NAME

Glib::MainLoop − An event source manager

DESCRIPTION

Event-driven programs need some sort of loop which watches for events and launches the appropriate actions. Glib::MainLoop provides this functionality.

Mainloops have context, provided by the MainContext object. For the most part you can use the default context (see `default`), but if you want to create a subcontext for a nested loop which doesn't have the same event sources, etc, you can.

Event sources, attached to main contexts, watch for events to happen, and launch appropriate actions. Glib provides a few ready-made event sources, the Glib::Timeout, Glib::Idle, and io watch (`Glib::IO->add_watch`).

Under the hood, Gtk+ adds event sources for GdkEvents to dispatch events to your widgets. In fact, Gtk2 provides an abstraction of Glib::MainLoop (See `Gtk2->main` and friends), so you may rarely have cause to use Glib::MainLoop directly.

Note: As of version 1.080, the Glib module uses a custom event source to ensure that perl's safe signal handling and the glib polling event loop play nicely together. It is no longer necessary to install a timeout to ensure that async signals get handled in a timely manner.

CONSTANTS

`SOURCE_REMOVE` and `SOURCE_CONTINUE` are designed for use as the return values from timeout, idle and I/O watch source functions. They return true to keep running or false to remove themselves. These constants can help you get that the right way around.

```
Glib::SOURCE_CONTINUE     # true
Glib::SOURCE_REMOVE       # false
```

METHODS

maincontext thingamabob = Glib::MainContext->new

mainloop = Glib::MainLoop->new ($context=undef, `$is_running`**=FALSE)**

- `$context` (Glib::MainContext thingamabob)
- `$is_running` (boolean)

integer = Glib::Timeout->add ($interval, `$callback`**,** `$data`**=undef,** `$priority`**=G_PRIORITY_DEFAULT)**

- `$interval` (integer) number of milliseconds
- `$callback` (subroutine)
- `$data` (scalar)
- `$priority` (integer)

Run *`$callback`* every *`$interval`* milliseconds until *`$callback`* returns false. Returns a source id which may be used with `Glib::Source->remove`. Note that a mainloop must be active for the timeout to execute.

integer = Glib::Idle->add ($callback, `$data`**=undef,** `$priority`**=G_PRIORITY_DEFAULT_IDLE)**

- `$callback` (subroutine)
- `$data` (scalar)
- `$priority` (integer)

Run *`$callback`* when the mainloop is idle. If *`$callback`* returns false, it will uninstall itself, otherwise, it will run again at the next idle iteration. Returns a source id which may be used with `Glib::Source->remove`.

integer = Glib::Timeout->add_seconds ($interval, `$callback`**,** `$data`**=undef,** `$priority`**=G_PRIORITY_DEFAULT)**

- `$interval` (integer)
- `$callback` (scalar)
- `$data` (scalar)
- `$priority` (integer)

Since: glib 2.14

integer = Glib::IO->add_watch ($fd, `$condition`**,** `$callback`**,** `$data`**=undef,** `$priority`**=G_PRIORITY_DEFAULT)**

- `$fd` (integer) file descriptor, e.g. fileno($filehandle)
- `$condition` (Glib::IOCondition)
- `$callback` (subroutine)
- `$data` (scalar)
- `$priority` (integer)

Run *`$callback`* when there is an event on *`$fd`* that matches *`$condition`*. The watch uninstalls itself if *`$callback`* returns false. Returns a source id that may be used with `Glib::Source->remove`.

Glib's IO channels serve the same basic purpose as Perl's file handles, so for the most part you don't see GIOChannels in Perl. The IO watch integrates IO operations with the main loop, which Perl file handles don't do. For various reasons, this function requires raw file descriptors, not full file handles. See `fileno` in perlfunc.

maincontext thingamabob = `$loop`**->get_context**
maincontext thingamabob = Glib::MainContext->default
boolean = `$context`**->is_owner**

Since: glib 2.12

boolean = `$loop`**->is_running**
boolean = `$context`**->iteration ($may_block)**

- `$may_block` (boolean)

integer = Glib::main_depth

Find the current main loop recursion level. This is handy in fringe situations, but those are very rare; see the C API reference for a more in-depth discussion.

Since: glib 2.4

boolean = `$context`**->pending**
`$loop`**->quit**
boolean = Glib::Source->remove ($tag)

- `$tag` (integer)

Remove an event source. *`$tag`* is the number returned by things like `Glib::Timeout->add`, `Glib::Idle->add`, and `Glib::IO->add_watch`.

`$loop`**->run**
integer = Glib::Child->watch_add ($pid, `$callback`**,** `$data`**=undef,** `$priority`**=G_PRIORITY_DEFAULT)**

- `$pid` (integer) child process ID
- `$callback` (subroutine)
- `$data` (scalar)
- `$priority` (integer)

Add a source to the default main context which will call

```
&$callback ($pid, $waitstatus, $data)
```

when child process `$pid` terminates. The return value is a source id which can be used with

`Glib::Source->remove`. When the callback is made the source is removed automatically.

In a non-threaded program Glib implements this source by installing a SIGCHLD handler. Don't change `$SIG{CHLD}` in Perl or the callback will never run.

Since: glib 2.4

ENUMS AND FLAGS

flags Glib::IOCondition

- 'in' / 'G_IO_IN'
- 'out' / 'G_IO_OUT'
- 'pri' / 'G_IO_PRI'
- 'err' / 'G_IO_ERR'
- 'hup' / 'G_IO_HUP'
- 'nval' / 'G_IO_NVAL'

SEE ALSO

Glib

COPYRIGHT

Copyright (C) 2003−2011 by the gtk2−perl team.

This software is licensed under the LGPL. See Glib for a full notice.