大家一起用 gtk 编程 3(初步使用 Makefile)

大家一起用 gtk 编程 3(初步使用 Makefile)

转载请注明出处：http://lvjinhua.cublog.cn
作者：lvjinhua at gmail dot com
(本文档最后由杨小邪编辑整理)
2009.1.7

5、初步使用 Makefile

上回说道，咱们使用 GTK 创建了一个能够截获事件的 GTK 程序，本节将简单地介绍一下如何使用 Makefile 来组织我们的源程序，使用 Makefile 是基于如下几个原因：

- 能实现从无限简单到无穷复杂的软件工程组织
- 能方便地与众多的 Linux 命令与实用程序集成，当然也包括 Vim 与 emacs
- 还有很多，就留给读者自己探索吧...

笔者在这里介绍Makefile的主要目的是想让大家知道Linux下有这样一个工具，使用它确实很方便，并且希望大家在有需要的时候能够想到它。

总的来说，Makefile 是使用一系列的依赖关系和时间值，来决定是否对一个目标进行重构建；不废话了，来看看我们的 Makefile 文件长什么样：

Makefile

注意：Makefile 文件如果从网页上直接拷贝，往往不能成功正常使用，请从下边的附件中下载可用的文件。

```
CC=gcc
PROG_NAME=hello_dubuntu2
INCS=
SRCS=hello_dubuntu2.c

#从 xx.c 文件得到 xx.o 文件
OBJS=${SRCS:.c=.o}

#编译 GTK 程序时要用到的库
LIBS=gtk+-2.0
#---- 用户修改区域 结束

# -O2
CFLAGS=`pkg-config --cflags ${LIBS}` -g -Wall
LDFLAGS=`pkg-config --libs ${LIBS}`   -g -Wall
```

```
all: ${PROG_NAME}

${PROG_NAME}:${OBJS}
	${CC} -o ${PROG_NAME} ${OBJS} ${LDFLAGS}
#注意：上边”${CC}″的前边有一个 TAB 键，而不是空格

#如果有头文件进行修改，则自动编译源文件
${OBJS}:${INCS}

.c.o:
	${CC} -c $<  ${CFLAGS}

clean:
	rm -f *.o  ${PROG_NAME}

rebuild: clean all
```

编译：将此 Makefile 文件与上一节的 hello_dubuntu2.c 文件放在同一个文件夹下，然后运行如下命令即可编译出 hello_dubuntu2 可执行文件：

```
make
```

文件:hello_dubuntu.tar
大小:10KB
下载:下载

程序注释：

1）刚开始的几行定义了几个变量，比如用 CC 来代替真正的 gcc 编译器，这样当想换别的编译器来编译我们的程序时直接改变 CC 变量的值就 OK 了。其它的：

- CC 指代用来编译程序的编译器，这里使用 gcc
- RPOG_NAME 指代最终要生成的可执行文件名，需手工填入
- INCS 指代工程文件中所自定义的所有头文件，需手工填入
- SRCS 指代所有使用到的源文件，需手工填入
- OBJS 是通过 .c 文件得到的 .o 文件，因为每个 .c 文件都将编译生成一个对应的 .o 文件，自动生成
- LIBS 用来指代编译 GTK 程序时需要使用到的 GTK 相关库，一般使用默认的 gtk+-2.0 即可，需手工填入
- CFLAGS 用来指代编译程序时使用到的一些编译选项， -g 表示生成调试信息以供 GDB 使用,-Wall 表示生成编译时的警告信息(W 表示 Warning，all 表示全部)
- LDFLAGS 用来指代进行程序连接时使用到的一些选项

以上这些变量，将在Makefile的剩余部分使用 ${xxx} 的形式进行引用。

2）关于“目标”：“依赖关系”
从“all: ${PROG_NAME}”这一句开始，Makefile就开始使用 目标：依赖 的关系来处理真正的程序编译了；而它们下边以 TAB 键开始的行就是满足依赖关系后要运行的程序（注意：是TAB键，不是空格）。具体笔者就不详细描述了，有兴趣的读者可以参考下边推荐的教程。

3）关于 “.c.o:” 语句：
这句话的意思就是说，当遇到一个 .c 文件，那么使用它下边的命令
“ ${CC} -c $< ${CFLAGS}” 将 .c 文件编译为 .o 文件；命令中 $< 用来代替对应的 .c 文件的文件名。

4）关于"clean" 和 "rebuild"：
这两个者伪目标，必须使用 "make clean" 或 "make rebuild" 来引用，"make clean" 用于删除生成的 .o 目标文件和可执行文件； "make rebuild" 用于对整个工程进行重构建。

5）小技巧：
在 Vim中，先按 ESC 键回到命令模式，再按 ":make" 命令即可对Makefile工程进行编译，如果有编译错误，那么Vim会自动定位到产生错误的源程序所对应的代码行，非常利于程序的修改，使用命 令":cn" 和 "cp" 还可以定位到“下一个错误”与“上一个错误”。

当然，更高级的用法是，使用 autoconf, automake 等工具来自动生成一个符合GNU标准的工程，然后通过它们来自动生成 Makefile 文件，这样做更佳方便快捷，自动化程度更高，并且能够使程序更具有可移值性，本文将在后续章节开辟专门一节来进行详细描述。

最后，推荐一篇好教程：跟我一起写Makefile