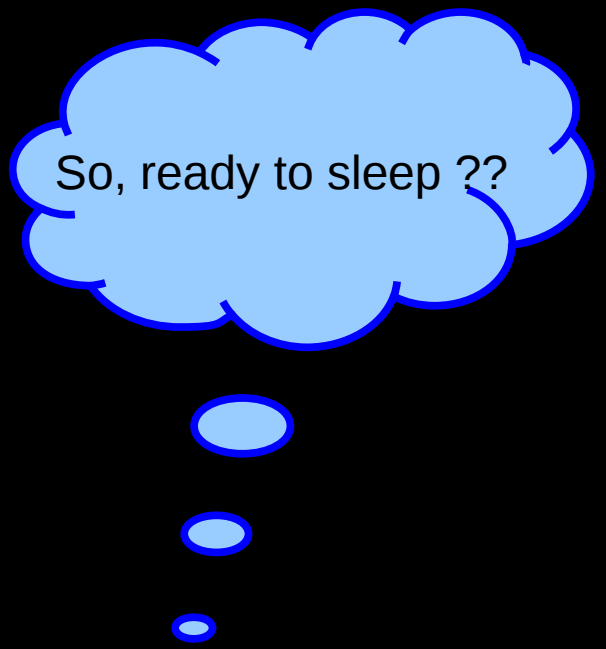
Hello Autotools!



Rakesh Pandit,
rpandit@redhat.com

Introduction

Why ??

Standard File System Hierarchy

Directory variable	Value
prefix	/usr/
exec-prefix	prefix
bindir	exec-prefix/bin
libdir	exec-prefix/lib
.....	.....
includedir	prefix/include/
datarootdir	prefix/share
datadir	datarootdir
mandir	datadir/man/
infodir	datarootdir/info/
.....	.....

Standard Installation Procedure

```
~ % tar zxf abcdef-1.0.tar.gz
~ % cd abcdef-1.0

~/abcdef-1.0 % ./configure
...

~/abcdef-1.0 % make
...

~/abcdef-1.0 % make check
...

~/abcdef-1.0 % su
Password:
/home/rakesh/abc-1.0 # make install
...

/home/rakesh/abcdef-1.0 # exit
~/abcdef-1.0 % make installcheck
...
```

Standard Configuration Variables

'./configure' automatically detects many settings.
You can force some of them using configuration variables.

```
        CC       C compiler command
    CFLAGS       C compiler flags
       CXX       C++ compiler command
   CXXFLAGS      C++ compiler flags
    LDFLAGS      linker flags
CPPFLAGS C/C++   preprocessor flags
```

... See './configure --help' for a full list.

Parallel Build Trees (a.k.a. VPATH Builds)

```
~ % tar zxf ~/abcdef-1.0.tar.gz
~ % cd abcdef-1.0
~/abcdef-1.0 % mkdir build && cd build
~/abcdef-1.0/build % ../configure
~/abcdef-1.0/build % make
...
```

Sources files are in ~/ abcdef-1.0/ ,
built files are all in ~/ abcdef-1.0/ build/ .

Parallel Build Trees For Multiple Architectures

Builds for multiple architectures can share the same source tree. Have the source on a (possibly read-only) shared directory.

```
~ % cd /nfs/src
/nfs/src % tar zxf ~/abcdef-1.0.tar.gz

Compilation on first host
~ % mkdir /tmp/host1 && cd /tmp/host1
/tmp/host1 % /nfs/src/abcdef-1.0/configure
/tmp/host1 % make && sudo make install

Compilation on second host
~ % mkdir /tmp/host2 && cd /tmp/host2
/tmp/host2 % /nfs/src/host2-1.0/configure
/tmp/host2 % make && sudo make install
```

Cross Compilation

```
~/abcdef-1.0 % ./configure

~/abcdef-1.0 % ./configure --build i686-pc-
linux-gnu --host i586-mingw32msvc
...

~/abcdef-1.0 % make
...

~/abcdef-1.0 % cd src; file hello.exe
hello.exe: MS Windows PE 32-bit Intel 80386
console executable not relocatable
```

Cross-compilation configure options:

‘--build=BUILD’ The system on which the package is built.

‘--host=HOST’ The system where built programs & libraries will run.

Building Binary Packages Using DESTDIR

~/abcdef-1.0 % ./configure --prefix /usr
...

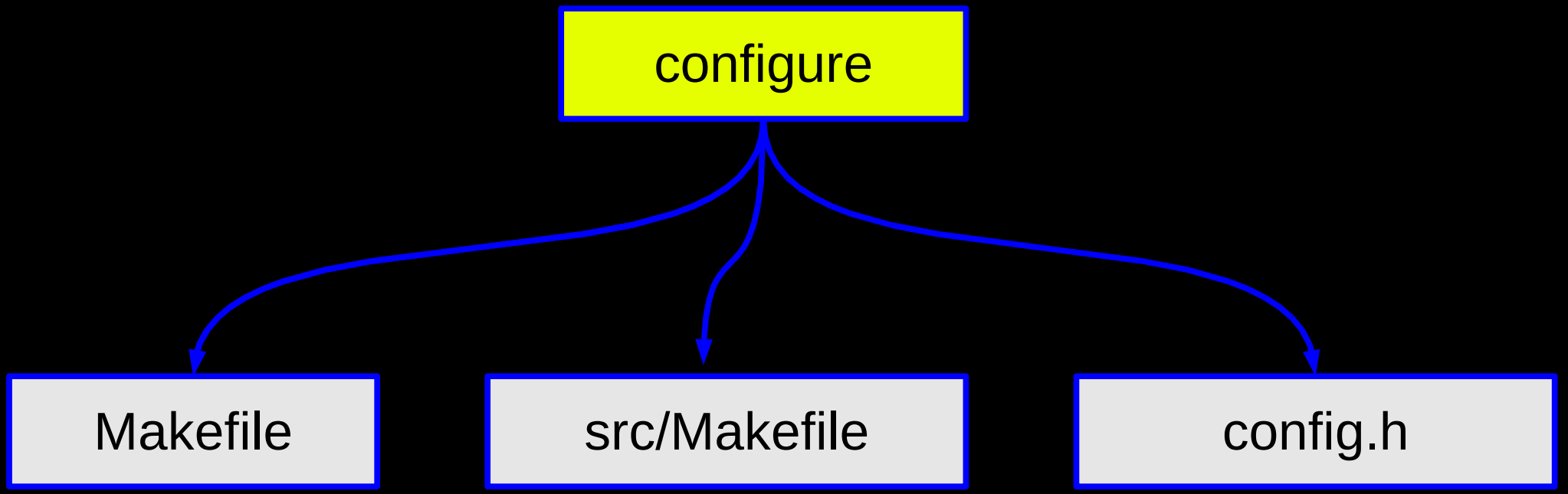
~/abcdef-1.0 % make
...

~/abcdef-1.0 % make DESTDIR=$HOME/inst install
...

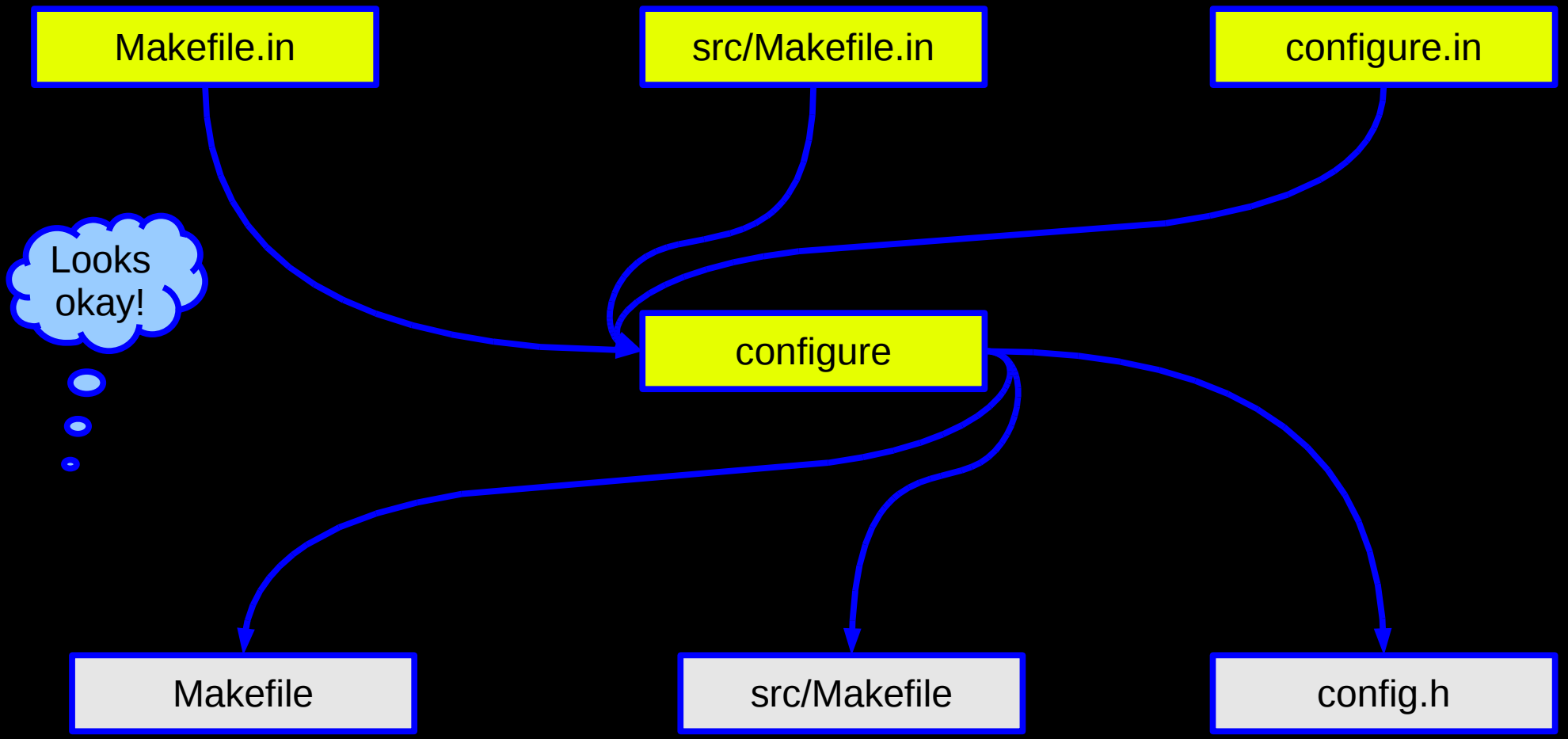
~/abcdef-1.0 % cd ~/inst

Will have all files installed.

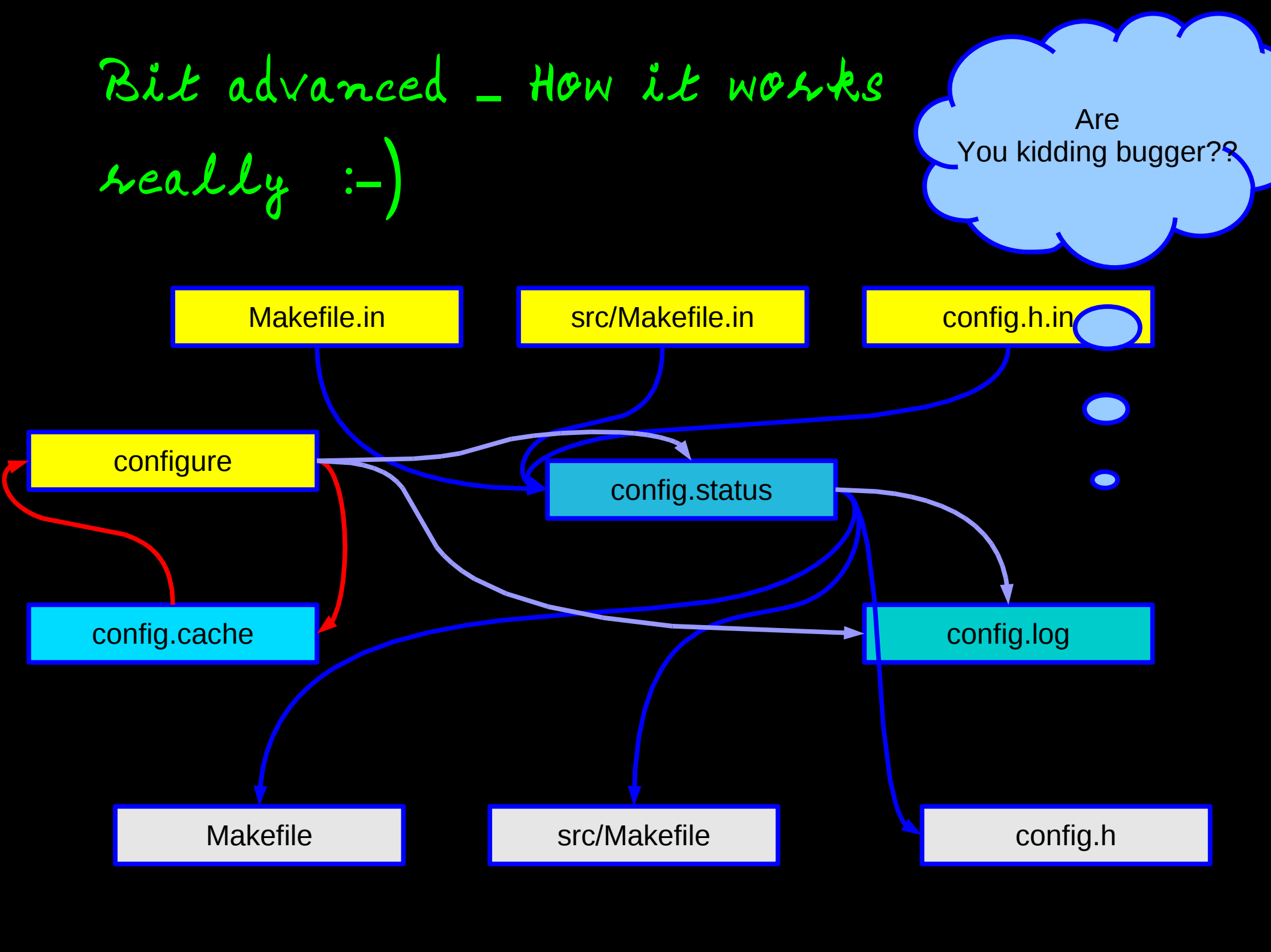
Project tarball configure file ?



How "simple configure" work ??



Bit advanced _ How it works really :-)



Where does these files came from in a project ??

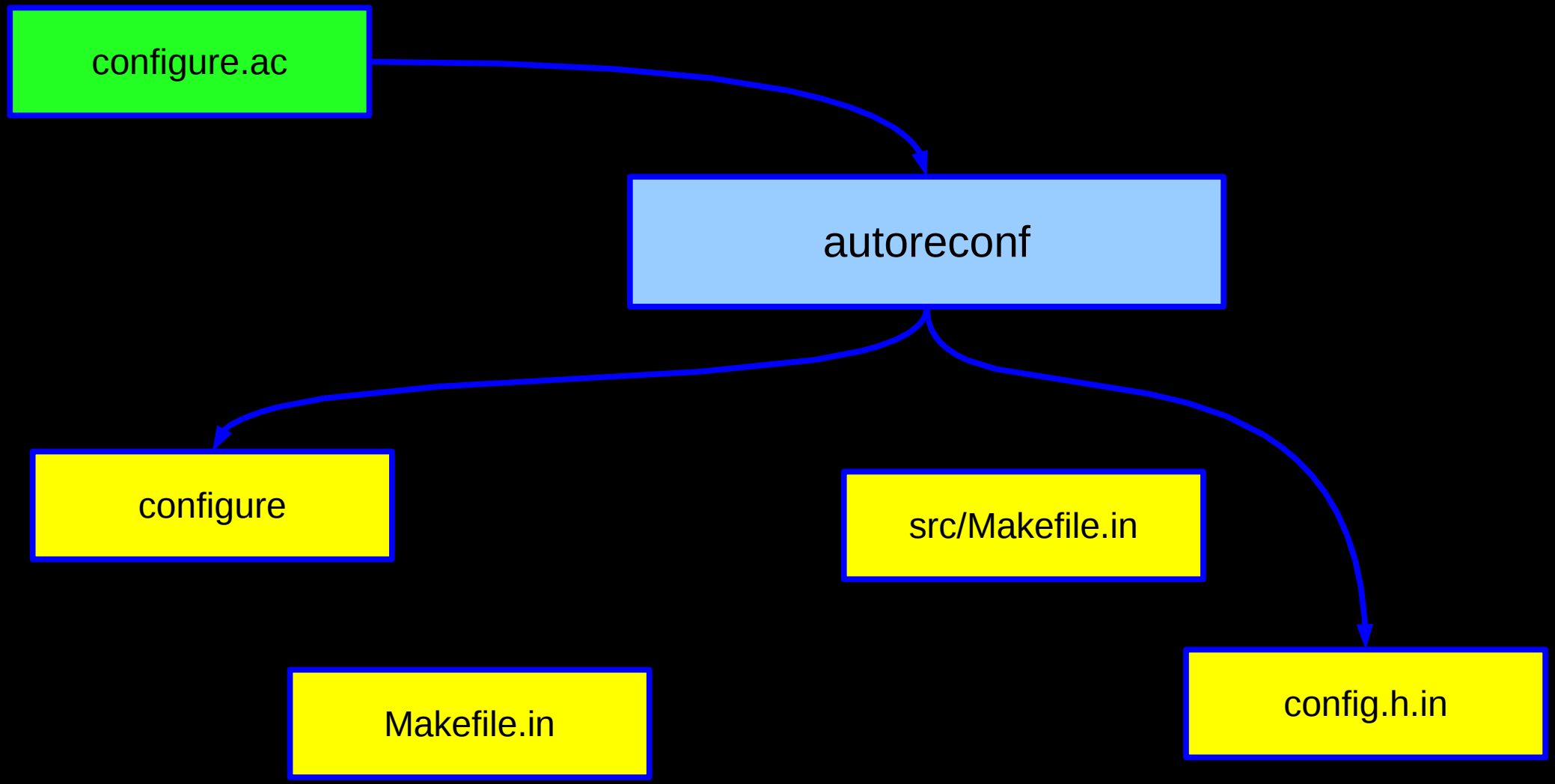
Makefile.in

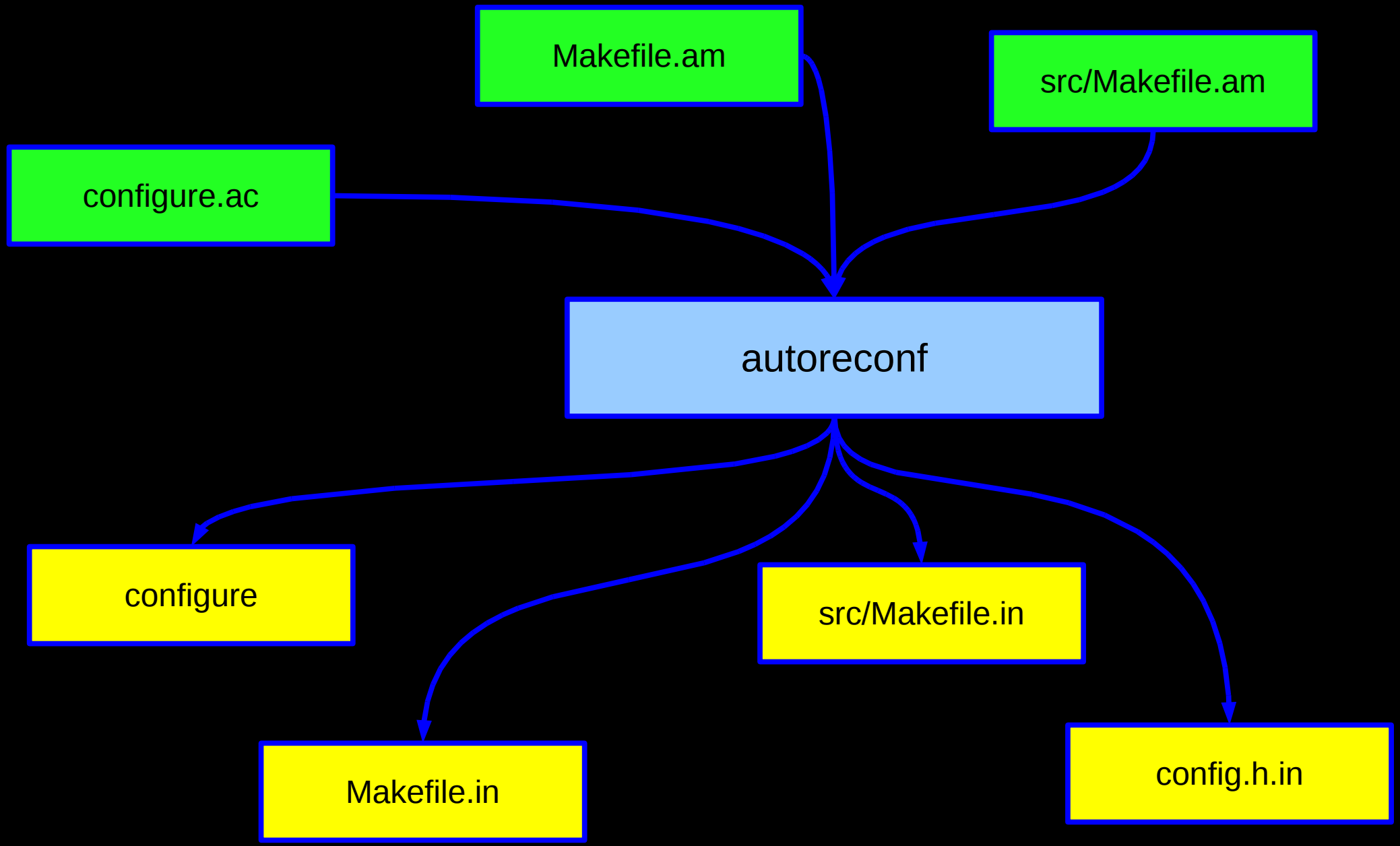
src/Makefile.in

config.h.in

configure

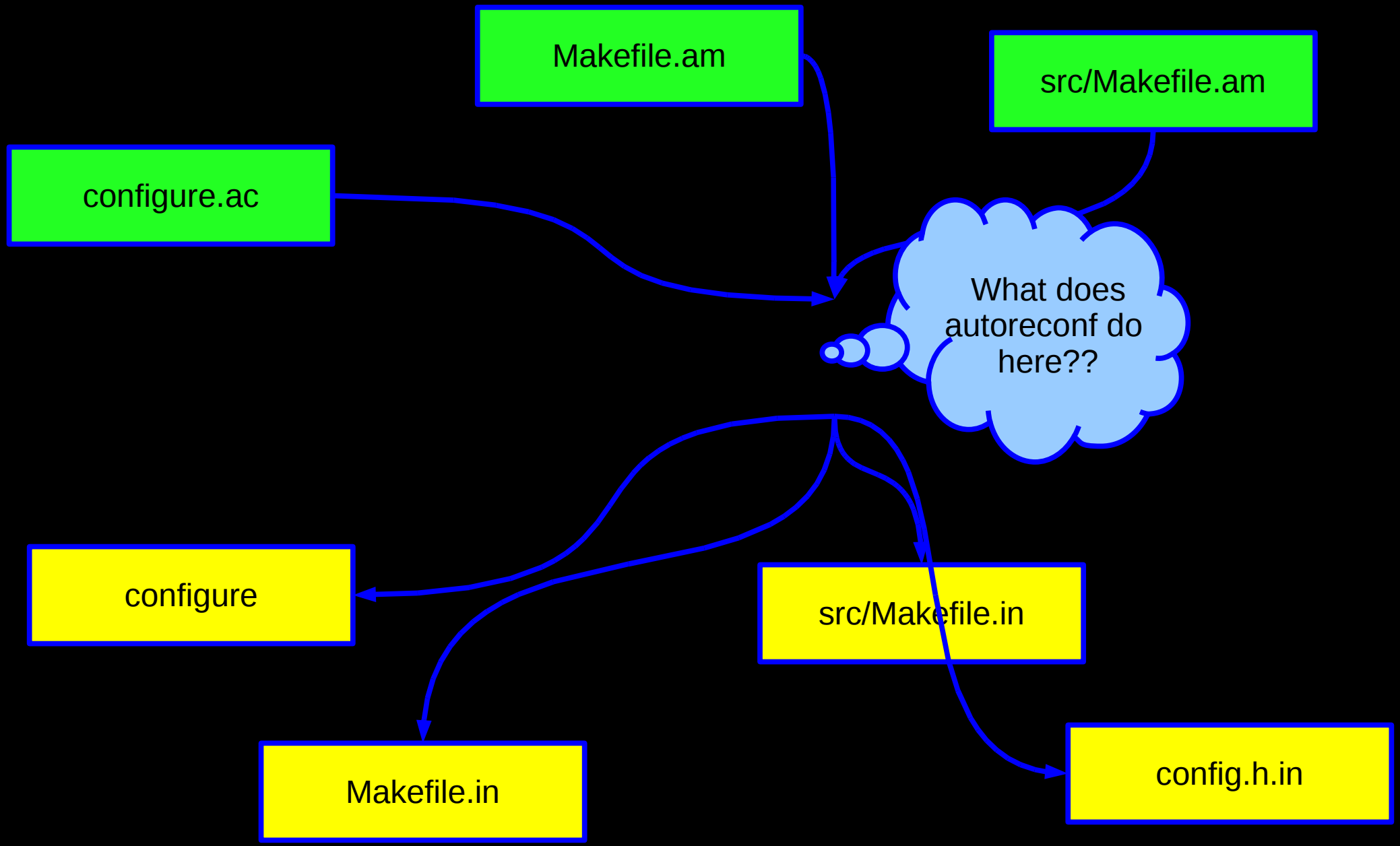
Yes AUTORECONF?




Makefile.am
src/Makefile.am
configure.ac
autoreconf
configure
src/Makefile.in
Makefile.in
config.h.in

Standard Makefile Targets

‘make all’	Build programs, libraries, documentation, etc. (Same as ‘make’.)
‘make install’	Install what needs to be installed.
‘make install-strip’	Same as ‘make install’, then strip debugging symbols.
‘make uninstall’	The opposite of ‘make install’.
‘make clean’	Erase what has been built
‘make distclean’	Additionally erase anything ‘./configure’ created.
‘make check’	Run the test suite, if any.
‘make installcheck’	Check the installed programs or libraries, if supported.
‘make dist’	Create PACKAGE-VERSION.tar.gz.
'make distcheck'	Same as make dist with testing of whether files packaged are same and installtion and unintallation of project works.

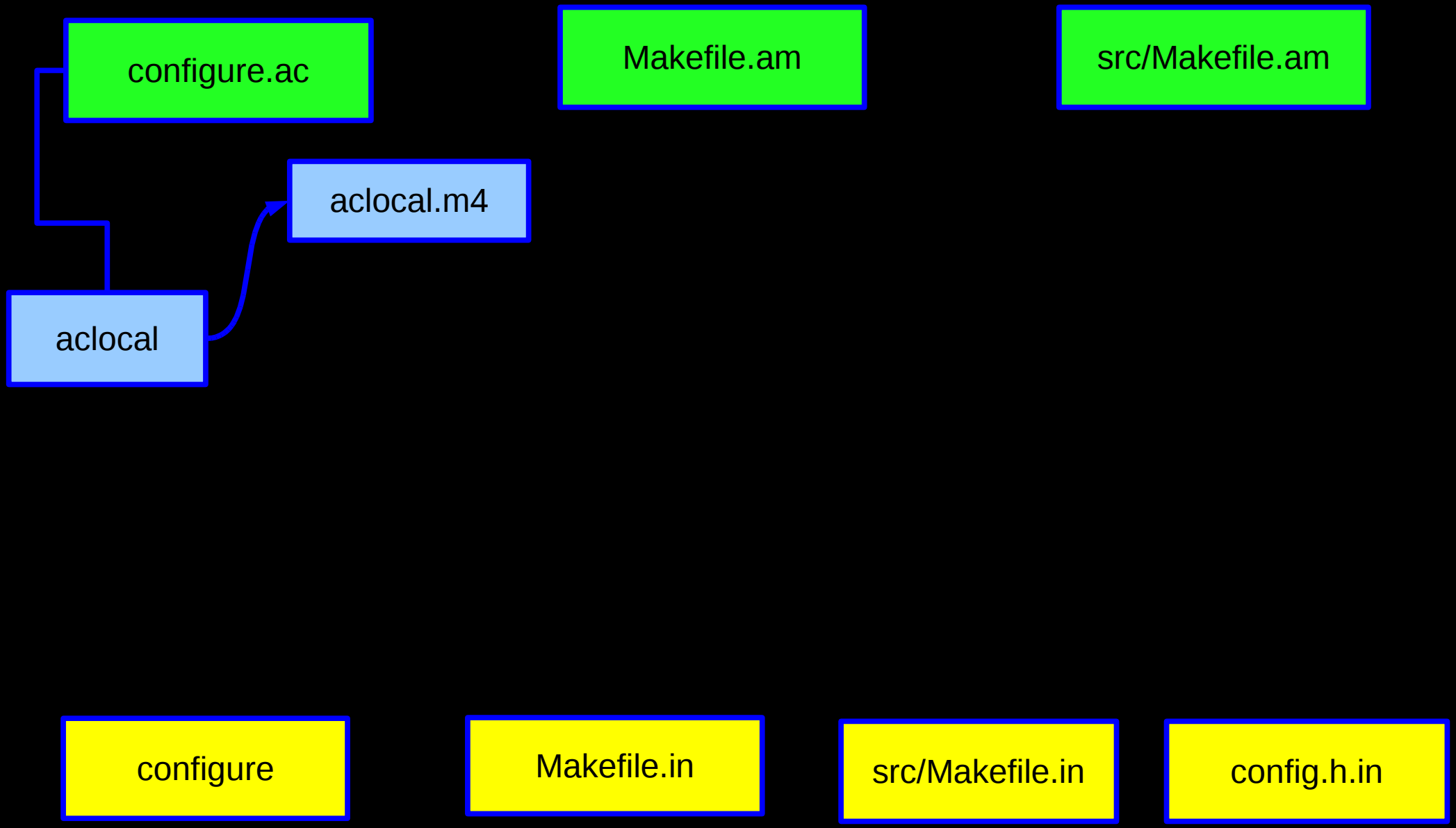

Makefile.am
src/Makefile.am
configure.ac
What does autoreconf do here??
configure
src/Makefile.in
Makefile.in
config.h.in

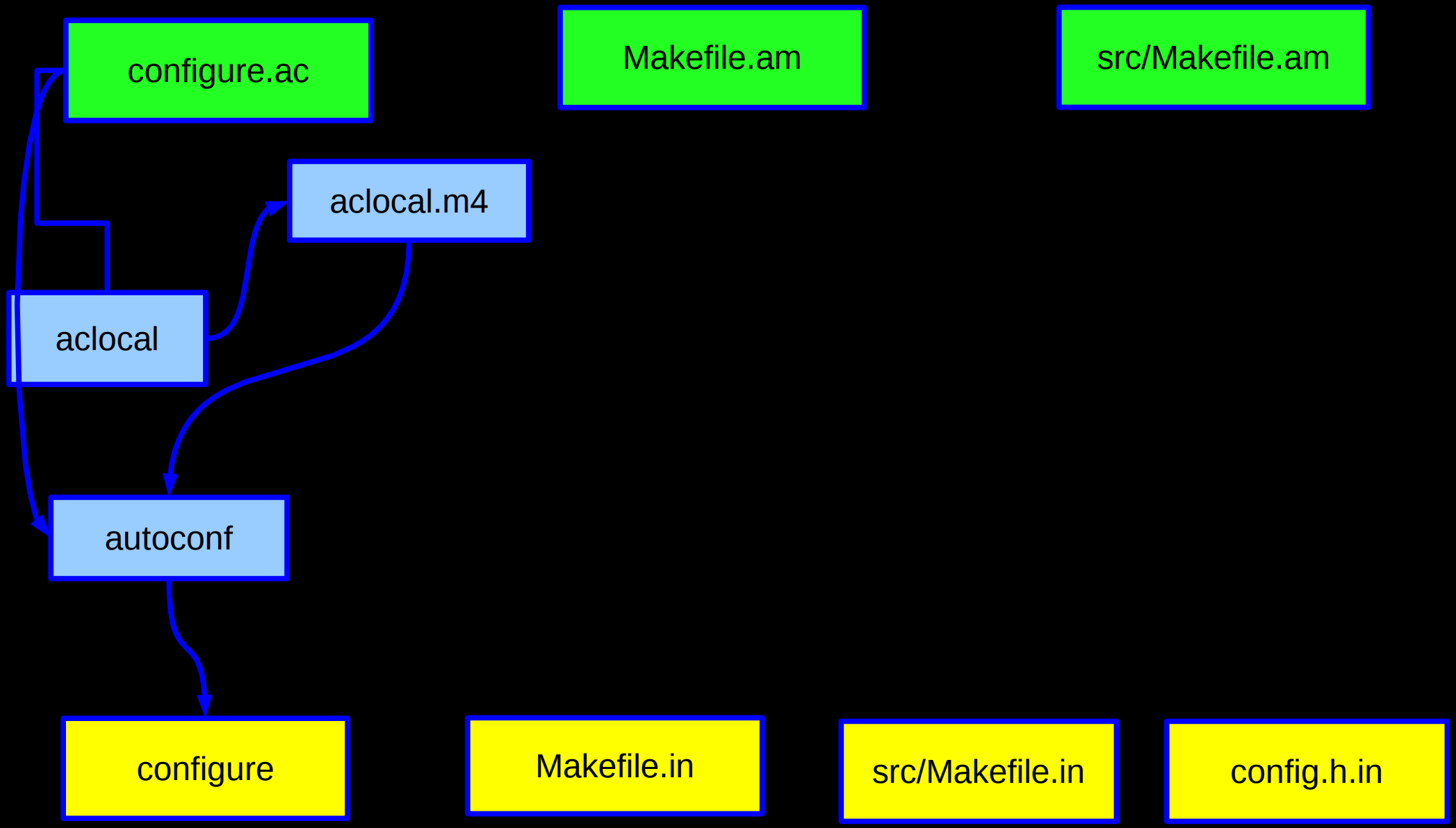
GNU AUTOCONF

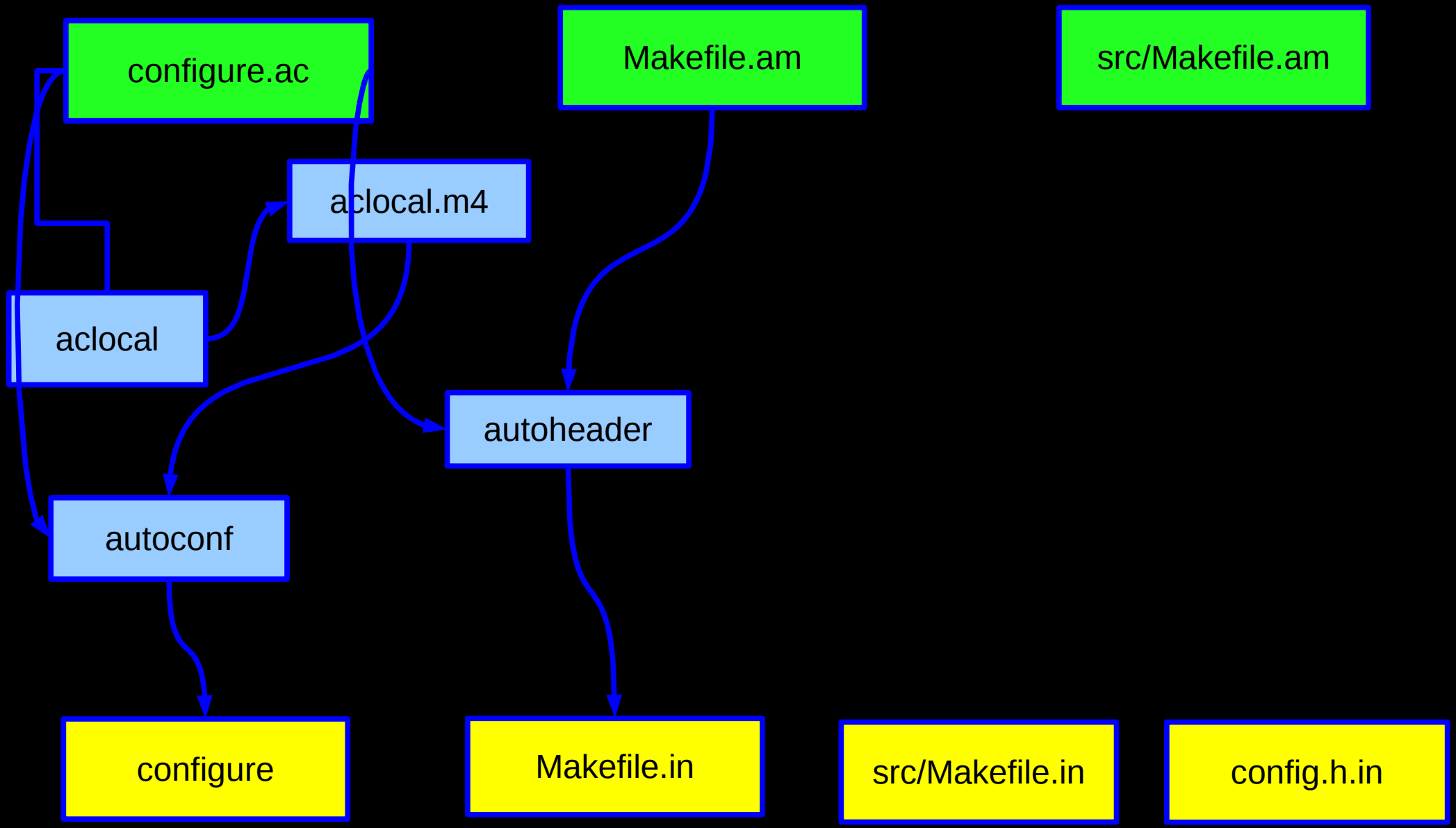
‘autoconf’	Create configure from configure.ac.
‘autoheader’	Create config.h.in fromconfigure.ac.
‘autoreconf’	Run all tools in the right order.
‘autoscan’	Scan sources for common portability problems, and related macros missing from configure.ac.
‘autoupdate’	Update obsolete macros in configure.ac.
‘ifnames’	Gather identifiers from directives.
‘autom4te’	The heart of Autoconf. It drives M4 and implements the features used by most of the above tools. Useful for creating more than just configure files.

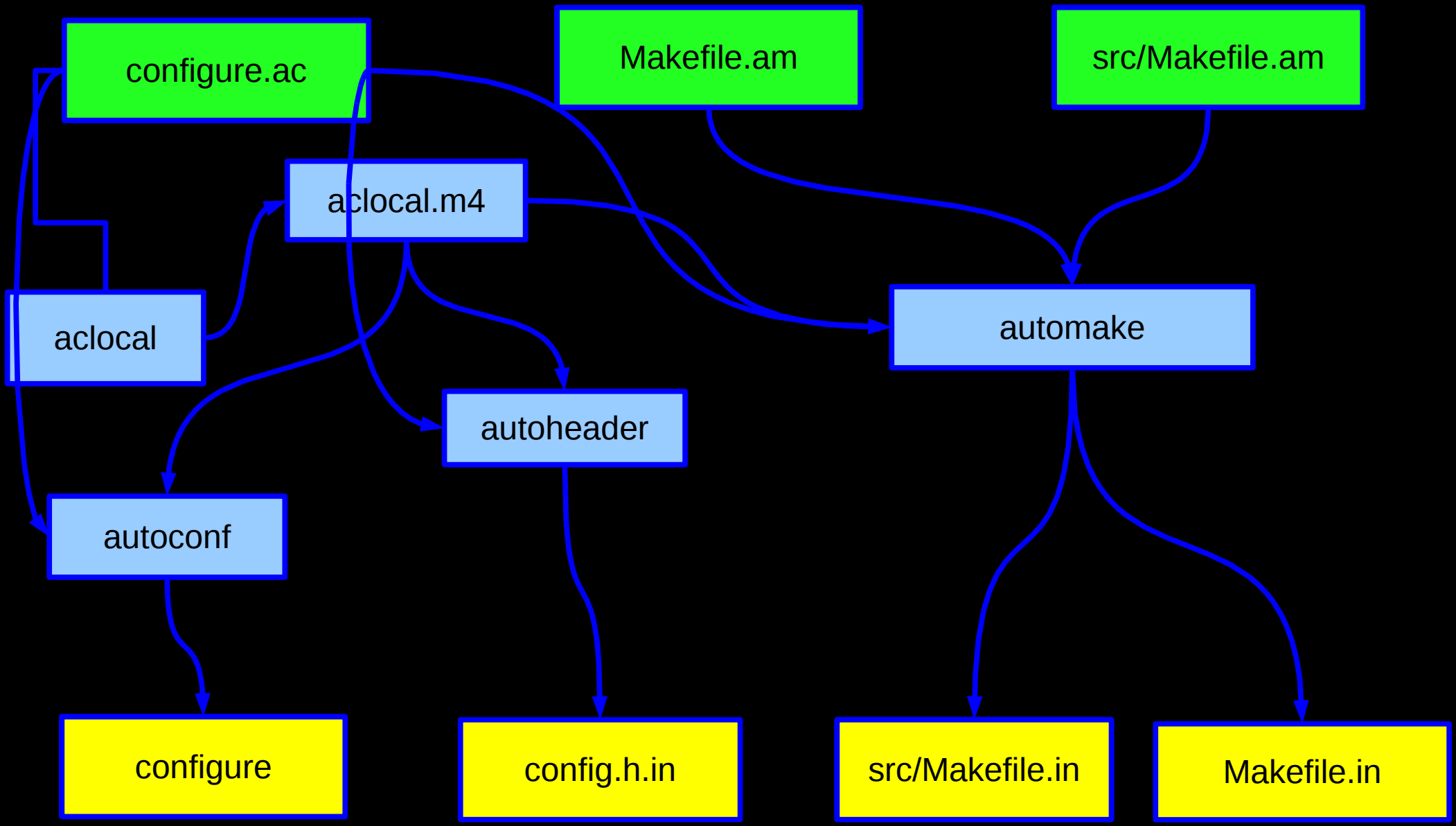
GNU Automake

‘automake’	Create Makefile.ins from Makefile.ams and configure.ac.
‘aclocal’	Scan configure.ac for uses of third-party macros, and gather definitions in aclocal.m4 .


configure.ac
Makefile.am
src/Makefile.am
aclocal.m4
aclocal
configure
Makefile.in
src/Makefile.in
config.h.in


configure.ac
Makefile.am
src/Makefile.am
aclocal.m4
aclocal
autoconf
configure
Makefile.in
src/Makefile.in
config.h.in


configure.ac
Makefile.am
src/Makefile.am
aclocal.m4
aclocal
autoheader
autoconf
configure
Makefile.in
src/Makefile.in
config.h.in


configure.ac
Makefile.am
src/Makefile.am
aclocal.m4
aclocal
automake
autoheader
autoconf
configure
config.h.in
src/Makefile.in
Makefile.in

Time for some explanation about abcdef project files and first session hands on!

Make

var1=arg1
Var2=arg2

.....

rule1
    cmd1
    cmd2

.....

rule2
    cmd3
     cmd4
.....

VARIABLES
RULES
COMMANDS

RULE:

targets: dependencies

$(var1) gives value of var1

For environment variables $${ENV_VAR}

$@ = targets

$+ = dependencies

$(MAKE) = make command

.PHONY rule or dot-rule are special rules

.PHONY: all clean dist

Thanks!