Pango: Text Handling for the World

Owen Taylor
Red Hat, Inc
otaylor@redhat.com

Ottawa Linux Symposium, 27 July 2001
http://people.redhat.com/otaylor/ols2001/

Introduction

Internationalization

- Expands users base
- Expands developer base

Parts of Internationalization

- Input
- Text processing
- Translation
- Rendering

Just like English with more characters?

- Just replace ASCII with Unicode?

Bidirectional rendering

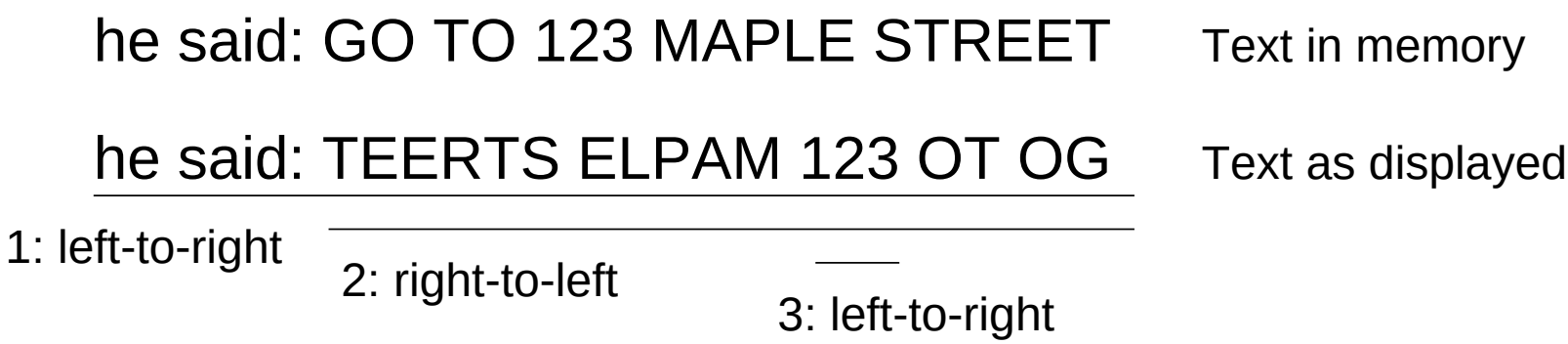
Some scripts written right-to-left (RTL)

- Arabic (Arabic, Farsi, Urdu), Hebrew, Syriac

Mixture of RTL and LTR common

Need reordering process

- From logical order (reading order)
- To visual order (order displayed on screen)



Characters and glyphs

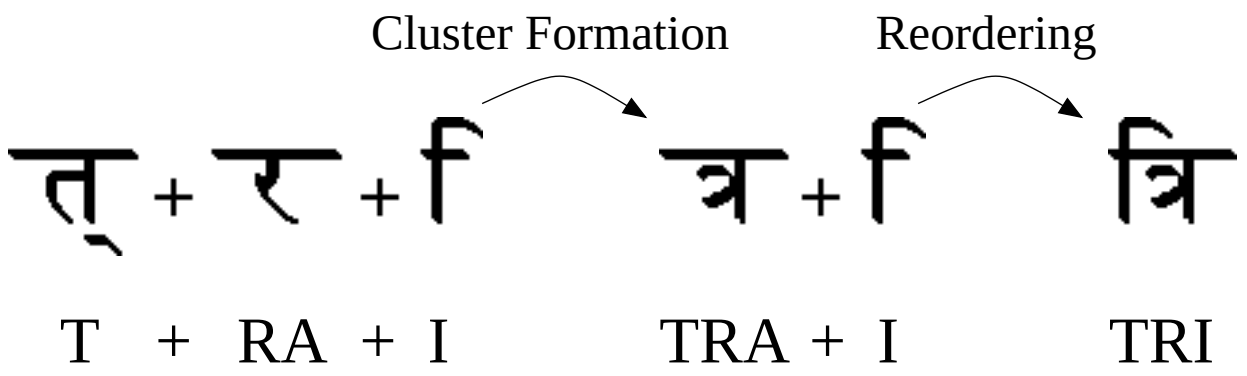
Characters in memory not 1:1 with glyphs on screen

Contextual variants (e.g., Arabic)

- Characters appear different at start, middle of word

Cluster formation

- Multiple characters may join to form a single glyph



Text boundaries

Can’t just break lines on whitespace

- Languages written without spaces
- Some languages (Thai) require dictionary lookup

Cursor positioning also requires care

- Don’t want to put cursor between ‘ and E of ¨
- Graphemes need to be preserved whole

Other boundaries:

- Sentence, word

Handling the complexity

Need high level interface: Pango

- Applications will never get it right otherwise

Same complexity shows up for high quality English text:

- Kerning
- Ligatures
- Hyphenation

Benefits not only for international users

Principles of design

Modularity

- Code for each language in a separate module
- Keeps core small
- Allows modules to be developed independently

Rendering system independence

- Same interface whether rendering to:
  - Windowing system (X, Win32, ...)
  - Locally (outlines in illustration program)
  - Printer

One character set and encoding everywhere

- Unicode

Unicode

Traditionally, one character set per language

- ISO-8859-1, ISO-8859-2, EUC-JP, EUC-TW, KOI8-R...

Unicode / ISO-10646 unifies character sets

- ~100,000 characters
  - Characters from all common encodings
  - Languages without previous standard character set

Encodings

- UCS-4: 32-bit words
- UTF-16: 16-bit words
  - Common in older systems
  - Need to use two words for some characters
- UTF-8: 8-bit words

```
0xxxxxxx
110xxxxx 10xxxxxx
1110xxxx 10xxxxxx 10xxxxxx
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

  - "Nice": efficient, stateless
  - Compatible with printf(), filesystems, etc.

Architecture

Pango Core

- Fundamental types
- Most public interfaces
- Driver routines
- Convenience functionality (PangoLayout)

Rendering Libraries

- PangoX, PangoXft, PangoWin32, PangoFT2...
- Subclasses of PangoFont, PangoFontmap
- Public interfaces for rendering
- Private interfaces used by shaper modules

Modules

Modules

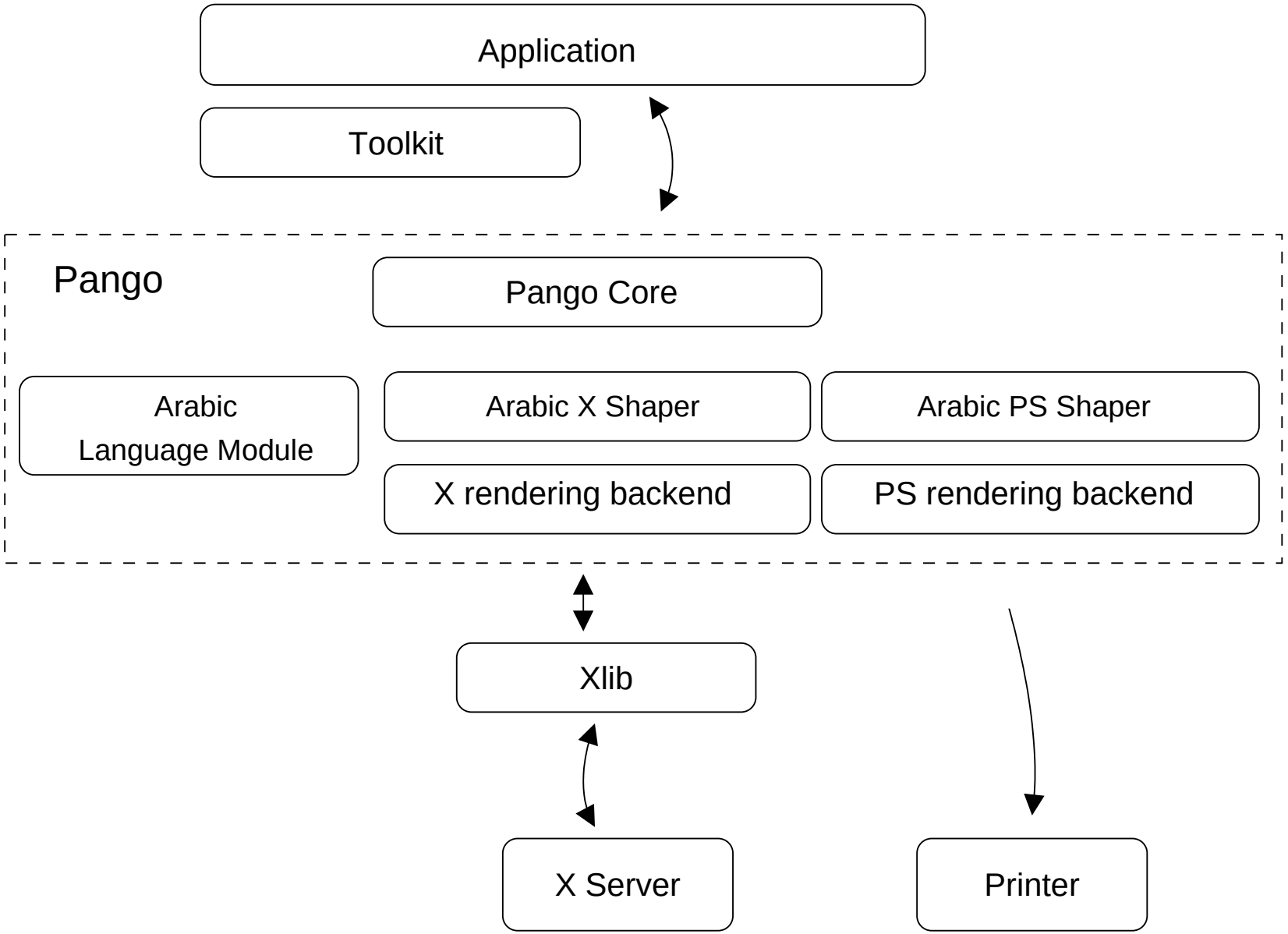
Language Modules

- Specific to a particular language
- Common to multiple rendering systems
- Mostly handle boundary resolution

Shape Modules (shapers)

- Specific to language and rendering system
  (arabic-xft shaper, thai-x shaper, ...)
- Do character to glyph conversion

Architecture



1. Itemization

|This is some |Arabic| text: |السلام عليكم|

Break input text into runs with:

- Shape engine
- Language engine
- Font
- Extra style attributes

2. Boundary resolution

This *is *some *Arabic *text: *السلام *عليكم

Allowed line breaks are determined
Implementation in language engine

(Default implementation works for most languages)

3. Shaping

Reordering:

السلام عليكم => مكيلع مالسلا

Contextual variants and ligatures:

السلام عليكم => السلام عليكم

The characters for each item are converted to glyphs

Done by shape engine

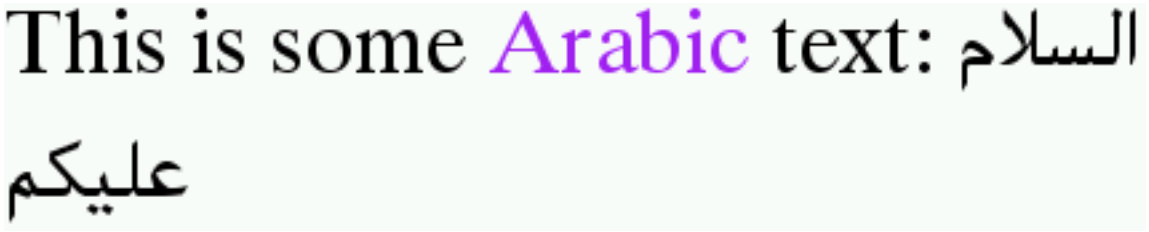
4. Line Breaking

|This is some |Arabic| text: |السلام عليكم|
|This is some |Arabic| text: |السلام |
|عليكم|

The shaped items are split to fit onto lines

The split pieces are reshaped

5. Rendering



The glyphs are rendered to the output device

Done in a way specific to the rendering device

(pango_x_render(), pango_ft2_render ())

Important types

PangoContext

- Information about a particular output device
- Available fonts, resolution, ...

PangoAttrList

- List of text attributes and ranges
  (Family, weight, size, color, ...)
- Used as input to rendering process

Important types (2)

PangoFontDescription

- Structure describing a particular font
  - (Independent of output device)
- Fields are:
  - Family
  - Style (normal, italic, oblique)
  - Variant (normal, small-caps)
  - Weight
  - Stretch (condensed, normal, expanded)
  - Size
- Has string form ("Times Bold 14")

PangoFont

- Describes loaded font
- Result of looking up FontDescription for a Context
- Implementation rendering system dependent
  - (PangoXFont, PangoFT2Font, ...)
- Methods to get metrics, coverage for a font

PangoLayout

Low level facilities hard to use

PangoLayout provides high-level driver

- Create layout
- Set text
- Set attributes
- Render layout

PangoLayout example

```
PangoContext *context = pango_x_get_context
(display);
PangoLayout *layout = pango_layout_new (context);

pango_layout_set_text (layout,
                       "This is some text");
pango_layout_set_width (layout,
                        300 * PANGO_SCALE);
pango_x_render_layout (display, drawable, gc,
                       layout, 0, 0);
```

PangoLayout and editing

Contains routines for:

- X,Y => character
- character => X, Y
- Determine visual ranges for logical range of text
- Visual movement

Complicated by

- Character to glyph mapping not one-to-one
- Bidirectional reordering
- Need to obey grapheme boundaries

Existing systems

Uniscribe (Microsoft)

- Has roughly the same pipeline
  - Pango’s is simplified and made nicer
- Only works with Win32 fonts

Java2D Text API

- Has high level object like PangoLayout
- Doesn’t provide as much low-level control
- Only works with Java2D rendering

Implementation

Written in C using GObject library

- Object orientation (inheritance, polymorphism)
- Language binding support
- (Also, signals, interfaces, properties, ...)
- Based on GtkObject in GTK+-2.0
- Separated out into GLib package so other libs can use it

Pango also uses many other facilities of GLib

- Data structures, Module loading, GMarkup, Unicode manipulation

Bidirectional reordering uses FriBidi library

- (Included as part of Pango)

Current rendering system support

- Traditional X fonts
- Anti-aliased X fonts using Xft and FreeType
  - Includes OpenType Support
- FreeType backend
  - Used for frame-buffer GTK+, libgnomecanvas
- Win32 fonts

Current language support

Depends on backend

X backend supports:

- "Basic" shaper: Roman, Greek, Cyrillic, East Asian Scripts
- Arabic, Hebrew, Korean, Thai, 7 Indic scripts

Other backends:

- At least "basic" scripts

Pango in GTK+-2.0

GTK+-2.0 primary user of Pango currently

- (But Pango not GTK+ specific)

All text rendering in GTK+ done through Pango

- GtkLabel (static text display)
- GtkEntry (single line text editing)
- GtkTextView (multi-line text editing)

GTK+ uses only PangoLayout, not low-level interfaces

- GtkLabel / GtkEntry use a single PangoLayout
- GtkTextView creates PangoLayout on the fly for each paragraph

Benefits for GTK+

Internationalization

(GTK+-2.0 also includes new input method framework and keyboard support, reverses widget layouts for RTL languages)

Sane fonts

No more XLFDs: `-*-times-bold-r-normal--*-140-*-*-*-*-iso8859-1`

Instead:

```
Times Bold 14
```

Fonts abstracted away from toolkit

Adding support for anti-aliased scalable fonts with Xrender, Xft, required no interface changes

Embedded formatting in labels

XML-like markup in labels

```
GtkWidget *label = gtk_label_new (NULL);

gtk_label_set_markup (GTK_LABEL (label),
                      "<span color=\"red\">Red</span>"
                      "<big><i>Text</i></big>");
```

Trivial to implement because GtkLabel uses

PangoLayout

Future plans

High Quality Printing

- Better hyphenation and line breaking
- Full Justification
- Vertical Text (Chinese, Japanese)

Better language support

- (especially for non-X backends)

Use OpenType fonts as normal font format

- Adobe/Microsoft standard extension of TrueType
- Info about Alternate glyphs, ligatures, etc.
- Allows Type1 outlines as well
- Basic support in Pango-1.0 (Arabic)

Conclusion

Acknowledgments

Shaper Authors

Changwoo Ryu, Sivaraj Doddannan, Karl Koehler, Robert Brady

FriBidi team

Dov Grobgeld, Roozbeh Pourander, ...

`http://fribidi.sourceforge.net`

Havoc Pennington

(markup, fixed PangoLayout mess)

More information

Pango web site

`http://www.pango.org`

GTK+ web site

`http://www.gtk.org`

API documentation

`http://developer.gnome.org`