# GETTEXT Howto

이승윤
INZEN (http://www.inzen.com)

sylee (at) inzen.com

$Date $


**gettext** 소개와 간단한 사용법



고친 과정

고침 2.02002년 11월 5일고친이 sylee
버전 2.0

**$Revision : $**

## 1. gettext 소개

사용자의 locale 에 따라 특정 message output 에 대한 다국어 처리를 할 수 있는 gettext 에 대해 살펴본다.

이를 위해 먼저 메세지가 처리될 프로그램을 작성하는 법을 shell script, C code 에서 살펴보고, 그 후에 변환될 메세지를 작성하는 방법을 살펴본다.

## 2. shell script 에서의 gettext 사용

shell 에서의 사용법을 살펴본다.

예제 프로그램으로 Hello 라는 shell script 를 작성한다. 단순히 Hello 와 World! 라고 출력하게 되어있다. 이를 locale 에 따라 출력하기 위해 gettext 로 둘러싸서 echo 시키고 있다. shell script 에서 특별히 다른 부분은 소스첫 부분에 TEXTDOMAINDIR 과 TEXTDOMAIN 을 전체 환경 변수로 export 시킨다는 것이다.

```
#! /bin/sh
# Hello

TEXTDOMAINDIR=/usr/local/share/locale
TEXTDOMAIN=Hello
export TEXTDOMAINDIR
export TEXTDOMAIN

echo "$(gettext -s "Hello")"
echo "$(gettext -s "World!")"
```

# 3. C code 에서의 gettext 사용

C code 에서의 사용법을 살펴본다.

예제 프로그램으로 Hello.c 라는 C code 를 작성한다. 이 프로그램은 Greeting, Hello, World 라고 출력을 하는데 각각의 message 들을 gettext 를 이용하여 사용자 locale 에 따라 출력토록 한다.

```
#include <libintl.h>
#include <locale.h>
#include <stdio.h>

int
main(void)
{
/* 현재 호스트의 locale 을 사용한다. */
        setlocale(LC_ALL,"");

/* Hello 의 message table 을 /usr/local/share/locale 아래에서 찾도록 한다. */
        bindtextdomain("Hello", "/usr/local/share/locale");
        textdomain("Hello");

        printf("origin message : %s\n", "Greeting");
        printf("gettext trans  : %s\n", gettext("Greeting"));

        printf("origin message : %s\n", "Hello");
        printf("gettext trans  : %s\n", gettext("Hello"));

        printf("origin message : %s\n", "World!");
        printf("gettext trans  : %s\n", gettext("World!"));

        return 0;
}
```

string 을 gettext 로 둘러싸게 되면 둘러싸인 message 가 해당 locale에 해당하는 message table 을 참조하여 출력하게 된다. 여기서 참조할 message table 을 프로그램에 연결시키기 위해서

bindtextdomain 을 사용하였다. 위 예제에서는 위 코드를 Hello 라는 이름으로 빌드하고, 이의 message table 은 /usr/local/share/locale 아래에서 textdomain 에서 지정한 Hello 프로그램의 message table 을 참조하도록 한것이다.

만약 해당하는 mesage table 이 없거나 정의된 값이 없을 경우에는 변환없이 원래 messgae가 출력된다.

# 4. gettext message table 작성법

gettext 를 통해 변활될 message 들을 담는 message table 을 작성하는 요령을 살펴본다.

gettext는 정해진 message 를 미리 정해진 message 로 바꾸어 주는 역할을 할뿐이다. 따라서 원본 message 와 이것이 어떤 message 로 변활될지를 나타내어 주는 table 이 필요하다. 이의 초안이 되는것이 *.po 에 해당하는 파일이다.

이 초안을 직접 작성할수도 있지만 xgettext 라는 프로그램이 그 초안을 만들어 준다.

```
# shell script
xgettext Hello
# C code
xgettext Hello.c
```

xgettext 를 사용하고 나면 messages.po 라는 파일이 생성된다. 이는 GNU version 의 gettext 인지 아닌지에 따라 약간 다르지만 뼈대는 비슷하다. msgid 와 msgstr 의 짝으로 이루어져 있으며, msgid 가 위 예제 프로그램 속에서 gettext 로 둘러싸인 message 들이다. 그리고 msgstr 이 변활될 메세지를 담게 된다. xgettext 를 하고 나면 msgid 들이 모두 추출되며, 사용자는 단지 msgstr 부분의 빈곳만 채워주면 된다. 다음과 같이 채워보자.

앞절의 프로그램들에서 Hello 와 World! 를 gettext 가 변환하기 위해서는 아래처럼 작성하면 된다.

```
msgid "Hello"
msgstr "헬로"

msgid "World!"
msgstr "월드!"
```

이제 이렇게 작성된 파일을 msgfmt 라는 프로그램을 이용하여 gettext 가 이해할수 있는 파일로 변환한다.

```
msgfmt -o Hello.mo messages.po
```

Hello.mo 라 한 이유는 프로그램의 이름과 같이 하기 위해서이다. 이제 이를 /usr/local/share/locale/ko/LC_MESSAGE/ 아래에 넣어두면 된다. 한국어로 locale 이 설정되어 있다면 ...locale/ko/ 아래를 참조하게 된다. 만약 프랑스어로 message translation 을 하고자 할 경우에는 해당 .po 파일의 msgstr 을 불어로 작성한다. 그리고 이를 msgfmt 로 Hello.mo 로 만

든후 /usr/local/share/locale/fr/LC_MESSAGE/ 아래에 넣어두면 된다. 시스템 로케일에 따라 영어, 한국어, 혹은 불어 message 가 출력된다.

# 5. Non GNU vs GNU

GNU version 의 gettext 인지 아닌지에 따라 차이점이 있다고 언급했다. 실제로는 이 차이때문에 message 변환이 안되기도 한다.

그 차이점은 messages.po 파일의 format 에 있다. non GNU version 의 gettext 를 사용하게 될때면 앞절에서 작성한 것처럼 msgid, msgstr 의 쌍으로도 충분하다. 그렇지만 GNU version 의 gettext 에서는 위처럼 해서는 원하는 결과를 얻을수없다.

xgettext 로 만들어진 messages.po 를 보자

non GNU messages.po

```
domain "messages"
# File:Hello.c, line:22, textdomain("Hello");
msgid  "Greeting"
msgstr
msgid  "Hello"
msgstr
msgid  "World!"
msgstr
```

GNU messages.po

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"POT-Creation-Date: 2002-11-13 15:09+0900\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: Hello.c:26
msgid "Greeting"
msgstr ""
```

```
#: Hello.c:29
msgid "Hello"
msgstr ""

#: Hello.c:32
msgid "Wolrd!"
msgstr ""
```

GNU messages.po 에서는 사용자가 수정해주어야 할 사항들이 몇가지 더 있다. 기본 정보 이외에 CHARSET 을 지정해주어야 한다. 한국어 일 경우는 euc-KR 로 지정해 주어야 안전하다.

GNU 이냐 아니냐에 따라 작성법은 그다지 다르지 않다. 중요한것은 빌드 방법이다. GNU gettext 를 사용한다면 xgettext, msgfmt 도 GNU version 을 사용하여야 한다. non GNU gettext 를 사용한다면 역시 xgettext, msgfmt 도 gettext 와 같은 것을 사용하여야 한다. 그리고 C code 일 경우에는 빌드 할때 링크될 라이브러리를 직접 명시해 주는것이 안전하다. (GNU 일때는 GNU 라이브러리, non GNU 라이브러리 일때는 해당 라이브러리)

```
# example
shell> gcc Hello.c -o Hello -lintl
```

# 6. 실행결과

이제 Hello 라는 프로그램이 실행될때 실행되고 있는 환경의 locale에 따라 xgettext, msgfmt 에 의해 만들어진, Hello.mo 를 참조할것인지 혹은 그냥 메세지를 출력할것인즐 결정하게 된다. 설정했던 message 가 출력되게 된다. C program 의 실행 예이다.

```
origin message : Greeting
gettext trans  : Greeting
origin message : Hello
gettext trans  : 헬로
origin message : World!
gettext trans  : 월드!
```

Greeting 은 message table 에 존재하지 않으므로 그냥 Greeting 이 찍히게 된다. 나머지 message 들은 message table 에 정의된 msgstr 값을 참조하여 출력하고 있다.

# 7. Miscellaneous

- 환경변수중 LC_MESSAGES 가 C 로 정의되어있다면 메세지 전환이 이루어 지지 않는다. 이를 변환하고자 하는 locale 로 바꾸어 주는 것이 좋다. setlocale(LC_MESSAGES, "ko_KR");
- non GNU 일때의 *.po 는 msgid, msgstr 의 쌍으로만 직접 작성하여도 된다.

- GNU gettext 를 사용할때는 흔히 gettext(String) 을 _(String) 으로 define 해서 사용한다. 허나 이렇게 하면 xgettext 에서 msgid 를 찾지는 못한다.