Go in Linux Desktop Environment

2015年4月

夏彬
武汉深之度

Linux桌面环境

- Gnome (Gtk)
- KDE (Qt)
- Lxde (Gtk) LxQt(Qt5)
- Xfce (Gtk)
- DDE (Gtk)
- .....

DE组件

- Windows Manager
- Input Method
- Launcher
- Panel (dock)
- Session Manager
- Appearance
- Configure Modules
- many service daemon, like

```
audio, network, volume, power, login
```

- Basic Applications

```
Document Reader、Browser、File Manager
```

Why golang?

- 简洁一致
- 开发效率与运行效率间的平衡
- 编译型语言相对来说更适合长期项目

Question: DE是否应该努力追求本身独立?

- 可以跑在更多发行版上
- 更多的受众，更多的反馈
- 但DE最终会与底层服务甚至特定版进行交互，完全独立很难走到完美。
- 造成更多的开发包袱

Linux不会被统一,所以DE最好还是拥有更好的内聚性。

但应该打包更多的系统在一起, 从更多的地方优化用户体验，无折腾。

```
基础文件系统 + 仓库 + DE + 软件商店(各种应用软件才是正常用户需要的,基础设施不应该放到用户面前)
```

相关实验性项目

```
Gnome SDK + sandboxed appliations
Deepin XX ?
```

遇到的问题

- 与其他组件的融合
- X11

解決方式

- using CGO to get anything
- Pure golang library

CGO Tips

- Go-C
- C-Go
- struct
- array

CGO Tips: Go call C functions && types convert

```
package main

//#include <stdlib.h>
//#include <time.h>
import "C"
import "fmt"

func main() {
    s := C.time(nil)
    C.srandom(C.uint(s))
    r := C.random()
    fmt.Println(r)
}
```

Run

Easy to use. Just using the **C** pseudo package.

The code of glue-C is far less than perl、python ...

CGO Tips: Even more convenient than C at make time

```
package main

/*
#cgo pkg-config: gtk+-3.0
#include <gtk/gtk.h>
void write_c_function_in_go_source(GtkWidget* w)
{
    GtkWidget* color = gtk_color_chooser_widget_new();
    gtk_window_set_position(GTK_WINDOW(w), GTK_WIN_POS_CENTER);
    gtk_container_add(GTK_CONTAINER(w), color);
}
*/
import "C"

func main() {
    C.gtk_init(nil, nil)
    w := C.gtk_window_new(C.GTK_WINDOW_TOPLEVEL)
    C.write_c_function_in_go_source(w)
    C.gtk_widget_show_all(w)
    C.gtk_main()
}
```

Run

CGO Tips: Struct is also easy to use

```
package main

/*
#include <time.h>
#include <stdlib.h>
struct tm* tms[2];
void init_array_times()
{
    struct tm* local, *gmt;
    time_t now = time(0);
    tms[0] = gmtime(&now);
    tms[1] = localtime(&now);
}
*/
import "C"
import "fmt"

func main() {
    C.init_array_times()
    fmt.Println("Tody is", C.tms[1].tm_year, C.tms[1].tm_mon+1, C.tms[1].tm_mday+1)
}
```

Run

CGO Tips: Passing array type is annoying

```
package main

/*
#include <time.h>
#include <stdlib.h>
struct tm** tms;
void init_array_times()
{
    struct tm* local, *gmt;
    tms = malloc(sizeof(struct tm*) * 2);
    time_t now = time(0);
    tms[0] = gmtime(&now);
    tms[1] = localtime(&now);
}
*/
import "C"
import "fmt"

func main() {
    C.init_array_times()
    fmt.Println("This will failed", C.tms[1].tm_year)
}
```

Run

CGO Tips: Passing array type is annoying

convert ****struct_tm** to ***[size]*C.struct_tm**

```
func main() {
    C.init_array_times()
    tms := (*[4](*C.struct_tm))(unsafe.Pointer(C.tms))
    fmt.Println("type of C.tms:", reflect.TypeOf(C.tms))
    fmt.Println("type of tms:", reflect.TypeOf(tms))
    fmt.Println("This will OK", tms[0].tm_year, len(tms))
}
```

Run

Why we choose **4** as the **array length**

CGO Tips: Passing array type is annoying

In practice, the array length must be given at compile time.

we can simply use a larger one and make a slice value, like

```
n := calacLength(C.tms)
tms := (*[1<<12](*C.struct_tm))(unsafe.Pointer(C.tms))[:n:n]
```

CGO Problems: We have more troubles

- GC

```
C.free(unsafe.Pointer(foo)) // malloc and free

foo.Ref() and foo.Unref() // C library reference system
```

ease the pain

```
runtime.SetFinalizer(obj interface{}, finalizer interface{})

runtime.GC()
```

- thread safe

```
goroutine with GUI (main loop、work loop)

goroutine with POSIX thread
```

CGO: automatically generate code

swig (http://www.swig.org/Doc2.0/Go.html)

go-cxxdict (https://github.com/sbinet/go-cxxdict)

gir-generator (https://github.com/snyh/gir-generator)

GIR: GObject Introspection repository (GIR)

The middleware layer between GObject libraries and language bindings.

After the first FooObject built, all C libraries which based on GObject
can be directly used by the Foo Language users.

The first guy in Golang is

nsf/gogobject (https://github.com/nsf/gogobject)

the first line of *README*

```
WARNING! This project is no longer maintained. Probably doesn't even compile.
```

And it's true. There has an workable and improved version.

snyh/gir-generator (https://github.com/snyh/gir-generator)

GIR: How GIR does this ?

- They has defined a *IDL* which is worked well with GObject.
- They offer the libgirepository.so to read the *IDL* file.
- Using the information offered by *IDL* file, you can communicate with any libraries based on GObject.

Bindings type
- static binding
- dynamic *binding*

GIR: How many GIR libraries we can use?

```
apt-file search /usr/lib/girepository-1.0/ | awk '{print $1}'  | uniq | wc
171     171    3513
```

- gstreamer
- appindicator
- anjuta
- atspi
- clutter
- gudev
- webkit
- wnck
- freedesktop (freetype2、xlib、fontconfig、DBus、xrandr)
- all libraries based on GObject (use *g-ir-scanner* to generate the IDL file)

GIR: use gir udev to inspect device information

```
package main

import "fmt"
import "pkg.linuxdeepin.com/lib/gudev"
import "pkg.linuxdeepin.com/lib/glib-2.0"

func main() {
    c := gudev.NewClient(nil)
    c.Connect("uevent", func(client *gudev.Client, action string, dev *gudev.Device) {
        fmt.Println("A:", action, "D:", dev.GetSysfsPath(), "Brightness:", dev.GetSysfsAttr("bri
ghtness"))
    })

    all := c.QueryBySubsystem("backlight")
    for _, bl := range all {
        fmt.Printf(`name: %v
path: %v
brightness: %v
type:%v

`, bl.GetName(), bl.GetSysfsPath(), bl.GetSysfsAttr("brightness"), bl.GetSysfsAttr("type"))
    }
    glib.StartLoop()
}
```

Run

GIR: create the new gudev binding in 5 minutes

```
{
    "namespace": "GUdev",
    "version": "1.0"
}
```

```
package gudev
/*
#include "gudev.gen.h"
#cgo pkg-config: gudev-1.0
*/
import "C"
import "unsafe"
import (
    "pkg.linuxdeepin.com/lib/gobject-2.0"
)
[<.go_utils_no_cb>]
[<.go_bindings>]
```

and run

```
gir-generator gudev.go.in
```

the result is an usable go package.

Pure golang packages:

X Go Binding

xgb (https://github.com/BurntSushi/xgb)

native Go client bindings for D-Bus message bus system

go.dbus (https://github.com/guelfey/go.dbus)

What is D-Bus ?

- a message bus system.
- advanced IPC
- support Method, Property and Signal
- support ACL
- automatically start services
- supported by GNOME, KDE, Systemd, upstart ...

D-Bus FreeDesktop (http://dbus.freedesktop.org)

others Kernel "dbus-like" code for the Linux kernel (https://github.com/gregkh/kdbus)

go-dbus: The Easy Frame-less D-Bus Go Interface

go-dbus (https://gitcafe.com/Deepin/go-lib/tree/master/dbus) us go.dbus as transport layer.

- Automatically export Properties/Methods/Signals from Go to D-Bus.
- Automatically update property value with other system, like GSettings.

go-dbus: example dbus-services, define service

```
type DService struct {
    Name    string
    Version string

    Enabled bool `access:"readwrite"`

    Changed func(int64) //signals

    service []string
}
```

go-dbus (https://gitcafe.com/Deepin/go-lib/tree/master/dbus) us go.dbus as transport layer.

go-dbus: example dbus-services, install service

```
//give the dbus service, path and interface information
func (*DService) GetDBusInfo() dbus.DBusInfo {
    return dbus.DBusInfo{
        "org.snyh.test",
        "/org/snyh/test",
        "org.snyh.test",
    }
}
```

```
func main() {
    d := &DService{"TestSerivce", "0.1", false, nil, []string{"Close", "Check"}}

    if err := dbus.InstallOnSession(d); err != nil {
        fmt.Println("Install failed:", err)
        return
    }
    if err := dbus.Wait(); err != nil {
        fmt.Println("Bus error:", err)
        return
    }
}
```

Run

go-dbus: defect

- lost struct filed name (D-Bus caused)
- lost argument name (go-dbus caused)

```
reflect package doesn't support query arguments name of a function at runtime.
```

Should we using *go generate* to generate stub code at build time ?

dbus-generator: Generate D-Bus binding codes

It can generate dbus-binding for QML/PyQt/Golang.

https://gowalker.org/github.com/linuxdeepin/go-dbus-generator

https://github.com/linuxdeepin/dbus-factory

Other Go Project in Deepin

- deepin id (online services for linux Desktop)
- Go for loongson (https://github.com/emb-team/loongson-golang)
- many IT infrastructures for internal use

hack project

dui (https://github.com/snyh/dui)

- webkit without javascript runtime
- golang play with DOM directly
- small memory footprint

dui: How it look like ?

```
    hello := f.NewElement("input")
    hello.Set("type", "button")
    hello.Set("value", "hello")
    hello.Set("id", "hello")
    f.Add(hello)

    txt.Set("style", "text-shadow: 2px 2px 2px red; font-size: 18px; -webkit-transform: rotate(-
20deg);")
    txt.SetContent("DUI Test")

    var flag bool
    hello.Connect("click", func(e dui.MouseEvent) {
        fmt.Printf("x:%d, y:%d\n", e.X, e.Y)
        flag = !flag
        if flag {
            txt.SetContent("one two three")
            println(txt.Content())
        } else {
            txt.SetContent("three two one")
            println(txt.Content())
        }
        runtime.GC()
    })
```

Run

湖北的同学们可以回家啦

Go to 武汉

Deepin 武汉光谷金融港

http://deepin.org

Slide源码 (https://github.com/snyh/slide/tree/master/gopherchina2015)

Thank you

2015年4月
Tags: golang, deepin, DE, linux (#ZgotmplZ)

夏彬
武汉深之度
snyh@snyh.org (mailto:snyh@snyh.org)