

Localizing GTK+

Theppitak Karoonboonyanan
thep@linux.thai.net

January 2004

Abstract

This paper summarizes information gathered by the author during contributing to Pango and GTK+ projects and developing own Thai modules. It is by no means a complete reference nor a universal guide for all languages. The author just hopes it would be useful for other localizers.

GTK+ 2 is more than internationalized. It is indeed a multilingual system. Language supports are modularized for input methods, text drawing and language processing. This paper describes the framework in localizer's point of view.

Localizing GTK+

- GTK+ I18N Framework
- GTK+ Input Method Modules
- Pango Engines
 - Pango Language Engines
 - Pango Shaping Engines

Note

1 GTK+ I18N Framework

GTK+ I18N Framework

I18N or Multilingual?

- **Unicode:** multilingual by nature
- **GTK+:**
 - Unicode-based multilingual
 - Multilingualized by dynamic L10N modules

GTK+ I18N Frameworks

- **GTK+ IM:** input method, dynamically selectable
- **Pango:** quality text layout engine, modularized as per character Unicode range
 - Language Engines
 - Shaping Engines

Note

2 GTK+ Input Method Modules

GTK+ Input Method Modules

GTK+ IM: text input via pure GTK+ interfaces

Clients:

- **GtkIMMulticontext**
 - **GtkIMContext** derivative
 - IM selectable by menu
- **GtkIMContext**
 - `gtk_im_context_filter_keypress()`
 - * passes key event to IM
 - * returns TRUE → discard the event, IM has consumed it
 - * otherwise → process the key as usual

Note

GTK+ Input Method Modules

Clients:

- **GtkIMContext** (cont.)
 - **signals** (from IM):
 - "preedit_changed"
uncommit (preedit) string changed
 - "commit"
characters commit from IM
 - "retrieve_surrounding"
IM wants to read text around cursor
 - "delete_surrounding"
IM wants to delete text around cursor

Note

GTK+ Input Method Modules

IM Implementation:

- **IM module entry functions:**
 - `im_module_init()`
initializes module – usually register IM context type
 - `im_module_exit()`
module clean-up
 - `im_module_list()`
info of all IM's provided
 - `im_module_create()`
creates a **GtkIMContext** instance
- The customized **GtkIMContext**
 - `(*filter_keypress)()` virtual function
 - etc.

Note

GTK+ Input Method Modules

IM Implementation:

- `(*filter_keypress)()` virtual function
 - return TRUE to consume the event
 - return FALSE to pass the event back to client
- Preedit strings
 - IM emits "preedit_changed" signal when uncommitted text changes
 - client handler retrieves it with `gtk_im_context_get_preedit_string()`, displays it at cursor
- Committing characters
 - IM emits "commit" signal, with UTF-8 string
 - client handler puts it into input buffer

Note

GTK+ Input Method Modules

IM Implementation:

- Retrieving context
 - IM issues `gtk_im_context_get_surrounding()`
 - `gtk_im_context_get_surrounding()` emits "retrieve_surrounding" signal
 - client handler reads text buffer; replies with `gtk_im_context_set_surrounding()`
- Deleting context
 - IM issues `gtk_im_context_delete_surrounding()`
 - `gtk_im_context_delete_surrounding()` emits "delete_surrounding" signal
 - client handler deletes text in buffer

Note

3 Pango Engines

Pango Overview

PangoLayout

- High-level layout processing
- Paragraph properties:
 - indent
 - spacing
 - alignment
 - justification
 - wrapping modes
 - tabs
- Text elements:
 - get lines and their extents
 - get runs and their extents
 - character search at (x, y) position
 - character logical attributes (is line break, is cursor pos, etc.)
 - cursor movements
- Text contents:
 - plain text
 - markup text

Note

Pango Overview

Middle-level Processing

- `pango_itemize()`
 - breaks text into chunks (items) as per language
- `pango_break()`
 - determines word, character, cursor break positions
 - usually handled by Pango Language Engine
- `pango_shape()`
 - converts text into glyphs, with proper positioning
 - usually handled by Pango Shaping Engine

Note

Pango Engine Implementation

Notice: A lot of changes in Pango 1.3.x

Module Entries:

- `script_engine_init()`
initializes module – usually register engine type
- `script_engine_exit()`
module cleanup
- `script_engine_list()`
info of all engines provided (**PangoEngineInfo**)
- `script_engine_create()`
creates a **PangoEngine** instance for the given ID

Note

Pango Engine Implementation

PangoEngineInfo

```
struct _PangoEngineInfo
{
    gchar *id;
    gchar *engine_type;
    gchar *render_type;
    PangoEngineScriptInfo *scripts;
    gint n_scripts;
};
```

Note

Pango Language Engines

PangoEngineLang

- Function: determine possible break positions in text
- Virtual function:

```
void
(*script_break) (PangoEngineLang *engine,
                 const char *text,
                 int len,
                 PangoAnalysis *analysis,
                 PangoLogAttr *attrs,
                 int attrs_len);
```

→ fills the **PangoLogAttr** array with character properties (is-line-break, is-char-break, etc.)

Note

Pango Language Engines

PangoLogAttr

```
struct _PangoLogAttr
{
    guint is_line_break : 1;
    guint is_mandatory_break : 1;
    guint is_char_break : 1;
    guint is_white : 1;

    guint is_cursor_position : 1;

    guint is_word_start : 1;
    guint is_word_end : 1;

    guint is_sentence_boundary : 1;
    guint is_sentence_start : 1;
    guint is_sentence_end : 1;

    guint backspace_deletes_character : 1;
};
```

Note

Pango Shaping Engines

PangoEngineShape

- Function: convert characters into positioned glyphs of given font
- Virtual function:

```
void
(*script_shape) (PangoEngineShape *engine,
                 PangoFont          *font,
                 const char         *text,
                 int                 length,
                 PangoAnalysis      *analysis,
                 PangoGlyphString *glyphs);
```

→ sets the **PangoGlyphString** with the converted glyphs.

Note

Pango Shaping Engines

PangoGlyphString

```
struct _PangoGlyphString {
    gint num_glyphs;
    PangoGlyphInfo *glyphs;
    gint *log_clusters;

    /*< private >*/
    gint space;
};
```

- `glyphs`: string [`num_glyphs`] of **PangoGlyphInfo**
- `log_clusters`: array [`num_glyphs`] of index to char in text for each glyph (useful for RTL texts, for example)
- `pango_glyph_string_*()` functions for client access
- Shaping engines: set the string with direct access

Note

Pango Shaping Engines

PangoGlyphInfo

```
struct _PangoGlyphInfo
{
    PangoGlyph    glyph;
    PangoGlyphGeometry geometry;
    PangoGlyphVisAttr attr;
};
```

- `glyph`: glyph index within the font
- `geometry`: width & positioning
- `attr`: visual attributes

Note

Pango Shaping Engines

PangoGlyphGeometry

```
struct _PangoGlyphGeometry
{
    PangoGlyphUnit width;
    PangoGlyphUnit x_offset;
    PangoGlyphUnit y_offset;
};
```

PangoGlyphVisAttr

```
struct _PangoGlyphVisAttr
{
    guint is_cluster_start : 1;
};
```

Note