Demystifying GObject

History...

GIMP → GTK+ → GNOME

ca. 1998

Primarily C world

Remember egcs?

UI toolkits ♥ OOP

⇒ Object system in C

glib, gio

gobject

Features...

Inheritance

Refcounting

Properties

Signals

Bindings

Portable

“Demo”

Let's start...

```
struct GstAudioRingBuffer {
  ...
};

struct GstAudioRingBufferClass {
  ...
};
```

```
G_DEFINE_TYPE (
    GstAudioRingBuffer,
    gst_audio_ring_buffer,
    GST_TYPE_ELEMENT);
```

```
void gst_audio_ring_buffer_class_init (GstAudioRingBufferClass *klass);
void gst_audio_ring_buffer_init (GstAudioRingBuffer *object);
void gst_audio_ring_buffer_finalize (GstAudioRingBuffer *object);
```

```
#define GST_TYPE_AUDIO_RING_BUFFER \
    gst_audio_ring_buffer_get_type()
```

```
GstAudioRingBuffer *rb;

rb = g_object_new (GST_TYPE_AUDIO_RING_BUFFER, NULL);
```

```
GType
g_type_register_static (
    parent,
    type_name,
    info,
    flags);
```

info

- class - size, init(), finalize()
- instance - size, init(), finalize()

```
static GType
gst_audio_ringbuffer_get_type (void)
{
  static GType gst_audio_ringbuffer_type = 0;

  if (!gst_audio_ringbuffer_type) {
    static const GTypeInfo gst_audio_ringbuffer_info = {
      sizeof (GstAudioRingbufferClass),
      NULL,
      NULL,
      (GClassInitFunc) gst_audio_ringbuffer_class_init,
      NULL,
      NULL,
      sizeof (GstAudioRingbuffer),
      0,
      (GInstanceInitFunc) gst_audio_ringbuffer_init,
      NULL
    };

    gst_audio_ringbuffer_type =
        g_type_register_static (GST_TYPE_ELEMENT, "GstAudioRingbuffer",
        &gst_audio_ringbuffer_info, 0);
  }
  return gst_audio_ringbuffer_type;
}
```

Everything is initialised on first invocation!

(how?)

Down the rabbit hole...

```
typedef gsize GType;

struct GTypeInstance {

};

struct GTypeClass {

};
```

```
typedef gsize GType;

struct GTypeInstance {
  GTypeClass *g_class;
};

struct GTypeClass {
  GType g_type;
};
```

```
struct MyChild {
  GTypeInstance parent;
};

struct MyChildClass {
  GTypeClass parent_class;
};
```

```
struct MyGrandChild {
  MyChild parent;
};

struct MyGrandChildClass {
  MyChildClass parent_class;
};
```

MyChild **is a** GTypeInstance

MyGrandChild **is a** MyChild

```
void my_child_goes_to_school (MyChild *child);

void foo()
{
  MyChild *c;
  MyGrandChild *g;

  g = g_type_create_instance (MY_GRAND_CHILD_TYPE);
  c = (MyChild *) g;

  my_child_goes_to_school (c);
}
```

```
#define MY_CHILD(o) \
    G_TYPE_CHECK_INSTANCE(o, MY_CHILD_TYPE, MyChildType)

c = MY_CHILD (g);
```

We inherited data

But what about functions?

```
struct MyChildClass {
  MyChildClass parent_class;
  void (*go_to_school) (MyChild *child);
};

struct MyGrandChildClass {
  MyChildClass parent_class;
};
```

```
void my_child_go_to_school (MyChild *child);

void my_child_class_init (MyChildClass *klass)
{
  klass->go_to_school = my_child_go_to_school;
}

void my_child_goes_to_school (MyChild *child)
{
  MyChildClass *klass =
    G_TYPE_INSTANCE_GET_CLASS(child, MY_CHILD_TYPE, MyChildClass);

  klass->go_to_school (child);
}
```

```
void my_grand_child_go_to_school) (MyChild *child);

void my_grand_child_class_init (MyGrandChildClass *klass)
{
  MyChildClass *child_class = MY_CHILD_CLASS (klass);

  child_class->go_to_school = my_grand_child_go_to_school;
}
```

Can get a copy of parent class

Helps while chaining up

Interface is a set of function pointers

Specified during type registration

That's the type system

Everything else goes on top

GObject

Ref-counted type

g_object_**new**
g_object_**ref**
g_object_**unref**

refcount == 0

klass->dispose()
klass->finalize()

Properties

Registered at class init

Example

Signals for observers

Signals for actions

Registered at class init

Specify argument and return types

Default implementation possible

GObject Introspection

Bindings made easy*

That's all folks!

Questions?

@louiswu / Ford_Prefect