
# GNU Autotools Are What Your Users Need

Ludovic Courtès


# GNU Autotools
# Do What Your Users Need

Ludovic Courtès

# agenda

1. intro
2. goals & design
3. on portability
4. action
5. wrap up

# 1

# intro

[redacted] installation instructions
==========

1) Mandatory tools
==========

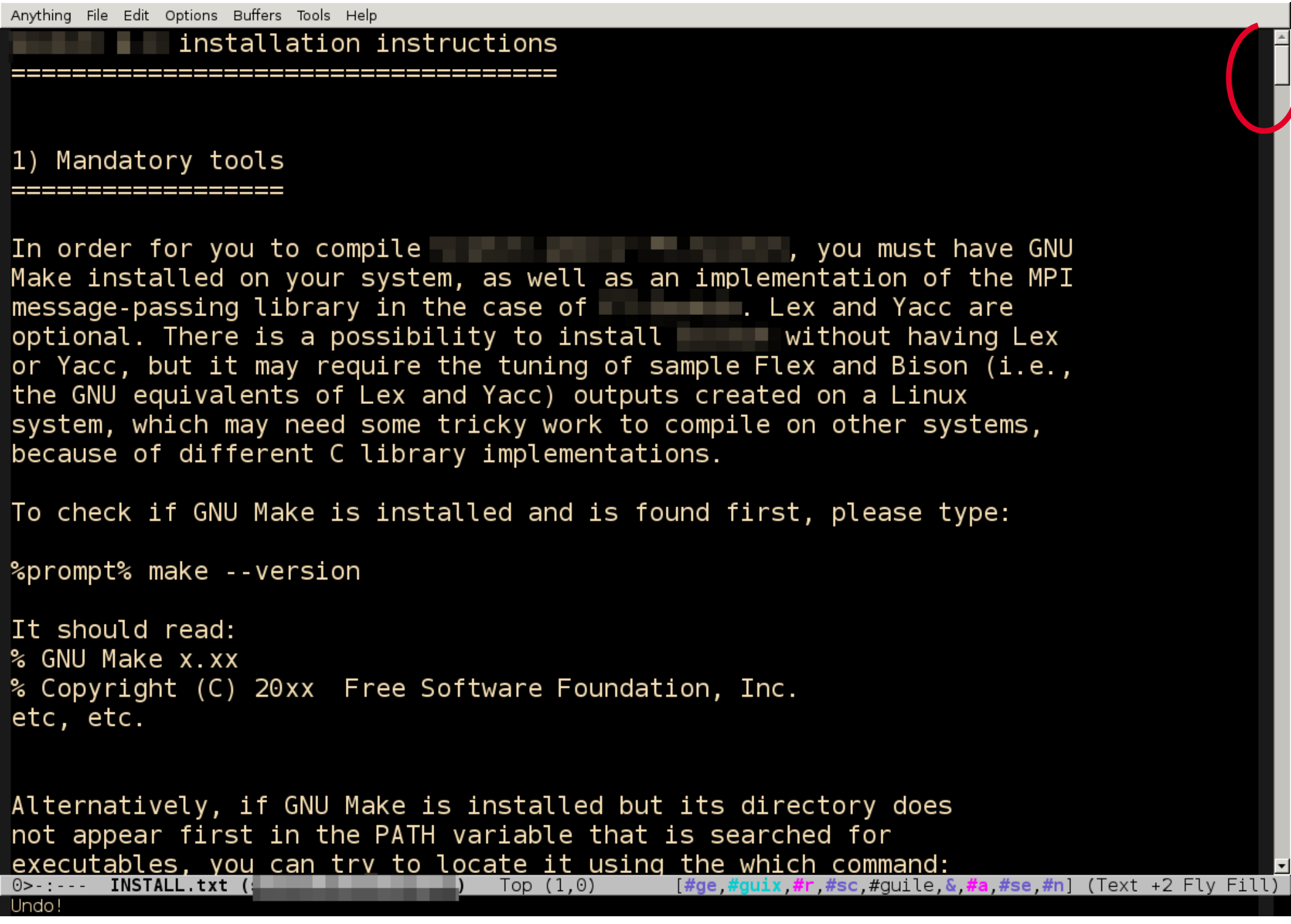
In order for you to compile [redacted], you must have GNU Make installed on your system, as well as an implementation of the MPI message-passing library in the case of [redacted]. Lex and Yacc are optional. There is a possibility to install [redacted] without having Lex or Yacc, but it may require the tuning of sample Flex and Bison (i.e., the GNU equivalents of Lex and Yacc) outputs created on a Linux system, which may need some tricky work to compile on other systems, because of different C library implementations.

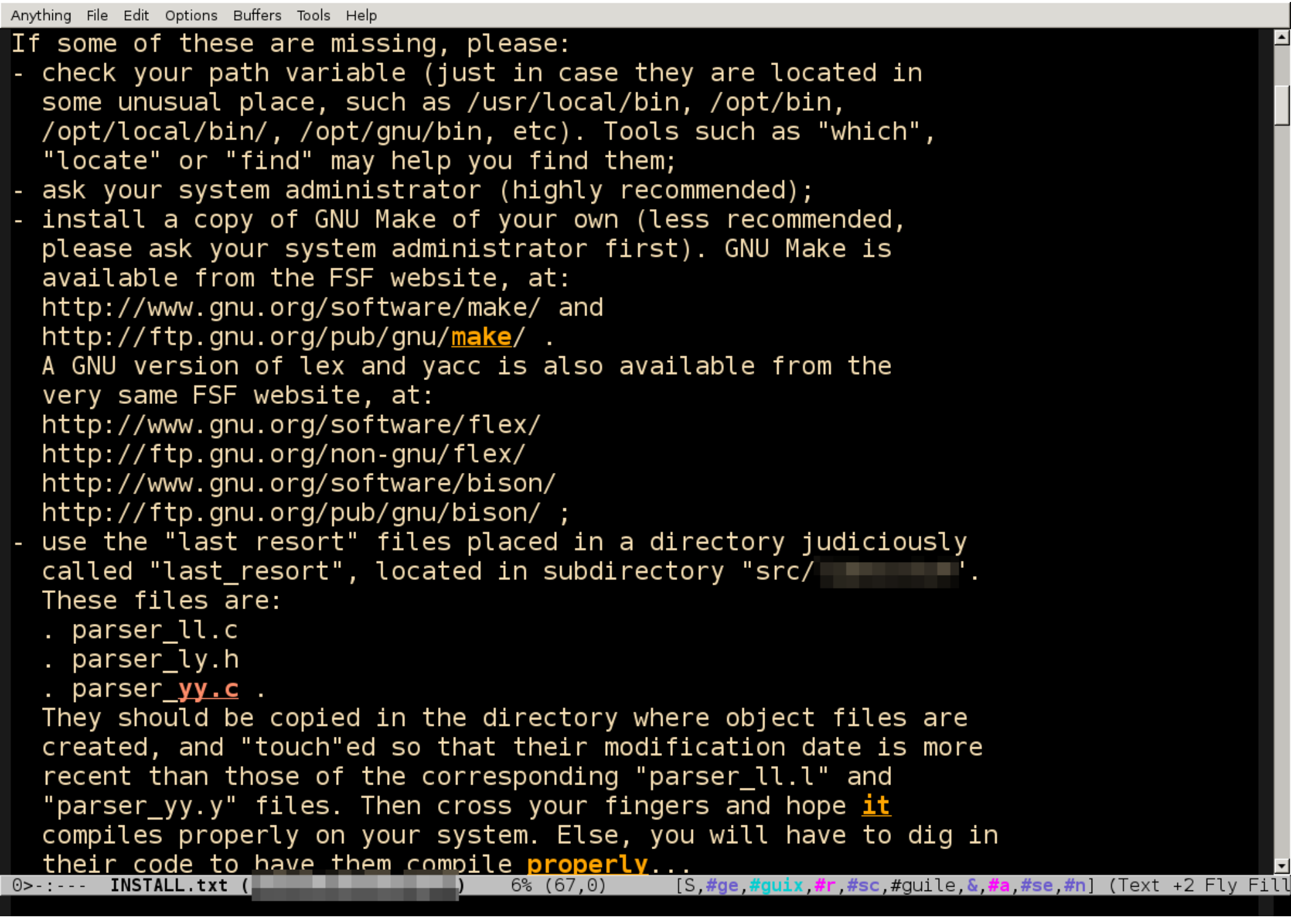
To check if GNU Make is installed and is found first, please type:

%prompt% make --version

It should read:
% GNU Make x.xx
% Copyright (C) 20xx  Free Software Foundation, Inc.
etc, etc.

Alternatively, if GNU Make is installed but its directory does not appear first in the PATH variable that is searched for executables, you can try to locate it using the which command:


Anything File Edit Options Buffers Tools Help
[redacted] [redacted] installation instructions
==========
1) Mandatory tools
==========
In order for you to compile [redacted], you must have GNU
Make installed on your system, as well as an implementation of the MPI
message-passing library in the case of [redacted]. Lex and Yacc are
optional. There is a possibility to install [redacted] without having Lex
or Yacc, but it may require the tuning of sample Flex and Bison (i.e.,
the GNU equivalents of Lex and Yacc) outputs created on a Linux
system, which may need some tricky work to compile on other systems,
because of different C library implementations.
To check if GNU Make is installed and is found first, please type:
%prompt% make --version
It should read:
% GNU Make x.xx
% Copyright (C) 20xx Free Software Foundation, Inc.
etc, etc.
Alternatively, if GNU Make is installed but its directory does
not appear first in the PATH variable that is searched for
executables, you can try to locate it using the which command:
0>-:--- INSTALL.txt ([redacted]) Top (1,0) [#ge,#guix,#r,#sc,#guile,&,#a,#se,#n] (Text +2 Fly Fill)
Undo!


Anything File Edit Options Buffers Tools Help
If some of these are missing, please:
- check your path variable (just in case they are located in
some unusual place, such as /usr/local/bin, /opt/bin,
/opt/local/bin/, /opt/gnu/bin, etc). Tools such as "which",
"locate" or "find" may help you find them;
- ask your system administrator (highly recommended);
- install a copy of GNU Make of your own (less recommended,
please ask your system administrator first). GNU Make is
available from the FSF website, at:
http://www.gnu.org/software/make/ and
http://ftp.gnu.org/pub/gnu/make/ .
A GNU version of lex and yacc is also available from the
very same FSF website, at:
http://www.gnu.org/software/flex/
http://ftp.gnu.org/non-gnu/flex/
http://www.gnu.org/software/bison/
http://ftp.gnu.org/pub/gnu/bison/ ;
- use the "last resort" files placed in a directory judiciously
called "last_resort", located in subdirectory "src/
These files are:
. parser_ll.c
. parser_ly.h
. parser_yy.c .
They should be copied in the directory where object files are
created, and "touch"ed so that their modification date is more
recent than those of the corresponding "parser_ll.l" and
"parser_yy.y" files. Then cross your fingers and hope it
compiles properly on your system. Else, you will have to dig in
their code to have them compile properly...
0>-:--- INSTALL.txt ( 6% (67,0) [S,#ge,#guix,#r,#sc,#guile,&,#a,#se,#n] (Text +2 Fly Fill

# 2) Configuration

## 2.1) Creating the "Makefile.inc" file

Go to the "src/" directory.

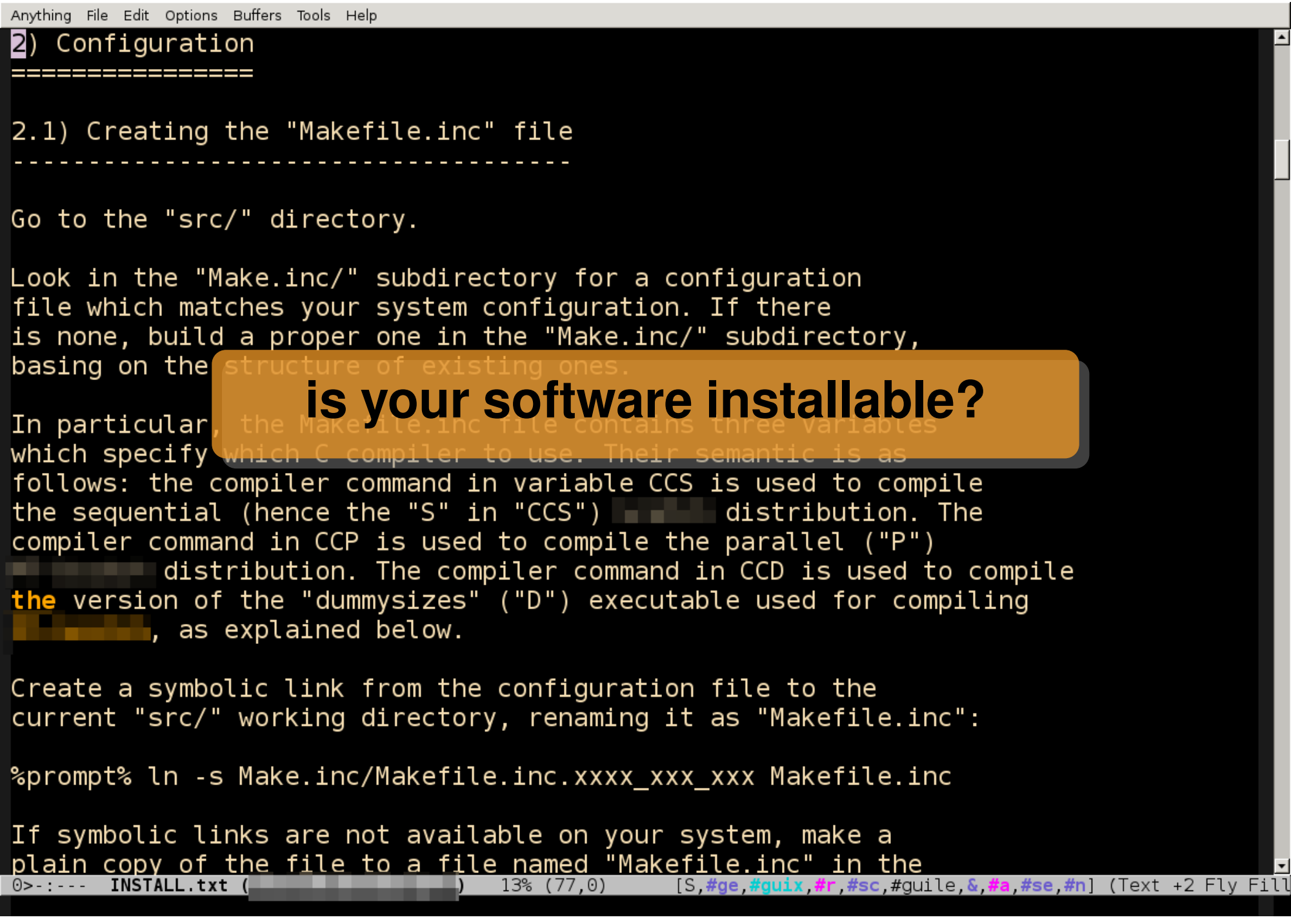
Look in the "Make.inc/" subdirectory for a configuration
file which matches your system configuration. If there
is none, build a proper one in the "Make.inc/" subdirectory,
basing on the structure of existing ones.

In particular, the Makefile.inc file contains three variables
which specify which C compiler to use. Their semantic is as
follows: the compiler command in variable CCS is used to compile
the sequential (hence the "S" in "CCS") [illegible] distribution. The
compiler command in CCP is used to compile the parallel ("P")
[illegible] distribution. The compiler command in CCD is used to compile
the version of the "dummysizes" ("D") executable used for compiling
[illegible], as explained below.

Create a symbolic link from the configuration file to the
current "src/" working directory, renaming it as "Makefile.inc":

```
%prompt% ln -s Make.inc/Makefile.inc.xxxx_xxx_xxx Makefile.inc
```

If symbolic links are not available on your system, make a
plain copy of the file to a file named "Makefile.inc" in the


Anything File Edit Options Buffers Tools Help
2) Configuration
==========
2.1) Creating the "Makefile.inc" file
Go to the "src/" directory.
Look in the "Make.inc/" subdirectory for a configuration
file which matches your system configuration. If there
is none, build a proper one in the "Make.inc/" subdirectory,
basing on the
is your software installable?
In particular,
which specify
follows: the compiler command in variable CCS is used to compile
the sequential (hence the "S" in "CCS")
distribution. The
compiler command in CCP is used to compile the parallel ("P")
distribution. The compiler command in CCD is used to compile
the version of the "dummysizes" ("D") executable used for compiling
, as explained below.
Create a symbolic link from the configuration file to the
current "src/" working directory, renaming it as "Makefile.inc":
%prompt% ln -s Make.inc/Makefile.inc.xxxx_xxx_xxx Makefile.inc
If symbolic links are not available on your system, make a
plain copy of the file to a file named "Makefile.inc" in the
0>-:--- INSTALL.txt 13% (77,0) [S,#ge,#guix,#r,#sc,#guile,&,#a,#se,#n] (Text +2 Fly Fill

# what do users want?

- `./configure`

- `make`
- `make check`
- `make install`

# what do users want?

- `./configure --prefix=$HOME/soft`

- `make`
- `make check`
- `make install`

# what do users want?

- `./configure --prefix=$HOME/soft --with-color`

- `make`
- `make check`
- `make install`

# what do users want?

- `./configure --prefix=$HOME/soft --with-color --disable-shared`
- `make`
- `make check`
- `make install`

# what do users want?

- `./configure --prefix=$HOME/soft --with-color --disable-shared CC=gcc-4.9`
- `make`
- `make check`
- `make install`

# 2

# goals & design

# autotools goals

- **portability aid**
- friendly **user interface**
- **no additional dependencies** for the user

# autotools goals

- **portability aid**
  - mostly among Unix variants
- friendly **user interface**

- **no additional dependencies** for the user

# autotools goals

- **portability aid**
  - mostly among Unix variants
- friendly **user interface**
  - easy way to install the software
  - standardized (GNU Coding Standards)
- **no additional dependencies** for the user

# autotools goals

- **portability aid**
  - mostly among Unix variants
- friendly **user interface**
  - easy way to install the software
  - standardized (GNU Coding Standards)
- **no additional dependencies** for the user
  - only requires POSIX `sh` and `make`!

*“The primary goal of Autoconf is making the **user’s** life easier; making the **maintainer’s** life easier is only a secondary goal.”*

— Autoconf manual

# “autotools”?

- Autoconf
- Automake
- Libtool
- Gnulib

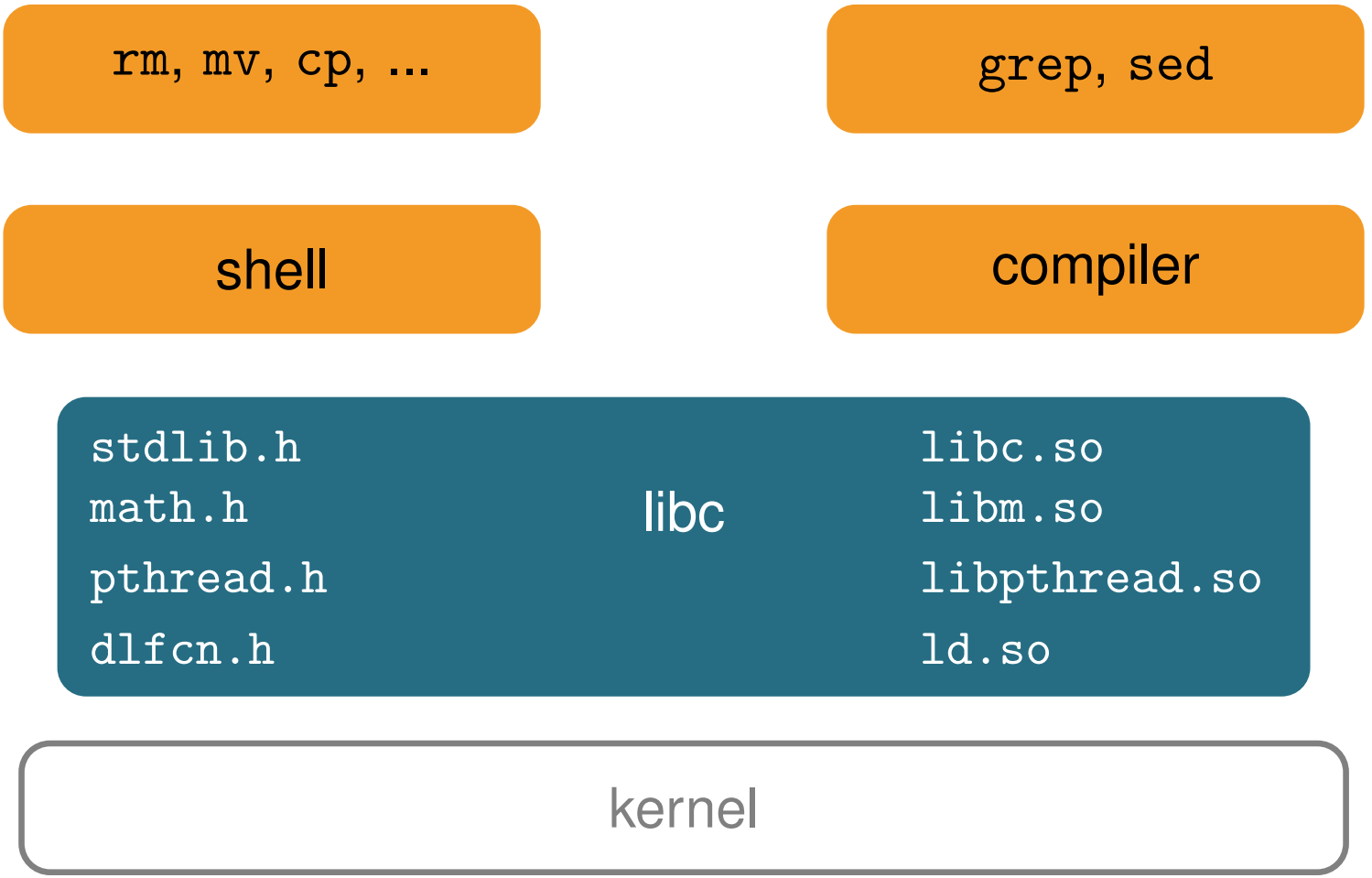
# 3

# on portability

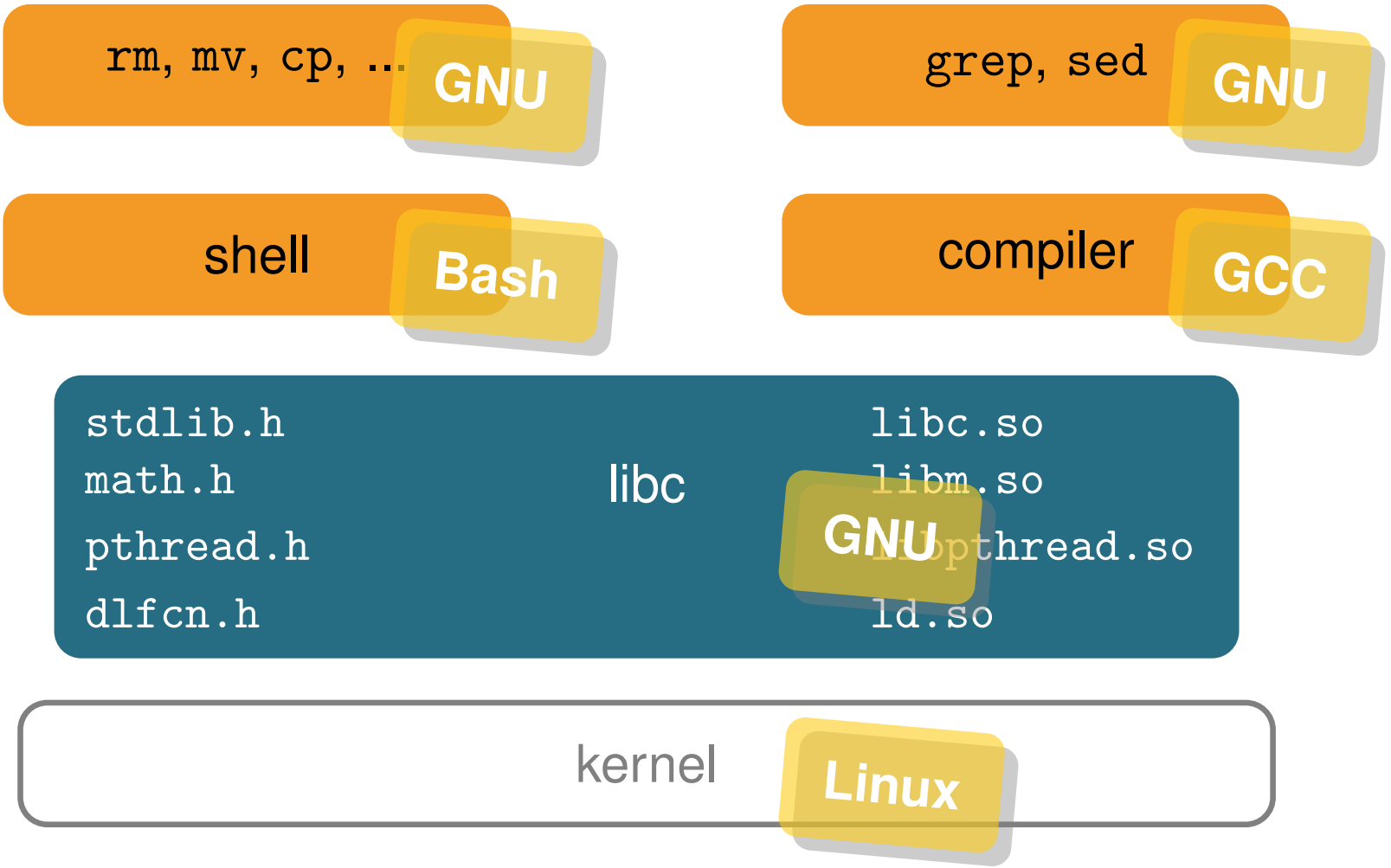
# being “portable” to different...

- shells
- `make` implementations
- C library features
- C library quirks (non-conformance, bugs, etc.)
- compilers
- linkers
- loaders
- ...

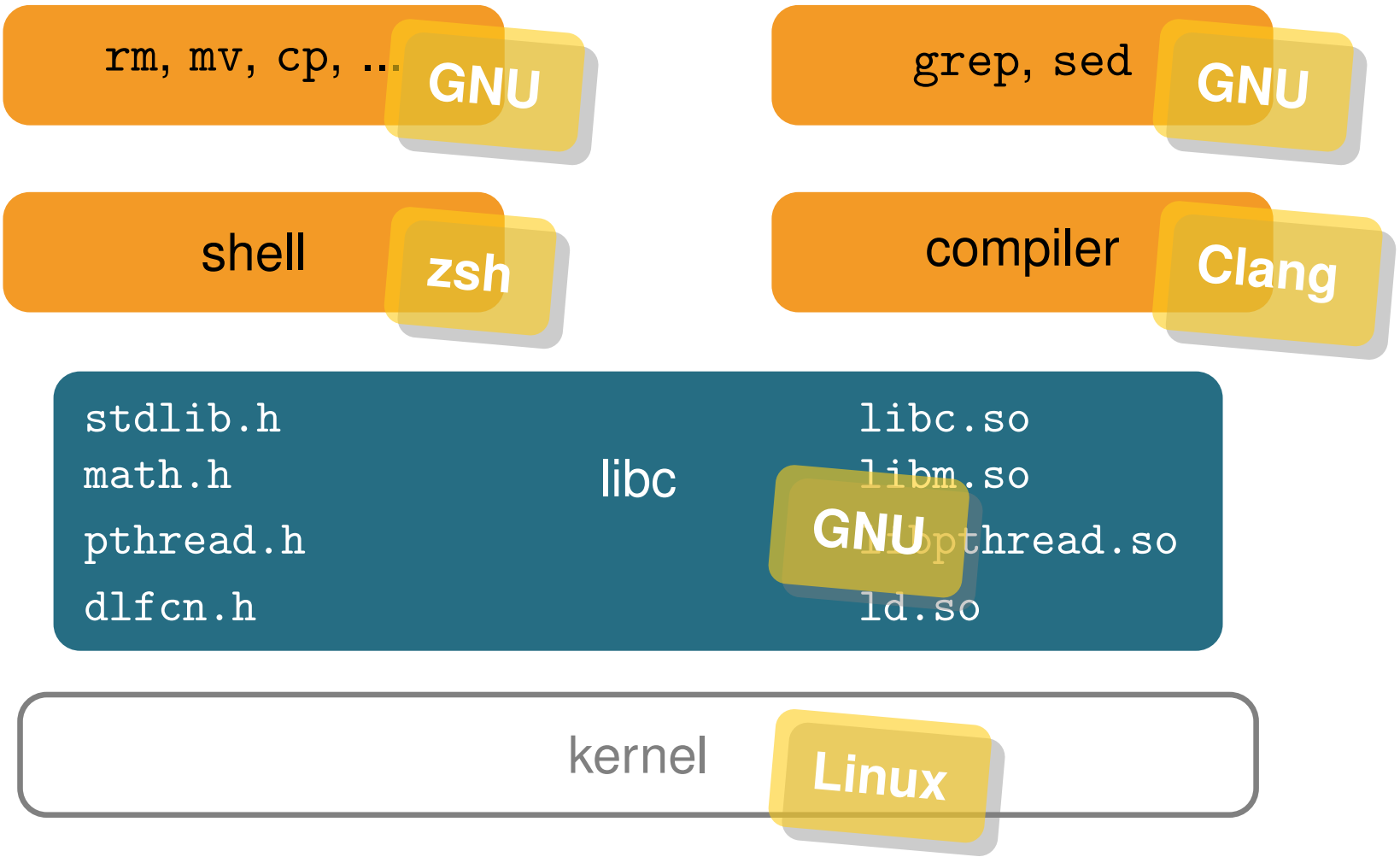
# systemology



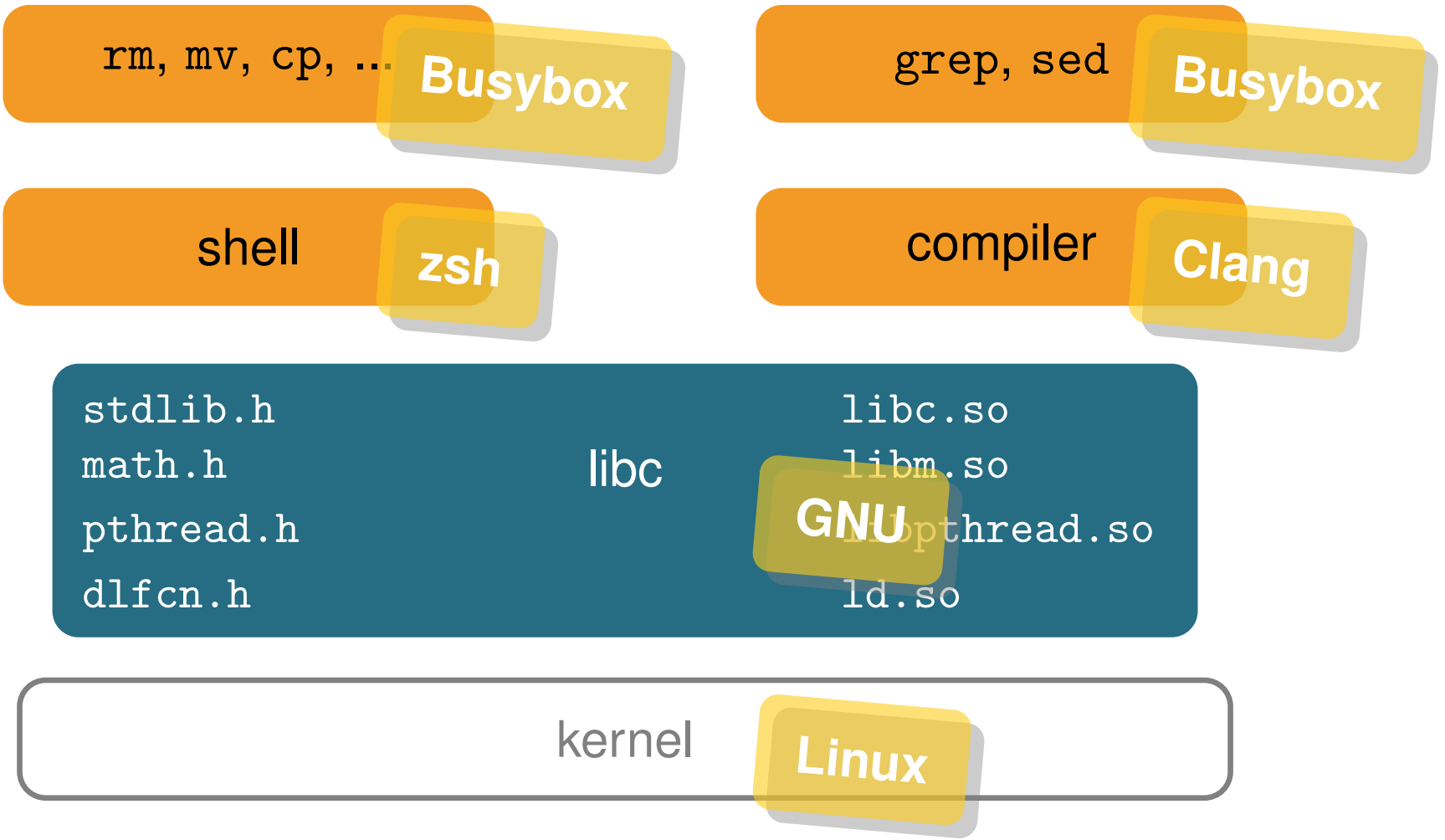
# systemology



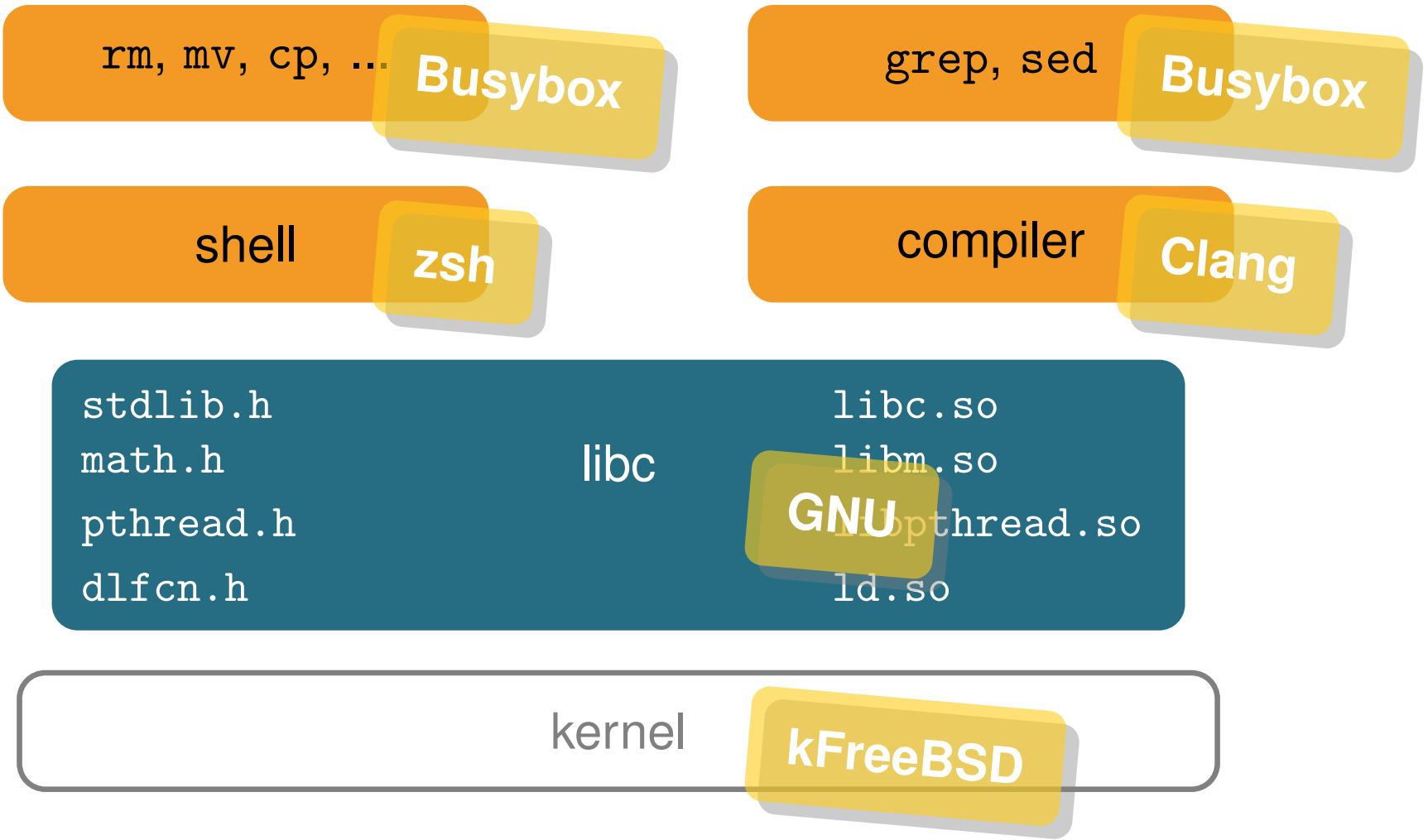
# systemology



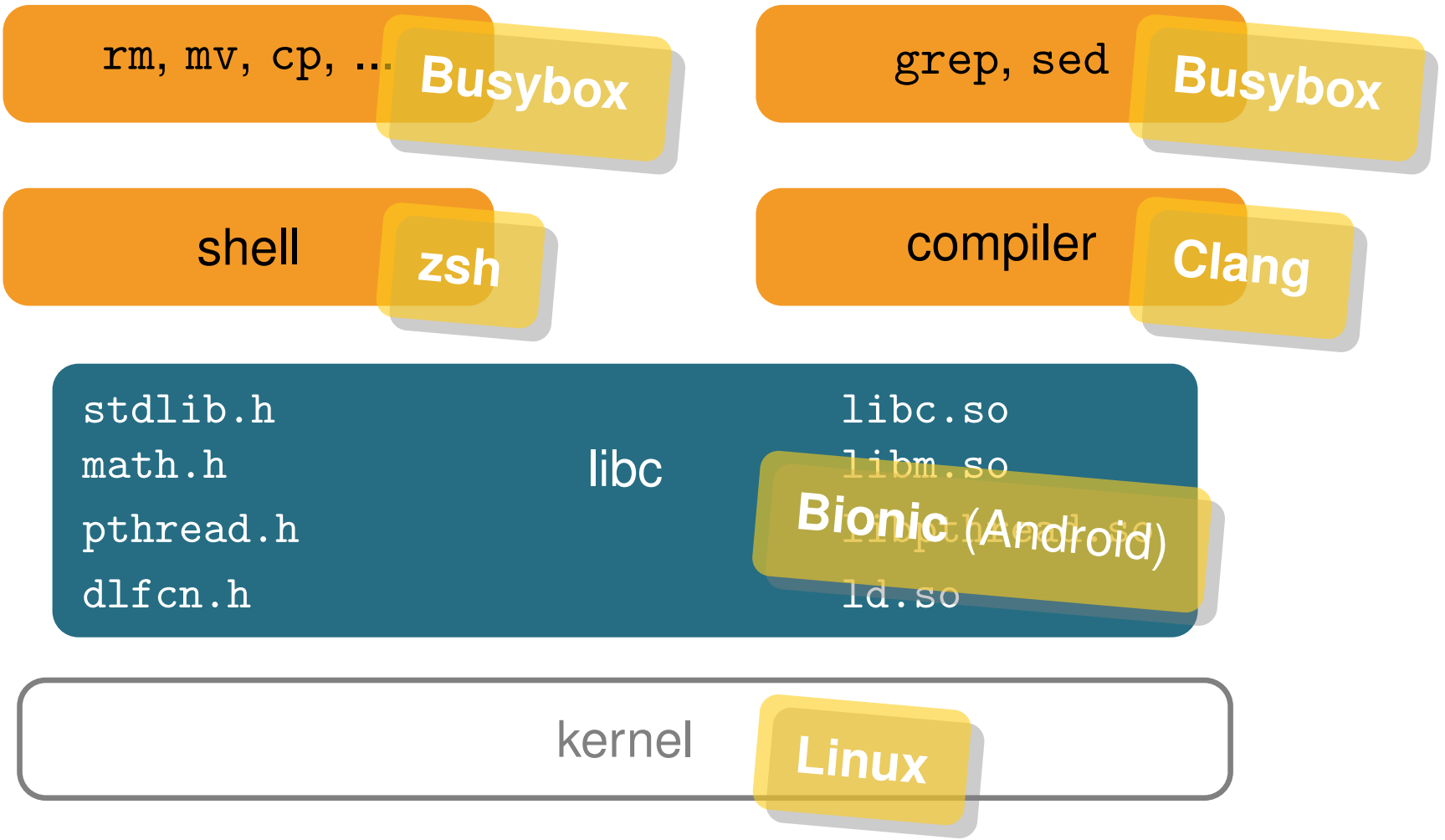
# systemology



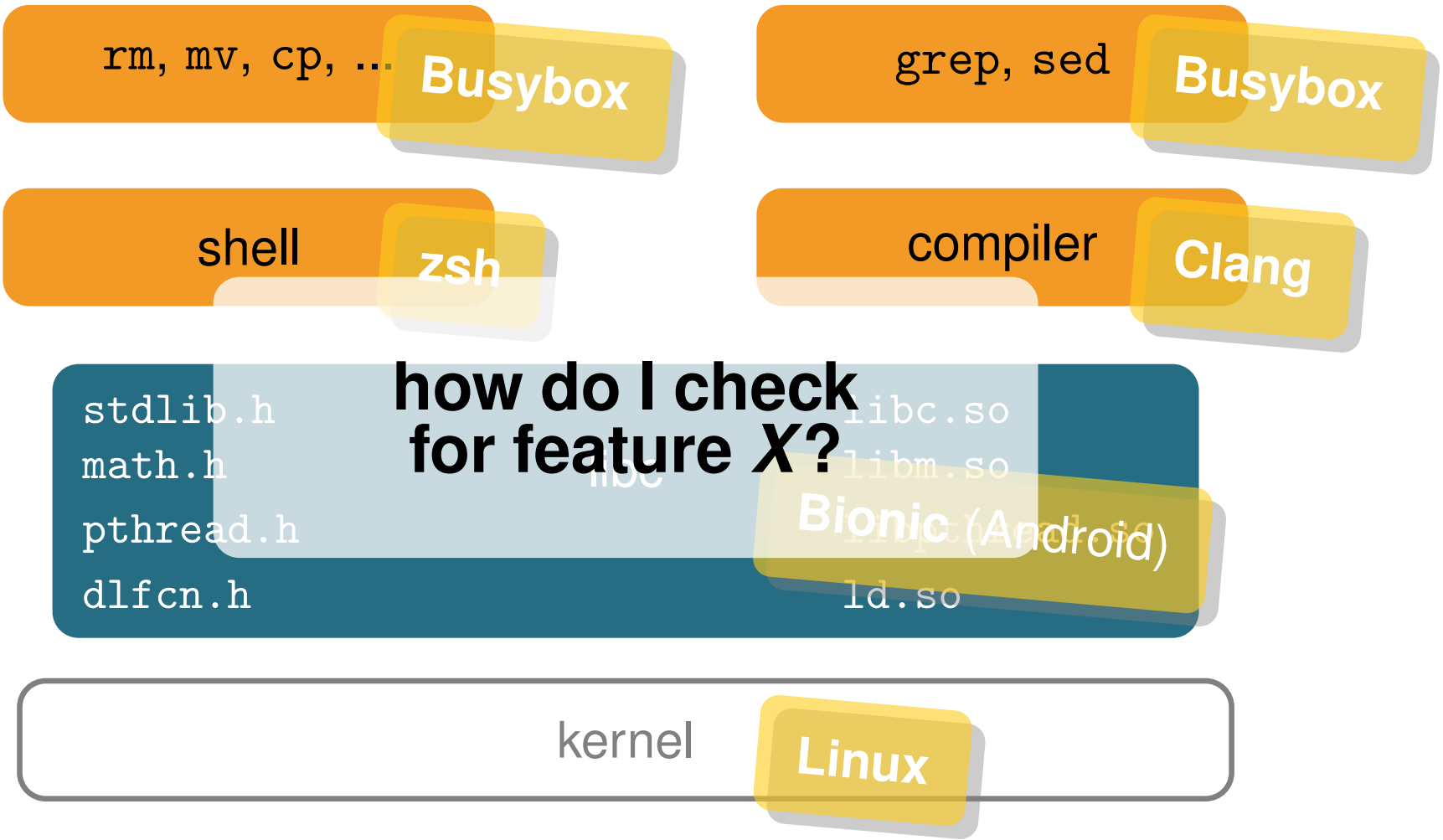
# systemology



# systemology



# systemology



# identity test

```
#ifdef __linux__
# include <langinfo.h>
#endif
```

# identity test

```
#ifdef __linux__
# include <langinfo.h>
#endif

if [ `uname` = Linux ]
then
  CC=gcc ; SHELL=/bin/bash
  ...
fi
```

# identity test

```
#ifdef __linux__
# include <langinfo.h>
#endif

if [ `uname` = Linux ]
then
  CC=gcc ; SHELL=/bin/bash
  ...
fi

gcc -DOS_LINUX=1 ...
```

# identity test

```
#ifdef __linux__
# include <langinfo.h>
#endif

if [ `uname` = Linux ]
then
  CC=gcc ; SHELL=/bin/bash
  ...
fi

gcc -DOS_LINUX=1 ...
```

wrong

# solution: feature tests

- AC_PATH_PROG
- AC_CHECK_LIB
- AC_CHECK_HEADER
- AC_COMPILE_IFELSE
- AC_LINK_IFELSE
- ...

check for **specific features**

# solution: feature tests

- `AC_PATH_PROG`
- `AC_CHECK_LIB`
- `AC_CHECK_HEADER`
- `AC_COMPILE_IFELSE`
- `AC_LINK_IFELSE`
- ...

# going further...

**Gnulib**: a portability library

`http://www.gnu.org/software/gnulib/`

# 4

# action

# running example

```
git://scm.gforge.inria.fr/autotools-demo/
                    autotools-demo.git
```

- C program
- requirements: GNU Scientific Library, GNU Plotutils
- computes an FFT, plots it

# road map

0. **hand-written** makefile
1. **Autoconf** + simple makefile template
2. Autoconf + **Automake**
3. + **optional** libplot support
4. + **tests**
5. + turn plotting into a **Libtool** library

each step has a corresponding branch in
`git://scm.gforge.inria.fr/.../autotools-demo.git`

# road map

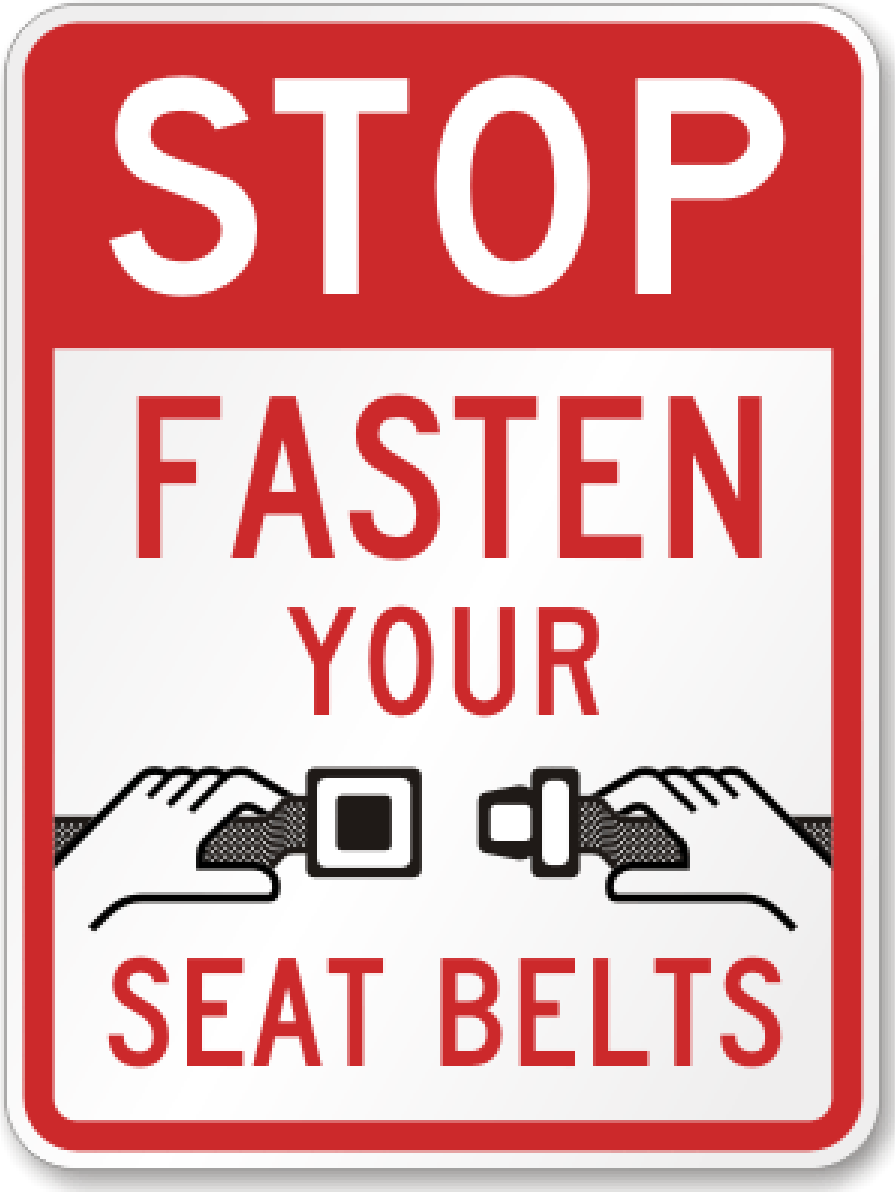
0. **hand-written** makefile
1. **Autoconf** + simple makefile template
2. Autoconf + **Automake**
3. + **optional** libplot support
4. + **tests**
5. + turn plotting into a **Libtool** library

**all that in 25mn!**

each step has a corresponding branch in
`git://scm.gforge.inria.fr/.../autotools-demo.git`

# road map

0. **hand-writt**
1. **Autoconf**
2. Autoconf +
3. + **optional**
4. + **tests**
5. + turn plott




each step h

git://scm.

s-demo.git

# step 0: hand-written makefile

**branch** `0-hand-written-makefile`

# step 0: hand-written makefile

**branch** `0-hand-written-makefile`

Yay it builds on *my* machine!

# step 1: our first `configure.ac`

**branch** `1-autoconf+simple-makefile-template`

- **tools**
  - commands: `autoscan`, `autoconf`
  - macros: `AC_CHECK_LIB`, `AC_CHECK_HEADERS`

# step 1: our first `configure.ac`

**branch** `1-autoconf+simple-makefile-template`

- **tools**
  - commands: `autoscan`, `autoconf`
  - macros: `AC_CHECK_LIB`, `AC_CHECK_HEADERS`
- **gains**
  - user interface to specify installation directories
  - tools to check for programming environment features
  - template instantiation: `foo.in` $\rightarrow$ `foo`
  - `config.h`, `config.log`

# step 2: using Automake

**branch** `2-autoconf+automake`

- **tools**
  - commands: `automake`, `autoreconf`
  - naming of `Makefile.am` variables, installation directories

# step 2: using Automake

**branch** `2-autoconf+automake`

- **tools**
  - commands: `automake`, `autoreconf`
  - naming of `Makefile.am` variables, installation directories
- **gains**
  - honor installation directories: `bindir`, etc.
  - honor user variables: `CFLAGS`, `DESTDIR`, etc.
  - rules: `all`, `install`, `uninstall`

# step 2: using Automake

**branch** `2-autoconf+automake`

- **tools**
  - commands: `automake`, `autoreconf`
  - naming of `Makefile.am` variables, installation directories
- **gains**
  - honor installation directories: `bindir`,
  - honor user variables: `CFLAGS`, `DESTDI`
  - rules: `all`, `install`, `uninstall`
- **bonuses!**
  - distribution: `dist` & `distcheck`
  - `--enable-silent-rules`
  - dependency tracking among `.c` and `.h`
  - rules to rebuild `configure` & co.
  - out-of-source builds
  - cross-compilation support
  - ...

# step 3: making a dependency optional

**branch** `3-autoconf+automake+conditional-libplot`

- **tools**
  - in `configure.ac`: `AM_CONDITIONAL`, `AC_ARG_ENABLE`
  - in `Makefile.am`: `if`, `+=`

# step 3: making a dependency optional

**branch** `3-autoconf+automake+conditional-libplot`

- **tools**
  - in `configure.ac`: `AM_CONDITIONAL`, `AC_ARG_ENABLE`
  - in `Makefile.am`: `if`, `+=`
- **gains**
  - optional part of the build
  - user can choose

# step 4: adding tests

**branch** `4-autoconf+automake+tests`

- **tools**
  - in `Makefile.am`: `TESTS`, `SH_LOG_EXTENSIONS`, `SH_LOG_COMPILER`

# step 4: adding tests

**branch** `4-autoconf+automake+tests`

- **tools**
  - in `Makefile.am`: `TESTS`, `SH_LOG_EXTENSIONS`, `SH_LOG_COMPILER`
- **gains**
  - portable makefile (GNU Make, BSD Make, etc.)
  - `make check`
  - `test-suite.log` and `TEST.log`

# step 4: adding tests

**branch** `4-autoconf+automake+tests`

- **tools**
  - in `Makefile.am`: `TESTS`, `SH_LOG_EXTENSIONS`, `SH_LOG_COMPILER`
  - in `configure.ac`: `color-tests`
- **gains**
  - portable makefile (GNU Make, BSD
  - `make check -j4`
  - `test-suite.log` and `TEST.log`
- **bonuses!**
  - color!
  - parallel tests!
  - `make recheck`
  - reStructuredText logs

# step 5: building a library

**branch** `5-autoconf+automake+libtool`

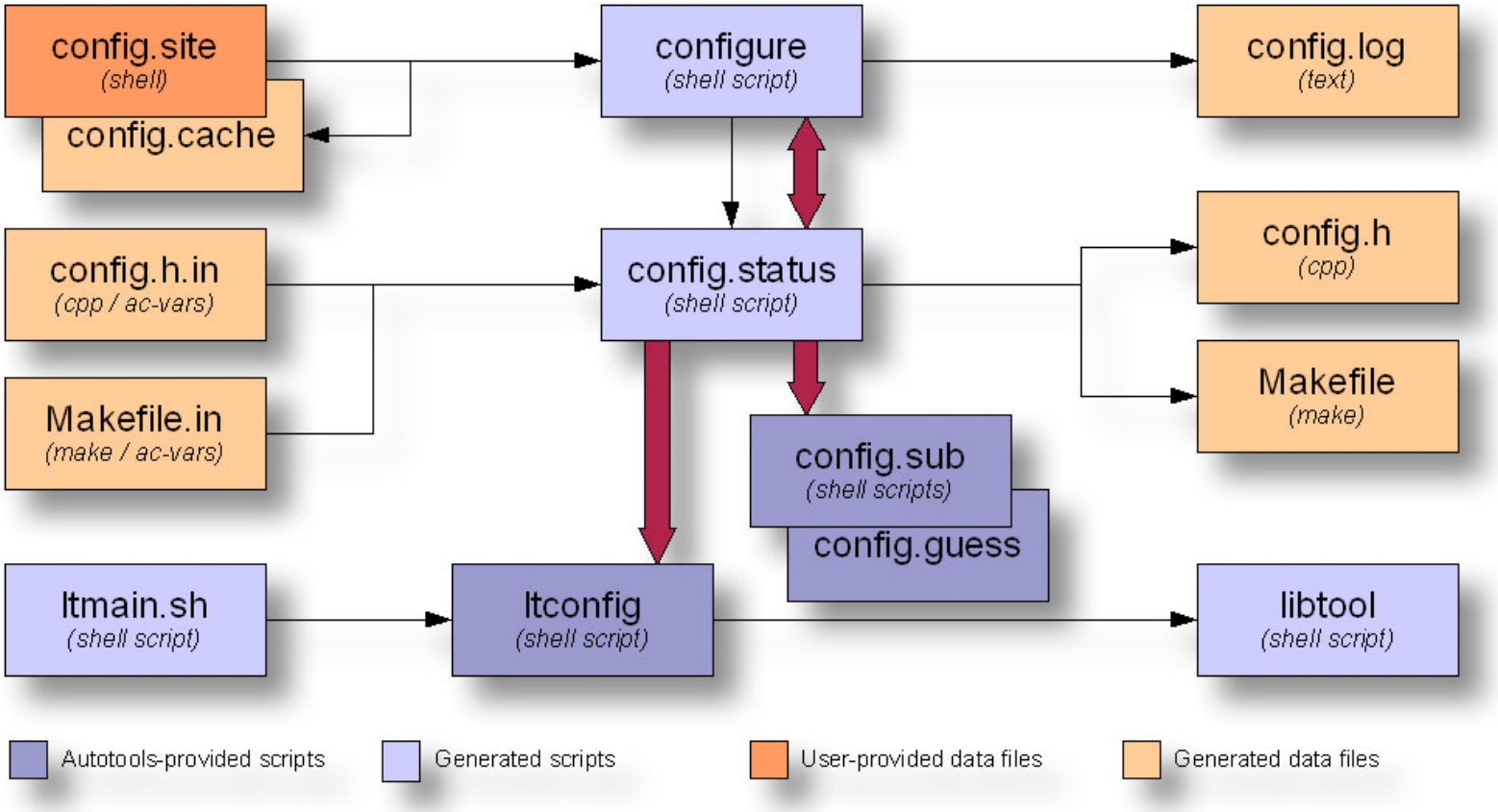
- **tools**
  - commands: `libtoolize`, `automake --add-missing`
  - in `Makefile.am`: `LT_LIBRARIES`, `.la`
  - in `configure.ac`: `LT_INIT`

# step 5: building a library

**branch** `5-autoconf+automake+libtool`

- **tools**
  - commands: `libtoolize`, `automake --add-missing`
  - in `Makefile.am`: `LT_LIBRARIES`, `.la`
  - in `configure.ac`: `LT_INIT`
- **gains**
  - portable shared libraries (woow!)
  - `--disabled-shared`, `--disable-static`, etc.
  - shared lib usable before installation
  - shared lib relinked with `RUNPATH` set upon install
  - `.la` contains dependent libs (for static linking)
  - `.la` contains extra linker flags

# the big picture




source: `http://www.freesoftwaremagazine.com/articles/brief_introduction_to_gnu_autotools`

# the big p





`make` and `autoreconf` know how to do that

source: `http://www.freesoftwaremagazine.com/articles/brief_introduction_to_gnu_autotools`

# 5

# wrap up

*“Those who do not understand Autoconf are condemned to reinvent it, poorly.”*

— Autoconf manual

# summary

- autotools make **the user's life** easier

# summary

- autotools make **the user's life** easier
- ... yet neatly support **maintenance work**

# summary

- autotools make **the user's life** easier
- ... yet neatly support **maintenance work**
- a *lot* of embedded **portability & UI expertise**

*ludovic.courtes@inria.fr*

`http://sed.bordeaux.inria.fr/`