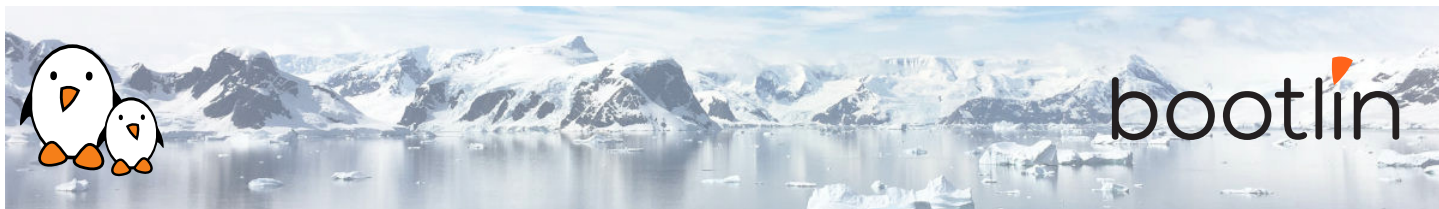


Autotools training

On-line seminar, 2 sessions of 4 hours

Latest update: May 03, 2024

Title	Autotools training
Training objectives	<ul style="list-style-type: none">• Be able to understand the role of the <i>autotools</i>• Be able to use the <i>autotools</i>• Be able to set up a basic project with <i>autoconf</i> and <i>automake</i>• Be able to use advanced <i>autoconf</i> features: configuration header, checking for functions, headers and libraries, writing custom tests, handling external software and optional features, pkg-config, etc.• Be able to use advanced <i>automake</i> features: subdirectories, conditionals, shared libraries with <i>libtool</i>, etc.
Duration	Two half days - 8 hours (4 hours per half day)
Pedagogics	<ul style="list-style-type: none">• Lectures delivered by the trainer, over video-conference. Participants can ask questions at any time.• Practical demonstrations done by the trainer, based on practical labs, over video-conference. Participants can ask questions at any time. Optionally, participants who have access to the hardware accessories can reproduce the practical labs by themselves.• Instant messaging for questions between sessions (replies under 24h, outside of week-ends and bank holidays).• Electronic copies of presentations, lab instructions and data files. They are freely available at https://bootlin.com/doc/training/autotools.
Trainer	Thomas Petazzoni. Thomas is a major Buildroot developer since 2009, an activity through which he has gained a good knowledge of <i>autoconf</i> , <i>automake</i> and <i>libtool</i> .
Language	Oral lectures: English, French. Materials: English.
Audience	Companies already using or interested in using <i>autotools</i> to build their software components.



Prerequisites	<ul style="list-style-type: none"> • Knowledge and practice of UNIX or GNU/Linux commands: participants must be familiar with the Linux command line. Participants lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides at bootlin.com/blog/command-line/.
Required equipment	<ul style="list-style-type: none"> • Computer with the operating system of your choice, with the Google Chrome or Chromium browser for videoconferencing. • Webcam and microphone (preferably from an audio headset) • High speed access to the Internet
Certificate	Only the participants who have attended all training sessions, and who have scored over 50% of correct answers at the final evaluation will receive a training certificate from Bootlin.
Disabilities	Participants with disabilities who have special needs are invited to contact us at training@bootlin.com to discuss adaptations to the training course.

Half day 1

Lecture - Overview and usage of <i>autotools</i>	Lab - Usage of an existing software component using the <i>autotools</i>
<ul style="list-style-type: none"> • What the <i>autotools</i> are, what the alternatives are, and what they are useful for. • Usage of an existing software component using the <i>autotools</i>: configuring and building the software component. • Standard Makefile targets, filesystem hierarchy, configuration variables • System types: build, host, target • Cross-compilation • Out of tree build • Diverted installation • Cache variables • Using <i>autoreconf</i> 	<ul style="list-style-type: none"> • First build of an <i>autotools</i> package • Out-of-tree build and cross-compilation • Overriding cache variables • Using <i>autoreconf</i>



Lecture - autoconf/automake: the basics

- `configure.ac` language and basic macros
- `AC_CONFIG_FILES` and *output variables*
- Minimal `Makefile.am`

Lab - autoconf/automake: the basics

- Your first `configure.ac`
- Adding and building a program
- Going further: `autoscan` and `make dist`

Half day 2

Lecture - Autoconf advanced

- Configuration header
- Checking for functions, headers, libraries
- Custom tests
- Handling external software and optional features
- `pkg-config`

Lecture - Automake advanced

- Subdirectories
- Conditionals
- Shared libraries
- Misc: variables, macro and auxiliary directories, silent rules, etc.

Lab - Implement more advanced options

- Use `AC_ARG_ENABLE` and `config.h`
- Implement a shared library
- Switch to multiple directories
- Make the compilation of programs conditional
- Use `pkg-config`