GNOME Lib

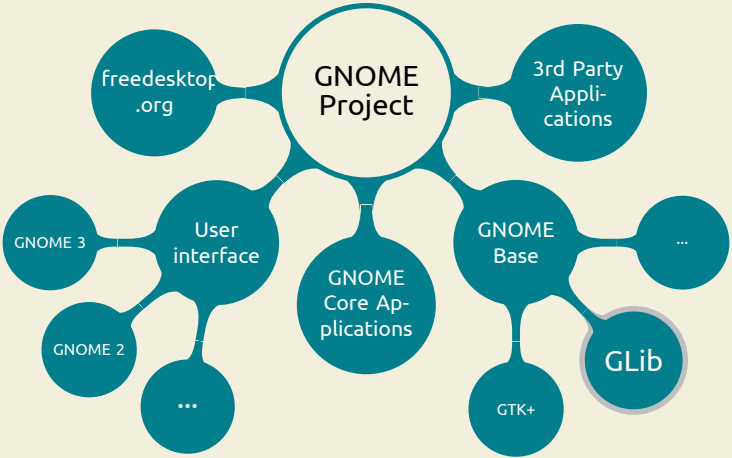
Sebastian Funck

Fachbereich Informatik
Universität Hamburg

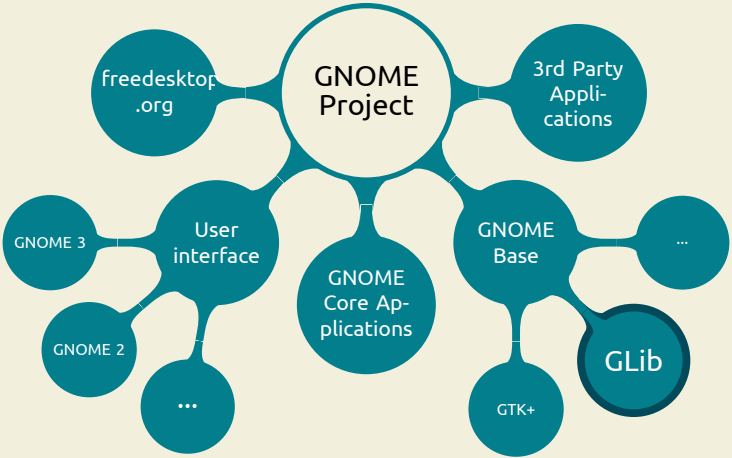
5. Juni 2014

Inhaltsverzeichniss

GNOME Project



GNOME Project



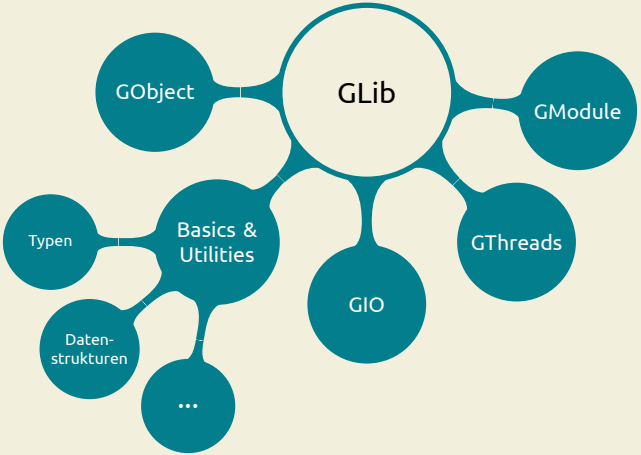
Einleitung

- GTK+ - UI-Framework
- Frei & Plattform unabhängig
- Abhängigkeit: libc
- Debian : apt-get install libglib2.0-dev
- Windows: http://www.gtk.org/download/win32.php

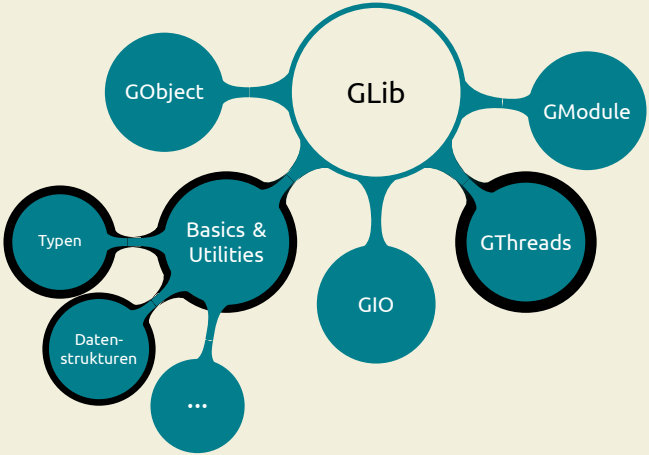
```
#include <glib.h>
```

```
gcc foobar.c -o foobar `pkg-config --cflags --libs glib-2.0`
```

GNOME Project



GNOME Project



Basic-Types

- gint8, gint16, gint32, guint32, gboolean, gpointer ...

gpointer NULL

gboolean FALSE, TRUE

- Benannte 'Basic-Types':

  gsize (uint)

  GPid (int)

  goffset (gint64)

GFunc

GFunc	Funktionszeiger
GCompareFunc	Zeiger auf eine Funktion, die zwei Datenstrukturen vergleicht (Siehe Java *Comparable*)
GHashFunc	Zeiger auf eine Hashfunktion. (Siehe Java *Object.hashcode()*)

...

GError

- Error Reporting

```
1  struct GError {
2    GQuark       domain;
3    gint         code;
4    gchar       *message;
5  };
```

Listen: GSList & GList

- GSList: Einfach verkettet
- GList: Doppelt verkettet

```
1  struct GList {
2          gpointer data;
3          GList *next;
4          GList *prev; // Nicht in Single-Linked List
5  };
```

Beispiel: GList

```
1  GList *liste = NULL;
2
3  liste = g_list_append(liste, "Erstes Element");
4  liste = g_list_append(liste, "Zweites Element");
5  liste = g_list_append(liste, "Drittes Element");
6
7  for (GList *li = liste; li != NULL; li = li->next) {
8          g_printf("%s\n", li->data);
9  }
```

Ausgabe:

Erstes Element
Zweites Element
Drittes Element

Beispiel: GList

```
1  GList *liste = NULL;
2
3  liste = g_list_append(liste, "Erstes Element");
4  liste = g_list_append(liste, "Zweites Element");
5  liste = g_list_append(liste, "Drittes Element");
6
7  for (GList *li = liste; li != NULL; li = li->next) {
8          g_printf("%s\n", li->data);
9  }
```

Ausgabe:

Erstes Element
Zweites Element
Drittes Element

Map & Set: GHashTable

- Key → Value
- siehe Java *Map* & *Set*

GHashTable erzeugen:

```
1  GHashTable* g_hash_table_new(GHashFunc hash_func,
2                               GCompareFunc comp_func);
```

Key-Value-Pair einfügen:

```
1  void g_hash_table_insert(GHashTable *table,
2                           gpointer key, gpointer value);
```

Value von einem Key abfragen:

```
1  gpointer g_hash_table_lookup(GHashTable *table,
2                               gconstpointer key);
```

Map & Set: GHashTable

- Key $\rightarrow$ Value
- siehe Java *Map* & *Set*

GHashTable erzeugen:

```
1  GHashTable* g_hash_table_new(GHashFunc hash_func,
2                               GCompareFunc comp_func);
```

Key-Value-Pair einfügen:

```
1  void g_hash_table_insert(GHashTable *table,
2                           gpointer key, gpointer value);
```

Value von einem Key abfragen:

```
1  gpointer g_hash_table_lookup(GHashTable *table,
2                               gconstpointer key);
```

GString

- UTF-8
- Einfachere Handhabung
- Viele Funktionen

```
1  struct GString {
2      gchar *str;
3      gsize len;
4      gsize allocated_len;
5  };
```

Ausgabe von Strings

```
#include <glib/gprintf.h>
```

```
1 GString * stringA = g_string_new("Ich␣bin␣ein␣");
2 GString * stringB = g_string_new("String!");
3 GString * stringC = g_string_append(stringA, stringB->str);
4
5 g_printf(stringC->str);
```

Ausgabe:

Ich bin ein String!

Ausgabe von Strings

```
#include <glib/gprintf.h>
```

```
1 GString * stringA = g_string_new("Ich␣bin␣ein␣");
2 GString * stringB = g_string_new("String!");
3 GString * stringC = g_string_append(stringA, stringB->str);
4
5 g_printf(stringC->str);
```

Ausgabe:

Ich bin ein String!

String Kodierung

gchar	UTF-8 character
gunichar2	UTF-16 character
gunichar	UTF-32 character

```
1 gpointer * g_convert (const gchar *str,
2         gssize len,
3         const gchar *to_charset, const gchar *from_charset,
4         gsize *bytes_read, gsize *bytes_written,
5         GError **error);
```

String Kodierung

gchar	UTF-8 character
gunichar2	UTF-16 character
gunichar	UTF-32 character

```
1  gpointer * g_convert (const gchar *str,
2          gssize len,
3          const gchar *to_charset, const gchar *from_charset,
4          gsize *bytes_read, gsize *bytes_written,
5          GError **error);
```

String Kodierung

gchar	UTF-8 character
gunichar2	UTF-16 character
gunichar	UTF-32 character

```
1 gpointer * g_convert (const gchar *str,
2         gssize len,
3         const gchar *to_charset, const gchar *from_charset,
4         gsize *bytes_read, gsize *bytes_written,
5         GError **error);
```

String Kodierung

gchar	UTF-8 character
gunichar2	UTF-16 character
gunichar	UTF-32 character

```
1 gpointer * g_convert (const gchar *str,
2         gssize len,
3         const gchar *to_charset, const gchar *from_charset,
4         gsize *bytes_read, gsize *bytes_written,
5         GError **error);
```

String Kodierung

gchar	UTF-8 character
gunichar2	UTF-16 character
gunichar	UTF-32 character

```
1  gpointer * g_convert (const gchar *str,
2          gssize len,
3          const gchar *to_charset, const gchar *from_charset,
4          gsize *bytes_read, gsize *bytes_written,
5          GError **error);
```

String Kodierung

gchar	UTF-8 character
gunichar2	UTF-16 character
gunichar	UTF-32 character

```
1  gpointer * g_convert (const gchar *str,
2          gssize len,
3          const gchar *to_charset, const gchar *from_charset,
4          gsize *bytes_read, gsize *bytes_written,
5          GError **error);
```

String Kodierung

gchar	UTF-8 character
gunichar2	UTF-16 character
gunichar	UTF-32 character

```
1 gpointer * g_convert (const gchar *str,
2         gssize len,
3         const gchar *to_charset, const gchar *from_charset,
4         gsize *bytes_read, gsize *bytes_written,
5         GError **error);
```

Beispiel: String Kodierung

```
1  GString* str_utf8;
2  gunichar2* str_utf16;
3
4  str_utf8  = g_string_new("Ich bin ein String!\n");
5  str_utf16 = g_convert(str_utf8->str, -1,
6                        "UTF-16BE", "UTF-8",
7                        NULL, NULL, NULL );
```

Memory:

0049	0063	0068	0020	0062	0069	006E	0020
0065	0069	006E	0020	0053	0074	0072	0069
006E	0067	0021	000A	0000			

Ich bin ein String!\n\0

GMainLoop

- Hauptbestandteil einer GLib-Anwendung
- Mögliche Event-Quellen:
  - Timeout (Zeitintervall)
  - 'File-Watch' (GI/O)
  - Signals
  - 'Child-Watch'
- Idle Funktionen

GMainLoop

GMainLoop erstellen:

```
1  GMainLoop* g_main_loop_new ( GMainContext context ,
2                               gboolean is_running );
```

GMainLoop starten:

```
1  void g_main_loop_run ( GMainLoop *loop );
```

GMainLoop beenden:

```
1  void g_main_loop_quit ( GMainLoop *loop );
```

GMainLoop

GMainLoop erstellen:

```
1  GMainLoop* g_main_loop_new ( GMainContext context ,
2                               gboolean is_running );
```

GMainLoop starten:

```
1  void g_main_loop_run ( GMainLoop *loop );
```

GMainLoop beenden:

```
1  void g_main_loop_quit ( GMainLoop *loop );
```

GMainLoop

GMainLoop erstellen:

```
1  GMainLoop* g_main_loop_new ( GMainContext context ,
2                               gboolean is_running );
```

GMainLoop starten:

```
1  void g_main_loop_run ( GMainLoop *loop );
```

GMainLoop beenden:

```
1  void  g_main_loop_quit ( GMainLoop *loop );
```

Hinzufügen von Quellen

Timeout Event-Quelle:

```
1  guint g_timeout_add ( guint intervall,
2                        GSourceFunc func,
3                        gpointer data );
```

Idle-Funktion:

```
1  guint g_idle_add ( GSourceFunc func,
2                     gpointer data);
```

Child-Watch:

```
1  guint g_child_watch_add ( GPid child_pid,
2                            GChildwatchFunc func,
3                            gpointer data);
```

Hinzufügen von Quellen

Timeout Event-Quelle:

```
1  guint g_timeout_add ( guint intervall,
2                        GSourceFunc func,
3                        gpointer data );
```

Idle-Funktion:

```
1  guint g_idle_add ( GSourceFunc func,
2                     gpointer data);
```

Child-Watch:

```
1  guint g_child_watch_add ( GPid child_pid,
2                            GChildwatchFunc func,
3                            gpointer data);
```

Hinzufügen von Quellen

Timeout Event-Quelle:

```
1  guint g_timeout_add ( guint intervall,
2                        GSourceFunc func,
3                        gpointer data );
```

Idle-Funktion:

```
1  guint g_idle_add ( GSourceFunc func,
2                     gpointer data);
```

Child-Watch:

```
1  guint g_child_watch_add ( GPid child_pid,
2                            GChildwatchFunc func,
3                            gpointer data);
```

Beispiel

```
1   int     main() {
2       GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3       g_timeout_add( 1000 , countdown, mainloop );
4       g_main_loop_run( mainloop );
5       g_main_loop_unref( mainloop );
6       return 0;
7   }
8
9   gboolean    countdown(gpointer data) {
10      static int counter = 5;
11      g_printf("%d...", counter);
12      if (counter <= 0) {
13          g_main_loop_quit( (GMainLoop*)data );
14          return FALSE;
15      }
16      counter--;
17      return TRUE;
18  }
```

Beispiel

```
1  int     main() {
2      GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3      g_timeout_add( 1000 , countdown, mainloop );
4      g_main_loop_run( mainloop );
5      g_main_loop_unref( mainloop );
6      return 0;
7  }
8
9  gboolean     countdown(gpointer data) {
10     static int counter = 5;
11     g_printf("%d...", counter);
12     if (counter <= 0) {
13         g_main_loop_quit( (GMainLoop*)data );
14         return FALSE;
15     }
16     counter--;
17     return TRUE;
18 }
```

Beispiel

```
1   int     main() {
2       GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3       g_timeout_add( 1000 , countdown, mainloop );
4       g_main_loop_run( mainloop );
5       g_main_loop_unref( mainloop );
6       return 0;
7   }
8
9   gboolean    countdown(gpointer data) {
10      static int counter = 5;
11      g_printf("%d...", counter);
12      if (counter <= 0) {
13          g_main_loop_quit( (GMainLoop*)data );
14          return FALSE;
15      }
16      counter--;
17      return TRUE;
18  }
```

Beispiel

```
1   int     main() {
2       GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3       g_timeout_add( 1000 , countdown, mainloop );
4       g_main_loop_run( mainloop );
5       g_main_loop_unref( mainloop );
6       return 0;
7   }
8
9   gboolean     countdown(gpointer data) {
10      static int counter = 5;
11      g_printf("%d...", counter);
12      if (counter <= 0) {
13          g_main_loop_quit( (GMainLoop*)data );
14          return FALSE;
15      }
16      counter--;
17      return TRUE;
18  }
```

Beispiel

```
1   int     main() {
2       GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3       g_timeout_add( 1000 , countdown, mainloop );
4       g_main_loop_run( mainloop );
5       g_main_loop_unref( mainloop );
6       return 0;
7   }
8
9   gboolean     countdown(gpointer data) {
10      static int counter = 5;
11      g_printf("%d...", counter);
12      if (counter <= 0) {
13          g_main_loop_quit( (GMainLoop*)data );
14          return FALSE;
15      }
16      counter--;
17      return TRUE;
18  }
```

Beispiel

```
1   int     main() {
2       GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3       g_timeout_add( 1000 , countdown, mainloop );
4       g_main_loop_run( mainloop );
5       g_main_loop_unref( mainloop );
6       return 0;
7   }
8
9   gboolean     countdown(gpointer data) {
10      static int counter = 5;
11      g_printf("%d...", counter);
12      if (counter <= 0) {
13          g_main_loop_quit( (GMainLoop*)data );
14          return FALSE;
15      }
16      counter--;
17      return TRUE;
18  }
```

Beispiel

```
1  int     main() {
2      GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3      g_timeout_add( 1000 , countdown, mainloop );
4      g_main_loop_run( mainloop );
5      g_main_loop_unref( mainloop );
6      return 0;
7  }
```

Ausgabe: 5..

Beispiel

```
1  int     main() {
2      GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3      g_timeout_add( 1000 , countdown, mainloop );
4      g_main_loop_run( mainloop );
5      g_main_loop_unref( mainloop );
6                  return 0;
7  }
```

Ausgabe: 5..4..

Beispiel

```
1  int     main() {
2      GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3      g_timeout_add( 1000 , countdown, mainloop );
4      g_main_loop_run( mainloop );
5      g_main_loop_unref( mainloop );
6                  return 0;
7  }
```

Ausgabe: 5..4..3..

Beispiel

```
1  int     main() {
2      GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3      g_timeout_add( 1000 , countdown, mainloop );
4      g_main_loop_run( mainloop );
5      g_main_loop_unref( mainloop );
6      return 0;
7  }
```

Ausgabe: 5..4..3..2..

Beispiel

```
1  int     main() {
2      GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3      g_timeout_add( 1000 , countdown, mainloop );
4      g_main_loop_run( mainloop );
5      g_main_loop_unref( mainloop );
6      return 0;
7  }
```

Ausgabe: 5..4..3..2..1..

Beispiel

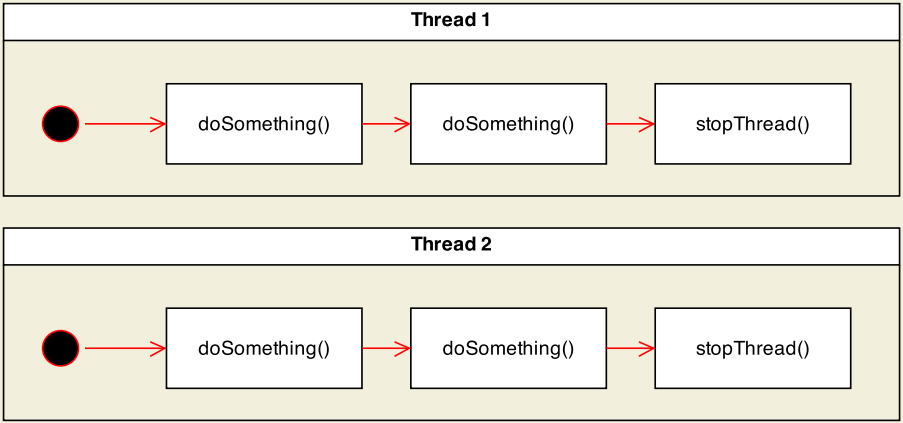
```
1  int     main() {
2      GMainLoop *mainloop = g_main_loop_new( NULL , FALSE );
3      g_timeout_add( 1000 , countdown, mainloop );
4      g_main_loop_run( mainloop );
5      g_main_loop_unref( mainloop );
6      return 0;
7  }
```

Ausgabe: 5..4..3..2..1..0..

Was geht noch?

- Mehrere Eventloops
- Eigene Eventquellen
- Verknüpfen mit einer GUI

GThread


Thread 1
doSomething()
doSomething()
stopThread()
Thread 2
doSomething()
doSomething()
stopThread()

GThread

GThread erstellen:

```
1  GThread * g_thread_create (GThreadFunc func,
2                             gpointer data,
3                             gboolean joinable,
4                             GError **error);
```

GThread 'joinen':

```
1  gpointer g_thread_join (GThread *thread);
```

GThreadFunc Signatur:

```
1  gpointer (*GThreadFunc) (gpointer data);
```

GThread

GThread erstellen:

```
1  GThread * g_thread_create (GThreadFunc func,
2                             gpointer data,
3                             gboolean joinable,
4                             GError **error);
```

GThread 'joinen':

```
1  gpointer g_thread_join (GThread *thread);
```

GThreadFunc Signatur:

```
1  gpointer (*GThreadFunc) (gpointer data);
```

GThread

GThread erstellen:

```
1   GThread * g_thread_create (GThreadFunc func,
2                              gpointer data,
3                              gboolean joinable,
4                              GError **error);
```

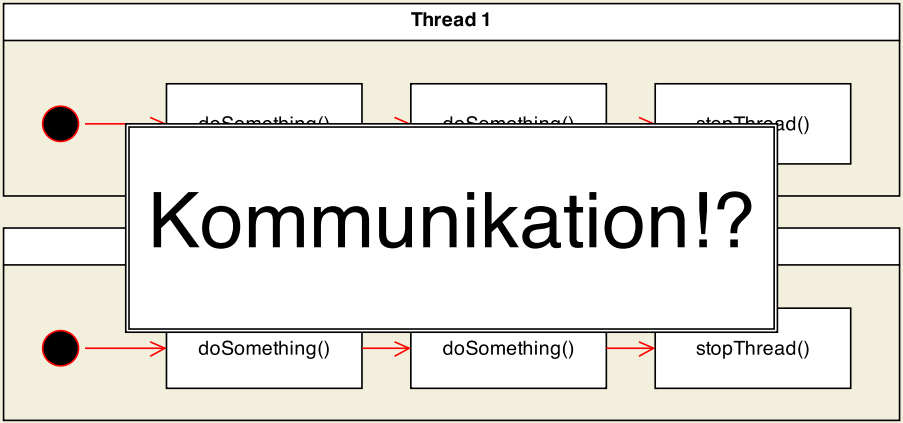
GThread 'joinen':

```
1 gpointer g_thread_join (GThread *thread);
```

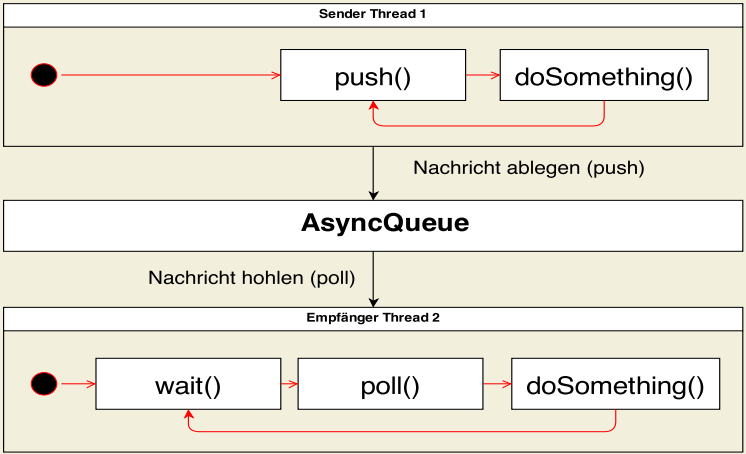
GThreadFunc Signatur:

```
1 gpointer (*GThreadFunc) (gpointer data);
```

GThread

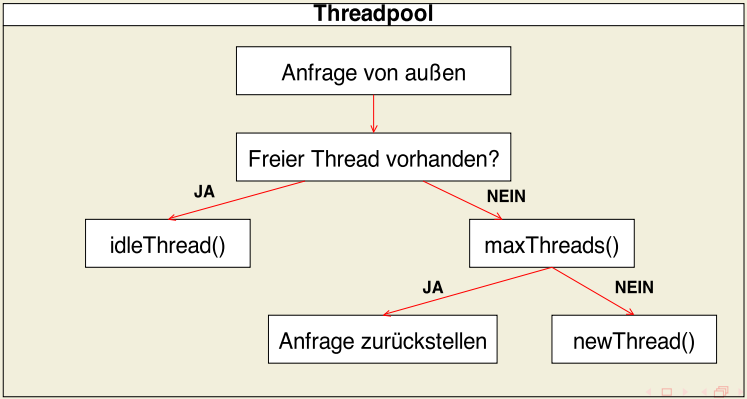

Thread 1
doSomething()
doSomething()
stopThread()
Kommunikation!?
doSomething()
doSomething()
stopThread()

GAsyncQueue



GThreadpool

- Thread-Verwalter
- Häufige Threaderstellung vermeiden



Zusammenfassung

- Einfache Datenstrukturen: Listen, Hashtable
- GString - String's und Kodierung
- GMainLoop - Eventloops
- GThread, GAsyncQueue, GThreadpools

Vor und Nachteile

- + Umfangreiche Dokumentation
- + Freie Software
- + Plattform unabhängig
- + Verwendung in großen Projekten (Gnome Desktop, GIMP, ...)

- - Wenige Beispiele in der Dokumentation
- - Umfangreichere Konkurrenz (Qt-Project)
- - GObject im Vergleich zu Hochsprachen sehr kompliziert

Ausblick

- Queues, Stacks
- Dynamische Arrays
- Binär- und '*n-way*'-Bäume
- Timer
- Pseudo-Zufallsgenerator
- ...

Ausblick: GObject

- Objekt- & Typsystem
- Objective-C

- GType
  - GTypeClass
  - GTypeInstance

- Vala basiert auf GObject

Ausblick

- GModule - Ein Pluginsystem
- GIO - Ein- und Ausgabe Funktionalitäten
- Testing-Framework (gtester, Assertions)
- Vorgefertigte Makros jeglicher Art (ByteOrder, Pointer-to-Int, ...)

Quellen

- http://developer.gnome.org/glib/stable/
  (Stand: 04.06.2014)
- http://book.huihoo.com/gtk+-gnome-application-development/ggad.html
  (Stand: 04.06.2014)
- http://docs.huihoo.com/symbian/nokia-symbian3-developers-library-v0.8/GUID-94D67092-5EB3-4D83-A164-CA628F2E2DB0.html
  (Stand: 04.06.2014)
- http://de.wikipedia.org/wiki/GObject
  (Stand: 03.06.2014)