent.be`

HPC-UGent, Ghent University, Belgium

Jülich Supercomputer Centre, October 21st 2014

## What is an easyconfig (file)?

- build specification for EasyBuild
- mostly key-value assignments to define *easyconfig parameters*
- plain text file, Python syntax (strings, lists, dictionaries, etc.)
- specified parameters (usually) override any default value
- easyconfigs typically follow a (fixed) strict naming scheme
  - <name>-<version>[-<toolchain>] [-<versionsuffix>].eb
  - toolchain label (name, version) is omitted for dummy toolchain
  - version suffix is omitted when empty
  - filename only important w.r.t. dependency resolution (--robot)

### Example

```
name = 'GCC'  
version = '4.8.3'  
...
```

## Available easyconfig parameters

- build specification is defined by easyconfig parameters
- about 50 generic different easyconfig parameters are supported
- see **eb --avail-easyconfig-params** or **eb -a** for full list
- parameters specific to a particular easyblock are indicated
- by default, those specific to ConfigureMake are included
- parameters for other easyblocks via --easyblock/-e

### Example

```
$ eb -a -e Binary | grep install_cmd  
install_cmd(*): Install command to be used. (default: None)
```

## Mandatory easyconfig parameters

- **name, version:** specify what software (version) to build
- **homepage, description:** metadata (used for module help)
- **toolchain:** specifies compiler toolchain to use (name, version)
- some others are planned to be required in the future
  - *docurls, software\_license, software\_license\_urls*

### Example

```
name = 'foo'  
version = '1.2.3'  
  
homepage = 'http://foo.org'  
description = "foo is a tool for doing foo"  
  
toolchain = {  
    "name": 'intel',  
    "version": '2014a',  
}
```

# Common easyconfig parameters

source files, patches

- **sources**: list of source files (filenames only)
- **source\_urls**: list of URLs where sources can be downloaded
- **patches**: list of patch files to be applied (.patch extension)
- sources are downloaded (best effort), unless already available
- patches need to be EasyBuild-compatible
  - unified diff format (diff -ru)
  - patched locations relative to unpacked sources

## Example

```
name = 'GROMACS'
version = '4.6.1'
...
source_urls = ['ftp://ftp.gromacs.org/pub/gromacs/']
sources = [SOURCELOWER_TAR_GZ]
patches = ['%(namelower)s-%(version)s_Makefile-fix.patch']
```

## Common easyconfig parameters

### dependencies

- **dependencies**: build/runtime dependencies
- **builddependencies**: build-only dependencies (not in module)
- **hiddendependencies**: dependencies via hidden modules
- **osdependencies**: system dependencies (package names)
- modules must exist for all (non-system) dependencies
- (non-system) dependencies can be resolved via --robot
- format: (<name>, <version>[, <versionsuffix>[, <toolchain>]])

## Example

```
name = 'GTI'  
...  
toolchain = {'name': 'goolf', 'version': '1.5.14'}  
dependencies = [('PnMPI', '1.2.0')]  
builddependencies = [('CMake', '2.8.12', '', ('GCC', '4.8.2'))]
```

## Common easyconfig parameters

configure/build/install command options

- **configopts**: options for configure command
- **preconfigopts**: options used as prefix for configure command

Analogous:

- **builddopts, prebuilddopts**: options for build command
- **installopts, preinstallopts**: options for install command

### Example

```
easyblock = 'ConfigureMake'  
...  
preconfigopts = "./autogen.sh && "  
builddopts = 'CC="$CC" CFLAGS="$CFLAGS"'  
installopts = 'PREFIX=%(installdir)s'
```

## Common easyconfig parameters

### sanity check

**sanity\_check\_paths**: files/directories that must get installed

- used to check whether installation (partly) failed unnoticed
- paths are *relative* to installation directory
- specified in Python dictionary syntax
- mandatory: *only* files and dirs keys
- values: lists of file/directory paths (one must be non-empty)
- default: non-empty bin *and* lib or lib64 directories

### Example

```
sanity_check_paths = {
    'files': ['bin/otfconfig', 'include/open-trace-format/otf.h'],
    'dirs': [('lib', 'lib64')],
}
```

# Common easyconfig parameters

## easyblock specification

**easyblock**: specify which easyblock must be used

- overrides easyblock derived from software name
- usually a generic easyblock, but there are exceptions
  - EB\_OpenFOAM for OpenFOAM and OpenFOAM-Extend
  - EB\_Score\_minus\_P for Score-P, Cube, OTF2, Scalasca, ...

## Example

```
easyblock = 'CMakeMake'  
  
name = 'GTI'  
version = '1.2.0'  
...
```

# Common easyconfig parameters

## module class

**moduleclass:** ‘category’ in which the software package fits

- only known module classes can be specified
- define list of known module classes via --moduleclasses
- see default list via --show-default-moduleclasses
- symlink for module class is created for module (by default)

### Example

```
name = 'GCC'  
...  
moduleclass = 'compiler'
```

## Tweaking existing easyconfig files

- modify easyconfig(s) straight from command line via --try-X
- --try-toolchain to try building with a different toolchain
- --try-software-version to try building a different version
- --try-amend to try tweaking a different parameter
  - currently only for parameters with string- or list-typed values
- see 'eb --help | grep try-' for all options
- cooperates as expected with --robot

### Example

*GCC version update:*

```
eb GCC-4.9.0.eb --try-software-version=4.9.1
```

*install WRF + dozen dependencies with another toolchain (!):*

```
eb WRF-3.5.1-ictce-5.3.0-dmpar.eb --try-toolchain=intel,2014b -r
```

# String templates & constants

Dynamic values for easyconfig parameters

- string templates are completed by easyconfig parameters
  - typically name and/or version
- help to avoid hardcoding values in multiple locations
- required for making --try-software-version behave as expected
- list of available templates via --avail-easyconfig-templates
- list of available constants via --avail-easyconfig-constants

## Example

```
name = 'GCC'  
version = '4.8.3'  
...  
source_urls = [  
    # http://ftpmirror.gnu.org/gcc/gcc-4.8.3  
    'http://ftpmirror.gnu.org/%(namelower)s/%(namelower)s-%(version)s',  
]  
sources = [SOURCELOWER_TAR_GZ]  # gcc-4.8.3.tar.gz  
...
```

## Use available generic easyblocks

- use available *generic* easyblocks where applicable
- avoids need for creating (and maintaining) new easyblocks
- (custom) easyconfig parameters allow tweaking their behavior
- overview via ‘eb --list-easyblocks | grep -v EB\_’

### Example

```
easyblock = 'CMakeMake'
name = 'GTI'

...
dependencies = [('PnMPI', '1.2.0')]
configopts  = '-DCMAKE_BUILD_TYPE=Release '
configopts += '-DPnMPI_INSTALL_PREFIX=${EBROOTPNMPI}'
buildopts = 'CXXFLAGS="$CXXFLAGS -fpermissive"'
...
```

## Contributing back

- contribute back your working easyconfig files!
- share your expertise with the community, avoid duplicate work
- especially if:
  - software package is not supported yet
  - existing easyconfig needs changes for new version/toolchain
  - frequently used software package (compilers, MPI, etc.)
- ~ 25% of easyconfigs by contributors outside of HPC-UGent
- requires a limited amount of knowledge on Git/GitHub
- contributions are reviewed & thoroughly tested before inclusion
- see EasyBuild wiki for detailed walkthrough:

<https://github.com/hpcugent/easybuild/wiki/Contributing-back>



*building software with ease*

## Writing Easyconfig Files: The Basics

Kenneth Hoste

`kenneth.hoste@ugent.be`

HPC-UGent, Ghent University, Belgium

Jülich Supercomputer Centre, October 21st 2014