

一、一般软件项目的开发

1.mkdir amhello

新建工作目录

2.在amhello目录下创建NEWS空文件；

3.在amhello目录下创建README空文件；

4.在amhello目录下创建AUTHORS空文件；

5.在amhello目录下创建ChangeLog空文件；

6.make src

新建放置源码的目录

7.在src中放置源文件

main.c say.c say.h 等

8.在src目录下编写Makefile.am :

`bin_PROGRAMS = hello` //bin 表示我们要编译的是可执行程序, 这些程序会安装在bindir目录下 (bindir目录通常为/usr/local/bin), 要编译的程序名字为hello, PROGRAMS表示要编译一个项目。

`hello_SOURCES = main.c say.c say.h` //hello表示要编译的可执行程序名, main.c say.c say.h为生成hello可执行程序所要用的源文件。

9.在amhello目录下编写Makefile.am:

`SUBDIRS = src` //在src目录下递归编译

10.在amhello目录下编写configure.ac : (可通过autoscan获得configure.scan文件, 修改后改为configure.ac)

`AC_PREREQ([2.69])` //需要的最低autoconf版本。

`AC_CONFIG_SRCDIR([src/main.c])` //一个安全的检查src/main.c 将是一个发布的源文件。这让configure脚本确保自己运行在正确的目录中。

`AC_INIT([amhello], [1.0], [liuhw@itep.com.cn])` //对autoconf进行初始化。三个参数分别是包名、版本号、报告bug的地址。

`AC_INIT_AUTOMAKE([-Wall -Werror])` //初始化automake,参数会传递给automake。

`AC_CONFIG_AUX_DIR([build-aux])` //保存辅助工具

`AC_PROG_CC` //检查C编译器

AC_CONFIG_HEADERS([config.h])//会从config.h.in中生成config.h

AC_CONFIG_FILES([Makefile src/Makefile]) //从Makefile.in文件中生成Makefile

AC_OUTPUT //输出所有声明过的文件

11.通过在amhello目录下执行autoreconf --install

amhello目录下生成：

aclocal.m4

autom4te.cache(目录)

compile

config.h.in

configure

COPYING

depcomp

INSTALL

install-sh

Makefile.in

missing

src目录下生成：

Makefile.in

文件说明：

Makefile.in config.h.in configure src/Makefile.in //预置配置模板文件

aclocal.m4 //定义第三方configure.ac文件中使用的宏

depcomp install-sh missing //构建中使用的辅助工具

autom4te.cache/ output.0 requests traces.1 output.1 traces.0 //autotools 缓存文件

12.在amhello目录下执行 ./configure

amhello目录下生成：

config.h

config.log

config.status

Makefile

stamp-h1

src目录下生成：

Makefile

13.在amhello目录下执行 make

src目录下生成：

hello

main.o

say.o

14.在amhello目录下执行make distcheck

amhello目录下生成：

amhello-1.0.tar.gz

注释：

Autotools生成的configure运行时从config.h.in生成config.h，从Makefile.in生成Makefile。

二、国际化软件项目的开发

1.在amhello目录下对configure.ac文件进行添加：

AC_INIT([amhello], [1.0], [liuhw@itep.com.cn]) //对autoconf进行初始化。三个参数分别是包名、版本号、报告bug的地址。

AC_INIT_AUTOMAKE([-Wall -Werror]) //初始化automake,参数会传递给automake。

AC_CONFIG_AUX_DIR([build-aux])//保存辅助工具

AM_GNU_GETTEXT_VERSION([0.17]) //使用的gettext版本。

AM_GNU_GETTEXT([external]) //使用外部*gettext*库。

AC_PROG_CC //检查C编译器

AC_CONFIG_HEADERS([config.h])//会从config.h.in中生成config.h

AC_CONFIG_FILES([Makefile src/Makefile]) //从Makefile.in文件中生成Makefile

AC_OUTPUT //输出所有声明过的文件

2.执行gettextize --copy --no-changelog //这将会把gettextize需要的基础文件拷贝到当前目录。

gettextize 会修改configure.ac

将： AC_CONFIG_FILES([Makefile src/Makefile])

改为： AC_CONFIG_FILES([Makefile src/Makefile po/Makefile.in])

gettextize 会修改amhello目录下的Makefile.am

将： SUBDIRS = src

改为： SUBDIRS = po src

ACLOCAL_AMFLAGE = -I m4

EXTRA_DIST = config.rpath

3.将/usr/share/gettext/gettext.h拷贝到src目录下

//也可以直接#include <glib/gi18n.h> 就不需要复制了

4.把po/Makevars.template重命名为po/Makevars.

修改COPYRIGHT HOLDER = liuhw

修改MSGIN_BUGS_ADDRESS = \$(PACKAGE_BUGREPORT)

5.往po/POTFILES.in中添加

src/main.c src/say.c

表示从src/main.c和src/say.c中提取需要翻译的字符串。

6.修改src/Makefile.am链需要的库：

`AM_CPPFLAGS = -DLOCALEDIR=\"$(localedir)\"` //为每个C文件增加了一个LOCALEDIR的宏，这个宏的内容就是po文件将要安装到的位置。

`bin_PROGRAMS = hello`

`hello_SOURCES = main.c say.c say.h gettext.h` //把gettext.h添加到编译文件中

`LDADD = $(LIBINTL)` //表示链接LIBINTL库

7.修改源代码支持国际化.

在src目录下，修改main.c文件

`#include <config.h>`

`#include <locale.h>`

`#include "gettext.h"` //也可以直接`#include <glib/gi18n.h>`

`#include "say.h"`

`int main()`

```
{  
  
    setlocale(LC_ALL, "");  
  
    bindtextdomain (PACKAGE, LOCALEDIR);  
  
    textdomain (PACKAGE);  
  
    say_hello();  
  
    return 0;  
  
}
```

在src目录下，修改print_hello.c文件

```
#include <stdio.h>  
  
#include "gettext.h"          //也可以直接#include <glib/gi18n.h>  
  
#define _(string) gettext(string)  
  
void say_hello()  
  
{  
  
    puts("Hello World!");  
  
    printf(_("This is %s\n"), PACKAGE_STRING);  
  
}
```

8.再次执行autoreconf --install (autoreconf -i -f -v)

9.执行 ./configure

10.执行 make

在po目录下就会生成amhello.pot。

11.本地化翻译。

在po目录下执行msginit -l zh_CN就会生成zh_CN.po

编辑该文件：

```
"Content-Type: text/plain; charset=UTF-8\n"
```

```
#: src/say.c:8
```

```
msgid "Hello World!"
```

```
msgstr "你好世界!"
```

```
#: src/say.c:9
```

```
#, c-format
```

```
msgid "This is %s\n"
```

```
msgstr "这是%s\n"
```

12.告诉gettext我们完成了zh_CN的翻译。

新建LINGUAS文件,并在文件中写入：

```
zh_CN
```

13.在amhello目录下执行./configure --prefix ~/test

14.在amhello目录下执行make

15.在po目录下执行 make update-po

生成zh_CN.gmo

16.在amhello目录下执行 make install

主目录 (~) 下生成test目录：

test目录下有两个子目录：

bin //bin目录下有可执行文件hello

share //share/local/zh_CN/LC_MESSAGES/目录下有amhello.mo文件，用来翻译用的。

17.到~/test/bin目录执行./hello

输出：

Hello World!

This is amhello 1.0

18.改变LANG=zh_CN，再次执行./hello

输出：

你好世界！

这是amhello 1.0