آموزش
نکات
برنامه‌نویسی
سیمبین

series60.blogfa.com

مقاله شماره ۱: نحوه طراحی برنامه‌های چندزبانه

از سری مقاله‌های «آموزش نکات برنامه‌نویسی سیمبین»

ویرایش ۱ – ۵ خرداد ۱۳۸۶

سیستم عامل سیمبین

موسی مرادی

mousamk@gmail.com

ایجاد برنامه‌های چندزبانه برای سیستم عامل سیمبین

ایجاد یک برنامه چند زبانه در سیمبین نسبتاً راحت است و این مقاله به شما نشان می‌دهد که چگونه این کار را انجام دهید. شما با مقداری تلاش می توانید برنامه‌ای داشته باشید که هم به انگلیسی باشد و هم این که زبان محلی شما را هم پشتیبانی کند. برای افزودن زبان‌های دیگر می‌توانید از دوستان اینترنتی‌تان که با زبان‌های مختلف آشنایی دارند، استفاده کنید.

مقدمه:

در این مثال فرض کنید که می‌خواهیم برنامه‌ای بنویسیم که سه زبان انگلیسی (English)، فرانسه (French) و آلمانی (German) را پشتیبانی می‌کند. و نیز فرض کنید که اسم برنامه‌مان MyApp است.

این برنامه بدون هیچ مشکلی باید در تمام محیط‌های گرافیکی کار کند. در اینجا ما برنامه را برای رابط UIQ می‌نویسیم ولی این موضوع مستقیماً به بحث ما مربوط نمی‌شود. شما می‌توانید فایل pkg را تغییر دهید و آن را برای رابط‌های دیگری از جمله سری ۶۰ هم کامپایل کنید.

در این مقاله شما را با کارهایی که باید برای تبدیل یک برنامه معمولی به یک برنامه چند زبانه انجام دهید آشنا می کنیم. برای انجام این کار شش مرحله در پیش داریم:

* جمع آوری تمام رشته ها و قرار دادن آنها در فایل های خاص
* اِعمال تغییرات لازم در منابع
* اِعمال تغییرات لازم در کد برنامه (تا بتواند با متغیرهای رشته ها کار کند)
* اعمال تغییرات لازم در فایل MMP (تا به کامپایلر بگوییم که کدام زبان ها پشتیبانی شده اند)
* مشخص کردن نام برنامه در هر زبان
* کامپایل کردن و ساخت برنامه

مرحله ۱) جمع آوری تمام رشته ها و قرار دادن آنها در فایل های خاص

در مرحله اول باید سه فایل با نام های MyApp.l01 و MyApp.l02 و MyApp.l03 ایجاد کنیم که به ترتیب حاوی زبان های انگلیسی، فرانسه و آلمانی هستند. کدهای 01 و 02 و 03 جزو قراردادهای سیمبین هستند که هر کدام یک زبان خاص را مشخص می کنند. کدهای زبان های مختلف را می توانید در پایان مقاله ببینید. همه آنها در فایل e32std.h تعریف شده اند.

در داخل سه فایلی که ایجاد کرده اید باید کدهای زیر را بنویسید:

فایل اول:

```
// MyApp.l01
// English localisation
#define ELanguage   ELangEnglish

// Localised strings
#define STRING_HOWAREYOU      "How are you ?"
#define STRING_APPLES         "You have %d apples."
#define STRING_EXIT           "Exit"
```

فایل دوم:

```
// MyApp.l02
// French localisation
#define ELanguage   ElangFrench

// Localised strings
#define STRING_HOWAREYOU      "Comment allez vous ?"
#define STRING_APPLES         "Vous avez %d pommes."
#define STRING_EXIT           "Sortir"
```

فایل سوم:

```
// MyApp.l3
// German localisation
#define ELanguage   ElangGerman

// Localised strings
#define STRING_HOWAREYOU      "Wie gehen Sie ?"
#define STRING_APPLES         "Sie haben %d Apfels."
#define STRING_EXIT           "Schliessen"
```

سپس یک فایل با نام MyApp.loc ایجاد کنید تا اطلاعات مربوط به سه فایل بالا را در آن قرار دهیم:

```
// MyApp.loc
// Localisation for MyApp

#ifdef LANGUAGE_01
#include "MyApp.l01"
#endif

#ifdef LANGUAGE_02
#include "MyApp.l02"
#endif

#ifdef LANGUAGE_03
#include "MyApp.l03"
#endif
```

اهمیت این فایل loc آن است که بتوانیم فایل زبانی مناسب را در فایل rss داخل کنیم؛ بنابراین استفاده کردن و تعریف این فایل فقط یک قرارداد است.

من معمولاً این فایل ها را همراه با فایل rss در یک پوشه به نام rsc قرار می دهم. البته این کار الزامی نیست. علت این کار هم این است که بعداً آنها را به راحتی به فایل mmp ضمیمه کنیم.

مرحله ۲) استفاده از رشته ها در منوها و پنجره های محاوره

اکنون ما رشته‌ها را در اختیار داریم. حالا باید آنها را به برنامه وارد کنیم. ابتدا باید فایل loc را به فایل تعاریف منابع (rss) ضمیمه کنیم.

```
// MyApp.rss
NAME HELL

#include <eikon.rh>
#include "MyApp.hrh"
#include "MyApp.loc"  // <<<< Here it is

RESOURCE RSS_SIGNATURE { }

RESOURCE TBUF { buf=""; }
```

...

حالا باید رشته‌ها موجود در کد را به نام در نظر گرفته شده برای رشته‌ها تغییر دهیم. مثلاً کد

```
//***************************************
RESOURCE MENU_PANE r_myapp_menu
//***************************************
{
 items=
 {
   MENU_ITEM { command=EMyAppHowAreYou;  txt="How are you?"; },
   MENU_ITEM { command=EEikCmdExit;      txt="Exit"; }
 };
}
```

باید به صورت

```
//***************************************
RESOURCE MENU_PANE r_myapp_menu
//***************************************
{
 items=
 {
   MENU_ITEM { command=EMyAppHowAreYou;  txt=STRING_HOWAREYOU; },
   MENU_ITEM { command=EEikCmdExit;      txt=STRING_EXIT; }
 };
}
```

تغییر کند.

اگر فقط می‌خواهید منوها را چندزبانه کنید، انجام کارهای بالا کافی است ولی اگر می‌خواهید رشته‌های موجود در کدها را هم چندزبانه کنید، باید برای هر کدام، یک بافر هم تعریف کنید. برای انجام این کار فقط کافی است که از برای هر رشته از دستور TBUF در انتهای فایل rss استفاده کنید. مثلاً به صورت زیر:

```
/***************************************
// STRINGS
//***************************************
RESOURCE TBUF r_string_apple  { buf=STRING_APPLE;  }
RESOURCE TBUF r_string_thanks { buf=STRING_THANKS; }
```

مرحله ۳) استفاده از متن‌ها در کد

شما هم‌اکنون می‌توانید خیلی از تعاریف LIT_ را از کدهایتان حذف کنید. مثلاً به سطر زیر، نیازی نداریم:

```
myDes.Format(_L("You have %d apples"),num);
```

به جای عبارت بالا، می‌توانید از یک format string که از فایل resource می‌خوانید، استفاده کنید. البته این کار مقداری پیچیده است ولی تابع **CCoeEnv::AllocReadResourceLC()** مقدار زیادی از کار را برای ما انجام می‌دهد:

```
HBufC* formatBuf=CCoeEnv::Static()->AllocReadResourceLC(R_STRING_APPLE);
myDes.Format(formatBuf,num);
CleanupStack::PopAndDestroy(formatBuf);
```

این کد یک واصف HBufC اختصاص می‌دهد و R_STRING_APPLE را در آن کپی می‌کند. این واصف به عنوان یک format template به کار می‌رود که می‌تواند یک عدد صحیح را داشته باشد. بسته به تنظیمات زبانی گوشی شما

(و همچنین مقدار num!) واصف myDes می‌تواند مقادیر "*You have 3 apples.*"، "*Vous avez 3 pommes.*" یا "*Sie haben 3 Apfels*" را داشته باشد.

مرحله ۴) آپدیت کردن فایل mmp

تا الان تغییرات لازم را در کدهایتان انجام داده‌اید. فقط یک آپدیت کوچک در فایل mmp لازم است تا به کامپایلر resource بفهمانیم که از کدام زبان استفاده کند. یک دستور LANG به فایل اضافه کنید و جلوی آن شماره زبان‌هایی که برنامه‌تان پشتیبانی می‌کند را وارد کنید.

```
TARGET        MyApp.app
TARGETTYPE    app
UID           0x100039CE 0x0EB00551

TARGETPATH    \system\apps\Myapp

// Language in which the application is translated
// (each supported language shall provide a MyApp.lxx file where
// xx is the code number of the language:
// 01 = english, 02 = french, 03 = german, etc... )
LANG        01 02 03

SOURCEPATH  ..\src
SOURCE      MyAppMain.cpp
SOURCE      MyAppDocument.cpp
SOURCE      MyAppUi.cpp
SOURCE      MyAppView.cpp
SOURCE      MyAppModel.cpp

SOURCEPATH  ..\rsc
RESOURCE    MyApp.rss

USERINCLUDE ..\inc
USERINCLUDE ..\rsc
...
```

البته فراموش نکنید که فایل resourceتان را هم include کنید. (توسط دستور RESOURCE) همچنین مسیر فایل‌های متن‌های برنامه در زبان‌های پشتیبانی شده را هم مشخص کنید. (توسط دومین دستور USERINCLUDE)

مرحله ۵) تنظیم نام برنامه در زبان‌های مختلف

نام برنامه به طور پیش‌فرض همان نام فایل app. است که نمی‌توان آن را به زبان‌های مختلف نوشت. البته با ایجاد کردن یک فایل aif. می‌توان نام برنامه در زبان‌های مختلف را مشخص کنید. ابتدا فایل تعریف را ایجاد و یا آپدیت کنید به صورت زیر:

```
//MyAppAif.rss
#include <aiftool.rh>

RESOURCE AIF_DATA
{
 app_uid=0x0EB00551;
 num_icons=2;
 embeddability=KAppNotEmbeddable;
 newfile=KAppDoesNotSupportNewFile;

 caption_list=
 {
   CAPTION { code=ELangEnglish; caption="MyApp";  },
   CAPTION { code=ELangFrench;  caption="MonApp"; },
   CAPTION { code=ELangGerman;  caption="MeinApp";}
 };
}
```

عبارت caption_list به هر زبان، یک نام را اختصاص می‌دهد. حالا دستور مربوطه را بـه آخـر فایـل mmpتـان اضـافه کنید (در صورت موجود نبودن):

```
AIF MyApp.aif . ..\aif\MyAppaif.rss
               c12 ..\aif\newlc42x35.bmp ..\aif\newlc42x35m.bmp
               ..\aif\newlc42x29.bmp ..\aif\newlc42x29m.bmp
```

این دستور، همچنین لوگوی برنامه را (که در Menu نمایش داده می‌شود) هم مشخص می‌کند.

مرحله ۶) فایل pkg را آپدیت کنید

هنوز هم دارید می‌خونید؟ خوبه! الان شما به پایان کار نزدیک هستید. الان وقت آن است که از شما یک سـؤال در مـورد نحوه نصب فایل‌های زبان‌های پشتیبانی شده در موبایل بپرسم. آیا شما می‌خواهید:

*) همه فایل‌ها را بی‌سروصدا روی موبایل کپی کنید. این کار در برنامه‌های با حجم زیـاد، جـای بسـیار زیـادی را اشـغال می‌کند؛ ولی دیگر لازم نیست از کاربر سؤالی بپرسید.
*) از کاربر بپرسید که کدام زبان را می‌خواهد نصب کند؟ این روش فضای کمتری روی موبایل اشغال می‌کنـد، ولـی یـه مرحله به مراحل نصب برنامه، اضافه می‌کند.

ابتدا مورد اول را توضیح می‌دهیم. به خاطر این که نمی‌خواهیم کاربر را با نشان دادن یک پنجره popup اذیـت کنـیم، بنابراین برای هر زبانی که پشتیبانی می‌کنیم، یک فایل resource نصب می‌کنیم و انتخاب زبـان رو بـه عهـده سیسـتم عامل می‌گذاریم. به صورت پیش‌فرض این فایل‌های resource دارای پسوند rXX. هستند که XX کد زبان مورد نظـر است و این فایل‌ها در مسیر زیر قرار می‌گیرند:

```
\epoc32\data\z\system\apps\MyApp\
```

البته باید همه آنها را در فایل pkg. لیست کنیم و در مسیر برنامه نصب کنیم.

```
;
; Installation file for MyApp example application
; "Silent" version: no language popup
;
; UID is the app's UID
;
#{"MyApp"},(0x0EB00551),1,0,0

; Target is UIQ 2.0 devices.
(0x101F617B), 2, 0, 0, {"UIQ20ProductID"}

;
; Main App
;
"\epoc32\release\armi\urel\MyApp.app"   -  "!:\system\apps\MyApp\MyApp.app"
"\epoc32\data\z\system\apps\MyApp\MyApp.r01"
-  "!:\system\apps\MyApp\MyApp.r01"
"\epoc32\data\z\system\apps\MyApp\MyApp.r02"
-  "!:\system\apps\MyApp\MyApp.r02"
"\epoc32\data\z\system\apps\MyApp\MyApp.r03"
-  "!:\system\apps\MyApp\MyApp.r03"
"\epoc32\data\z\system\apps\MyApp\MyApp.aif"   -"!:\system\apps\MyApp\MyApp.aif"
```

شما باید توجه داشته باشید که با وجود این که ما یک برنامه چندزبانه مـی‌نویسـیم، ولـی هـیچ زبـانی را در فایـل pkg. مشخص نمی‌کنیم. به همین خاطر است که از کاربر چیزی موقع نصب برنامه پرسیده نمی‌شود. و موضوع جالب این است

که برنامه شما برای یک کاربر انگلیسی‌زبان، به صورت انگلیسی، برای یک فرانسوی به زبان فرانسه و برای یک آلمانی، به زبان آلمانی، طبق تنظیمات محلی گوشی مورد نظر، ظاهر خواهد شد. (که به صورت پیش‌فرض انگلیسی است.) و اگر شما زبان گوشی‌تان را عوض کنید، زبان برنامه شما هم به صورت اتوماتیک عوض خواهد شد. یکی از مشکلات این روش این است که شما نمی‌توانید کاری کنید که نام برنامه‌تان در Installer در زبان‌های مختلف نشان داده شود!

روش دیگر این است که یک پنجره popup برای انتخاب زبان توسط کاربر، موقع نصب برنامه، نشان دهیم و فقط زبان انتخاب شده توسط کاربر را نصب کنیم. در این روش نسبت به روش قبل لازم نیست تغییر در کد برنامه انجام دهید. فقط باید چند خط را در فایل pkg. تغییر دهید:

```
;
; Installation file for MyApp example application
; Standard version: language popup
;
;Languages
&EN,FR,GE

; UID is the app's UID
;
#{"MyApp","MonApp","MeinApp"},(0x0EB00551),1,0,0

; Target is UIQ 2.0 devices.
(0x101F617B), 2, 0, 0, {"UIQ20ProductID", "UIQ20ProductID", "UIQ20ProductID"}

;
; Main App
;
"\epoc32\release\armi\urel\MyApp.app"-"!:\system\apps\MyApp\MyApp.app"
{
"\epoc32\data\z\system\apps\MyApp\MyApp.r01",
"\epoc32\data\z\system\apps\MyApp\MyApp.r02",
"\epoc32\data\z\system\apps\MyApp\MyApp.r03"
}
-  "!:\system\apps\MyApp\MyApp.rsc"
"\epoc32\data\z\system\apps\MyApp\MyApp.aif"   -"!:\system\apps\MyApp\MyApp.aif"
```

این فایل pkg. فایل sisی تولید می‌کند که که موقع نصب شدن، کمی متفاوت‌تر از قبل عمل می‌کند.

* این برنامه در Installer همان نامی را خواهد داشت که برای زبان‌های مورد نظر نوشته‌اید. (در اینجا MyApp، MonApp یا MeinApp)
* یک پنجره popup ظاهر خواهد شد که از کاربر زبان مورد نظر را خواهد پرسید.
* فقط یکی از فایل‌های rXX. در گوشی نصب خواهد شد و نامش هم به rsc. تغییر خواهد یافت.
* بعد از نصب، برنامه فقط در این زبان نمایش داده خواهد شد. بدون توجه به تنظیمات زبانی گوشی.

لینک‌های مرتبط

اگر شما مایل به مطالب بیشتری در این مورد هستید، لینک‌های زیر برای شما مفید خواهند بود:

* برنامه نمونه Nokia language در پوشه series60Ex\Language در مسیری که SDK سری ۶۰ را نصب کرده‌اید.
* مقاله Localization.
* How to make a multilingual application installer از SavaaZ
* برنامه نمونه localisation در سایت برنامه‌نویسی Sendo. (ثبت‌نام لازم است.)

جدول زبان‌ها

Code	TLanguageEnum	Language
01	ELangEnglish	**UK English**
02	ELangFrench	**French**
03	ELangGerman	**German**
04	ELangSpanish	**Spanish**
05	ELangItalian	**Italian**
06	ELangSwedesh	**Swedesh**
07	ELangDanish	**Danish**
08	ELangNorwegian	**Norwegian**
09	ELangFinish	**Finnish**
10	ELangAmerican	**American**
11	ELangSwissFrench	**Swiss French**
12	ELangSwissGerman	**Swiss German**
13	ELangPortuguese	**Portuguese**
14	ELangTurkish	**Turkish**
15	ELangIcelandic	**Icelandic**
16	ELangRussian	**Russian**
17	ELangHungarian	**Hungarian**
18	ELangDutch	**Dutch**
19	ELangBelgianFlemish	**Belgian Flemish**
20	ELangAustralian	**Australian English**
21	ELangBelgianFrench	**Belgian French**
22	ELangAustrian	**Austrian German**
23	ELangNewZealand	**New Zealand English**
24	ELangInternationalFrench	**International French**
25	ELangCzech	**Czech**
26	ELangSlovak	**Slovak**
27	ELangPolish	**Polish**
28	ELangSlovenian	**Slovenian**
29	ELangTaiwanChinese	**Taiwanese Chinese**
30	ELangHongKongChinese	**Hong Kong Chinese**
31	ELangPrcChinese	**PRC Chinese**
32	ELangJapanese	**Japanese**
33	ELangThai	**Thai**
34	ELangAfrikaans	**Afrikaans**
35	ELangAlbanian	**Albanian**
36	ELangAmharic	**Amharic**
37	ELangArabic	**Arabic**
38	ELangArmenian	**Armenian**
39	ELangTagalog	**Tagalog**
40	ELangBelarussian	**Belarussian**
41	ELangBengali	**Bengali**
42	ELangBulgarian	**Bulgarian**
43	ELangBurmese	**Burmese**
88	ELangTelugu	**Telugu**
89	ELangTibetan	**Tibetan**
90	ELangTigrinya	**Tigrinya**
91	ELangCyprusTurkish	**Cyprus Turkish**
92	ELangTurkmen	**Turkmen**
93	ELangUkrainian	**Ukrainian**
94	ELangUrdu	**Urdu**
96	ELangVietnamese	**Vietnamese**
97	ELangWelsh	**Welsh**

Code	TlanguageEnum	Language
44	ELangatalan	**Catalan**
45	ELangCroation	**Croation**
46	ELangCanadianEnglish	**Canadian English**
47	ELangInternationalEnglish	**International English**
48	ELangSouthAfricanEnglish	**South African English**
49	ELangStonian	**Estonian**
50	ELangFarsi	**Farsi**
51	ELangCanadianFrench	**Canadian French**
52	ELangScotsGaelic	**Gaelic**
53	ELangGeorgian	**Georgian**
54	ELangGreek	**Greek**
55	ELangCyprusGreek	**Cyprus Greek**
56	ELangGujarati	**Gujarati**
57	ELangHebrew	**Hebrew**
58	ELangHindi	**Hindi**
59	ELangIndonesian	**Indonesian**
60	ELangIrish	**Irish**
61	ELangSwissItalian	**Swiss Italian**
62	ELangKannada	**Kannada**
63	ELangKazakh	**Kazakh**
64	ELangKhmer	**Kmer**
65	ELangKorean	**Korean**
66	ELangLao	**Lao**
67	ELangLatvian	**Latvian**
68	ELangLithuanian	**Lithuanian**
69	ELangMacedonian	**Macedonian**
70	ELangMalay	**Malay**
71	ELangMalayalam	**Malayalam**
72	ELangMarathi	**Marathi**
73	ELangMoldavian	**Moldovian**
74	ELangMongolian	**Mongolian**
75	ELangNorwegianNynorsk	**Norwegian Nynorsk**
76	ELangBrazilianPortuguese	**Brazilian Portuguese**
77	ELangPunjabi	**Punjabi**
78	ELangRomanian	**Romanian**
79	ELangSerbian	**Serbian**
80	ELangSinhalese	**Sinhalese**
81	ELangSomali	**Somali**
82	ELangInternationalSpanish	**International Spanish**
83	ELangLatinAmericanSpanish	**American Spanish**
84	ELangSwahili	**Swahili**
85	ELangFinlandSwedish	**Finland Swedish**
87	ELangTamil	**Tamil**

درباره این مقاله:

این مقاله اولین مقاله از سری مقاله‌های **آموزش نکات برنامه‌نویسی سیمبین** می‌باشد که در زمینه «نحوه طراحی برنامه‌های چندزبانه» نوشته شده است.

این مقاله، ترجمه مقاله انگلیسی Multilingual از سایت newlc است. که برای انتشار در این وبسایت، تهیه شده بود.

برای دریافت مقالات دیگر این سری می‌توانید به وبلاگ بنده و یا فروم مخصوص برنامه نویسی سیمبین مراجعه کنید.

مترجم: موسی مرادی کندلجی (mousamk@gmail.com)

در صورتی که هرگونه سؤال، پیشنهاد، انتقادی و یا مطلبی داشتید، می‌توانید از طریق ایمیل تماس بگیرید. ضمناً می‌توانید از طریق وبلاگ و یا فروم هم با من ارتباط داشته باشید.