自动化编译

在前面的标准编译安装中，第一步是./configure[30]，它会根据Makefile.in生成Makefile文件，然后make根据Makefile自动编译软件

通常在一个源码包中，已经包含了configure脚本和Makefile文件，作为课外知识，我们大致了解一下怎么生成这两个文件

autoconf

autoconf用来生成configure脚本，它可以检查系统特性、编译环境、环境变量、软件参数、依赖关系等

autoconf需要用到 m4

1. 用autoscan描源代码目录生成configure.scan文件
2. 将configure.scan改名为configure.in
3. 用aclocal根据configure.in文件的内容，自动生成aclocal.m4文件
4. 使用autoconf，根据configure.in和aclocal.m4来产生configure文件

automake

automake可以从Makefile.am文件自动生成Makefile.in，它主要用来配置源代码

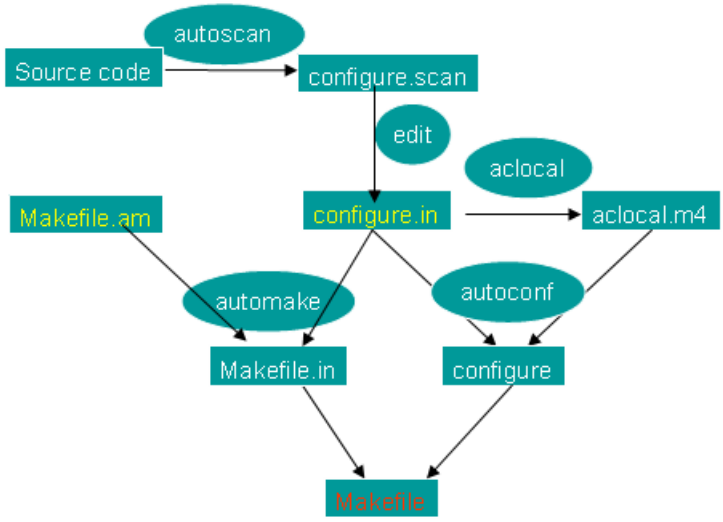
automake需用到perl

- 手工写Makefile.am
- 使用automake，根据configure.in和Makefile.am来产生Makefile.in

Makefile

使用configure脚本，配合Makefile.in可以生成Makefile文件，然后用make自动化的编译软件

这里有一张生成Makefile的流程图：



Makefile的用途不只是编译软件，还可以利用它完成一些琐碎的工作，只要最后输出一个文件，都可以用make来完成

这是一个最简单的Makefile

```
filelist❶:❷*❸❹    ls -lF > filelist❺
```

❶ 输出的目标文件，不能省略。如果有多个文件，可以使用all
❷ 分隔符，不能省略
❸ 输入文件，可以省略

❹ 这一行必须以**TAB**字符起始，不能使用空格代替
❺ make的命令

可以使用变量代替命令，便于维护

```
TARGET = filelist❶SOURCE =  *ARG = -lFAPPLICATION = ls$(TARGET):$(SOURCE)❷    $(APPLICATION) $(ARG
```

❶ 定义变量，传统上用大写
❷ 使用变量写Makefile

Makefile可以有多个目标文件，我们前面提到，gcc编译时先生成目标文件，再把目标文件链接成可执行文件，Makefile应该是这样的：

```
OBJECTS = main.o kbd.o command.o display.o \❶            insert.o search.o files.o utils.oexe : $
```

❶ 如果写在多行，要用脱字符换行
❷ 如何生成中间文件
❸ 伪目标文件，**make clean**并不生成clean文件，而是清理编译结果

Makefile还有很多强大的机制，这里就不详细介绍了

---

[30] 执行当前目录下的configure脚本