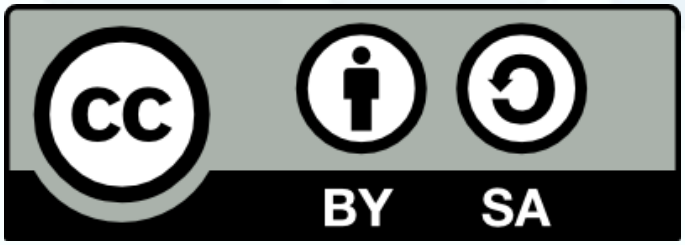
Procesos «i18n»

Miguel Anxo Bouzada
para:
Proxecto Trasno
GALPon



trasno

Internacionalización «i18n»

A internacionalización é a serie de tarefas que se deben levar a cabo para que un determinado elemento póidase rexionalizar. No caso dos programas informáticos débese **retocar o código para que permita mostrar mensaxes en varios idiomas**, por exemplo. Tamén leva toda a serie de tarefas sobre definición de estándares comúns, procedemento de traballo, etc.

trasno

Localización «l10n»

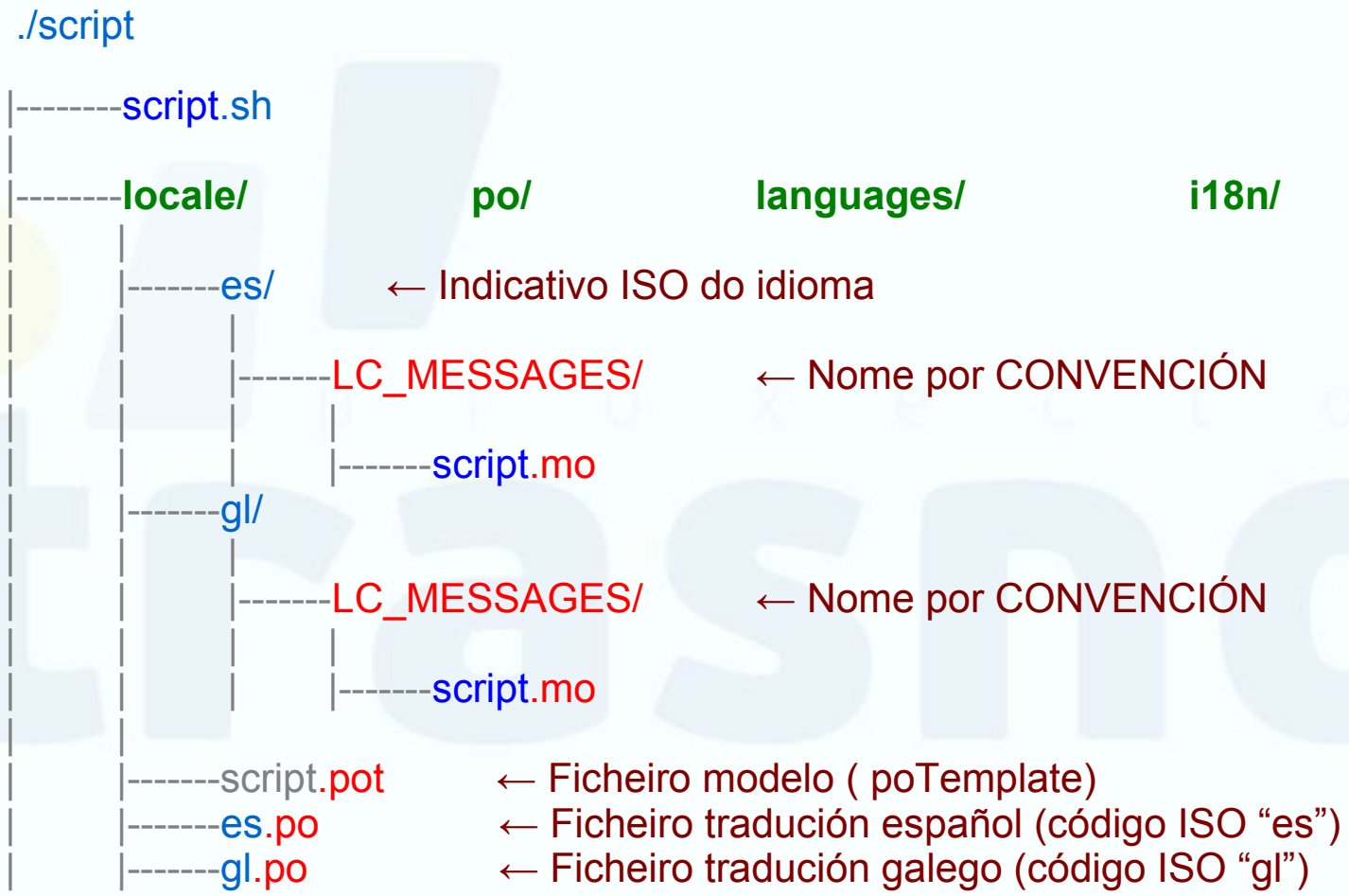
A localización ou rexionalización consiste en tomar elementos internacionalizados e adaptalos a unha determinada rexión. **A maior parte deste traballo reside na tradución**, pero existen outras tarefas como o cambio de formatos de datas, moeda, calendario e calquera outro elemento susceptible de afectar ao entendemento dun usuario dun determinado lugar.

trasno

i18n pautas xerais

- Todos os procesos en todas as linguaxes de programación teñen varios elementos en común:
  - Definir a estrutura de ficheiros
  - Definición de cabeceiras
  - Preparación dos textos a localizar
  - Obtención dos ficheiros de localización
    - Ficheiros de modelo
    - Ficheiros de tradución
  - Xeración de binarios localizados

Definir a estrutura de ficheiros



Definir a estrutura de ficheiros

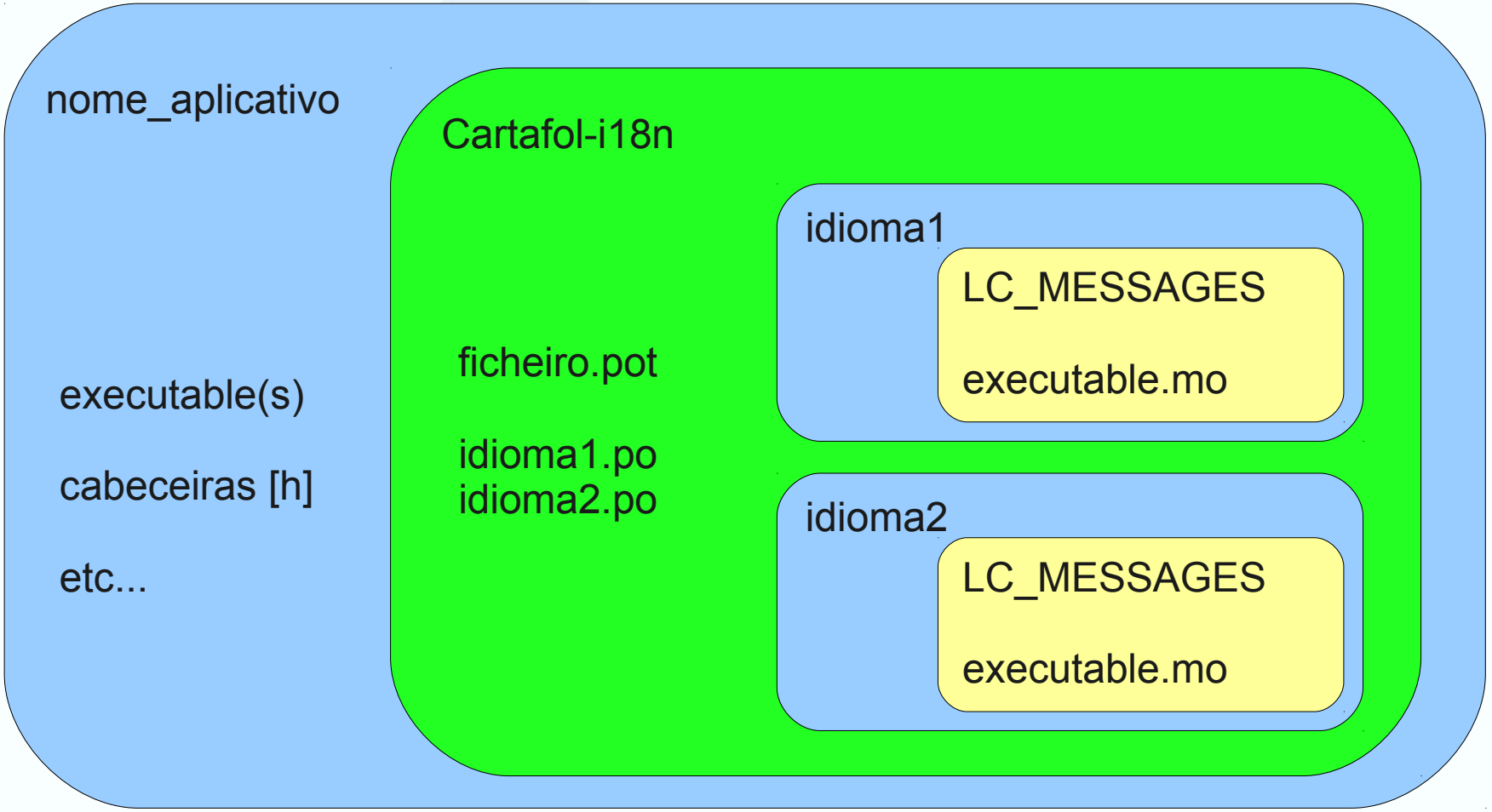
```
./script
|--------script.sh
|
|--------/usr/local/share/locale/        ← PREDEFINIDA DE USUARIO
|        |
|        |-------es/          ← Indicativo ISO do idioma
|        |        |
|        |        |-------LC_MESSAGES/        ← Nome por CONVENCIÓN
|        |        |        |
|        |        |        |-------script.mo
|        |-------gl/
|        |        |
|        |        |-------LC_MESSAGES/        ← Nome por CONVENCIÓN
|        |        |        |
|        |        |        |-------script.mo
|        |
|        |-------script.pot        ← Ficheiro modelo ( poTemplate)
|        |-------es.po             ← Ficheiro tradución español (código ISO “es”)
|        |-------gl.po             ← Ficheiro tradución galego (código ISO “gl”)
```

Definir a estrutura de ficheiros

```
./script
|--------script.sh
|
|--------/usr/share/locale/        ← PREDETERMINADA DO SISTEMA
|        |
|        |-------es/          ← Indicativo ISO do idioma
|        |        |
|        |        |-------LC_MESSAGES/        ← Nome por CONVENCIÓN
|        |        |        |
|        |        |        |-------script.mo
|        |-------gl/
|        |        |
|        |        |-------LC_MESSAGES/        ← Nome por CONVENCIÓN
|        |        |        |
|        |        |        |-------script.mo
|        |
|        |-------script.pot        ← Ficheiro modelo ( poTemplate)
|        |-------es.po             ← Ficheiro tradución español (código ISO “es”)
|        |-------gl.po             ← Ficheiro tradución galego (código ISO “gl”)
```

Definir a estrutura de ficheiros

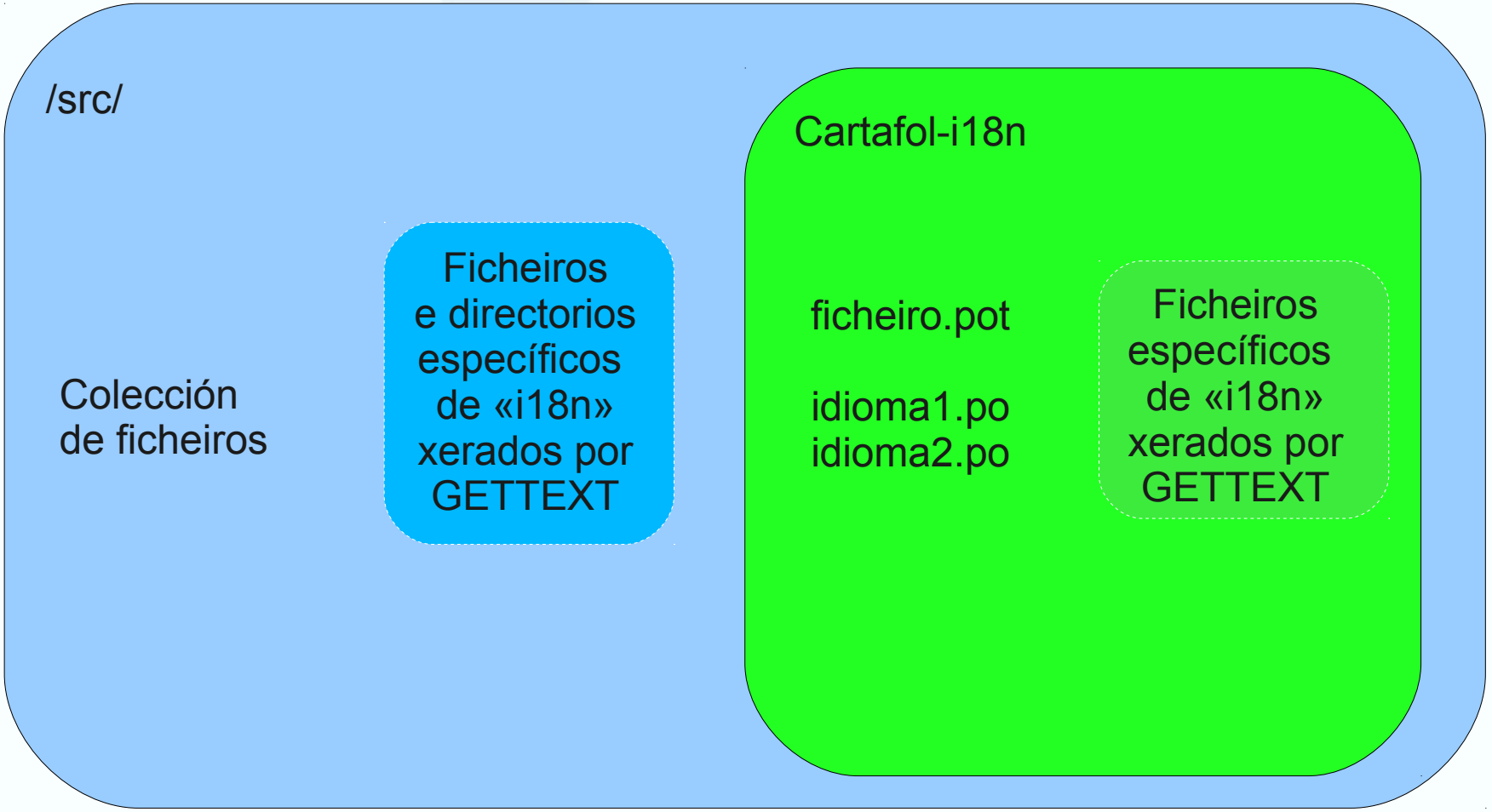
Desenvolvemento



trasno

Definir a estrutura de ficheiros

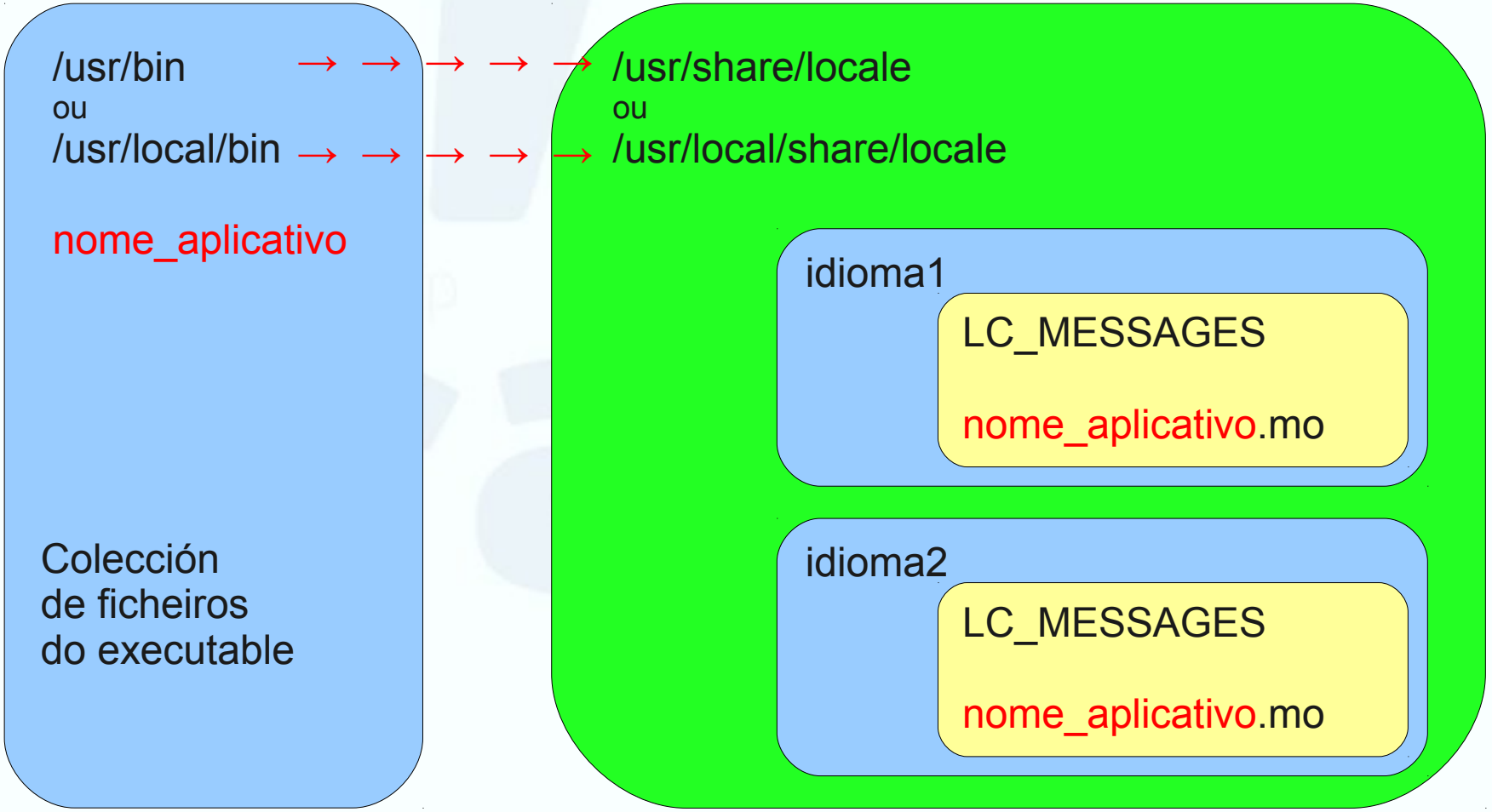
Desenvolvemento en C



trasno

Definir a estrutura de ficheiros

Final



trasno

Definición de cabeceiras

Teremos que definir o parámetro

TEXTDOMAIN=" ? "

Definición de cabeceiras

Bash Scripting

```
export TEXTDOMAIN="nome_executable"
export TEXTDOMAINDIR="./locale"
```

Definición de cabeceiras

nome_executable

é o nome que o executable vai ir a buscar como binario_de_tradución.mo.
NOTA: aquí indicaráse **sen** extensión.

Como NORMA (ou preferible) debe chamarse igual, ainda que non é estritamente necesario.

O que si está OBLIGADO e que aquí indiquemos o nome do ficheiro **.mo** que se vai xerar.

Definición de cabeceiras

Phyton

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton+Glade

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gtk.glade.textdomain(APP)
gtk.glade.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton+Glade

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gtk.glade.textdomain(APP)
gtk.glade.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

Phyton+Glade

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gtk.glade.textdomain(APP)
gtk.glade.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Definición de cabeceiras

"C"

Utilizando gettext("cadea_de_texto_orixinal")

Este método de marcaxe considerase «obsoleto»

```
/* includes para localizar con gettext */
#include <libintl.h>
#include <locale.h>

// Definición de ficheiros e rutas para gettext
trans_text(void)
{
    setlocale( LC_ALL, "" );
    bindtextdomain( "nome_executable", "/usr/share/locale" );
    bind_textdomain_codeset ( "nome_executable", "UTF-8" );
    textdomain( "nome_executable" );
}
```

Definición de cabeceiras

“C”

Utilizando _("cadea_de_texto_orixinal")

```
/* includes para localizar con gettext */
#include <libintl.h>
#include <locale.h>

#define _(String) gettext(String)    // cadeas traducibles
#define N_(String) (String)          // cadeas non traducibles

// Definición de ficheiros e rutas para gettext
trans_text(void)
{...
```

Definición de cabeceiras

"C"

Utilizando _("cadea_de_texto_orixinal")

```
/* includes para localizar con gettext */
#include <libintl.h>
#include <locale.h>

#define _(String) gettext(String)    // cadeas traducibles
#define N_(String) (String)          // cadeas non traducibles

// Definición de ficheiros e rutas para gettext
trans_text(void)
{
    setlocale( LC_ALL, "" );
    bindtextdomain( "nome_executable", "/usr/share/locale" );
    bind_textdomain_codeset ("nome_executable", "UTF-8");
    textdomain( "nome_executable" );
}
```

Definición de cabeceiras

"C"

```
#define _(String) gettext(String)    // cadeas traducibles
#define N_(String) (String)          // cadeas non traducibles
```

Cadeas _(String)
As cadeas que se marquen así, serán traducidas non só como visibles na IGU, senón tamén nos procesos internos.

Cadeas N_(String)
As cadeas que se marquen así, serán traducidas só como visibles na IGU, máis respectarase a cadea orixinal nos procesos internos.

Definición de cabeceiras

"C"

```
trans_text(void)
{
    setlocale( LC_ALL, "" );
    bindtextdomain( "nome_executable", "/usr/share/locale" );
    bind_textdomain_codeset ( "nome_executable", "UTF-8" );
    textdomain( "nome_executable" );
}

main (int argc, char *argv[])
{
  trans_text();
...
```

Preparación dos textos a localizar

Teremos que facer que os textos teñan unha estrutura determinada a fin de que poidan seren «discriminados» por GETTEXT

trasno

Preparación dos textos a localizar

Bash Scripting

```
mode=$(Xdialog --title "MAC Change" --no-tags  \
      --item-help \
      --icon $HOME/.icewm/icons/sysinfo.xpm \
    --radiolist "Change mode" 0 0 3  \
    "-r" "Set fully random MAC."    on   "1"      \
    "-e" "Dont change the vendor bytes."    off  "2"  2>&1)
```

Preparación dos textos a localizar

Bash Scripting

```
mode=$(Xdialog --title $(gettext "MAC Change") --no-tags  \
      --item-help \
      --icon $HOME/.icewm/icons/sysinfo.xpm \
    --radiolist "$(gettext "Change mode")" 0 0 3  \
    "-r" "$(gettext "Set fully random MAC.")"    on   "1"    \
    "-e" "$(gettext "Dont change the vendor bytes.")"    off  "2"  2>&1)
```

Preparación dos textos a localizar

Bash Scripting

```
--title $(gettext "MAC Change")
--radiolist "$(gettext "Change mode")"
"-r" "$(gettext "Set fully random MAC.")"
"-e" "$(gettext "Dont change the vendor bytes.")"
```

Preparación dos textos a localizar

Bash Scripting

Recomendo (así o fago eu) empregar sempre o delimitador externo xa que non muda o comportamento e evita erros

```
"$(gettext "Dont change the vendor bytes.")"
```

Preparación dos textos a localizar

Bash Scripting

"$(gettext "Dont change the vendor bytes.")"

Espazo

Preparación dos textos a localizar

“C”, e Python

Tomemos este exemplo en “C”

```
void create_file_selection(void)
{
    // file_selection_box = gtk_file_selection_new("Select MIXER application");
    file_selection_box = gtk_file_chooser_dialog_new("Select MIXER application", NULL,
    ....
```

ímos a analizalo

Preparación dos textos a localizar

```
// file_selection_box = gtk_file_selection_new("Select MIXER application");
file_selection_box = gtk_file_chooser_dialog_new("Select MIXER application", NULL,
```

A primeira liña comeza cun comentario (//) polo que iso é de uso interno para informacion dos/aos desenvolvedores. Polo tanto ignoramos esa liña

Na segunda liña xa vemos unha cadea (por estar entre comiñas «”» ou «'») de texto, segundo as convencións.

Preparación dos textos a localizar

```
// file_selection_box = gtk_file_selection_new("Select MIXER application");
file_selection_box = gtk_file_chooser_dialog_new("Select MIXER application", NULL,
```

É esta cadea a que teremos que modificar para que quede así:

```
file_selection_box = gtk_file_chooser_dialog_new(_("Select MIXER application"), NULL,
```

Preparación dos textos a localizar

```
..._new("Select MIXER application", NULL,

..._new(_("Select MIXER application"), NULL,
```

Preparación dos textos a localizar

Exemplo clásico de uso de N_("_String")

```
static GtkActionEntry _menu_entries[] = {
  { "GameMenu",     NULL,          N_("_Game") },
  { "EditMenu",     NULL,          N_("_Edit") },
  { "SettingsMenu", NULL,          N_("_Settings") },
  { "HelpMenu",     NULL,          N_("_Help") },
```

Preparación dos textos a localizar

```
static GtkActionEntry _menu_entries[] = {
  { "GameMenu",     NULL,                N_("_Game") },
  { "EditMenu",     NULL,                N_("_Edit") },
  { "SettingsMenu", NULL,                N_("_Settings") },
  { "HelpMenu",     NULL,                N_("_Help") },
```

Neste caso marcamos con N_(?) aquelas entradas de texto que son «constantes» do proceso, pero que queremos que cara ao exterior (IGU) estén traducidas, pero que se se traduciran «totalmente» o(s) proceso(s) non atoparian.

Preparación dos textos a localizar

Este caso de uso vese moi ben en:

GNU gettext II - Un ejemplo en C
http://www.alu.ua.es/p/psp4/Documentacion/Febrero_2002/gettext2.html

Preparación dos textos a localizar

```
N_("_Game")
N_("_Edit")
N_("_Settings")
N_("_Help")
```

Outra cousa a ter en conta é non confundir a «_» (barra baixa) N_(?) coa tamen «_» (barra baixa) N_("_Help") esta segunda corresponde ao atallo de teclado (ou tecla quente) que vemos na IGU como Help ou xa traducido como Axuda (Ax_uda).

Preparación dos textos a localizar

```
_("Select MIXER application")
```

En Python usase exactamente o mesmo esquema, xa que temos definido que así sexa en:

```
import gettext
APP="nome_executable"
DIR="./locale" #/usr/local/share/locale" #/usr/share/locale"
gettext.textdomain(APP)
gettext.bindtextdomain(APP, DIR)
gettext.install('nome_executable', 'po', True)
_ = gettext.gettext
```

Preparación dos textos a localizar

Exemplos de liñas en Python

```
self.btn_test.set_label(_(" Test Drive Speed"))
column = gtk.TreeViewColumn(_("New name"), renderer, text = 1)
```

Preparación dos textos a localizar

En Glade obriga a que no IDE de desenvolvemento xa se teña defido a liña como translatable

```
<property name="title" translatable="yes">Rename files</property>
<property name="label" translatable="yes">Rename</property>
<property name="label" translatable="yes">_Rename</property>
<property name="label" translatable="yes">_Help</property>
<property name="tooltip" translatable="yes">Rename selected ...
```

Obtención dos ficheiros de l10n

Se ainda non o fixemos, é o momento de crear a árbore de cartafoles

mkdir -p locale/es/LC_MESSAGES
mkdir -p locale/gl/LC_MESSAGES

etc..

Obtención dos ficheiros de l10n

Bash Scripting e Python

Xerar o modelo ou ficheiro .pot

```
xgettext script.sh -o locale/script.pot --from-code=utf-8
```

Obtención dos ficheiros de l10n

```
xgettext                  → aplicativo ou orde
script.sh                 → ficheiro do que extraer o modelo
-o                        → opción de saída  - output
locale/script.pot
     locale/              → directorio onde irá o modelo
     script.pot           → nome do modelo
--from-code=utf-8         → definir (forzar) que empregue utf-8
```

Obtención dos ficheiros de l10n

Bash Scripting e Python

Xeramos o ficheiro de tradución .po

```
msginit -o locale/gl.po -i locale/script.pot
msginit -o locale/es.po -i locale/script.pot
```

Obtención dos ficheiros de l10n

```
msginit                    → aplicativo ou orde
-o                         → opción de saída  - output
locale/gl.po
    gl.po                  → ficheiro de idioma a xerar
-i                         → opción de entrada  - intput
locale/script.pot          → ficheiro do que extraer o modelo
```

Obtención dos ficheiros de l10n

Bash Scripting e Python

Xerar o ficheiro de tradución .po

Xa que estamos no mesmo directorio, podemos empregar a orde sinxela

```
msginit -i script.pot -l gl
```

e/ou

```
msginit -i script.pot -l es
```

Obtención dos ficheiros de l10n

```
msginit -i script.pot -l gl
msginit -i script.pot -l es
```

Nestas dúas liñas fixemonos na opción -l

```
-l gl
-l es
```

estamos a indicar que locale empregar

Obtención dos ficheiros de l10n

```
-l gl
-l es
```

Esta é unha opción imprescindible cando o SO no que estamos a traballar emprega un «locale» diferente ao «locale» que queremos obter na saída da orde `msginit`

Obtención dos ficheiros de l10n

Supoñamos que o noso SO emprega como idioma (locale) o gl

executamos

```
msginit -o es.po -i ficheiro.pot
```

obteremos un ficheiro chamado es.po, pero a súa estructura interna será de gl.

O máis correcto é que executemos:

```
msginit -l es -o es.po -i ficheiro.pot
```

Obtención dos ficheiros de l10n

A xeito de resumo

A orde `xgettext` obten o ficheiro modelo .pot

A orde `msginit` xera o ficheiro de tradución .po

Obtención dos ficheiros de l10n

A xeito de resumo

En ambos casos:
Indicamos o ficheiro orixe ou de entrada -i
Indicamos o ficheiro resultante ou de saída -o

Obtención dos ficheiros de l10n

Xerando os .po dun idioma (locale) distinto ao do noso SO

Indicaremos o idioma destino (ou de saída) coa opción `-l codigo_idioma`

```
msginit -l gl -o gl.po -i ficheiro.pot
```

Obtención dos ficheiros de l10n

Glade

Extraer cadeas dun ficheiro .glade

Precisamos dispor da ferramenta «intltool»

```
apt-get install intltool
```

Extraemos con:

```
intltool-extract --type="gettext/glade" xxx.glade
```

Obtención dos ficheiros de l10n

Glade

```
intltool-extract --type="gettext/glade" xxx.glade
```

crea un ficheiro chamado xxx.glade.h

Obtención dos ficheiros de l10n

Python + Glade

Agora teremos no cartafol das fontes (src), entre outros:

ficheiro1.py
ficheiro(n).py
meuprograma.glade.h

Obtención dos ficheiros de l10n

Python + Glade

ficheiro1.py
ficheiro(n).py
meuprograma.glade.h

Extraemos o modelo con:

```
xgettext -k_ -kN_ -o mensaxes.pot *.py *.h
```

Obtención dos ficheiros de l10n

Python + Glade

```
xgettext -k_ -kN_ -o mensaxes.pot *.py *.h
```

Con isto obteremos un ficheiro mensajes.pot

Obtención dos ficheiros de l10n

```
xgettext -k_ -kN_ -o mensaxes.pot *.py *.h
```

`-k_` → extraer cadeas marcadas (mark) como _

`-kN_` → extraer cadeas marcadas (mark) como N_

`*.py` → * comodín «todos» os .py (executables Python)

`*.h` → * comodín «todos» os .h (cabeceiras - header)

Obtención dos ficheiros de l10n

Python + Glade

Xerar o ficheiro de tradución .po

Xa que estamos no mesmo directorio, podemos empregar a orde sinxela

```
msginit -i mensaxes.pot -l gl
```

e/ou

```
msginit -i mensaxes.pot -l es
```

Obtención dos ficheiros de l10n

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR Free Software Foundation, Inc.
# FIRST AUTHOR , YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"POT-Creation-Date: 2002-02-28 21:52+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME \n"
"Language-Team: LANGUAGE \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: adivinar.c:21
msgid "winner"
msgstr ""
```

Obtención dos ficheiros de l10n

```
#: adivinar.c:21
msgid "winner"
msgstr ""

#: adivinar.c:22
msgid "loser"
msgstr ""

#: adivinar.c:41
#, c-format
msgid "%d - Enter a number (0-20) : "
msgstr ""

#: adivinar.c:44
msgid "You must enter a value lesser than 21 and greater than -1\n"
msgstr ""
```

Obtención dos ficheiros de l10n

Este é o momento de facer a tradución, a partires do ficheiro de tradución .po

Para isto temos dúas posibilidades:

Empregar ferramentas IGU como

- Poedit
- GTranslator
- Lokalize
- etc...

que xa nos xeran o correspondente ficheiro .mo

Traducir directamente en modo texto cun editor.

Obtención dos ficheiros de l10n

```
# Galician translations for FONTpage package.
# Traducións ao galego para o paquete FONTpage.
# Copyright (C) 2009 THE FONTpage_src'S COPYRIGHT HOLDER
# This file is distributed under the same license as the FONTpage package.
# Miguel Anxo Bouzada <mbouzada@gmail.com>, 2009.
#
msgid ""
msgstr ""
"Project-Id-Version: FONTpage 2.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2009-11-12 22:32+0100\n"
"PO-Revision-Date: 2009-11-13 12:03+0100\n"
"Last-Translator: Miguel Anxo Bouzada <mbouzada@gmail.com>\n"
"Language-Team: Galician <proxecto@trasno.net>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n != 1);\n"
"X-Poedit-Language: Galician\n"

msgid "pyGTK version 2.6 or greater is required for this application"
msgstr "Este aplicativo precisa pyGTK versión 2.6 ou superior"

msgid "Cannot load Pango"
msgstr "Non se puido cargar Pango"

msgid "You need FontConfig to run FONTpage"
msgstr "Para executar FONTpage precisase FontConfig"

msgid "Information"
msgstr "Información"

msgid "Error"
msgstr "Erro"

msgid "Sample Text"
msgstr "Texto de exemplo"

msgid "Enter Text Here"
msgstr "Introduza texto aquí"
```

Obtención dos ficheiros de l10n

```
# Galician translations for FONTpage package.
# Traducións ao galego para o paquete FONTpage.
# Copyright (C) 2009 THE FONTpage_src'S COPYRIGHT HOLDER
# This file is distributed under the same license as the FONTpage package.
# Miguel Anxo Bouzada <mbouzada@gmail.com>, 2009.
#
msgid ""
msgstr ""
```

Obtención dos ficheiros de l10n

```
msgid ""
msgstr ""
"Project-Id-Version: FONTpage 2.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2009-11-12 22:32+0100\n"
"PO-Revision-Date: 2009-11-13 12:03+0100\n"
"Last-Translator: Miguel Anxo Bouzada <mbouzada@gmail.com>\n"
"Language-Team: Galician <proxecto@trasno.net>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n != 1);\n"
"X-Poedit-Language: Galician\n"
```

Obtención dos ficheiros de l10n

msgid "pyGTK version 2.6 or greater is required for this application"
msgstr "Este aplicativo precisa pyGTK versión 2.6 ou superior"

msgid "Cannot load Pango"
msgstr "Non se puido cargar Pango"

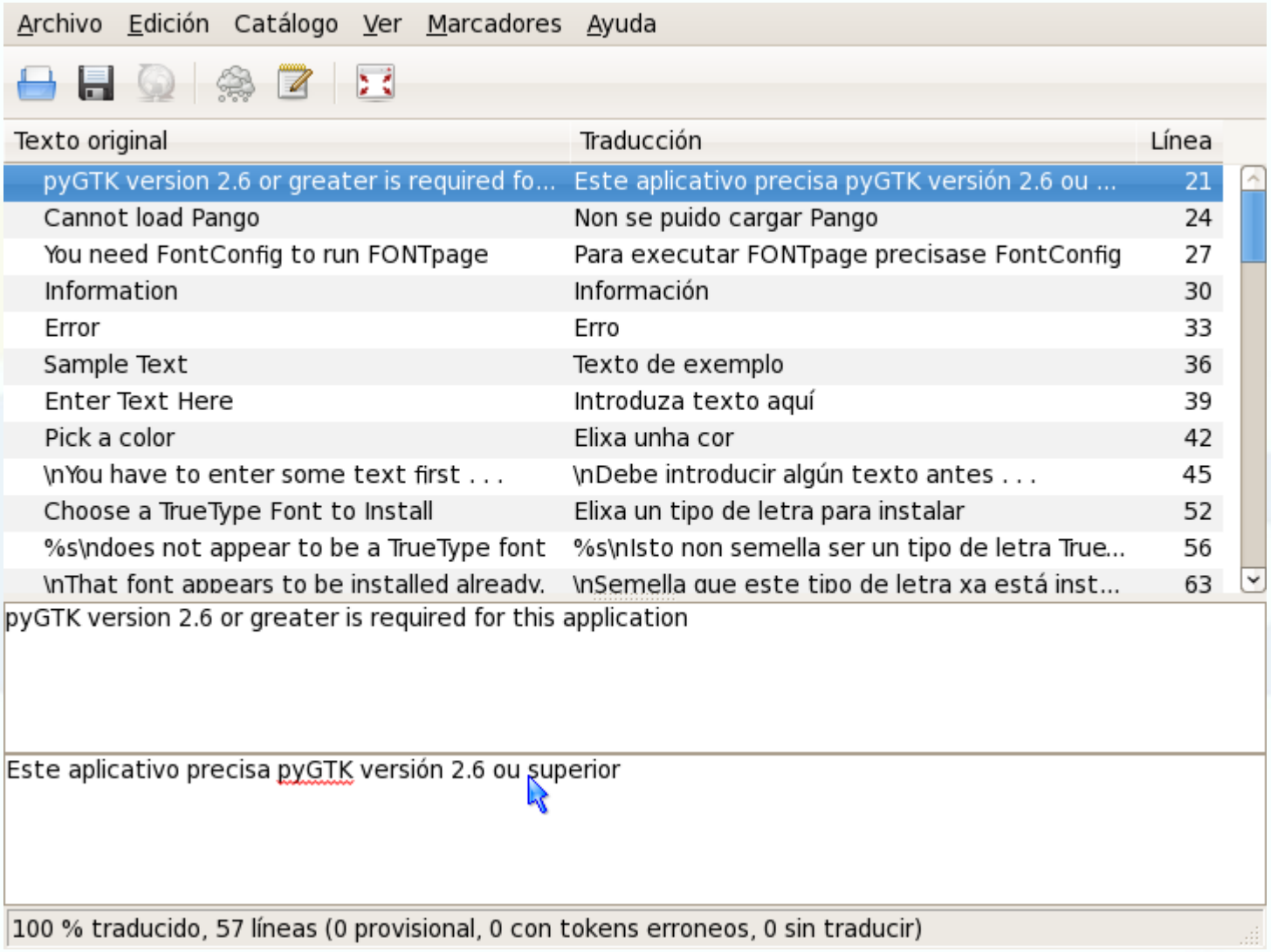
msgid "You need FontConfig to run FONTpage"
msgstr "Para executar FONTpage precisase FontConfig"

msgid "Information"
msgstr "Información"

msgid "Error"
msgstr "Erro"

msgid "Sample Text"
msgstr "Texto de exemplo"

Obtención dos ficheiros de l10n



trasno

Obtención dos binarios de tradución

Xa temos o ficheiro .po traducido.

Compre agora construir (xerar) o binario de tradución .mo

Obtención dos binarios de tradución

Xerar o binario de tradución .mo

```
msgfmt -o locale/gl/LC_MESSAGES/script.mo locale/gl.po
```

e/ou

```
msgfmt -o locale/es/LC_MESSAGES/script.mo locale/es.po
```

Obtención dos binarios de tradución

Xerar o binario de tradución .mo

```
msgfmt -c -v -o po/nome_executable.mo po/gl.po
```

e/ou

```
msgfmt -c -v -o po/nome_executable.mo po/es.po
```

Obtención dos binarios de tradución

Ímos ver a diferencia entre as dúas ordes.

```
msgfmt -o locale/gl/LC_MESSAGES/script.mo locale/gl.po
```

```
msgfmt -c -v -o po/nome_executable.mo po/es.po
```

Obtención dos binarios de tradución

```
msgfmt -o locale/gl/LC_MESSAGES/script.mo locale/gl.po
msgfmt -c -v -o po/nome_executable.mo po/es.po
```

Na primeira liña de ordes vemos só a opción **-o** que como xa vimos antes é a opción de saída (output)

Na segunda liña temos ademáis outras dúas opcións:

-c → Comproba todo o proceso (check)

-v → Fornece unha saída detallada de todo o proceso de xeración (verbose)

Obtención dos binarios de tradución

En “C” é recomendable empregar:

```
msgfmt -c -v -o po/nome_executable.mo po/es.po
```