Linux 下 C 语言的本地化/国际化实现

Linux 下 C 语言的本地化／国际化实现

中标普华 李东东 2010

在 **linux** 系统下，以 **c 语言**程序为例来实现程序的**国际化**，即让程序根据 Linux 系统不同的语言环境的不同来显示出对应该语言的文字，即先让 c 程序支持**国际化**然后再进行**本地化翻译**。

Linux 上实现这个过程需要用到 **xgettext** 和 **msgfmt** 这两个工具。

Xgettext 是国际化的工具，用来提取程序中的字符串，生成＊.po 或是*.pot 的文件，

msgfmt 是本地化的工具，用来编译翻译后的.po 文件为.mo 文件,这样系统在启动时候会扫描系统环境提取对应名字的.mo 文件中的字符串替代原来的英文，实现本地化。

如我们来做一个简单的 rpm 包，包文件的目录树如下：

hello.c /*我们用来测试的 c 语言程序*/

po /zh_CN.po /*放在该包根目录下的 po 目录。对应于该程序进行的中文翻译*/

第一步、支持国际化的 C 程序(hello.c)如下：

一个支持本地化的 C 程序，需要用 setlocale()来设定翻译选择的语言环境，一组或多组 textdomain()和 bindtextdomain()函数来指定用到的翻译文件和该文件所在路径,其 textdomain()函数用来指定翻译文件的名字，而 bindtextdomain()函数用来指定该名称的翻译文件所在路径（如果存在多组翻译文件和路径，则 textdomain()函数下面的待翻译语句对应查找该函数所指定的路径和名称）

```
#include <stdio.h>
#include <libintl.h>  // gettext 库支持
#include <locale.h>  // 本地化 locale 的翻译支持

#define _(STRING)  gettext(STRING)  //为了减少代码量和输入量，用别名_(STRING)替换 gettext(STRING)
#define PACKAGE "hello"      // 定义软件包的名字为 hello
#define LOCALEDIR "/usr/share/locale/"   //定义系统语言环境的目录，该目录下存放各种国际化的语言，不同系统可能有差异。


int main (int argv, char  *argc[] )
{
      setlocale (LC_ALL, "");  //设置 locale 为系统默认语言环境，gtk 程序不需要,因为 gtk_init()已经帮我们做了
      bindtextdomain (PACKAGE, LOCALEDIR); //关联包名称及其对应的国际化翻译语言所在路径
      textdomain (PACKAGE); //定义.mo 文件的名字，省略“.mo”后缀（可以多个路径下的不同 mo 文件）
      printf(_("Hello, World!\n"));       //国际化字符串支持，结合上面的替换，实际应是：printf(gettext("Hello, World!\n"));
      printf(_("This is a example of locale&rpm @ Neoshine Desktop 5!\n"));
      return 0;
}
```

关于不同路径下面的多个翻译文件

对于一个 C 语言程序允许对应于系统中不同位置的多个 mo 文件，用一组 bindtextdomain()和 textdomain()函数来定义紧跟在该组中 textdomain()函数后的支持国际化的翻译语言包所在位置，当程序运行时根据指定的位置来查找翻译文件并显示。如下：

```
bindtextdomain (PACKAGE, LOCALEDIR);
```

By：沉思小屋 http://leedd.com/2010/03/linux-c-i18n-l10n-xgettext-msgfmt-rpmbuild/

Linux 下 C 语言的本地化/国际化实现

```
textdomain (PACKAGE);
    printf(_("Hello, World!\n"));
    printf(_("test 001\n"));
        /*上面的翻译用的是 LOCALEDIR 路径下 PACKAGE.mo 的翻译*/
bindtextdomain (P2, L2);
textdomain (P2);
    printf(_("Hello, World!\n"));        /*该行的翻译用的是 L2 路径下 P2.mo*/
```

当然也可以先用 bindtextdomain()来定义系统语言包所在位置，然后再分别指定翻译文件在哪个路径下的语言包：

```
bindtextdomain (PACKAGE, LOCALEDIR);
bindtextdomain (P2, L2);

textdomain (PACKAGE);
    printf(_("Hello, World!\n"));
    printf(_("test 001\n"));
        /*上面的翻译用的是 LOCALEDIR 路径下 PACKAGE.mo 的翻译*/
textdomain (P2);
    printf(_("Hello, World!\n"));      /*该行的翻译用的是 L2 路径下 P2.mo*/
```

第二步、提取待翻译 po 并进行本地化

先用下面的命令提取出 c 程序中需要进行本地化的语言文件。

```
[Lee@leedd.com hello]$ xgettext -k --keyword=_  hello.c  -o  hello.pot
```

再用 poedit 或是 vim 翻译 hello.pot 中对应英文为中文并另存为 zh_CN.po，作为中文本地化的翻译文件。

对于 po 文件的头文件格式

首先来简单介绍下 po 内容的头文件格式(详细的可以参考“**Mos 翻译注意事项—内部版**”)：

```
 1 msgid ""                          //待翻译的文字，msgid 字符串是由 GNU gettext 工具产生和管理，并不允许翻译人员改动
 2 msgstr ""                         //对应的翻译后文字
 3 "Project-Id-Version: \n"                    //项目名称版本，一般为该包名称 如:hello，亦可空缺
 4 "Report-Msgid-Bugs-To: \n"                  //Bugs 信息报告方法
 5 "POT-Creation-Date: 2010-03-11 11:52+0800\n"   //pot 文件创建时间
 6 "PO-Revision-Date: \n"                      //生成 po 文件的时间
 7 "Last-Translator: Leedd <dongdong.li@cs2c.com.cn>\n"  //最后翻译者信息，格式：名称<邮箱>\n
 8 "Language-Team: \n"          //翻译小组名称，上游社区本地化群组的翻译成员填入其组织信息，亦可以不填
 9 "MIME-Version: 1.0\n"       //MIME 版本，一般取默认即可，不影响翻译文件
10 "Content-Type: text/plain; charset=UTF-8\n"        // 刚创建的 pot 文件默认为“charset=CHARSET”，编辑 po 文件时，注意要将字符调整为可移植的编码格式，如：UTF-8，才可以得到正确结果。否则会有如下警告：（hello.po：警告：字符集“CHARSET”不是可移植的编码名称。将消息转换为用户字符集可能不工作。）
11 "Content-Transfer-Encoding: 8bit\n"
12 "Plural-Forms: nplurals=1; plural=0;\n"       // nplurals= 表示单复数变化形式的数量,中文中单复数相同所以为 1
                                    // plural=  表示对第几种单复数变化取相应的第几种译文,中文中只取 msgstr[0] 即可
```

如下面翻译后的 zh_CN.po 文件的内容：

By：沉思小屋　http://leedd.com/2010/03/linux-c-i18n-l10n-xgettext-msgfmt-rpmbuild/

Linux 下 C 语言的本地化/国际化实现

```
msgid ""
msgstr ""
"Project-Id-Version: \n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2010-03-11 11:52+0800\n"
"PO-Revision-Date: \n"
"Last-Translator: Leedd <dongdong.li@cs2c.com.cn>\n"
"Language-Team: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=1; plural=0;\n"
#: hello.c:15
#, c-format
msgid "Hello, World!\n"
msgstr "你好，世界！\n"

#: hello.c:16
#, c-format
msgid "This is a example of locale&rpm @ Neoshine Desktop 5!\n"
msgstr "这是中标普华桌面 5 下的关于\"本地化\"和\"rpm 打包\"的测试!\n"
```

第三步、把翻译后的文件打包

制作 rpm 包相当于给源程序添加一个外衣，让用户很容易的安装、使用，而不用理会源程序的编译过程等细节。如果要进行 rpmbuild 我们需要先写个 SPEC 文件，如下：

Neoshine-hello-wrold.spec 文件内容

```
Name:          neoshine-hello-world
Version:       5.0
Release:       1
License:       GPL+
Group:         Applications/System
URL:           http://www.cs2c.com.cn
Source0:       %{name}-%{version}.tar.bz2
BuildRoot:     %{_tmppath}/%{name}-%{version}-%{release}-root

Summary:       A applet for Neoshine Desktop (or the other similar Linux system).
BuildRequires:  gettext
BuildRequires:  gcc


%description
        A hello world examples for the locale&rpm

%prep
%setup -q

##,%build

%install
```

By：沉思小屋　http://leedd.com/2010/03/linux-c-i18n-l10n-xgettext-msgfmt-rpmbuild/

Linux 下 C 语言的本地化/国际化实现

```
24
25      mkdir -p ${RPM_BUILD_ROOT}/usr/share/locale/zh_CN/LC_MESSAGES
26      mkdir -p ${RPM_BUILD_ROOT}/var/tmp/hello
27      xgettext -k --keyword=_ hello.c -o po/hello.pot
28      pushd ./po
29          msgmerge zh_CN.po hello.pot
30          msgfmt zh_CN.po -o hello.mo
31          cp hello.mo ${RPM_BUILD_ROOT}/usr/share/locale/zh_CN/LC_MESSAGES
32      popd
33
34      gcc hello.c -o ${RPM_BUILD_ROOT}/var/tmp/hello/hello.out
35
36 %clean
37 rm -rf $RPM_BUILD_ROOT
38
39 %files
40 %defattr(-,root,root)
41 /var/tmp/hello/*
42 /usr/share/locale/zh_CN/LC_MESSAGES
43
44 %changelog
45 * Thu Mar 11 2010 Leedd <dongdong.li@cs2c.com.cn> 5.0-1
46 - Inital
```

已经有了源文件和 SPEC 文件后，需要把他们分别放在 rpmbuild 目录下对应的 SOURCES 和 SPECS 目录中，然后打包：

```
[Lee@leedd.com SPECS]$ rpmbuild -bs neoshine-hello-world.spec
Wrote: /home/Lee/rpmbuild/SRPMS/neoshine-hello-world-5.0-1.src.rpm
[Lee@leedd.com SPECS]$ mock -r neoshine-5-i686 --rebuild
/home/Lee/rpmbuild/SRPMS/neoshine-hello-world-5.0-1.src.rpm
```

本文用到的源程序:

SOURCES 源文件：neoshine-hello-world-5.0.tar.bz2

SPEC 文件：neoshine-hello-world.spec

编译后的源码包：neoshine-hello-world-5.0-1.src.rpm

参考文章：

[1] Hello World http://zh.wikipedia.org/wiki/Hello_World#C

[2] Linux 国际化编程 http://riuliyu.blog126.fc2.com/blog-entry-20.html

[3] 认识下 linux 下程序的国际化（C 语言实例）

http://swinging-breeze.blogspot.com/2009/05/linuxc.html （需要翻墙）

Google 的缓存： http://203.208.39.132/search?q=cache:E-Os3g0yHjsJ:swinging-breeze.blogspot.com/2009/05/linuxc.html+hello+world+%E5%9B%BD%E9%99%85%E5%8C%96+c%E8%AF%AD%E8%A8%80&cd=1&hl=zh-CN&ct=clnk&gl=cn&st_usg=ALhdy2_j1VcwccRlk394Xz6n81ZnDCvbuQ

By：沉思小屋 http://leedd.com/2010/03/linux-c-i18n-l10n-xgettext-msgfmt-rpmbuild/

Linux下C语言的本地化/国际化实现

－－

Changelog:

* Fri Mar 12 2010 Leedd beta1
- 创建

* Thu Mar 18 2010 Leedd beta2
- 增加了多个mo文件的处理介绍和po头文件的介绍

By：沉思小屋 http://leedd.com/2010/03/linux-c-i18n-l10n-xgettext-msgfmt-rpmbuild/