Hello World

... in C

hello.c

```c
01 #include <stdio.h>
02
03 int main (int argc, char* argv[]) {
04     printf ("hello world.\n");
05     return 0;
06 }
```

```
$ gcc -o hello hello.c
$ ./hello
hello world.
$
```

... in vala

hello.vala

```vala
01 int main (string[] args) {
02     stdout.printf ("hello world.\n");
03     return 0;
04 }
```

```
$ valac hello.vala
$ ./hello
hello world.
$
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

... ein kleiner Vorgeschmack

main.vala

```
01 namespace My {
02
03    public class Main : Object {
04
05       // PROPERTY
06       public string message { get; set; default="hello world.\n"; }
07
08       public void hello () {
09          stdout.printf ("%s", this.message);
10       }
11
12       public static void main (string[] args) {
13          Main m = new Main ();
14          m.hello ();
15
16          m.message = "goodbye world.\n";
17          m.hello ();
18       }
19    }
   }
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Wert Typen

... erst einmal klein anfangen ...

vala	glib	C
char, uchar unichar	gchar, guchar gunichar	char, unsigned char
short, ushort int, uint long, ulong	gshort, gushort gint, guint glong, gulong	short, unsigned short int, unsigned int long, unsigned long
int8, uint8 int16, uint16 int32, uint32 int64, uint64	gint8, guint8 gint16, guint16 gint32, guint32 gint64, guint64	int8_t, uint8_t * int16_t, uint16_t int32_t, uint32_t int64_t, uint64_t
size_t, ssize_t	gsize, gssize	size_t, ssize_t **
float, double	gfloat, gdouble	float, double
bool true | false	gboolean TRUE | FALSE	

- in: <stdint.h> ** in: <sys/types.h>

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Arrays

Deklaration

- *typ*[ ] *name*;

Initialisierung

- int a[ ] = new int[4];
- int b[ ] = { 1, 2, 3 };

Größenänderung

- array.resize(n)
- array += element;

"Slicing"

von start (inklusiv) bis end (exklusiv):

- array[start:end]

Iteration

- foreach (... in ...)

array.vala

```
01 void print (int[] array, string name) {
02     stdout.printf ("%s:\n", name);
03     foreach (int i in array)
04         stdout.printf (" %d\n", i);
05 }
06
07 int main (string[] args) {
08     int[] array = new int[200];
09     print (array[0:4], "array");
10
11     int[] array2 = { 1, 2 };
12     array2 += 3;
13     print (array2, "array2");
14
15     array2.resize (1);
16     print (array2, "array2");
17
18     return 0;
19 }
20
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Strings

Spezielle Varianten

- verbatim string: """ ... """
- template string: @" ... $ .... "

Operatoren

- Konkatenation: +
- Vergleich: == != < >
- Suche: ... in ...

Properties

- length
- data

(einige) Methoden

- up ()
- down ()
- split (...)

string.vala

```
01 int main (string[] args) {
02   stdout.printf ("Please enter your name: ");
03   string name = stdin.read_line();
04
05   if (name == "cpu")
06     stdout.printf ("Oh, no, that's me!\n");
07   else {
08     stdout.printf ("%s", """Hello """"
09       + name.up() + """"\n"""
10       + "\n" + "nice \\n, isn't it ? \n"
11       + " ... as you see verbatim strings"
12       + "really seem to work ...\n");
13     stdout.printf ("%s",
14       @"Goodbye $(name.down())\n");
15   }
16
17   return 0;
18 }
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Programmieraufgabe III - revisited

Es soll ein Programm words geschrieben werden, das eine von der Standardeingabe gelesene Eingabe in einzelne Wörter zerlegt. Diese Wörter sollen dann Zeilenweise ausgegeben werden.

Auch hier soll die Aufrufsyntax denkbar einfach sein:

words

Folgende Anforderungen sollen dabei erfüllt werden:

Als ein "Wort" soll wie in der Aufgabe zuvor die jeweils die längste zusammenhängende Folge von Eingabezeichen gelten, die ausschließlich aus alphabetischen Zeichen besteht.

Neben dem Wort selbst soll auch die Zeile, auf der es gefunden wurde ausgegeben werden.

Tipp

zum Trennen von Strings kann split() verwendet werden:

```
public string[] split (string delimiter, int max_tokens = 0)
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Lösungsvorschlag

words.vala

```
01 int main (string[] args) {
02
03    var line_number = 0;
04    string line;
05
06    while (true) {
07       line = stdin.read_line ();
08       if (line == null)
09          break;
10
11       ++ line_number;
12       string[] words = line.split (" ");
13       foreach (string s in words) {
14          if (s == "")
15             continue;
16          stdout.printf ("%d: %s\n", line_number, s);
17       }
18    }
19    return 0;
20 }
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Klassen und Objekte

Deklaration

- **class** Klasse : BasisKlasse

Zugriffsmodifikatoren

... die üblichen Verdächtigen:

- **public**
- **private**
- (**protected**)

Konstruktor(en)

- Klasse ()
- Klasse.name ()
  üblicherweise with_...

Destruktor

- ~Klasse ()

abstrakte Klassen / Methoden

- Deklaration mit **abstract**
- Implementierung mit **override**

zoo.vala

```
01 public abstract class Animal : Object {
02
03    protected string name = "unknown";
04
05    public Animal () {
06       stdout.printf ("new animal\n");
07    }
08
09    ~Animal () {
10       stdout.printf ("animal destroyed\n");
11    }
12
13     public abstract void hello ();
14 }
15
16 public class Dog : Animal {
17
18     public Dog.with_name (string name) {
19         this.name = name;
20     }
21
22     public override void hello () {
23         stdout.printf ("%s: woof!\n",  name);
24     }
25 }
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

manuelle Erstellung im Vergleich

Typ-Definition

A: Klassen-Struktur B: Klassen-Struktur C: Makros

typ.h

A
```
typedef struct {
  parent parent;
} typ;
```

B
```
typedef struct {
  parentClass parent_class;
} typClass;
```

C
```
#define TYPE_typ
#define typ_GET_CLASS(instance)

#define typ(instance)
#define typ_CLASS(class)

#define IS_typ(instance)
#define IS_typ_CLASS(class)
```

Typ-Registrierung

D: Anmeldung beim Typ-System

typ.c

D
```
G_DEFINE_TYPE (typ, typ, G_TYPE_OBJECT)
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Der Zoo in Aktion

```
$ ./zoo
new animal
new animal
unknown: woof!
Hasso: woof!
animal destroyed
animal destroyed
$
```

zoo.vala

```
27 int main (string[] args) {
28
29    Animal[] zoo = {};
30    zoo += new Dog ();
31    zoo += new Dog.with_name ("Hasso");
32
33    foreach (Animal a in zoo) {
34       a.hello ();
35    }
36
37    return 0;
38 }
```

Programmieraufgabe

Dem Zoo sollen weitere Tiere hinzugefügt werden.

Wenigstens ein weiteres (sprechendes) Tier soll in den Zoo aufgenommen werden und sich dabei mit möglichst typischen Laut bemerkbar machen.

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Zuwachs im Zoo

Der Direktor erwartet eine ganze Reihe nicht-sprechender Tiere. Um diese mit weniger Arbeit aufnehmen zu können wünscht er sich eine kleine Änderung.

Ausprobier-Aufgabe

Die bisherige Zoo-Implementierung soll an die Wünsche des Direktors angepasst und in diese die neue Tierart Schmetterling aufgenommen werden.

Was funktioniert?
... und was geht dabei schief?

zoo2.vala

```
01 public abstract class Animal : Object {
02
03   protected string name = "unknown";
04
05   public Animal () {
06     stdout.printf ("new animal\n");
07   }
08
09   ~Animal () {
10     stdout.printf ("animal destroyed\n");
11   }
12
13   public abstract void hello () { }
14 }
```

(In Zeile 13 ist „abstract" durchgestrichen.)

```
16 public class Dog : Animal {
17
18   public Dog.with_name (string name) {
19     this.name = name;
20   }
21
22   public override void hello () {
23     stdout.printf ("%s: woof!\n", name);
24   }
25 }
```

(In Zeile 22 ist „override" durchgestrichen.)

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

... mit .. wäre das nicht passiert

zoo2.vala

Bindung in vala

frühe Bindung

solange nicht anders angegeben

- immer !

späte Bindung

nur wenn explizit angefordert

- Schlüsselwort **virtual**
- **override** zum überschreiben

```
01 public class Animal : Object {
02 
03    protected string name = "unknown";
04 
05    public Animal () {
06       stdout.printf ("new animal\n");
07    }
08 
09    ~Animal () {
10       stdout.printf ("animal destroyed\n");
11    }
12 
13    public virtual void hello () { }
14 }
15 
16 public class Dog : Animal {
17 
18     public Dog.with_name (string name) {
19         this.name = name;
20     }
21 
22     public override void hello () {
23         stdout.printf ("%s: woof!\n",  name);
24     }
25 }
```

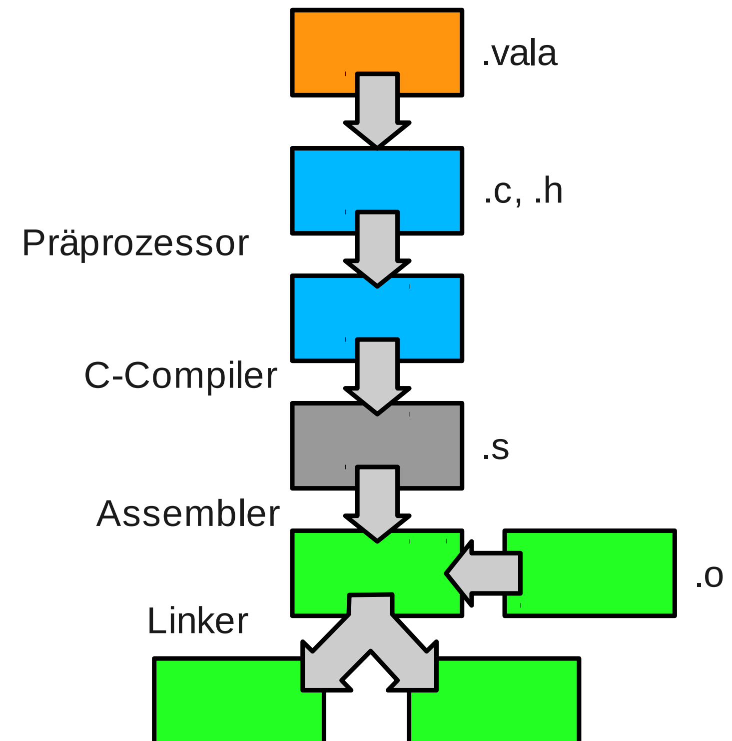
To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

valac – Bedienung und Funktionsweise

Arbeitsweise

... mit weiterem Übersetzungsschritt

- Transformation von .vala in C-Quellcode
- dann: Übersetzung mit C-Compiler



Übersetzen

eine Quelldatei

valac source.vala

mehrere Quelldateien

valac source-1.vala source-2.vala ...

Schalter

-o out-name	Ausgabedatei mit Namen out-name erzeugen
-C	C-Quellcode erzeugen
-H header.h	C-Header header.h erzeugen

PUBLIC DOMAIN
To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Properties

Worum geht's ?

Zur Erinnerrung

"Eigenschaften" eignes Objekts

GtkButton* button

"focus-on-click"	gboolean
"image"	GtkWidget*
"image-position"	GtkPositionType
"label"	gchar*
"relief"	GtkReliefStyle
"use-stock"	gboolean
"use-underline"	gboolean
"xalign"	gfloat
"yalign"	gfloat

Zugriff

Klassisch

- gtk_button_set_focus_on_click (button, TRUE)
- bool = gtk_button_get_focus_on_click (button)
- ...

gobject

- g_object_set (button, ..., NULL)
- g_object_get (button, …, NULL)

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Properties – manuelle Erstellung

Typ-Definition

A: eindeutig Numerieren B / C: getter/setter implementieren
D: Property anmelden + getter/setter der Basisklasse überschreiben

typ.c

A	`enum {` `  ...`

B	`static void typ_set_property (GObject *object, guint property_id,`
	`                   const GValue *value, GParamSpec *pspec) {`
	`  switch (property_id) {`
	`    ...`

C	`static void typ_get_property (GObject *object, guint property_id,`
	`                   GValue *value, GParamSpec *pspec) {`
	`  switch (property_id) {`
	`    ...`

D	`static void typ_class_init (typClass *class) {`
	
	`  GParamSpec *pspec = ...`
	`  g_object_class_install_property ( ... );`
	
	`  G_OBJECT_CLASS (class)->get_property = typ_get_property;`
	`  G_OBJECT_CLASS (class)->set_property = typ_set_property;`
	`}`

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Properties – mit vala

Erstellen

- manuell:
  set / **get** Block
- automatisch:
  set; / **get;**

Implementierung

- **set** nutzt Paramter **value**
- Vorgabewert **default**

Verwenden

- wie gewöhnliches Feld:
  instanz.property

prop.vala

```
01 public class Main : Object {
02
03   private int _percent = 0;
04
05   public int percent {
06     get { return _percent; }
07     set { if (value >= 0 && value <= 100)
08             _percent = value;
09     }
10   }
11
12   public int any_number { get; set; default=0; }
13
14   [Description(nick = "counter",
15                blurb = "just a counter, right?")]
16   public int counter { get; private set; default=0; }
17
18   public void inc () {
19     ++counter;
20   }
21 }
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Rückblick auf das Fahrzeug-Beispiel

... und das wäre Ihr Preis gewesen:

fahrzeug.vala

```
01 public class Fahrzeug : Object {
02     private int _speed = 0;
03 
04     [Description(nick = "Speed",
05                  blurb = "Die Geschwindigkeit des Fahrzeugs")]
06     public int speed {
07         get { return _speed; }
08         set { if (value >= 0 && value <= 300)
09                 _speed = value;
10         }
11     }
12 }
```

Übersetzen mit:

valac -c -H fahrzeug.h
gcc $(pkg-config –cflags –libs gtk+-2.0) fahrzeug.vala.o main.c -o main

... fertig !!

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Signale

Worum geht's

GtkButton* button;

"activate"
"clicked"
"enter"
"leave"
"pressed"
"released"

Zur Erinnerung

"Sockel" für Aktionen

Verbinden

manuell

- g_signal_connect (button, "clicked", G_CALLBACK (button_clicked), NULL)

GtkBuilder

- gtk_builder_connect_signals (builder, NULL);

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Signale

Deklarieren

- Schlüsselwort **signal**

Verbinden

- instanz.signal.**connect** (handler)
- instanz.signal **+=** handler

Trennen

- instanz.signal.**disconnect** (handler)
- Instanz.signal **-=** handler

Lambda Ausdrücke

- (p, ...) => { ... }

Typ-Inferenz

... auch bei lokalen Variablen:

- **var** number = 10;

signal.vala

```
01 public class Main : Object {
02   public signal void hello (string message);
03 }
04
05 void my_handler (string s) {
06   stdout.printf ("my_handler: %s\n", s);
07 }
08
09 int main (string[] args) {
10   Main m = new Main ();
11
12   m.hello += my_handler;
13   m.hello.connect ((s) =>
14     {stdout.printf ("lambda_handler: %s\n", s);});
15   m.hello ("message_1");
16
17   m.hello -= my_handler;
18   m.hello ("message_2");
19
20   return 0;
21 }
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

GTK mit vala

main.vala

```
01 using Gtk;
02
03 int main (string[] args) {
04
05    Gtk.init (ref args);
06
07    var window = new Window ();
08    window.destroy.connect (Gtk.main_quit);
09
10    var vbox = new VBox (true, 4);
11    window.add (vbox);
12    var label = new Label ("hello world.");
13    vbox.add(label);
14
15    var button = new Button.from_stock ("gtk-ok");
16    vbox.add(button);
17    button.clicked.connect ( () => {label.label = "goodbye.";} );
18
19    window.show_all ();
20
21    Gtk.main ();
22
23    return 0;
24 }
```

Übersetzen mit:

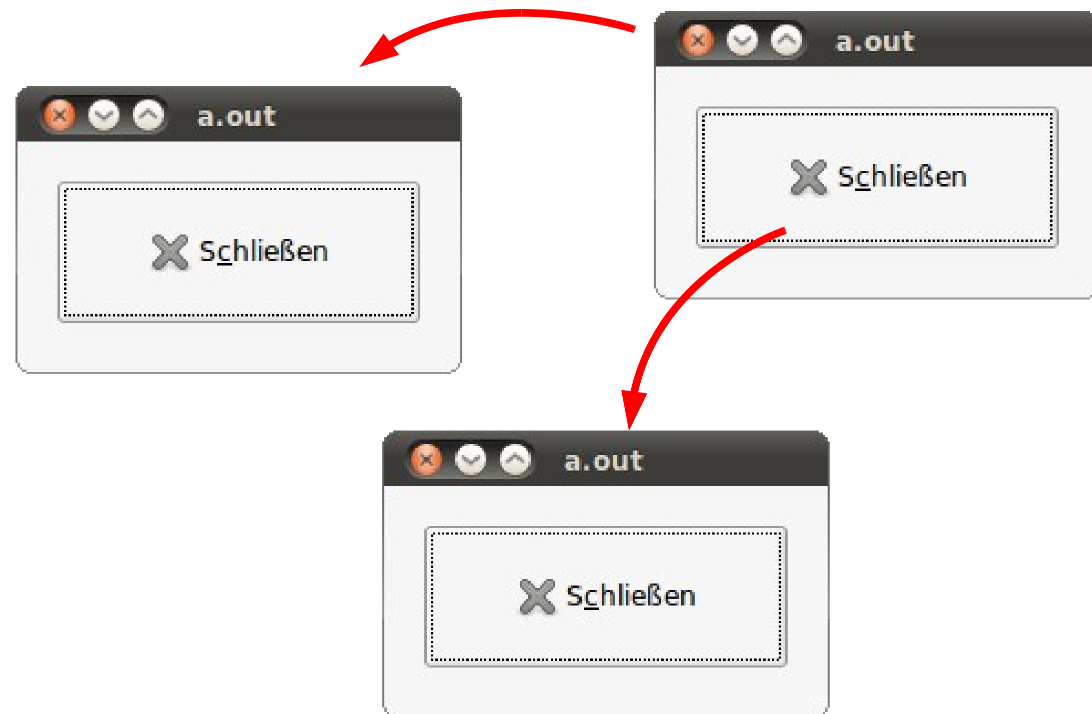
```
valac  --pkg=gtk+-2.0 main.vala
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Spawn-Window - revisited

Programmieraufgabe

Das Programm spawn window soll unter zu Hilfenahme von vala neu implementiert werden.



PUBLIC DOMAIN
To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.

Lösungsvorschlag

spawn.vala

```
01 using Gtk;
02
03 void spawn () {
04     var window = new Window ();
05     window.border_width = 8;
06     window.destroy.connect (Gtk.main_quit);
07
08     var button = new Button.from_stock ("gtk-close");
09     button.clicked.connect (spawn);
10
11     window.add (button);
12     window.show_all ();
13 }
14
15 int main (string[] args) {
16
17     Gtk.init (ref args);
18
19     spawn ();
20
21     Gtk.main ();
22     return 0;
23 }
```

To the extent possible under law, the person who associated CC0 with this work has waived all copyright and related or neighboring rights to this work. This work is published from: Germany.