从零开始为 RISC-V 构建一个 Linux 系统

第 4 章 制作公共的本地构建工具

汪辰

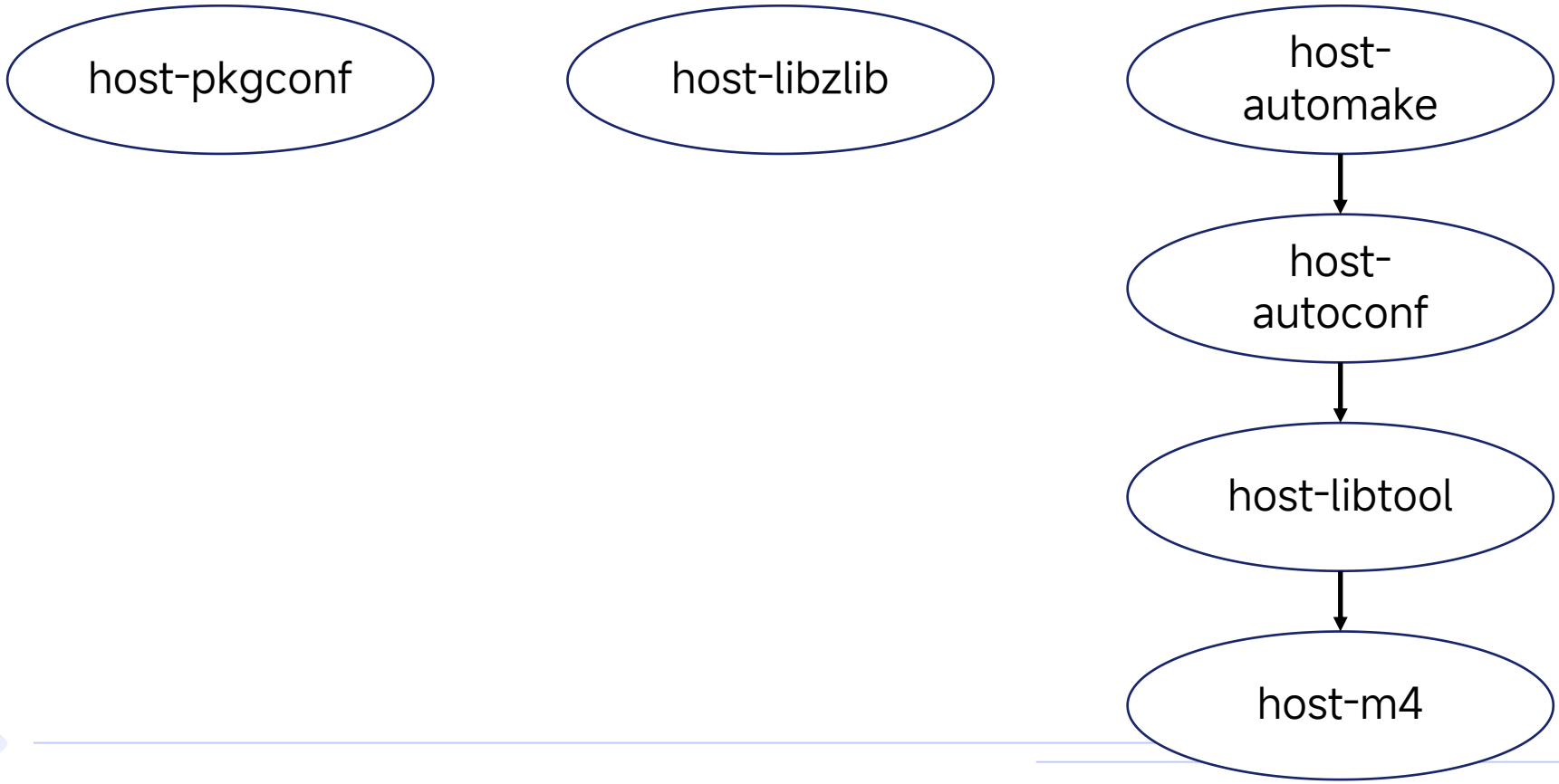
目录

01 pkgconf

02 libzlib

03 autotools

制作公共的本地构建工具



01

Pkgconf

Pkgconf - 简介

pkgconf 是经典工具 pkg-config 的一个现代化实现和兼容替代品。其作用是在我们编译软件时帮助我们查找和正确使用已安装的软件库的相关信息。可以把它想象成一个“**库信息查询中心**”。

Pkgconf - 简介

pkgconf 是经典工具 pkg-config 的一个现代化实现和兼容替代品。其作用是在我们编译软件时帮助我们查找和正确使用已安装的软件库的相关信息。可以把它想象成一个“**库信息查询中心**”。

```
$ cd ${HOST_DIR}
$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --modversion zlib
1.3.1

$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --cflags zlib
-I/home/wangchen/ws/test-buildroot/build-linux-system-from-scratch/output/host/include

$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --libs zlib
-L/home/wangchen/ws/test-buildroot/build-linux-system-from-scratch/output/host/lib -lz
```

Pkgconf - 简介

pkgconf 是经典工具 pkg-config 的一个现代化实现和兼容替代品。其作用是在我们编译软件时帮助我们查找和正确使用已安装的软件库的相关信息。可以把它想象成一个“**库信息查询中心**”。

```
$ cd ${HOST_DIR}
$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --modversion zlib
1.3.1

$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --cflags zlib
-I/home/wangchen/ws/test-buildroot/build-linux-system-from-scratch/output/host/include

$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --libs zlib
-L/home/wangchen/ws/test-buildroot/build-linux-system-from-scratch/output/host/lib -lz
```

```
${HOST_DIR/lib/pkgconfig/
├──── blkid.pc
├──── com_err.pc
├──── e2p.pc
├──── expat.pc
├──── ......
├──── uuid.pc
└──── zlib.pc
```

Pkgconf - 简介

pkgconf 是经典工具 pkg-config 的一个现代化实现和兼容替代品。其作用是在我们编译软件时帮助我们查找和正确使用已安装的软件库的相关信息。可以把它想象成一个“**库信息查询中心**”。

```
$ cd ${HOST_DIR}
$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --modversion zlib
1.3.1

$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --cflags zlib
-I/home/wangchen/ws/test-buildroot/build-linux-system-from-scratch/output/host/include

$ PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_PATH=./lib/pkgconfig/ ./bin/pkg-config --libs zlib
-L/home/wangchen/ws/test-buildroot/build-linux-system-from-scratch/output/host/lib -lz
```

```
prefix=/home/wangchen/ws/build-linux-system-from-scratch/output/host
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
sharedlibdir=${libdir}
includedir=${prefix}/include

Name: zlib
Description: zlib compression library
Version: 1.3.1

Requires:
Libs: -L${libdir} -L${sharedlibdir} -lz
Cflags: -I${includedir}
```

```
${HOST_DIR/lib/pkgconfig/
├──── blkid.pc
├──── com_err.pc
├──── e2p.pc
├──── expat.pc
├──── ......
├──── uuid.pc
└──── zlib.pc
```

Pkgconf - 简介

pkg-config/pkgconf 命令常用环境变量

环境变量名	用途	例子
PKG_CONFIG	指定 pkg-config 程序的路径	PKG_CONFIG="${HOST_DIR}/bin/pkg-config"
PKG_CONFIG_LIBDIR	指定在使用 pkg-config 程序时搜索 pc 文件的路径	"${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig"
PKG_CONFIG_SYSROOT_DIR	设置 pkg-config 工具查找库和头文件的根目录。所有从 .pc 文件里得到的绝对路径都自动在前面加该环境变量指定的路径前缀。	PKG_CONFIG_SYSROOT_DIR="/"
PKG_CONFIG_ALLOW_SYSTEM_CFLAGS	控制 pkg-config 在根据 “--cflags” 输出编译选项时是否包含指向系统标准头文件目录的路径（如 `-I/usr/include`）	PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1
PKG_CONFIG_ALLOW_SYSTEM_LIBS	控制 pkg-config 在根据 “--libs” 输出链接选项时是否包含指向标准系统库路径（如 `-L/usr/lib`）	PKG_CONFIG_ALLOW_SYSTEM_LIBS=1

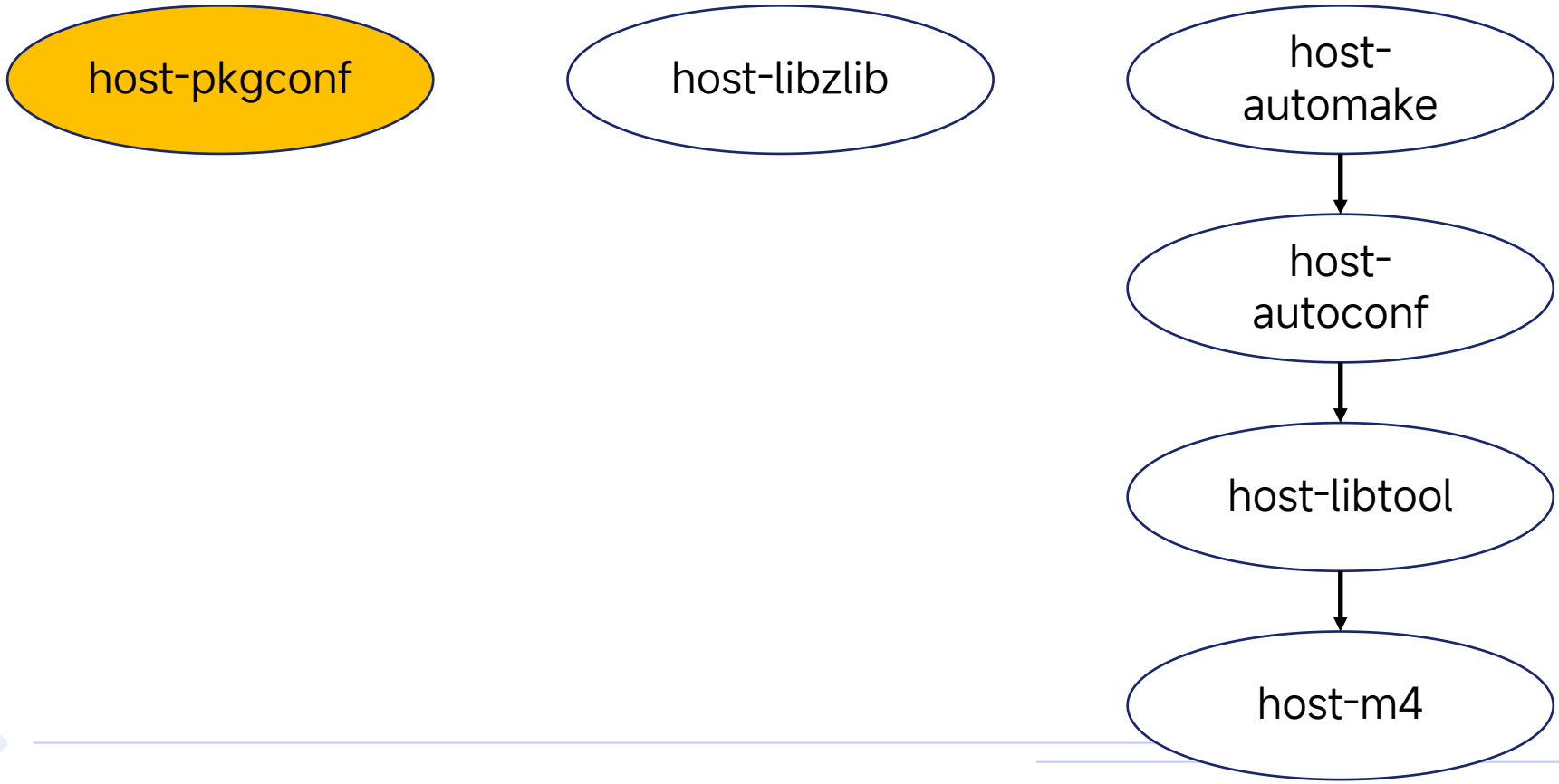
Pkgconf - 简介

pkgconf 使用举例

- PKG_CONFIG=“${HOST_DIR}/bin/pkg-config” PKG_CONFIG_SYSROOT_DIR=“/” PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1 PKG_CONFIG_LIBDIR=“${HOST_DIR}/host/lib/pkgconfig:${HOST_DIR}/host/share/pkgconfig” ./configure ……

- PKG_CONFIG="${HOST_DIR}/bin/pkg-config" PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1 PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig " /usr/bin/make ……

Pkgconf - 构建


host-pkgconf
host-libzlib
host-automake
host-autoconf
host-libtool
host-m4

Pkgconf – 下载

```
mkdir -p ${DL_DIR}/pkgconf
(cd ${DL_DIR}/pkgconf; https://distfiles.ariadne.space/pkgconf/pkgconf-2.3.0.tar.xz)
```

Pkgconf - 下载

```
mkdir -p ${DL_DIR}/pkgconf
(cd ${DL_DIR}/pkgconf; https://distfiles.ariadne.space/pkgconf/pkgconf-2.3.0.tar.xz)
```

对所有的软件组件，下载操作都是类似的，往后将不再赘述。

Pkgconf - 解压缩

```
mkdir -p ${PKGBUILD_DIR}
xzcat ${DL_DIR}/pkgconf/pkgconf-2.3.0.tar.xz | tar --strip-components=1 -C ${PKGBUILD_DIR} -xf -
chmod -R +rw ${PKGBUILD_DIR}
```

Pkgconf - 解压缩

```
mkdir -p ${PKGBUILD_DIR}
xzcat ${DL_DIR}/pkgconf-2.3.0.tar.xz | tar --strip-components=1 -C ${PKGBUILD_DIR} -xf -
chmod -R +rw ${PKGBUILD_DIR}
```

```
PKGNAME=pkgconf
PKGVERSION=2.3.0

PKGBUILDNAME=host-${PKGNAME}
PKGBUILD_DIR=${BUILD_DIR}/${PKGBUILDNAME}-${PKGVERSION}
```

package/pkgconf/make-host.sh

Pkgconf - 解压缩

```
mkdir -p ${PKGBUILD_DIR}
xzcat ${DL_DIR}/pkgconf/pkgconf-2.3.0.tar.xz | tar --strip-components=1 -C ${PKGBUILD_DIR} -xf -
chmod -R +rw ${PKGBUILD_DIR}
```

${BUILD_DIR}/host-pkgconf-2.3.0

```
PKGNAME=pkgconf
PKGVERSION=2.3.0

PKGBUILDNAME=host-${PKGNAME}
PKGBUILD_DIR=${BUILD_DIR}/${PKGBUILDNAME}-${PKGVERSION}
```

package/pkgconf/make-host.sh

Pkgconf – 解压缩

```
mkdir -p ${PKGBUILD_DIR}
xzcat ${DL_DIR}/pkgconf/pkgconf-2.3.0.tar.xz | tar --strip-components=1 -C ${PKGBUILD_DIR} -xf -
chmod -R +rw ${PKGBUILD_DIR}
```

- 管道左边 xzcat 将压缩的归档文件解压后输出到标准输出。
- 管道将左边 xzcat 的输出作为右边 tar 的输入（标准输入）
- 管道右边的 tar 命令中：
  - ✓ -x 表示解开归档（extract）
  - ✓ -f – 表示从标准输入读取归档数据
  - ✓ --strip-components=1 作用是：在解开归档（extract）时移除归档文件中的顶层目录结构，和 –C 配合使用。

Pkgconf - 解压缩

```
mkdir -p ${PKGBUILD_DIR}
xzcat ${DL_DIR}/pkgconf/pkgconf-2.3.0.tar.xz | tar --strip-components=1 -C ${PKGBUILD_DIR} -xf -
chmod -R +rw ${PKGBUILD_DIR}
```

对所有的软件组件，解压缩操作都是类似的，区别仅在于对于不同的压缩格式会采用不同的解压缩工具，譬如对于 *.tar.gz，对应会使用 gzip。
往后将不再赘述。

Pkgconf - 打补丁

```
TAR="tar" PATH=${HOST_DIR}/bin:$PATH ${PROJECT_DIR}/support/scripts/apply-patches.sh ${PKGBUILD_DIR} ${PROJECT_DIR}/package/pkgconf \*.patch
```

Pkgconf - 打补丁

```
TAR="tar" PATH=${HOST_DIR}/bin:$PATH ${PROJECT_DIR}/support/scripts/apply-patches.sh ${PKGBUILD_DIR} ${PROJECT_DIR}/package/pkgconf \*.patch
```

```
${PROJECT_DIR}/package/pkgconf/
├──── 0001-Only-prefix-with-the-sysroot-a-subset-of-variables.patch
├──── ......
└──── make-host.sh
```

Pkgconf - 打补丁

```
TAR="tar" PATH=${HOST_DIR}/bin:$PATH ${PROJECT_DIR}/support/scripts/apply-patches.sh ${PKGBUILD_DIR} ${PROJECT_DIR}/package/pkgconf \*.patch
```

```
${PROJECT_DIR}/package/pkgconf/
├── 0001-Only-prefix-with-th
├── .....
└── make-host.sh
```

对所有的软件组件，打补丁操作都是类似的，往后将不再赘述。

Pkgconf - 配置

```
GIT_DIR=. PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}"
PKG_CONFIG="${HOST_DIR}/bin/pkg-config" PKG_CONFIG_SYSROOT_DIR="/"
PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1
PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig"
AR="/usr/bin/ar" AS="/usr/bin/as" LD="/usr/bin/ld" NM="/usr/bin/nm" CC="/usr/bin/gcc"
GCC="/usr/bin/gcc" CXX="/usr/bin/g++" CPP="/usr/bin/cpp" OBJCOPY="/usr/bin/objcopy"
RANLIB="/usr/bin/ranlib" CPPFLAGS="-I${HOST_DIR}/include" CFLAGS="-O2 -
I${HOST_DIR}/include" CXXFLAGS="-O2 -I${HOST_DIR}/include" LDFLAGS="-L${HOST_DIR}/lib
-Wl,-rpath,${HOST_DIR}/lib" INTLTOOL_PERL=/usr/bin/perl CC="/usr/bin/gcc"
CXX="/usr/bin/g++" CONFIG_SITE=/dev/null ./configure --prefix="${HOST_DIR}" --
sysconfdir="${HOST_DIR}/etc" --localstatedir="${HOST_DIR}/var" --enable-shared --disable-
static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-
documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-
dependency-tracking
```

Pkgconf - 配置

```
GIT_DIR=. PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}"
PKG_CONFIG="${HOST_DIR}/bin/pkg-config" PKG_CONFIG_SYSROOT_DIR="/"
PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1
PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig"
AR="/usr/bin/ar" AS="/usr/bin/as" LD="/usr/bin/ld" NM="/usr/bin/nm" CC="/usr/bin/gcc"
GCC="/usr/bin/gcc" CXX="/usr/bin/g++" CPP="/usr/bin/cpp" OBJCOPY="/usr/bin/objcopy"
RANLIB="/usr/bin/ranlib" CPPFLAGS="-I${HOST_DIR}/include" CFLAGS="-O2 -
I${HOST_DIR}/include" CXXFLAGS="-O2 -I${HOST_DIR}/include" LDFLAGS="-L${HOST_DIR}/lib
-Wl,-rpath,${HOST_DIR}/lib" INTLTOOL_PERL=/usr/bin/perl CC="/usr/bin/gcc"
CXX="/usr/bin/g++" CONFIG_SITE=/dev/null ./configure --prefix="${HOST_DIR}" --
sysconfdir="${HOST_DIR}/etc" --localstatedir="${HOST_DIR}/var" --enable-shared --disable-
static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-
documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-
dependency-tracking
```

Pkgconf - 配置

```
GIT_DIR=. PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}"
PKG_CONFIG="${HOST_DIR}/bin/pkg-config" PKG_CONFIG_SYSROOT_DIR="/"
PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1
PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig"
AR="/usr/bin/ar" AS="/usr/bin/as" LD="/usr/bin/ld" NM="/usr/bin/nm" CC="/usr/bin/gcc"
GCC="/usr/bin/gcc" CXX="/usr/bin/g++" CPP="/usr/bin/cpp" OBJCOPY="/usr/bin/objcopy"
RANLIB="/usr/bin/ranlib" CPPFLAGS="-I${HOST_DIR}/include" CFLAGS="-O2 -
I${HOST_DIR}/include" CXXFLAGS="-O2 -I${HOST_DIR}/include" LDFLAGS="-L${HOST_DIR}/lib
-Wl,-rpath,${HOST_DIR}/lib" INTLTOOL_PERL=/usr/bin/perl CC="/usr/bin/gcc"
CXX="/usr/bin/g++" CONFIG_SITE=/dev/null /configure --prefix="${HOST_DIR}" --
sysconfdir="${HOST_DIR}/etc" --localstatedir="${HOST_DIR}/var" --enable-shared --disable-
static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-
documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-
dependency-tracking
```

构建 Host 上运行的软件。

Pkgconf - 配置

```
GIT_DIR=. PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}"
PKG_CONFIG="${HOST_DIR}/bin/pkg-config" PKG_CONFIG_SYSROOT_DIR="/"
PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1
PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig"
AR="/usr/bin/ar" AS="/usr/bin/as" LD="/usr/bin/ld" NM="/usr/bin/nm" CC="/usr/bin/gcc"
GCC="/usr/bin/gcc" CXX="/usr/bin/g++" CPP="/usr/bin/cpp" OBJCOPY="/usr/bin/objcopy"
RANLIB="/usr/bin/ranlib" CPPFLAGS="-I${HOST_DIR}/include" CFLAGS="-O2 -
I${HOST_DIR}/include" CXXFLAGS="-O2 -I${HOST_DIR}/include" LDFLAGS="-L${HOST_DIR}/lib
-Wl,-rpath,${HOST_DIR}/lib" INTLTOOL_PERL=/usr/bin/perl CC="/usr/bin/gcc"
CXX="/usr/bin/g++" CONFIG_SITE=/dev/null ./configure --prefix="${HOST_DIR}" --
sysconfdir="${HOST_DIR}/etc" --localstatedir="${HOST_DIR}/var" --enable-shared --disable-
static --disable-gtk-doc --disable-gtk-doc-html --disable-
documentation --disable-debug --with-xml-catalog=/etc/xml/catalog --with-
dependency-tracking
```

对于需要安装到 ${HOST_DIR} 的软件组件，这部分环境变量的设置操作都是类似的，往后将统一定义为 HOST_CONFIGURE_OPTS 。

Pkgconf - 配置

```
eval "${HOST_CONFIGURE_OPTS} CC=\"/usr/bin/gcc\" CXX=\"/usr/bin/g++\"
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\"
--localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --
with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

Pkgconf - 配置

```
eval "${HOST_CONFIGURE_OPTS} CC=\"/usr/bin/gcc\" CXX=\"/usr/bin/g++\" CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

Pkgconf - 配置

```
eval "${HOST_CONFIGURE_OPTS} CC=\"/usr/bin/gcc\" CXX=\"/usr/bin/g++\"
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\"
--localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --
with-xmlto=no --with-fop=no --disable-dependency-tracking"
```

选项	含义/用途	例子
--prefix	指定软件安装目录的根路径	--prefix="${HOST_DIR}"
--sysconfdir	指定配置文件的安装位置	--sysconfdir="${HOST_DIR}/etc"
--localstatedir	指定运行时可变数据的安装位置	--localstatedir="${HOST_DIR}/var"

Pkgconf - 配置

```
eval "${HOST_CONFIGURE_OPTS} CC=\"/usr/bin/gcc\" CXX=\"/usr/bin/g++\"
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\"
--localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-doc disable-documentation --disable-debug --
with-xmlto=no --with-fop=no --disable-nls -- dependency-tracking"
```

选项	含义/用途
--enable-shared	如果软件项目会生成库则构建为共享库 (.so) 形式，与之相对的是 --disable-shared
--disable-static	如果软件项目会生成库则禁止构建静态库 (.a) 形式, 与之相对的是 --enable-static

Pkgconf - 配置

```
eval "${HOST_CONFIGURE_OPTS} CC=\"/usr/bin/gcc\" CXX=\"/usr/bin/g++\"
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\"
--localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --
with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

选项	含义/用途
--disable-gtk-doc	禁止在构建过程中采用 GTK-Doc 方式生成 API 参考文档。--disable-gtk-doc-html 只禁用 HTML，其他格式仍可生成
--disable-doc	通用的禁止生成文档参数，--disable-docs 和 --disable-documentation 是 --disable-doc 的别名
--with-xmlto=no	明确指定不使用 xmlto 工具生成文档（即使检测出系统安装了 xmlto）
--with-fop=no	明确指定不使用 FOP 工具生成文档（即使检测出系统安装了 FOP）

Pkgconf - 配置

```
eval "${HOST_CONFIGURE_OPTS} CC=\"/usr/bin/gcc\" CXX=\"/usr/bin/g++\"
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\"
--localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --
with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

选项	含义/用途
--disable-debug	禁止在构建过程中生成调试信息
--disable-nls	禁用本地化语言支持（Native Language Support）
--disable-dependency-tracking	关闭支持自动依赖跟踪（Dependency Tracking）

Pkgconf – 构建

```
GIT_DIR=. PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}"
PKG_CONFIG="${HOST_DIR}/bin/pkg-config" PKG_CONFIG_SYSROOT_DIR="/"
PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1
PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig
"  /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}
```

Pkgconf - 构建

```
GIT_DIR=. PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}"
PKG_CONFIG="${HOST_DIR}/bin/pkg-config" PKG_CONFIG_SYSROOT_DIR="/"
PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1
PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig
"  /usr/bin/make -j${MAXNUM_CPUS}  -C ${H          D_DIR}
```

对于需要安装到 ${HOST_DIR} 的软件组件，环境变量的设置操作都是类似的，往后将统一定义为 HOST_MAKE_ENV 。

Pkgconf - 构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

Pkgconf - 构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

-j：支持多线程并发构建

Pkgconf – 安装（install-host）

```
GIT_DIR=. PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}" PKG_CONFIG="${HOST_DIR}/bin/pkg-config" PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1 PKG_CONFIG_ALLOW_SYSTEM_LIBS=1 PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pkgconfig:${HOST_DIR}/share/pkgconfig"  /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}

/usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/pkgconf/pkg-config.in ${HOST_DIR}/bin/pkg-config

/usr/bin/sed -i -e "s,@STAGING_SUBDIR@,${STAGING_SUBDIR},g" ${HOST_DIR}/bin/pkg-config

/usr/bin/sed -i -e 's,@STATIC@,,' ${HOST_DIR}/bin/pkg-config
```

Pkgconf - 安装（install-host）

```
GIT_DIR=. PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}" PKG_CONFIG="${HOST_DIR}/bin/pkg-
config" PKG_CONFIG_SYSROOT_DIR="/" PKG_CONFIG_ALLOW_SYSTEM_CFLAGS=1
PKG_CONFIG_ALLOW_SYSTEM_LIBS=1
PKG_CONFIG_LIBDIR="${HOST_DIR}/lib/pk      nfig:${HOST_DIR}/share/pkgconfig"  /usr/bin/make -
j${MAXNUM_CPUS} install -C ${PKGBUILD_D

/usr/bin/install -m 0755 -D ${PROJECT_DIR}/pac                  f/pkg-config.in ${HOST_DIR}/bin/pkg-
config

/usr/bin/sed -i -e "s,@STAGING_SUBDI

/usr/bin/sed -i -e 's,@STATIC@,,' ${HOS
```

对于需要安装到 ${HOST_DIR} 的软件组件，环境变量的设置操作都是类似的，往后将统一定义为 HOST_MAKE_ENV 。

Pkgconf – 安装（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
/usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/pkgconf/pkg-config.in ${HOST_DIR}/bin/pkg-config
/usr/bin/sed -i -e "s,@STAGING_SUBDIR@,${STAGING_SUBDIR},g" ${HOST_DIR}/bin/pkg-config
/usr/bin/sed -i -e 's,@STATIC@,,' ${HOST_DIR}/bin/pkg-config
```

Pkgconf - 安装（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
/usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/pkgconf/pkg-config.in ${HOST_DIR}/bin/pkg-config
/usr/bin/sed -i -e "s,@STAGING_SUBDIR@,${STAGING_SUB                ${HOST_DIR}/bin/pkg-config
/usr/bin/sed -i -e 's,@STATIC@,,' ${HOST_DIR}/bin/p
```

pkg-config.in 是一个脚本，内部封装调用 ${HOST_DIR}/bin/pkgconf ，实现了 pkgconf 对 pkg-config 的兼容性。

Pkgconf – 安装（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
/usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/pkgconf/pkg-config.in ${HOST_DIR}/bin/pkg-config
/usr/bin/sed -i -e "s,@STAGING_SUBDIR@,${STAGING_SUBDIR},g" ${HOST_DIR}/bin/pkg-config
/usr/bin/sed -i -e 's,@STATIC@,,' ${HOST_DI      /pkg-config
```

利用 sed 对 ${HOST_DIR}/bin/pkg-config 中的一些字符串进行替换，因为这些字符串涉及的路径信息只有在完成构建后才能确定。

Pkgconf - 安装（install-host）

```
$ ls ${PKGBUILD_DIR}/.files-list* -l
-rw-rw-r-- 1 wangchen wangchen 820 Dec 15 09:58 .files-list-host.txt
-rw-rw-r-- 1 wangchen wangchen   0 Dec 15 09:58 .files-list-images.txt
-rw-rw-r-- 1 wangchen wangchen   0 Dec 15 09:58 .files-list-staging.txt
-rw-rw-r-- 1 wangchen wangchen   0 Dec 15 09:58 .files-list.txt
```

Pkgconf - 安装（install-host）

```
$ ls ${PKGBUILD_DIR}/.files-list* -l
-rw-rw-r-- 1 wangchen wangchen 820 Dec 15 09:58 .files-list-host.txt
-rw-rw-r-- 1 wangchen wangchen   0 Dec 15 09:58 .files-list-images.txt
-rw-rw-r-- 1 wangchen wangchen   0 Dec 15 09:58 .files-list-staging.txt
-rw-rw-r-- 1 wangchen wangchen   0 Dec 15 09:58 .files-list.txt
```

.files-list-host.txt: install-host
.files-list-images.txt: install-image
.files-list-staging.txt: install-staging
.files-list.txt: install-target

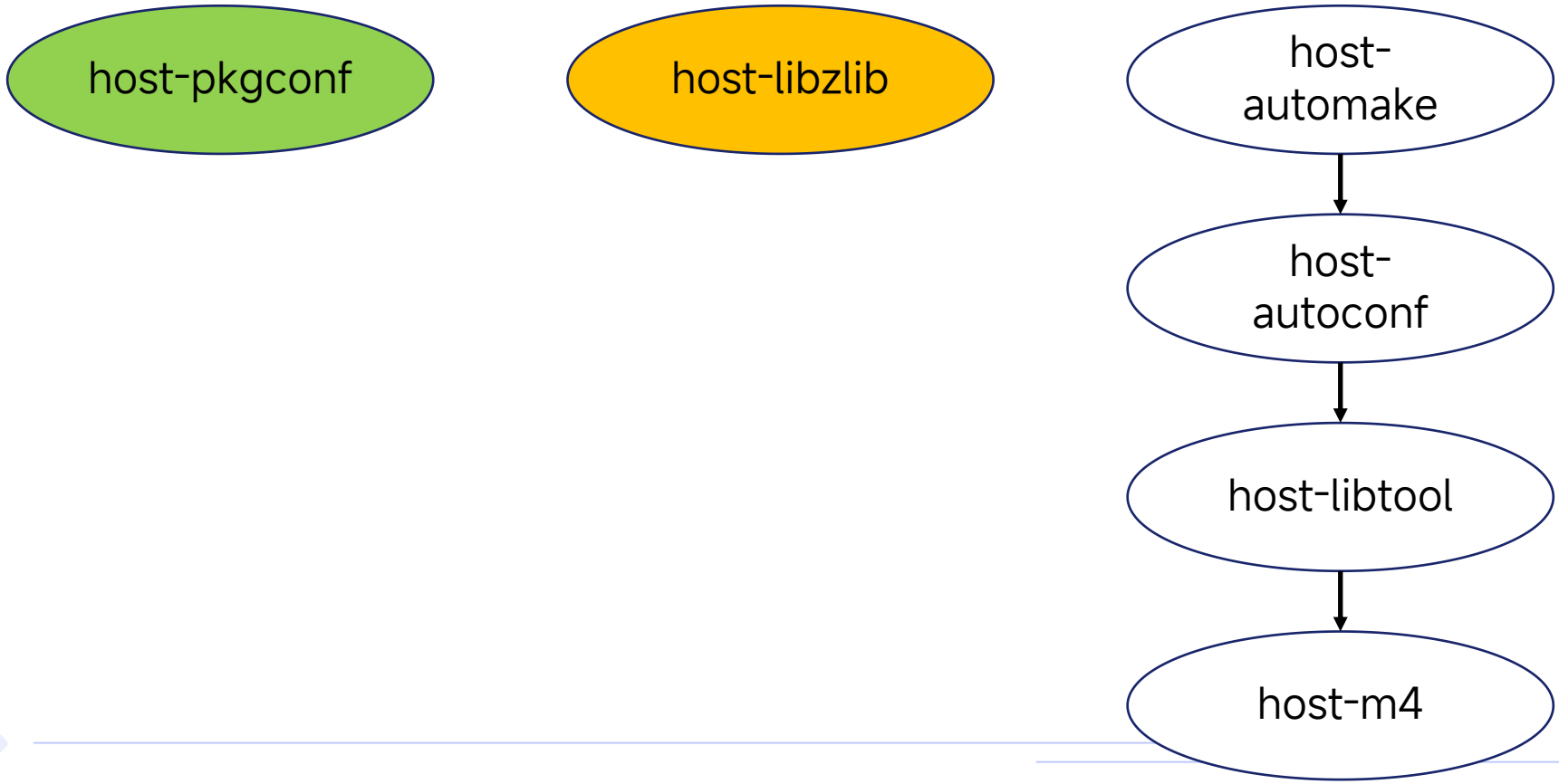
Pkgconf – 安装（install-host）

host-pkgconf,./bin/bomtool
host-pkgconf,./bin/pkg-config
host-pkgconf,./bin/pkgconf
host-pkgconf,./include/pkgconf/libpkgconf/bsdstubs.h
host-pkgconf,./include/pkgconf/libpkgconf/iter.h
host-pkgconf,./include/pkgconf/libpkgconf/libpkgconf-api.h
host-pkgconf,./include/pkgconf/libpkgconf/libpkgconf.h
host-pkgconf,./include/pkgconf/libpkgconf/stdinc.h
host-pkgconf,./lib/libpkgconf.la
host-pkgconf,./lib/libpkgconf.so
host-pkgconf,./lib/libpkgconf.so.5
host-pkgconf,./lib/libpkgconf.so.5.0.0
host-pkgconf,./lib/pkgconfig/libpkgconf.pc
host-pkgconf,./share/aclocal/pkg.m4
host-pkgconf,./share/doc/pkgconf/AUTHORS
host-pkgconf,./share/doc/pkgconf/README.md
host-pkgconf,./share/man/man1/pkgconf.1
host-pkgconf,./share/man/man5/pc.5
host-pkgconf,./share/man/man5/pkgconf-personality.5
host-pkgconf,./share/man/man7/pkg.m4.7

02

Libzlib

Libzlib


host-pkgconf
host-libzlib
host-automake
host-autoconf
host-libtool
host-m4

Libzlib – 简介

libzlib（通常简称为 zlib）是一个广泛使用的、开源的数据压缩库。https://www.zlib.net/

- 使用纯 C 语言编写，可在几乎所有操作系统（Linux、Windows、macOS 等）和硬件平台上运行。
- 支持无损压缩，采用 DEFLATE 压缩算法（结合了 LZ77 和霍夫曼编码），压缩后的数据可以完全还原。
- 本身对其他软件包没有依赖。
- 被其他软件和工具广泛使用，譬如 util-linux， python3

Libzlib – 制作步驟

配置

```
eval "${HOST_CONFIGURE_OPTS} ./configure --prefix=\"${HOST_DIR}\" --
sysconfdir=\"${HOST_DIR}/etc\""
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j1  -C ${PKGBUILD_DIR}"
```

安装 （install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j1  -C ${PKGBUILD_DIR} LDCONFIG=true
install"
```

Libzlib – 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} ./configure --prefix=\"${HOST_DIR}\" --
sysconfdir=\"${HOST_DIR}/etc\""
```

构建

```
eval "${HOST_MAKE_ENV} /u
```

覆盖 Makefile 中原 LDCONFIG 的定义（ LDCONFIG=ldconfig ），从而跳过 ldconfig 的执行，因为这里只是安装到 ${HOST_DIR} 而不是本地构建主机的系统。

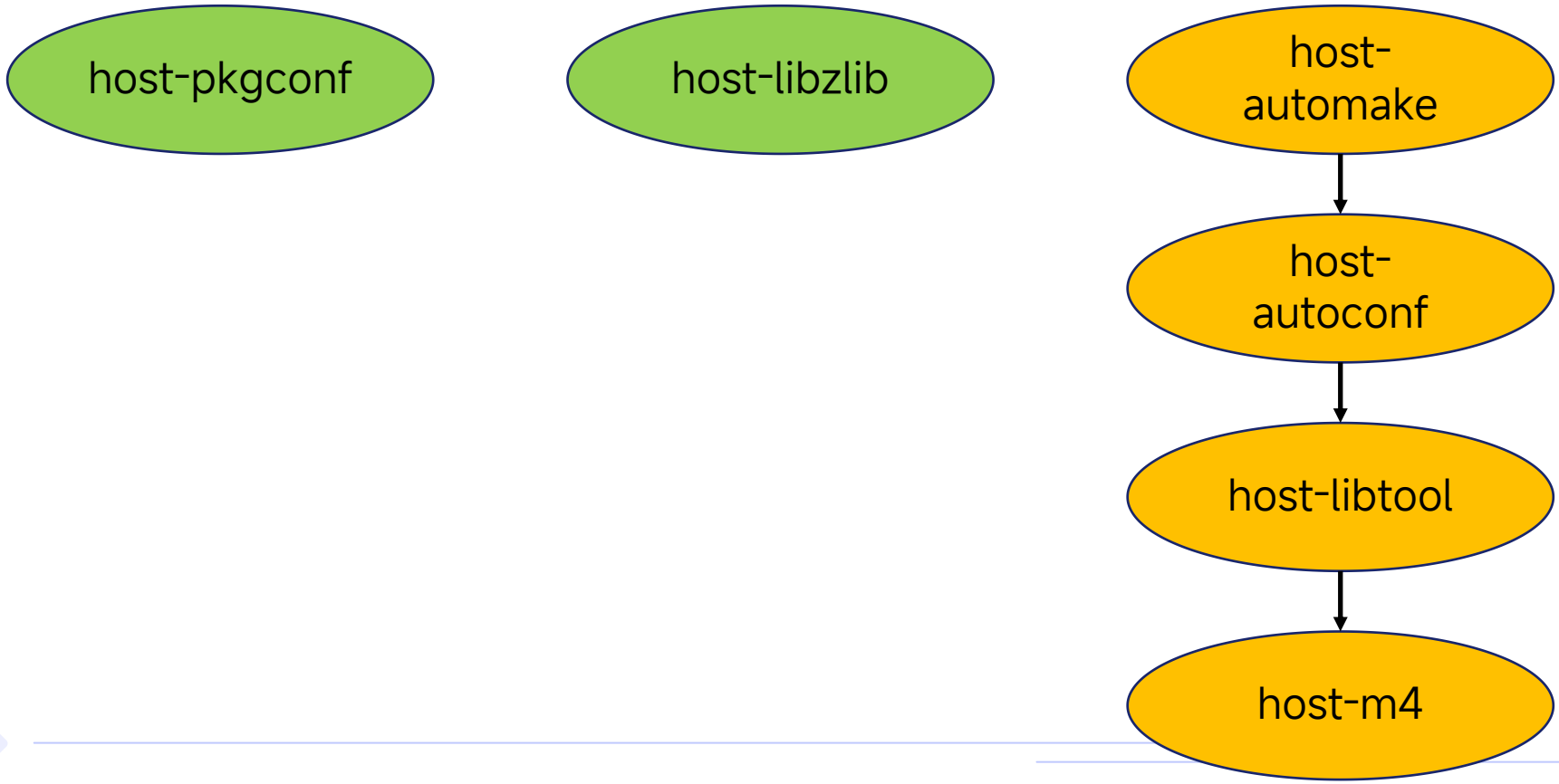
安装 （install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j1  -C ${PKGBUILD_DIR} LDCONFIG=true
install"
```

03

Autotools

Autotools


host-pkgconf
host-libzlib
host-automake
host-autoconf
host-libtool
host-m4

M4 - 简介

- M4 是一个“宏处理器”（Macro Processor）专门用于处理和扩展（展开）“宏文本”（Macros）。这个过程是递归的，即展开后的文本如果包含其他宏，会继续被处理。
- M4 是 Autotools 的“引擎”，用于生成最终由开发者和用户运行的配置脚本。LibTool、Autoconf 和 Automake 则是在此引擎基础上构建的、服务于不同构建阶段的具体工具。譬如 configure.ac 以及 libtool 中的宏文件都是使用 M4 宏编写的。
- M4 宏处理器诞生于 1977 年，已有近 50 年历史。尽管现在看来有些“古老”，但它凭借独特的早期设计定位和在关键软件（如 Autoconf）中的核心地位，至今仍在特定的开发和构建领域中活跃使用。

M4 – 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS}  CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
```

Libtool - 简介

Libtool 的目标是封装不同系统（Linux、MacOS 、Windows ...）上生成和使用库文件的差异。开发者只需要写一份通用的“库描述”，Libtool 就会在构建时，自动调用适合当前平台的正确命令和参数来生成静态库或共享库。它解决了如下问题：

- 不同系统平台上构建库的命令差异。
- 不同系统平台上库文件命名差异。
- 链接与运行时路径问题：Libtool 能自动处理编译时的库路径和安装后的库路径，减少“库未找到”的错误。
- 统一静态库与共享库处理，让开发者可以用相同的方式处理静态库和共享库。

作为 Autotools 的一部分，Libtool 通常与 Autoconf 和 Automake 协同工作，实现对库操作的封装。

Libtool - 简介

Libtool 的目标是封装不同系统（Linux、MacOS 、Windows ...）上生成和使用库文件的差异。开发者只需要写一份通用的“库描述”，Libtool 就会在构建时，自动调用适合当前平台的正确命令和参数来生成静态库或共享库。它解决了如下问题：

- 不同系统平台上构建库的命令差异。
- 不同系统平台上库文件命名差异。
- 链接与运行时路径问题：Libtool 能自动
  减少“库未找到”的错误。
- 统一静态库与共享库处理，让开发者可以用相同的方式处理静态库和共享库。

  - Linux：gcc -shared -fPIC 生成共享库。
  - MacOS：clang -dynamiclib 生成动态库。

作为 Autotools 的一部分，Libtool 通常与 Autoconf 和 Automake 协同工作，实现对库操作的封装。

Libtool - 简介

Libtool 的目标是封装不同系统（Linux、MacOS 、Windows ...）上生成和使用库文件的差异。开发者只需要写一份通用的“库描述”，Libtool 就会在构建时，自动调用适合当前平台的正确命令和参数来生成静态库或共享库。它解决了如下问题：

- 不同系统平台上构建库的命令差异。
- 不同系统平台上库文件命名差异。
  - Linux: libfoo.so.1.2.3
  - macOS: libfoo.1.2.3.dylib
  - Windows: foo.dll
- 链接与运行时路径问题：Libtool 能自动
  减少“库未找到”的错误。
- 统一静态库与共享库处理，让开发者可以用相同的方式处理静态库和共享库。

作为 Autotools 的一部分，Libtool 通常与 Autoconf 和 Automake 协同工作，实现对库操作的封装。

Libtool - 简介

Libtool 的目标是封装不同系统（Linux、MacOS 、Windows ...）上生成和使用库文件的差异。开发者只需要写一份通用的“库描述”，Libtool 就会在构建时，自动调用适合当前平台的正确命令和参数来生成静态库或共享库。它解决了如下问题：

- 不同系统平台上构建库的命令差异。
- 不同系统平台上库文件命名差异。
- 链接与运行时路径问题：Libtool 能自动处理编译时的库路径和安装后的库路径，减少“库未找到”的错误。
- 统一静态库与共享库处理，让开发者可以

作为 Autotools 的一部分，Libtool 通常与 Au
对库操作的封装。

Libtool 提供一系列的脚本（例如：ltmain.sh）M4 宏（例如 libtool.m4）来处理不同平台的可移植性问题

Libtool – 简介

Libtool 的目标是封装
文件的差异。开发者
用适合当前平台的正

- 不同系统平台上构
- 不同系统平台上库
- 链接与运行时路径
  减少“库未找到”的错误。
- 统一静态库与共享库处理，让开发者可以用相同的方式处理静态库和共享库。

作为 Autotools 的一部分，Libtool 通常与 Autoconf 和 Automake 协同工作，实现对库操作的封装。

通过“.la”文件（库元信息文件）封装动态库和静态库的不同处理。

生成 foo 库时： libtool --mode=link gcc -o libfoo.la ……

链接 foo 库时： libtool --mode=link gcc -o myapp main.o libfoo.la

具体生成什么形式的 foo 库，以及采用何种方式链接 foo 库，则由具体配置时的 (--enable/--disable)-(static/shared) 选项决定。

Libtool - 制作步驟

配置

```
eval "${HOST_CONFIGURE_OPTS}  CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

構建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安裝（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
```

autoconf - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} EMACS=\"no\" ac_cv_path_M4=${HOST_DIR}/bin/m4 ac_cv_prog_gnu_m4_gnu=no CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
```

autoconf - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} EMACS=\"no\" ac_cv_path_M4=${HOST_DIR}/bin/m4
ac_cv_prog_gnu_m4_gnu=no CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\"
--sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --
disable-static --disable-gtk-doc --disable-html --disable-doc --disable-docs --
disable-documentation --disable-debug --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} make -C ${PKGBUILD_DIR}"
```

安装（install-host）

```
eval "${HOST_MAKE_ENV} make -C ${PKGBUILD_DIR} DESTDIR=${HOST_DIR} install
${PKGBUILD_DIR}"
```

选项	含义/用途
EMACS=“no”	跳过对 emacs 的依赖检查，autoconf 本身不需要 emacs
ac_cv_path_M4=${HOST_DIR}/bin/m4	指定 autoconf 所使用的 m4 的位置路径
ac_cv_prog_gnu_m4_gnu=no	明确告知 configure 脚本不要把当前 m4 按照 GNU M4 方式处理，主动禁用 GNU 扩展，确保生成的 m4 宏使用兼容模式，确保构建过程的可重复性和跨平台兼容。

automake - 制作步驟

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --
prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --
localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-
debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  \
-C ${PKGBUILD_DIR}"
```

安装（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}“

/usr/bin/install -D -m 0644 ${PROJECT_DIR}/package/automake/gtk-doc.m4
${HOST_DIR}/share/aclocal/gtk-doc.m4

mkdir -p ${STAGING_DIR}/usr/share/aclocal
```

automake - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --
prefix=\"${HOST_DIR}\" --sysconfdir=\"${
localstatedir=\"${HOST_DIR}/var\" --ena
disable-gtk-doc-html --disable-doc --di
debug --with-xmlto=no --with-fop=no
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/mak
-C ${PKGBUILD_DIR}"
```

安装（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -i    AXNUM_CPUS} install -C ${PKGBUILD_DIR}“

/usr/bin/install -D -m 0644 ${PROJECT_DIR}/package/automake/gtk-doc.m4
${HOST_DIR}/share/aclocal/gtk-doc.m4

mkdir -p ${STAGING_DIR}/usr/share/aclocal
```

从 automake 1.16 版本开始，不再默认包含某些第三方宏文件（如 gtk-doc.m4）。

但有些版本的软件包在使用 autotools 进行构建时仍然依赖 gtk-doc.m4 中定义的宏，为了避免对每个有此依赖的软件包单独打补丁，统一提供一份备用。

automake - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --
prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --
localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-
debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  \
-C ${PKGBUILD_DIR}"
```

安装（install-host）

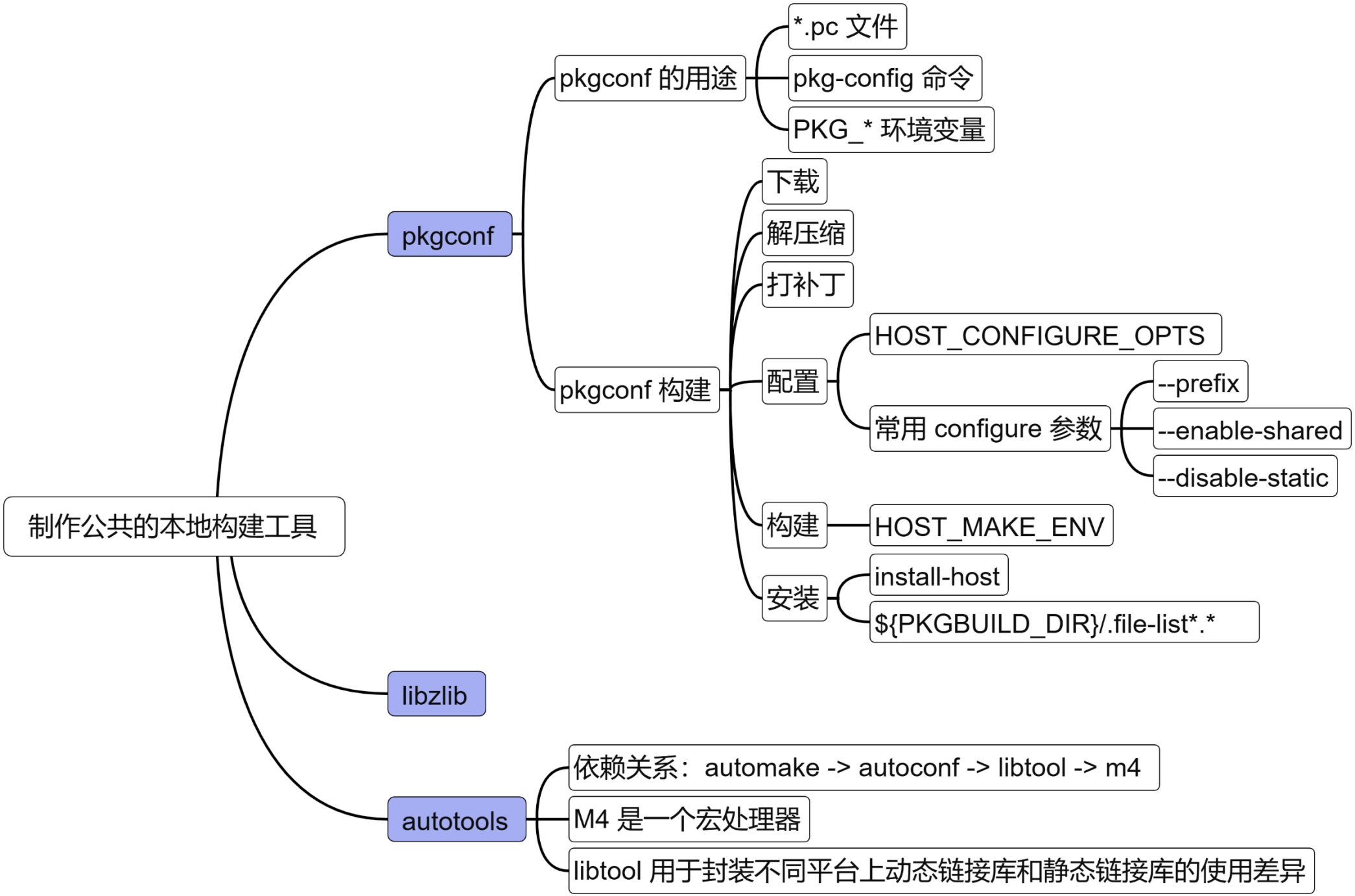
```
eval "${HOST_MAKE_ENV} /usr/bin/make                                    UILD_DIR}“

/usr/bin/install -D -m 0644 ${PROJECT_DIR}/                ge/automake/gtk-doc.m4
${HOST_DIR}/share/aclocal/gtk-doc.m4

mkdir -p ${STAGING_DIR}/usr/share/aclocal
```

为交叉编译 Target 软件组件时提供存放系统级 M4 宏文件的目录

本章总结


制作公共的本地构建工具
pkgconf
pkgconf 的用途
*.pc 文件
pkg-config 命令
PKG_* 环境变量
pkgconf 构建
下载
解压缩
打补丁
配置
HOST_CONFIGURE_OPTS
常用 configure 参数
--prefix
--enable-shared
--disable-static
构建
HOST_MAKE_ENV
安装
install-host
${PKGBUILD_DIR}/.file-list*.*
libzlib
autotools
依赖关系：automake -> autoconf -> libtool -> m4
M4 是一个宏处理器
libtool 用于封装不同平台上动态链接库和静态链接库的使用差异

谢谢