LiNUX ALTiNDA GRAFiK PROGRAMLAMA TEKNiKLERi

Önsöz

Linux atında yazılım geliştirmeyi düşünüyorsunuz veya geliştiriyorsunuz.Artık programlarınızı görsel bileşenler ile süslemek istiyorsanız bu doküman sizlere hitap ettiğini düşünüyorum.Bu dokümanı ele alırken linux altında 3 tane çok kullanılan grafik programlama birimi olduğunu belirtmek istiyorum.

1-) X Window API'lerini kullanabilirsiniz.Bu biraz daha karışık yöntem.Aslında diğer tüm grafik arabirimleri bu API'lerini daha anlaşılır bir şekilde kullanılmasında ibarettir.

2-) GTK Arabirimi linux üzerinde en çok tercih edilen görsel geliştirme araçlarından biridir.Çoklu dil desteği mevcuttur.C,C++, Guile, Perl, Python, TOM, Ada95, Free Pascal ve Eiffel diyebilirim.GTK arabiriminin kullanımının en iyi örneklerinden biri GNOME masaüstü diyebilirim.GNOME masaüstü tamamen GTK arabirimi ile yazılmıştır.

3-) Qt Arabirimi linux üzerinde çokça kullanılan özellikle C++ ile yazılım geliştirenlerin kullandıkları bir arabirimdir.Qt kütüphanesi ile KDE masaüstü yazılmıştır.

Bu dokümanın ilk önce GTK ile program geliştirmeyi sizlere açıklayacağım. İlerleyen Bölümlerde Qt ara birimini ve yeri geldiğince X windows apilerinden bahsetmeye çalışacağım.

iskender atasoy

GTK Arabirimi ile Program Geliştirme

Gtk ara birimi program geliştiricilerin en çok tercih ettikleri ara birimlerden biridir .Başlıca nedenleri:
Hızlıdır çünkü tamamen C dili ile yazılmıştır. Kolay anlaşılır bir yapısı mevcuttur. Ve geliştirmeye açık bir yapısı mevcuttur. Ayrıca GTK ara birimi üzerinde kolay program geliştirmek için Glade adında görsel bileşen tasarımı yapabileceğiniz bir program geliştirme aracıda mevcuttur. Glade sadece programınızın grafik ara birimini oluşturabiliyorsunuz. Kodlama için en çok tercih edilen editör Anjuta'dır. Kodlarımızı yazacağımız editor olarak ***emacs*** editörünü tercih etmekle birlikte her hangi bir editör kullanabilirsiniz. Daha fazla bilgi için GTK'nın resmi web sitesine www.gtk.org 'a gidebilirsiniz.
GTK ile program yazarken yaygın olarak kullanılan GLIB kütüphanesi sizin için oldukça faydalı olacağına inanmaktayım.GLIB kütüphanesi içimde kanal programlama fonksiyonları,bellek,veri yapıları ve kritik ve özel fonksiyonları ve makrolar içerir.GTK içinde tümleşik olarak veya kendi konsol uygulamalarında kullanabileceğiniz bir kütüphandedir. İçerisinde yararlı fonksiyonlar bulunmaktadır.

GLIB Kütüphanesi

Glib kütüphanesinde yazılım geliştirirken ilk önce glib içinde kullanılan veri tiplerine bakmamız gerekiyor.

Glib Nasıl Derlenir ?

Glib kütüphanesinde yazdığınız kodu unix ortamlarda pkg-config arabirimi ile derleyebilirsiniz.

```
$ pkg-config --cflags glib-2.0
-I/usr/include/glib-2.0 -I/usr/lib/glib-2.0/include
$ pkg-config --libs glib-2.0
-lglib-2.0
```

Eğer diğer Glib modüllerini derlemek istiyorsak ör: kanal eklemek vb.

```
$ pkg-config --cflags --libs gmodule-2.0
$ pkg-config --cflags --libs gthread-2.0
$ pkg-config --cflags --libs gobject-2.0
```

tarzında ekleyebiliriz.

En basit Glib kodunu derleyebilmek için

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

tarzında derleyebiliriz.

Ayrıca ***pkg-config --list-all*** komutu ile diğer kütüphaneleride göre bilirsiniz.

Basit Tipler:

Bunlar Glib kütüphanesinde tanımlanmış olan tip tanımlamalarıdır.Çoğu Standard C'de kullanılmak ile birlikte yeni tür tanımlamalarıda yapılşmıştır.

```
#include <glib.h>

typedef gboolean;
typedef gpointer;
typedef gconstpointer;
typedef gchar;
typedef guchar;

typedef gint;
typedef guint;
typedef gshort;
typedef gushort;
typedef glong;
typedef gulong;

typedef gint8;
typedef guint8;
typedef gint16;
typedef guint16;
typedef gint32;
typedef guint32;

#define G_HAVE_GINT64
typedef gint64;
typedef guint64;
#define G_GINT64_CONSTANT (val)

typedef gfloat;
typedef gdouble;

typedef gsize;
typedef gssize;
```

Açıklama :

Glib kütüphanesi ile 4 çeşit veri tipi kullanılmaktadır.Bunlar:
1-) Yeni Veri Türleri Standard C'nin bir parçası değillerdir: gboolean, gsize, gssize.
2-) Platform Bağımsızlığı için tanımlanmış sayısal veri tipleridir .Bu sayede 16,32 veya 64 bitlik platformlar için tanımladığınız veri boyutları sabit olacaktır. Bunlar : g***int8, guint8, gint16, guint16, gint32, guint32, gint64, guint64.***
3-) C dilinde bir parçası olan ve Okunulabilirliği arttırmak için tanımlanmış olan veri tipleridir.
Bunlar : ***gpointer, gconstpointer, guchar, guint, gushort, gulong.***
4-) C dilinde bir parçası olan ve Glib'de yeniden tür dönüşümü yapılan C tipleridir.
Bunlar : ***gchar, gint, gshort, glong, gfloat, gdouble.***

Veri Tipleri ve Detayları

gboolean

typedef gint gboolean;
Standart TRUE veya FALSE ifadesidir.Bunlarda makro olarak tanımlanmış değişkenlerdir.Yani
#define FALSE (0)
#define TRUE (!FALSE)
tarzında glib kütüphaneside tanımlanmışlardır.

gpointer

typedef void* gpointer;
Bu tamamen okunulabilirliği arttırmak için oluşturulmuş olan void * türününde tipi olmayan göstericidir.Bu sayede gpointer türünde bir değişken tanımladığımızda bunun void * olduğunu kolaylıkla anlarız.

gconstpointer

typedef const void *gconstpointer;
Bu içeriği değiştirilemez olan void * türünde bir değişkeni temsil eder.

gchar

typedef char gchar;
Bu C deki char türünden başka birşey değildir.glib kütüphanesi ile yazılım geliştirirken okunulabilirliği arttırmak için bu türleri tercih etmenizde fayda var.

guchar

typedef unsigned char guchar;
C deki unsigned char karşılığı.

gint

typedef int gint;
C deki int tipinin karşılığıdır.Bunun sınırıda G_MININT ile G_MAXINT arasındadır.
#define G_MININT INT_MIN
#define G_MAXINT INT_MAX
tarzında tanımlanmıştır.Daha detaylı bilgi için limits.h inceleyebilirsiniz.

guint

typedef unsigned int guint;
C deki unsigned int tipinin karşıtıdır.Bunun sınırı ise 0 ile G_MAXUINT arasındadır.
#define G_MAXUINT UINT_MAX
tarzında tanımlanmıştır.Daha detaylı bilgi için limits.h inceleyebilirsiniz.

gshort

typedef short gshort;
C deli short tipinin karşılığıdır.Bunun sınırıda G_MINSHORT ile G_MAXSHORT arasındadır.
#define G_MINSHORT SHRT_MIN
#define G_MAXSHORT SHRT_MAX
tarzında tanımlanmıştır.Daha detaylı bilgi için limits.h inceleyebilirsiniz.

gushort

typedef unsigned short gushort;
C deki unsigned short tipinin karşılığıdır.Bunun sınırıda 0 ile G_MAXUSHORT arasındadır.
#define G_MAXUSHORT USHRT_MAX
tarzında tanımlanmıştır.Daha detaylı bilgi için limits.h inceleyebilirsiniz.

glong

typedef long glong;
C deki long tipinin karşılığıdır.Bunun sınırıda G_MINLONG ile G_MAXLONG arasındadır.
#define G_MINLONG LONG_MIN
#define G_MAXLONG LONG_MAX
tarzında tanımlanmıştır.Daha detaylı bilgi için limits.h inceleyebilirsiniz.

gulong

typedef unsigned long gulong;
C deki unsigned long tipinin karşılığıdır.Bunun sınırıda 0 ile G_MAXULONG arasındadır.
#define G_MAXULONG ULONG_MAX
tarzında tanımlanmıştır.Daha detaylı bilgi için limits.h inceleyebilirsiniz.

gint8

typedef signed char gint8;
İşaretsiz ve 8 bitlik sayı türüdür.Tüm platformlarda limitleri sabittir.Değer aralığı -127 ile 128 arasındadır.

guint8

typedef unsigned char guint8;
İşaretli ve 8 bitlik sayı türüdür.Tüm platformlarda limitleri sabittir.Değer aralığı 0 ile 255 arasındadır.

gint16

typedef signed short gint16;
İşaretli ve 16 bitlik sayı türüdür.Tüm platformlarda limitleri sabittir.Değer aralığı -32,768 ile 32,767 arasındadır.

guint16

typedef unsigned short guint16;
İşaretsiz ve 16 bitlik sayı türüdür.Tüm platformlarda limitleri sabittir.Değer aralığı 0 ile 65,535 arasındadır.

gint32

typedef signed int gint32;
İşaretli ve 32 bitlik sayı türüdür.Tüm platformlarda limitleri sabittir.Değer aralığı -2,147,483,648 ile 2,147,483,647 arasındadır.

guint32

typedef unsigned int guint32;
İşaretsiz ve 32 bitlik sayı türüdür.Tüm platformlarda limitleri sabittir.Değer aralığı 0 ile 4,294,967,295 arasındadır.

G_HAVE_GINT64

#define G_HAVE_GINT64 1 /* deprecated, always true */
Bu 64 bitlik işaretli ve işaretsiz sayıları sisteminizde desteklenip desteklenmediğine bakarız.

gint64

G_GNUC_EXTENSION typedef signed long long gint64;
İşaretli ve 64 Bitlik sayı türüdür.Aralığı ise -9,223,372,036,854,775,808 ile 9,223,372,036,854,775,807 arasındadır.

guint64

G_GNUC_EXTENSION typedef unsigned long long guint64;
İşaretsiz ve 64 Bitlik sayı türüdür.Aralığı ise 0 ile 18,446,744,073,709,551,615 arasındadır.

G_GINT64_CONSTANT()

#define G_GINT64_CONSTANT(val) (G_GNUC_EXTENSION (val##LL))
64 bitlik sayıları literatüre uygun bir şekilde kodumuza yerleştiririz.
val:uygun bir sayı ör: 0x1d636b02300a7aa7U.

gfloat

typedef float gfloat;
C deki float karşılığındadır.Bunun sınırıda G_MINFLOAT ile G_MAXFLOAT arasındadır.
#define G_MINFLOAT FLT_MIN
#define G_MAXFLOAT FLT_MAX
tarzında tanımlanmıştır.Daha detaylı bilgi için limits.h inceleyebilirsiniz.

gdouble

typedef double gdouble;
C deki double karşılığındadır.Bunun sınırıda G_MINDOULE ile G_MAXDOUBLE arasındadır.
#define G_MINDOUBLE DBL_MIN
#define G_MAXDOUBLE DBL_MAX
tarzında tanımlanmıştır.Daha detaylı bilgi için limits.h inceleyebilirsiniz.
gsize
typedef unsigned int gsize;
32 bit işaretli int değerindedir.
gssize
typedef signed int gssize;
32 bit işaretsiz int değerindedir.

Standard Makrolar

#include <glib.h>
#define GLIB_MAJOR_VERSION
#define GLIB_MINOR_VERSION
#define GLIB_MICRO_VERSION

#define G_OS_WIN32
#define G_OS_BEOS
#define G_OS_UNIX

#define GLIB_CHECK_VERSION (major,minor,micro)

#define G_DIR_SEPARATOR
#define G_DIR_SEPARATOR_S
#define G_SEARCHPATH_SEPARATOR
#define G_SEARCHPATH_SEPARATOR_S

#define TRUE
#define FALSE

#define NULL

#define MIN (a, b)
#define MAX (a, b)

#define ABS (a)
#define CLAMP (x, low, high)

#define G_STRUCT_MEMBER (member_type, struct_p, struct_offset)
#define G_STRUCT_MEMBER_P (struct_p, struct_offset)
#define G_STRUCT_OFFSET (struct_type, member)

#define G_MEM_ALIGN

#define G_CONST_RETURN

Açıklamalar:

GLIB_MAJOR_VERSION
#define GLIB_MAJOR_VERSION 2
Glib kütüphanesinin major versionu.

GLIB_MINOR_VERSION
#define GLIB_MINOR_VERSION 2
Glib kütüphanesinin minor versionu.

GLIB_MICRO_VERSION
#define GLIB_MICRO_VERSION 1
Glib kütüphanesinin mikro versionu.

G_OS_WIN32
#define G_OS_WIN32
Windows tabanlı kodlarda kullanabileceğimiz bir makrodur.
#ifdef G_OS_WIN32 tarzında bir kullanım mevcuttur.

G_OS_BEOS
#define G_OS_BEOS
BEOS tabanlı kodlarda kullanabileceğimiz bir makrodur.
#ifdef G_OS_BEOS tarzında bir kullanım mevcuttur.

G_OS_UNIX
#define G_OS_UNIX
UNIX tabanlı kodlarda kullanabileceğimiz bir makrodur.
#ifdef G_OS_UNIX tarzında bir kullanım mevcuttur.

GLIB_CHECK_VERSION()
#define GLIB_CHECK_VERSION(major,minor,micro)
Glib kütüphanesinin versi yonunu kontrol eder.Eğer istenilen versiyon numarası

veya bir üstü bulunursa TRUE döndürür.
Örneğin:
```
if(!GLIB_CHECK_VERSION(1,2,10))
{
g_error("Glib Versiyonunda 1.2.10 veya Üstü gerekiyor.");
exit(0);
}
```

G_DIR_SEPARATOR
#define G_DIR_SEPARATOR
Dizin ayırıcı karakteridir.Bu Unix makinelerde '/' ve Windows makinelerde ise '\' dir.

G_DIR_SEPARATOR_S
#define G_DIR_SEPARATOR_S
Dizin ayırıcı kelimesidir.Bu Unix makinelerde "/" ve Windows makinelerde ise "\" dir.

G_SEARCHPATH_SEPARATOR
#define G_SEARCHPATH_SEPARATOR
Aranan dizin ayırıcı karakteridir.Bu Unix makinelerde ':' ve Windows makinelerde ise ';' dir.

G_SEARCHPATH_SEPARATOR_S
#define G_SEARCHPATH_SEPARATOR_S
Aranan dizin ayırıcı kelimesidir.Bu Unix makinelerde ":" ve Windows makinelerde ise ";" dir.

TRUE
#define TRUE (!FALSE)
gboolean tipi için TRUE değeri içerir.

FALSE
#define FALSE (0)
gboolean tipi için FALSE değeri içerir.

NULL
#define NULL
Standart NULL değerini içirir.

MIN()
#define MIN(a, b) (((a) < (b)) ? (a) : (b))
a ile b arasında minumum olanı hesaplar.

MAX()
#define MAX(a, b) (((a) > (b)) ? (a) : (b))
a ile b arasında maksimum olanı hesaplar.

ABS()
#define ABS(a) (((a) < 0) ? -(a) : (a))
Mutlak değerini hesaplar.Yani
ABS(-139) =139
ABS(139) =139
şeklidedir.

CLAMP()
#define CLAMP(x, low, high) (((x) > (high)) ? (high) : (((x) < (low)) ? (low) : (x)))
x değerinin low ve high arasında olup olmadığına bakar.Yani
CLAMP(15,10,40) sonucu 15
CLAMP(45,10,40) sonucu 45
CLAMP(10,15,45) sonucu 15

G_STRUCT_MEMBER()
#define G_STRUCT_MEMBER(member_type, struct_p, struct_offset)
Verilen struct yapısından gereki offset ve tipteki elemanı dönürür.

G_STRUCT_MEMBER_P()
#define G_STRUCT_MEMBER_P(struct_p, struct_offset)
Verilen struct yapısının offset alanından Tipi olmayan (void *) bir gösterici geri döndürür.

G_STRUCT_OFFSET()
#define G_STRUCT_OFFSET(struct_type, member)
Verilen struct yapısından verilen byte'lardan offset oranını döndürür.

G_MEM_ALIGN
#define G_MEM_ALIGN
Mevcut platformda ayrılan byte sayını gösterir.

G_CONST_RETURN
#define G_CONST_RETURN
Eger G_DISABLE_CONST_RETURNS tanımlı ise bu makro hiçbir işlev yapmaz.Bu makro ile const değerlere ulaşırız.Yani geri dönüş değeri const olan fonksiyonlarda geri dönüş değerlerinde işlem yapmak için kullanışlıdır.