Adapting binary software with multiple object layouts

Stephen Kell

Stephen.Kell@cl.cam.ac.uk

Multiple object layouts

Linking together mismatched binaries. Why?

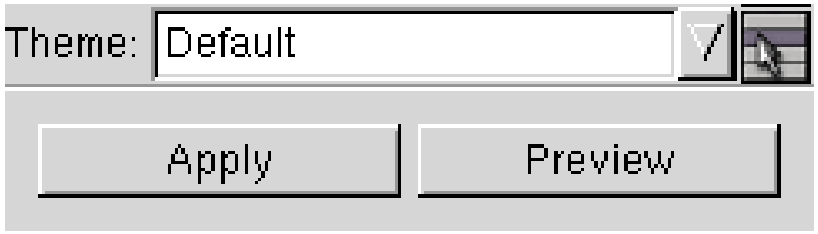
- late composition
- no source code?

What's mismatched?

- symbol names (easy)
- assignment of meanings to values (tractable)
- function signatures (tractable)
- **object layout**
- object model of application domain
- communication structure, paradigm, . . .

Software evolution

Experimental work right now: `gtk-theme-switch`



- two functionally-identical C programs, ∼1000 lines
- one uses Gtk+ 1.2, the other Gtk+ 2.0
- small number of API changes, but. . .
- . . . diff (`-U3`) is ∼500 lines
- forked codebase (maintenance overhead)
- why can’t one source program work with both libraries?
- why can’t one *binary* work with both libraries?

Two approaches

"Static" approach (a bit like RPC within one VAS):

- find points where control-flow crosses "rep domains"
- interpose code to
  - copy + transform into required representation...
  - ...any heap objects needed on the other side.

"Dynamic" approach (a bit like pointer swizzling):

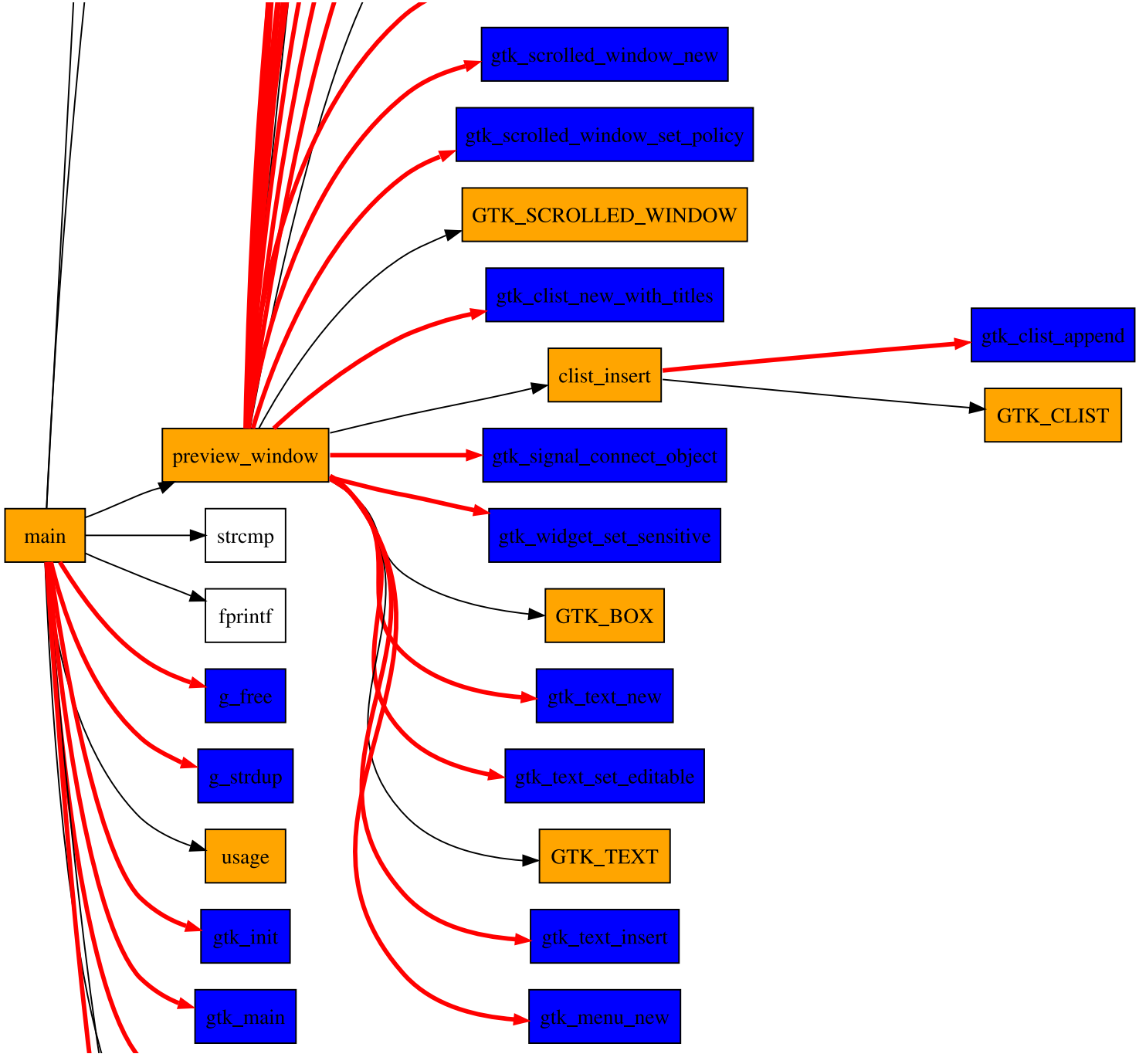
- allocate multi-rep objects in special memory area...
- ...segregated by rep-domain at page granularity
- keep copies for each rep behind the scenes
- trap and copy/map on rep-mismatched accesses

The static approach

What's tricky:

- function pointers
- object identity
- consistency (in multithreaded context)
- knowing how much of the object graph to copy
- knowing when to sync copies
- objects of vague length
- forked object modifications
- deallocation

A picture of the static approach



The dynamic approach

The dynamic approach seems cleaner, but

- need to handle SIGSEGV from user-space
- want to do better than trapping on every access. . .
- . . . trap transitions by protecting text pages?
  - still faults too much in some multithread cases
  - need access to NX bit from user-space
- object identity and function pointers not a problem
- sync problems somewhat eased
- deallocation, sync frequency and forking still a problem

A picture of the dynamic approach

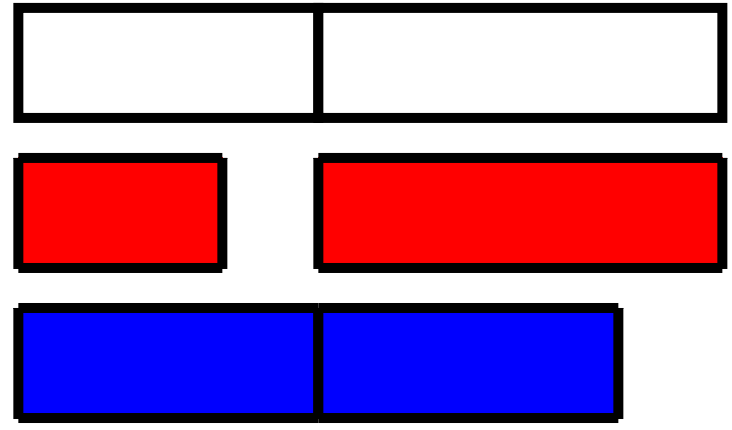
...
obj = malloc(42);
...

magic_malloc(size_t)

issued pointer block

rep 1 objects

rep 2 objects



A possible third way

One could conceive other approaches:

- intra-object relocations?
- new point on fully-compiled – bytecode – . . . spectrum

. . . and a more general version of the problem:

- no longer 1:1 correspondences between object reps. . .
- instead must reformulate whole graphs at a time
  - wanted: a neat abstraction of graphs (grammars?)
  - pattern-matching / rewrite rules for the above

Vote, and thank you

On my first whistle, you will start voting. . .

Thanks for your attention. Any questions?

Source code

```
#include <gtk/gtk.h>

#define INIT_GTK if (!using_gtk) {  gtk_init  (&argc, &argv); using_gtk = 1;  }

/* globals */
GtkWidget *dockwin, *box;
int  using_gtk = 0;
static  void quit(void);

static  void
dock (void)
{
        GdkColormap *colormap;
        dockwin = gtk_dialog_new();
        gtk_widget_realize(dockwin);
```