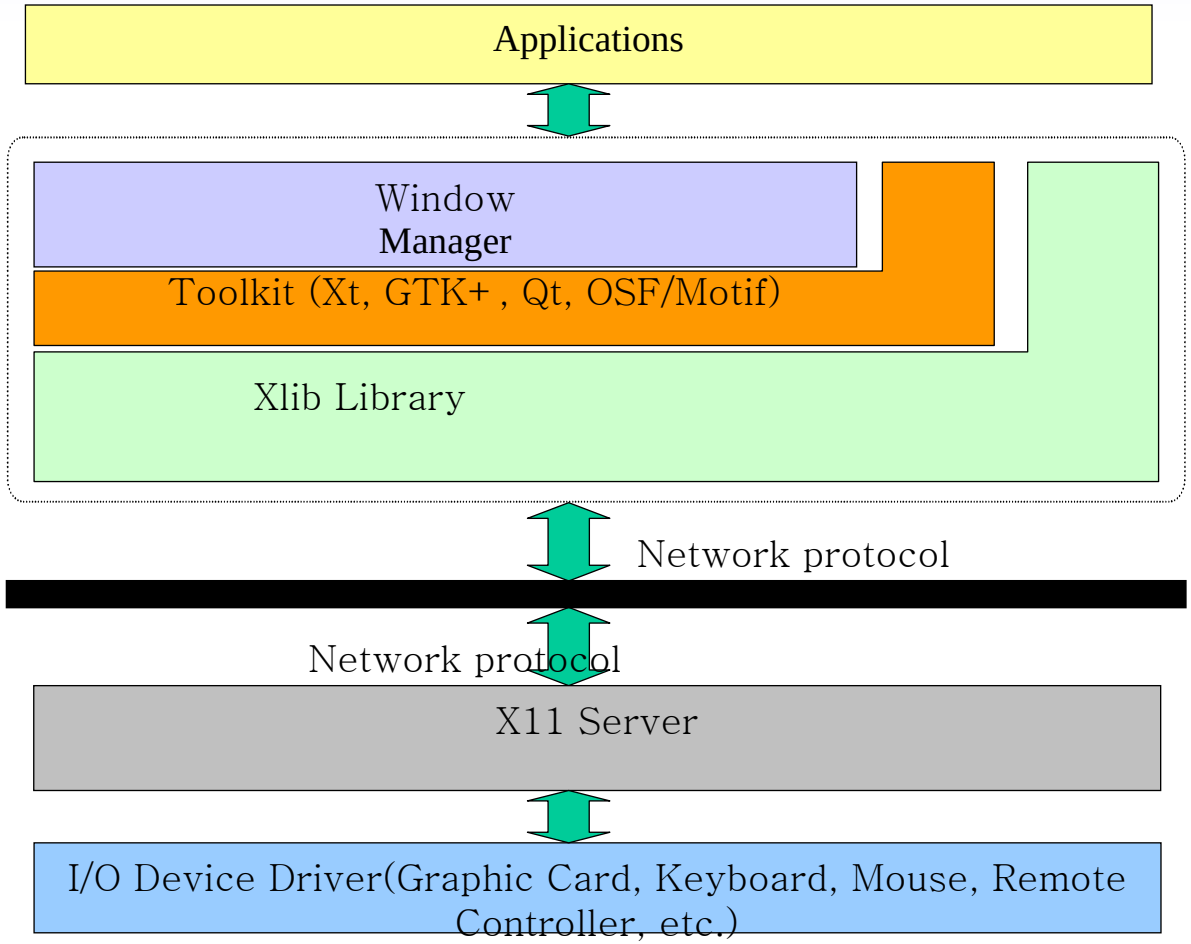
GUI 프로그래밍

GUI System Model

- 단일 프로세스 모델
  - 한 프로세스가 모든 그래픽 장치를 전용해서 사용
  - 속도가 빠르지만, 다양한 **GUI** 프로그램들을 실행하기 어려움
  - **GtkFB**
- 서버-클라이언트 모델
  1. 서버가 그래픽 장치를 전용해서 사용하는 경우
     - 클라이언트로부터의 그래픽 요청을 처리
     - 안정된 처리를 보장하지만, 복잡하고 크기가 큼(수 **MB** 이상)
     - **X-Windows, Microwindows**
  2. 클라이언트와 그래픽 장치를 공유해서 사용하는 경우
     - 서버는 그래픽 장치 사용을 허가
     - 클라이언트가 그래픽 처리를 책임져야 함
     - **Qtopia**

X Windows 개요


Applications
Window
Manager
Toolkit (Xt, GTK+ , Qt, OSF/Motif)
Xlib Library
Network protocol
Network protocol
X11 Server
I/O Device Driver(Graphic Card, Keyboard, Mouse, Remote Controller, etc.)

X Windows 개요

- 서버-클라이언트 구조
  - 서버가 실제 그래픽 처리 등을 담당
  - 클라이언트는 입력(키보드, 마우스) 에 대한 정보를 요구하고, 터미널 출력을 서버로 보내 터미널 변경 요구
- 계층적 구조
  - **Xlib Library**
    - 윈도우 생성, 이동, 크기 조정, 폰트, 커서, 도형 관련 작업들
    - 직접 **X** 프로토콜을 생성하여 서버로 전송
  - **X Toolkit**
    - 버튼, 풀다운 메뉴 등 고급 프로그래밍 지원
    - **X** 서버와 클라이언트 간의 통신과 **Widget(위젯) GUI** 객체 처리
    - **Adena(MIT), OpenLook(AT&T), Motif(OSF), GTK+(GNU), Qt(Trolltech)** 등 다양한 위젯 집합을 지원
  - **Window Manager**
    - 데스크톱 화면을 나누어 워크스페이스를 늘리거나, 각 윈도우의 주변 프레임, 최소화, 최대화, 닫기 아이콘, 제목 표시줄, 이동 등의 기능
    - 또 하나의 **X Client** 중 하나

X 윈도우의 역사

- Creation in 1987 by MIT X Consortium
- X11R6 released in 1992
- Dark ages : no significant change by X.org since then
- XFree86 gaining importance
  - LINUX (and BSD) drive X
  - Two new toolkits with applications (Qt and GTK+)
  - Numerous commercial and free applications, games etc
- New challenges for X
  - Hardware changes since 1987
    - CPU infinitely fast, Bus writes dominate performance, graphic cards converged for 2D ops
  - Application expectations are much higher
  - X moving down to handheld computers

X 윈도우의 특징

- **Pros**
  - **Network Transparency**
    - 네트워크가 투명하여 사용자 입장에서는 없는 것처럼 보임
  - **Modularity and Extensibility**
    - 서버-클라이언트 구조로 장치 독립성이 보장
    - 많은 윈도우 자원들이 정해지지 않아 다양한 인터페이스로 적용 가능
  - **Open Source**
    - 소스가 공개
- **Cons**
  - **Too slow**
  - **Too much burden on the programmer**
  - **No standard toolkit**
  - **Too old**
  - **Too complex**

진보된 기능들

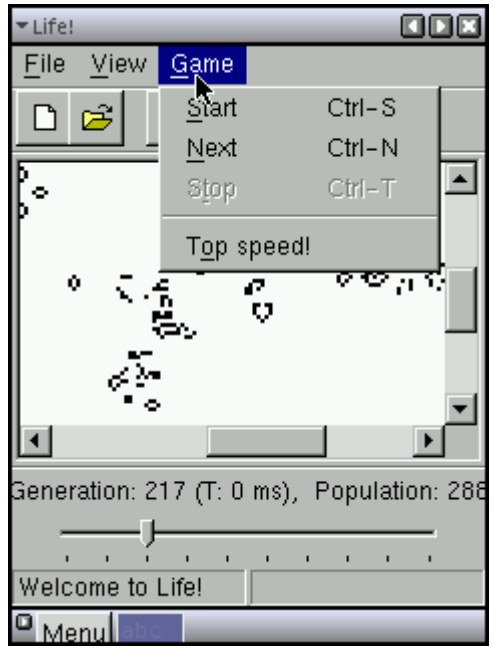
- 이미지 합성 (Image Composition)
  - X Render Extension
    - Porter/Duff image composition
    - Support for anti-aliased characters.
- 폰트 지원 (Font support)
  - Xft Extension
    - Anti-aliased outline font support
    - No more server side fonts
    - Applications' direct access to font files and window system cache
  - Fontconfig Extension
    - font installation, naming, substitution
- 크기 재설정 및 회전 (Resize and Rotation)
  - X Resize and Rotation Extension

Qt

- 멀티플랫폼 **GUI** 응용을 위한 **C++ GUI Toolkit**
  - 객체 지향 프로그래밍 구현 – 시그널과 슬롯 등 제공
  - 풍부한 위젯**(Widget)** 제공
  - 다양한 플랫폼 지원 – **Windows, Mac OS X, Linux, Solaris,HP-UX**
  - **Unicode** 와 **i18n** 등의 다양한 언어 지원
- 유용한 도구 제공
  - **Qt Designer(GUI** 빌더**) , Qt Linguist, qmake** 등
- 대표적인 리눅스 데스크탑 환경인 **KDE** 의 기반
  - **Qt/X11** 사용
- 노르웨이의 **Trolltech** 사에서 개발
- 다양한 배포판으로 제공됨
  - **Qt Enterprise Edition and Qt Professional Edition**
  - **Qt Free Edition**
  - **Qt Embedded Edition**

WxWidgets

- 오픈 소스 크로스 플랫폼 **C++ GUI** 라이브러리
  - **Windows, Linux/GTK, Linux/X11, MacOS** 등에 포팅
  - 응용을 재컴파일하면 여러 플랫폼에서 실행 가능
- 임베디드 용으로 설계된 것은 아님
- http://www.wxwidgets.org



그래픽 응용 프로그램 개발

- 응용 프로그래밍
  - 그래픽 라이브러리 학습
    - 라이브러리 관련 출판된 책이나 사이트의 매뉴얼 등 참고
  - **IDE, GUI** 빌더 등의 소프트웨어 사용
    - **Qt - Kdevelop, Qt Designer(GUI Builder)**
    - **GTK+ - Glade-2(GUI Builder)**
    - **wxWindows – MinGW Developer Studio, wxDesigner**
- 개발 환경
  - 그래픽 라이브러리 컴파일 및 설치
  - 이를 이용하여 그래픽 응용 프로그램 컴파일

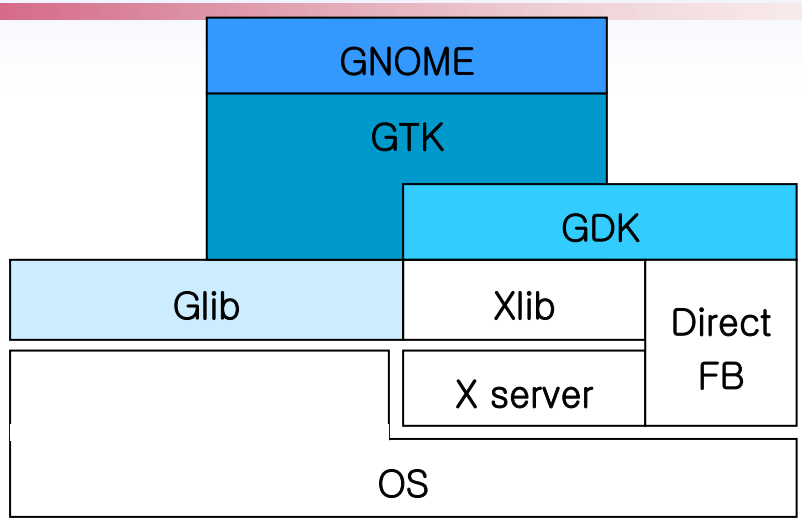
GTK+ 프로그래밍

GTK+

- 원래 **GIMP** 툴을 위해 개발된 **GUI toolkit**
  - **Qt** 와 함께 가장 많이 사용하는 오픈 소스 그래픽 라이브러리
- 특징
  - 전부 **C** 로 작성되었으며, 대부분의 **GTK+** 소프트웨어도 **C** 로 작성
  - 다양한 위젯들**(Button, List, Tree, Menu** 등**)** 제공
  - 다양한 언어 바인딩 제공 – **Python, PHP, Perl, Tcl/tk, Java** 등
  - 다양한 개발 도구 제공 **- Glade-2 GUI** 생성 도구 등
- 홈 페이지 - http://www.gtk.org

GTK+ 구조

GTK+ 구조



구성 라이브러리

- **Glib** – 기본 데이터 타입과 오류처리, 메모리 관리 함수들
- **GDK** – 플랫폼의 그래픽 **API** 와 **GTK+** 사이에 위치
- **GNOME** – 다양한 고급 **GUI** 객체(파일 선택 창 등) 제공
- **Pango** – 다중 언어를 위한 텍스트와 폰트 지원
- **libpng, libjpeg, libtiff** – 이미지 처리 지원
- **FreeType** – **Truetype** 및 **Type1** 폰트 지원 라이브러리

GLib

- 개요
  - 데이터 형식, 함수, 메모리 관리, 공통 작업을 처리하는 이식 가능한 C 라이브러리
- 특징
  - 여러 플랫폼에서 개발 가능하도록 형식을 제공
  - 데이터 형식은 표준 C 형식을 대체
    - **gint, guint, gchar, guchar, glong, gulong, gfloat, gdouble**
    - **gpointer**
    - **gboolean**
    - **Gint8, guint8, gint16, guint16, gint32, guint32**

Signal 과 Callback

- **GTK+** 는 이벤트 기반 라이브러리
  - 사용자가 키보드나 마우스를 통해 입력할 때마다 이벤트 발생
  - 이벤트가 발생하면 적절한 함수로 처리하고, 이벤트가 발생하지 않으면 계속 대기
- **Signal** 과 **Callback** 매커니즘으로 이벤트 처리
  - 어떤 위젯에 이벤트가 발생하면, 위젯은 시그널을 발생하고 그 시그널과 연결된 콜백 함수 호출
  - 사용 방법
    - 미리 시그널을 처리할 함수를 만들고 **g_signal_connect()** 함수를 이용하여 시그널과 처리 함수를 연결
    - 예) 버튼의 **clicked()** 시그널과 **callback_func()** 함수를 연결하여 사용자가 버튼을 클릭하면 **callback_func()** 함수가 호출
    - ➔ **g_signal_connect (GTK_OBJECT(button), "clicked",**
      GTK_SIGNAL_FUNC (callback_func), func_data);

Signal 과 Callback

▣ 간단한 Signal 과 Callback 예제

```
#include <gtk/gtk.h>

static int count = 0;
/* callback */
void button_clicked(GtkWidget *widget, gpointer data)
{
   printf("%s pressed %d time(s)\n", data, ++count);
}

int main( int argc, char *argv[] )
{
   GtkWidget *window, *button;

   gtk_init (&argc, &argv);

   window =
     gtk_window_new(GTK_WINDOW_TOPLEVEL);
   button = gtk_button_new_with_label ("Hello
     World!");
   gtk_container_add (GTK_CONTAINER(window),
     button);

   g_signal_connect (GTK_OBJECT (button), "clicked",
            GTK_SIGNAL_FUNC (button_clicked),
            "Button1");
   gtk_widget_show (button);
   gtk_widget_show (window);

   gtk_main();

   return (0);
}
```

Clicked 시그널에 대한 콜백 함수임

clicked 시그널이 발생하여 해당 콜백 함수 button_clicked 를 호출

GTK+ Widgets

- **GTK+** 는 버튼, 스크롤바 등의 풍부한 위젯들 제공
  - 위젯들은 **GtkWidget** 의 인스턴스들

  - 컨테이너 **(Container)** 위젯 **GtkContainer**
    - 컨테이너 위젯은 많은 수의 자식 위젯들을 포함
    - 자식 위젯들은 컨테이너 위젯 영역 안에 표시
    - 여러 자식 위젯들을 배치하기 위해 패킹상자**(packing box)** 사용
      - 수평 배치 박스 hbox, 수직 배치 박스 vbox
    - **gtk_hbox_new(), gtk_vbox_new()**
      - homogeneous(gboolean) : 상자의 모든 위젯이 가장 큰 위젯과 같은 공간 여부
      - Spacing(gint) : 이웃한 위젯 사이의 폭
    - **gtk_box_pack_start(), gtk_box_pack_end()**
      - Expand(gboolean) : 패킹상자가 남은 공간을 채우기 위해 커지는지 여부
      - Fill(gboolean) : 위젯들이 남은 공간을 채우기 위해 커지는지 여부
      - Padding(gint) : 위젯을 둘러싼 공백 폭

GNOME

- 개요
  - **GNU GPL** 하에 개발된 완전한 데스크톱 환경
  - **GNU Network Object Model Environment** 의 약자
- 특징
  - 응용 프로그램 실행 패널, 프로그램과 유틸리티 모음, 프로그래밍 라이브러리, 개발자를 위한 기능으로 구성
    - **GNOME** 라이브러리 – 파일 선택 창과 같은 고급 **GUI** 툴킷
  - 레드햇 리눅스 기본 데스크톱의 기반

GTK+ 개발 환경

- **GTK 개발에 필요한 소스**
  - **Xfree86 또는 Xorg 소스코드 (XFree86 4.5.0, xorg 6.8)**
    - **XFree86 홈페이지(http://www.xfree86.org) 또는 X.org 홈페이지 (www.x.org) 로부터 다운로드**
  - **GTK+ 소스코드 (GTK+ 2.x)**
    - **GTK+ 홈페이지(http://www.gtk.org) 로부터 다운로드**
  - **LIBGNOMEUI-2 소스코드 (libgnomeui-2.x)**
    - **GNOME UI 라이브러리**
    - **GNOME 홈페이지(http://www.gnome.org) 로부터 다운로드**
  - **Glade-2 소스코드 (Glade 2.x)**
    - **GTK+ GUI 빌더**
    - **Glade 홈페이지(http://glade.gnome.org) 로부터 다운로드**

GTK+ Programming

- 기본 GTK+ 프로그램의 7 단계
  - 환경 초기화
  - 위젯 생성 및 속성 설정
  - 콜백 루틴 등록
  - 인스턴스 계층 정의
  - 위젯을 화면에 보이게 함
  - 시그널과 이벤트 처리
  - 종료

GTK+ Programming

Hello GTK 예제

```
#include <gtk/gtk.h>

/* callback */
static void destroy(GtkWidget *widget,
                    gpointer data )
{
   gtk_main_quit ();
}

int main(int argc, char *argv[] )
{
   /* GtkWidget is the storage type for widgets */
   GtkWidget *window, *button;

   gtk_init (&argc, &argv);

   /* create a new window and sets it's border width. */
   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   gtk_container_set_border_width (GTK_CONTAINER (window), 10);
   /* Creates a new button with the label "Hello World". */
   button = gtk_button_new_with_label ("Hello World");

   g_signal_connect (GTK_OBJECT (button), "clicked",
                     GTK_SIGNAL_FUNC (destroy), NULL);

   /* This packs the button into the window (a gtk container). */
   gtk_container_add (GTK_CONTAINER (window), button);

   /* The final step is to display this newly created widget. */
   gtk_widget_show (button);
   gtk_widget_show (window);

   gtk_main ();

   return 0;
}
```

GTK+ Programming

▣ 환경 초기화

◈ 환경 초기화 함수

➢ **gtk_init()**

```
void gtk_init(int *argc_addr, char **argv_addr)
```

- argc_addr : 명령 행의 인자 수를 나타내는 변수 argc 의 주소
- argv_addr : 명령 행의 인자 값을 가지는 변수 argv 의 주소

▣ 위젯 생성 및 속성 설정

◈ 위젯 생성 함수

➢ **gtk_{widget_name}_new{tailopt}({arguments})**

```
GtkWidget *gtk_window_new(GtkWindowType type)
GtkWidget *gtk_button_new_with_label(const gchar *text)
```

- type : 생성할 윈도우의 형태. GTK_WINDOW_TOPLEVEL, GTK_WINDOW_DIALOG, GTK_WINDOW_POPUP
- Text : 버튼의 레이블 텍스트

GTK+ Programming

◈ 속성 설정 함수

➢ **gtk_{widget_name}_set_{property}({arguments})**

```
void gtk_container_set_border_width(GtkContainer *container, guint width)
```

- container : 해당하는 container 위젯
- width : 위젯을 둘러싸는 테두리의 폭

▣ 콜백 루틴 등록

◈ 콜백 루틴 – 특정 시그널을 처리할 함수

◈ 콜백 루틴 등록 함수

➢ **g_signal_connect()**

```
void g_signal_connect (gpointer instance, gchar *name, GCallback call_routine,
                       gpointer data);
```

- Instance : 시그널을 발생시킬 위젯
- Name : 시그널의 이름
- Call_routine : 시그널이 탐지되었을 때 호출할 함수
- Data : 콜백 함수에 넘겨줄 정보

GTK+ Programming

▣ 인스턴스 계층 정의

◈ 계층 정의 함수

➢ **gtk_container_add() 또는 gtk_box_pack_start()**

```
void gtk_container_add(GtkContainer *container, GtkWidget *child)
void gtk_box_pack_start(GtkBox *box, GtkWidget *child, gboolean expand,
    gboolean fill, guint padding)
```

- Container, box : 자식 위젯을 포함할 container 또는 box 위젯
- Child : container 또는 box 위젯의 자식으로 포함되는 위젯
- expand : 패킹상자가 남은 공간을 채우기 위해 커지는지 여부
- fill : 위젯들이 남은 공간을 채우기 위해 커지는지 여부
- padding : 위젯을 둘러싼 공백 폭

▣ 위젯을 화면에 보이게 함

◈ 위젯 화면 표시 함수

➢ **gtk_widget_show()**

```
void gtk_widget_show(GtkWidget *widget);
```

➢ **gtk_widget_show_all() : 모든 자식 위젯들을 화면에 보이게 함**

```
void gtk_widget_show_all(GtkWidget *pwidget);
```

GTK+ Programming

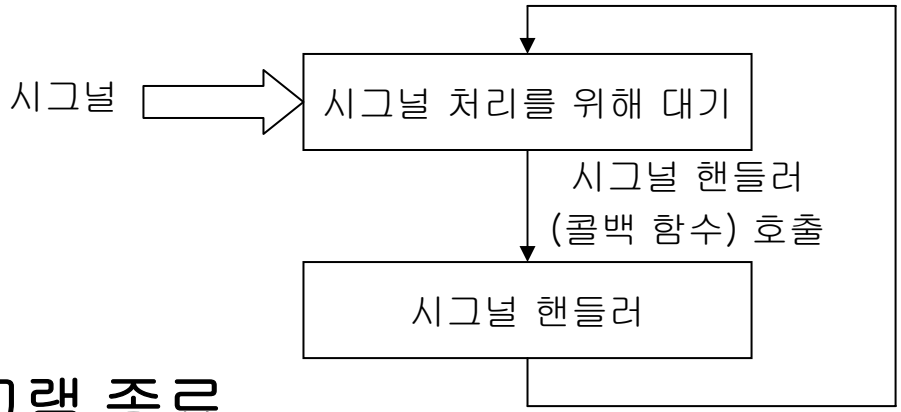
▣ 시그널과 이벤트 처리

◈ 시그널과 이벤트 처리 함수

➢ **gtk_main()**

```
void gtk_main();
```

- 루프를 돌면서 시그널을 받아들이고 적절한 콜백 루틴을 호출
- 콜백 루틴이 종료하면 다시 시그널 처리 계속



▣ 프로그램 종료

◈ 프로그램 종료 함수

➢ **gtk_main_quit() : gtk_main()** 함수를 종료시킴. 주로 콜백 함수 이용

```
void gtk_main_quit();
```

GTK+ Programming

▣ GTK+ 프로그램 컴파일

◈ Helloworld.c 프로그램 컴파일

```
$ gcc -Wall -g helloworld.c -o helloworld `pkg-config --cflags gtk+-2.0` \
`pkg-config --libs gtk+-2.0`
```

◈ pkg-config

- 설치된 패키지(프로그램/도구) 의 정보(컴파일 옵션 등)를 추출하는 도구
  - 컴파일 옵션 : $ pkg-config --cflags {pkg_name}
  - 링크 옵션 : $ pkg-config --libs {pkg_name}

◈ GTK+ 관련 링크 라이브러리

- **GTK 라이브러리 (-lgtk) : GDK 에 기반한 위젯 라이브러리.**
- **GDK 라이브러리 (-lgdk) : Xlib wrapper 의 일종.**
- **gdk-pixbuf 라이브러리 (-lgdk_pixbuf) : 이미지 처리 라이브러리.**
- **Pango 라이브러리 (-lpango) : 국제화 문자 처리 라이브러리.**
- **gobject 라이브러리(-lgobject) : GTK 를 위한 타입 체계.**
- **gmodule 라이브러리 (-lgmodule) : 실행 시간 확장의 적재를 위한 라이브러리.**
- **GLib 라이브러리 (-lglib) : GTK 가 이용하는 자료 구조 등을 포함.**
- **Xlib 라이브러리 (-lX11) : GDK 가 이용하는 X 윈도우 라이브러리.**
- **Xext 라이브러리 (-lXext) : 공유 메모리 픽스맵 등 X 윈도우 확장 라이브러리.**
- **math 라이브러리 (-lm) : 다양한 목적으로 GTK가 사용하는 수학 라이브러리.**

GtkBox Widget

- 패킹 상자(packing box) 위젯
- 종류
  - **GtkHBox** – 수평 패킹 상자 위젯
  - **GtkVBox** – 수직 패킹 상자 위젯
- 생성
  - **GtkWidget * gtk_hbox_new (gboolean homogeneous, gint spacing);**
  - **GtkWidget * gtk_vbox_new (gboolean homogeneous, gint spacing);**
    - ***homogeneous :*** **TRUE** 이면 포함 위젯이 모두 같은 공간 차지
    - ***spacing :*** 위젯 간의 간격을 픽셀 단위로 지정
- 사용 함수
  - **void gtk_box_pack_start (GtkBox *box, GtkWidget *child, gboolean expand, gboolean fill, guint padding);**
    - **GtkHBox** 의 왼쪽과 **GtkVBox**의 아래쪽에 위젯 추가
  - **void gtk_box_pack_end (GtkBox *box, GtkWidget *child, gboolean expand, gboolean fill, guint padding);**
    - **GtkHBox** 의 오른쪽과 **GtkVBox**의 위쪽에 위젯
    - ***box :*** 채워질 패킹 상자, ***child :*** 패킹 상자에 놓을 위젯
    - ***expand :*** **TRUE** 이면 다른 위젯 공유 가용 공간을 채움
    - ***fill :*** **TRUE** 이면 자신에게 할당된 공간을 모두 채움
    - ***padding :*** 위젯 주변 공간의 픽셀 단위 여백

GtkBox Widget

GtkBox 예제 프로그램

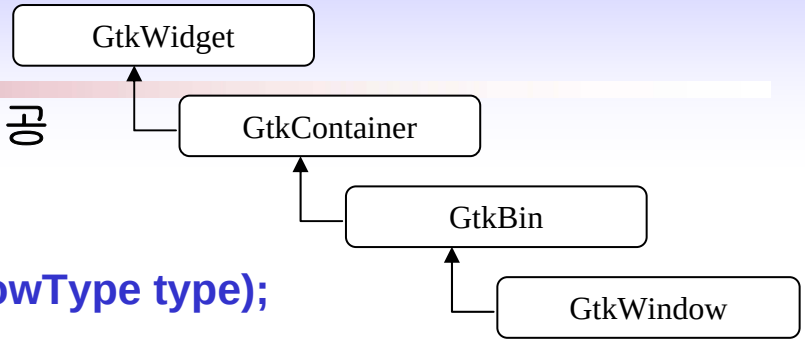
```
#include <gtk/gtk.h>

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *label1, *label2, *label3;
  GtkWidget *hbox;
  GtkWidget *vbox;

  gtk_init(&argc, &argv);
  window =
     gtk_window_new(GTK_WINDOW_TOPLEVEL);

  gtk_window_set_title(GTK_WINDOW(window), "The
     Window Title");
  gtk_window_set_position(GTK_WINDOW(window),
     GTK_WIN_POS_CENTER);
  gtk_window_set_default_size(GTK_WINDOW(window),
     300, 200);
```

```
/* The "destroy" signal is emitted when the window is
     closed  */
  g_signal_connect (GTK_OBJECT(window), “destroy”,
                    GTK_SIGNAL_FUNC ( closeApp), NULL);

  label1 = gtk_label_new("Label 1");
  label2 = gtk_label_new("Label 2");
  label3 = gtk_label_new("Label 3");

  hbox = gtk_hbox_new ( TRUE, 5 );
  vbox = gtk_vbox_new ( FALSE, 10);

  gtk_box_pack_start(GTK_BOX(vbox), label1, TRUE,
     FALSE, 5);
  gtk_box_pack_start(GTK_BOX(vbox), label2, TRUE,
     FALSE, 5);

  gtk_box_pack_start(GTK_BOX(hbox), vbox, FALSE,
     FALSE, 5);
  gtk_box_pack_start(GTK_BOX(hbox), label3, FALSE,
     FALSE, 5);

  gtk_container_add(GTK_CONTAINER(window), hbox);
  gtk_widget_show_all(window);
  gtk_main ();

  return 0;
}
```

GtkBox Widget

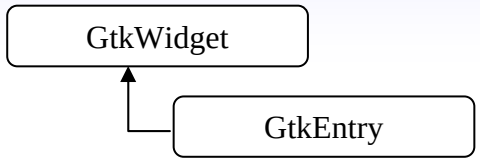
- **GtkBox** 예제 프로그램 실행 결과

GtkWindow Widget



- 모든 GTK+ 응용 프로그램의 기본 윈도우 제공
  - TOPLEVEL, DIALOG 등의 종류가 있음
- 생성
  - GtkWidget * gtk_window_new (GtkWindowType type);
- 사용 함수
  - void gtk_window_set_title (GtkWindow *window, const gchar *title);
    - 제목 표시줄의 텍스트 변경
  - void gtk_window_set_position (GtkWindow *window, GtkWindowPosition position);
    - 윈도우의 화면 상의 초기 위치 제어, GTK_WIN_POS_NONE, GTK_WIN_POS_CENTER, GTK_WIN_POS_MOUSE, GTK_WIN_POS_CENTER_ALWAYS, GTK_WIN_POS_CENTER_OR_PARENT 의 5가지 위치 가능
  - void gtk_window_set_default_size (GtkWindow *window, gint width,gint height);
    - 윈도우의 디폴트 크기 설정
  - void gtk_window_resize (GtkWindow *window, gint width, gint height);
    - 윈도우의 크기 변경
  - void gtk_window_set_resizable (GtkWindow *window, gboolean resizable);
    - 윈도우의 크기 변경 가능성 설정
  - void gtk_window_present (GtkWindow *window);
    - 윈도우가 최소화되거나 숨겨지지 않도록 설정
  - void gtk_window_maximize (GtkWindow *window);
    - 윈도우를 최대화
  - void gtk_window_minimize (GtkWindow *window);
    - 윈도우를 최소화

GtkEntry Widget



- 사용자가 키보드로 입력할 수 있는 단일-라인 영역
- 생성
  - **GtkWidget * gtk_entry_new (void);**
  - **GtkWidget * gtk_entry_new_with_max_length (gint max);**
- 사용 함수
  - **void gtk_entry_set_max_length (GtkEntry *entry, gint max);**
    - 입력 텍스트의 최대 길이 설정
  - **G_CONST_RETURN gchar * gtk_entry_get_text (GtkEntry *entry);**
    - 입력 텍스트 획득
  - **void gtk_entry_set_text (GtkEntry *entry, const gchar *text);**
    - 입력 텍스트 설정
  - **void gtk_entry_append_text (GtkEntry * entry, const gchar *text);**
    - 입력 텍스트 추가
  - **void gtk_entry_prepend_text (GtkEntry * entry, const gchar *text);**
    - 입력 텍스트 추가
  - **void gtk_entry_set_visibility (GtkEntry * entry, gboolean visible);**
    - 입력 텍스트의 가시성 설정
  - **void gtk_entry_set_invisible_char (GtkEntry * entry, gchar invch);**
    - 원하는 문자를 보이지 않도록 변경
  - **void gtk_entry_set_editable (GtkEntry * entry, gboolean editable);**
    - 텍스트의 입력 가능성 설정

GtkEntry Widget

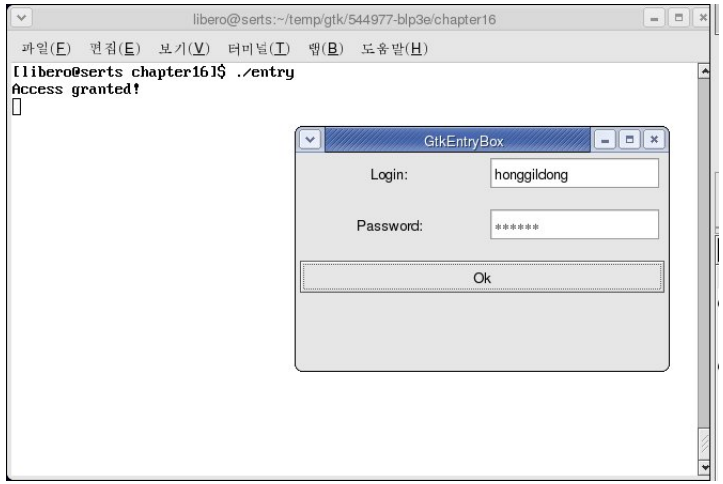
GtkEntry 예제 프로그램

```
#include <gtk/gtk.h>
#include <stdio.h>
#include <string.h>

const char * password = "secret";

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

void button_clicked (GtkWidget *button, gpointer data)
{
  const char *password_text = gtk_entry_get_text
      (GTK_ENTRY( (GtkWidget *) data));

  if (strcmp(password_text, password) == 0)
    printf("Access granted!\n");
  else
    printf("Access denied!\n");
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *username_label, *password_label;
  GtkWidget *username_entry, *password_entry;
  GtkWidget *ok_button;
  GtkWidget *hbox1, *hbox2;
  GtkWidget *vbox;

  gtk_init(&argc, &argv);

  window =
     gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title(GTK_WINDOW(window),
     "GtkEntryBox" );
  gtk_window_set_position(GTK_WINDOW(window),
     GTK_WIN_POS_CENTER);
  gtk_window_set_default_size(GTK_WINDOW(window),
     200, 200);

  g_signal_connect (GTK_OBJECT (window), "destroy",
                GTK_SIGNAL_FUNC ( closeApp), NULL);

  username_label = gtk_label_new("Login:");
  password_label = gtk_label_new("Password:");

  username_entry = gtk_entry_new();
  password_entry = gtk_entry_new();
  gtk_entry_set_visibility(GTK_ENTRY (password_entry),
     FALSE);

  ok_button = gtk_button_new_with_label("Ok");

  g_signal_connect (GTK_OBJECT(ok_button), "clicked",
     GTK_SIGNAL_FUNC(button_clicked),
     password_entry);
```

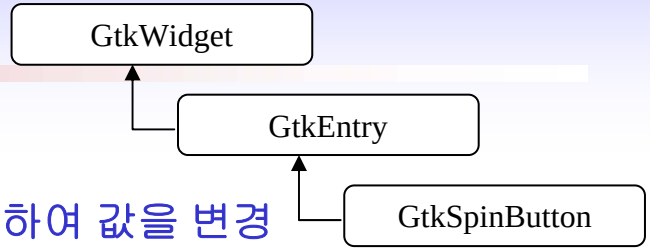
GtkEntry Widget

GtkEntry 예제 프로그램

```
  hbox1 = gtk_hbox_new ( TRUE, 5 );
  hbox2 = gtk_hbox_new ( TRUE, 5 );

  vbox = gtk_vbox_new ( FALSE, 10);

  gtk_box_pack_start(GTK_BOX(hbox1),
     username_label, TRUE, FALSE, 5);
  gtk_box_pack_start(GTK_BOX(hbox1),
     username_entry, TRUE, FALSE, 5);

  gtk_box_pack_start(GTK_BOX(hbox2),
     password_label, TRUE, FALSE, 5);
  gtk_box_pack_start(GTK_BOX(hbox2),
     password_entry, TRUE, FALSE, 5);

  gtk_box_pack_start(GTK_BOX(vbox), hbox1, FALSE,
     FALSE, 5);
  gtk_box_pack_start(GTK_BOX(vbox), hbox2, FALSE,
     FALSE, 5);
  gtk_box_pack_start(GTK_BOX(vbox), ok_button,
     FALSE, FALSE, 5);

  gtk_container_add(GTK_CONTAINER(window), vbox);
  gtk_widget_show_all(window);
  gtk_main ();

  return 0;
}
```



GtkSpinButton Widget

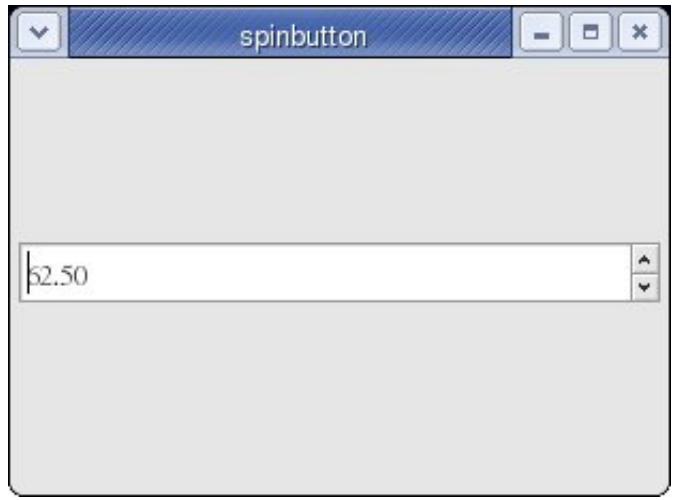


- 숫자 문자만을 입력하는 위젯
  - 지정된 최소/최대값에 대한 화살표를 마우스로 클릭하여 값을 변경
  - **GtkAdjustment** 위젯을 사용하여 제한된 범위의 값 제어
- 생성
  - **GtkWidget * gtk_spin_button_new (GtkAdjustment *adjustment, gdouble climb_rate, guint digits);**
  - **GtkWidget * gtk_spin_button_new_with_range (gdouble min, gdouble max, gdouble step);**
  - **GtkObject *gtk_adjustment_new (gdouble value, gdouble lower, gdouble upper, gdouble step_increment, gdouble page_increment, gdouble page_size);**
    - **GtkAdjustment** 위젯 생성하며 초기값, 최소값, 최대값, 최소/최대 증감 크기 설정
- 사용 함수
  - **void gtk_spin_button_set_digits (GtkSpinButton *spin_button, guint digits);**
  - **void gtk_spin_button_set_increments (GtkSpinButton *spin_button, gdouble step, gdouble page);**
  - **void gtk_spin_button_set_range (GtkSpinButton *spin_button, gdouble min, gdouble max);**
    - 각각 스핀버튼의 값, 최소/최대 증감 크기, 최소값/최대값 설정
  - **gdouble gtk_spin_button_get_value (GtkSpinButton *spin_button);**
    - 스핀버튼의 값 획득
  - **gint gtk_spin_button_get_value_as_int (GtkSpinButton *spin_button);**

GtkSpinButton Widget

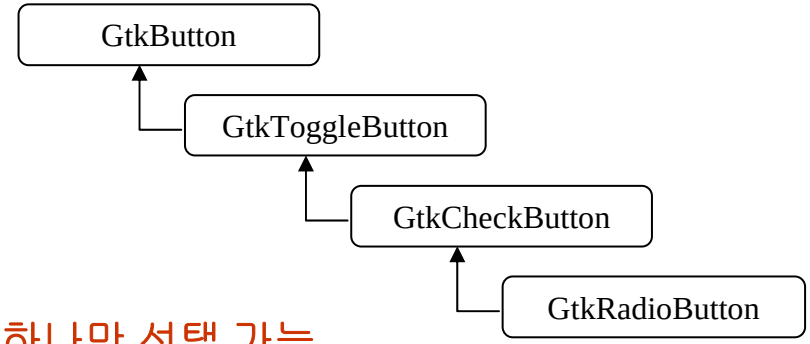
GtkSpinButton 예제 프로그램

```
#include <gtk/gtk.h>

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *spinbutton;
  GtkObject *adjustment;

  gtk_init (&argc, &argv);
  window =
     gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_window_set_default_size
     ( GTK_WINDOW(window), 300, 200);
  g_signal_connect ( GTK_OBJECT (window), "destroy",
                     GTK_SIGNAL_FUNC ( closeApp), NULL);

  adjustment = gtk_adjustment_new(100.0, 50.0,
     150.0, 0.5, 0.05, 0.05);
  spinbutton =
     gtk_spin_button_new(GTK_ADJUSTMENT(adjustme
     nt), 0.01, 2);
  gtk_container_add(GTK_CONTAINER(window),
     spinbutton);
  gtk_widget_show_all(window);
  gtk_main ();

  return 0;
}
```



GtkButton Widget

- 사용자가 클릭하여 특정 동작을 유발하는 위짓
- 종류
  - 명령(command) 버튼 - **GtkButton**
  - 상태(state) 버튼 - **GtkToggleButton**
    - 사용자가 상태 값으로 선택 가능
    - **GtkCheckButton :** 체크가 토글됨
    - **GtkRadioButton :** 그룹으로 묶여 그 중 하나만 선택 가능



- **GtkToggleButton Widget**
  - 사용자가 토글버튼을 클릭하면 **"clicked"** 신호 발생하고 자신의 상태 변경
  - 생성
    - **GtkWidget * gtk_toggle_button_new (void);**
    - **GtkWidget * gtk_toggle_button_new_with_label (const gchar *label);**
  - 사용 함수
    - **gboolean gtk_toggle_button_get_active (GtkToggleButton *toggle_button);**
      - 토글버튼의 상태 획득
    - **void gtk_toggle_button_set_active (GtkToggleButton *toggle_button, gboolean is_active);**
      - 토글버튼의 상태 설정

GtkButton Widget

- **GtkCheckButton Widget**
  - **GtkToggleButton** 과 거의 같으나 텍스트가 옆에 있는 체크 박스 형태 가능
  - 생성
    - **GtkWidget * gtk_check_button_new (void);**
    - **GtkWidget * gtk_check_button_new_with_label (const gchar *label);**
- **GtkRadioButton Widget**
  - 같은 형식의 버튼을 여러 개를 묶어서 하나만을 선택
  - **RadioButton** 그룹은 **GSList** 라는 **Glib** 리스트 객체 사용
  - 생성
    - **GtkWidget * gtk_radio_button_new (GSList *group);**
    - **GtkWidget * gtk_radio_button_new_from_widget (GtkRadioButton *group);**
    - **GtkWidget * gtk_radio_button_new_with_label (GSList *group, const gchar *label);**
  - 사용 함수
    - **gboolean gtk_radio_button_set_group (GtkRadioButton *radio_button, GSList *group);**
      - 라디오버튼의 그룹 설정
    - **void gtk_radio_button_get_group (GtkRadioButton *radio_button);**
      - 라디오버튼의 그룹 획득

GtkButton Widget

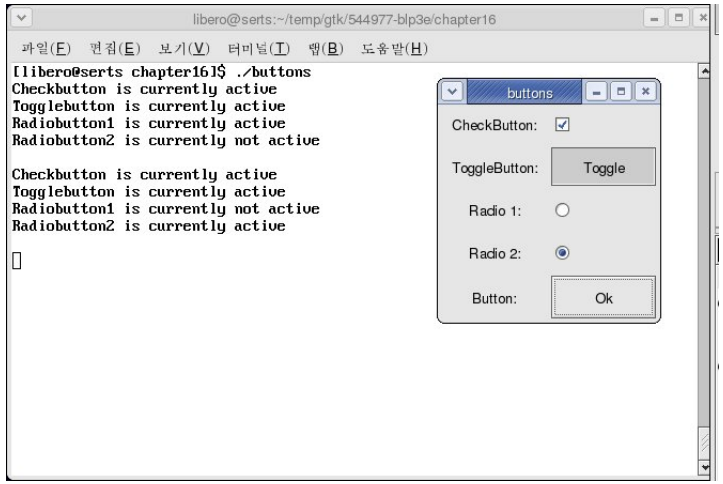
GtkButton 예제 프로그램

```
#include <gtk/gtk.h>
#include <stdio.h>

GtkWidget *checkbutton;
GtkWidget *togglebutton;
GtkWidget *radiobutton1, *radiobutton2;

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

void add_widget_with_label ( GtkContainer * box, gchar
     * caption, GtkWidget * widget)
{
  GtkWidget *label = gtk_label_new (caption);
  GtkWidget *hbox = gtk_hbox_new (TRUE, 4);

  gtk_container_add(GTK_CONTAINER (hbox), label);
  gtk_container_add(GTK_CONTAINER (hbox), widget);

  gtk_container_add(box, hbox);
}

void print_active(char * name, GtkToggleButton
     *button)
{
  gboolean active =
     gtk_toggle_button_get_active(button);
  printf("%s is currently %s\n", name,
     active?"active":"not active");
}

void button_clicked(GtkWidget *button, gpointer data)
{
  print_active("Checkbutton",
     GTK_TOGGLE_BUTTON(checkbutton));
  print_active("Togglebutton",
     GTK_TOGGLE_BUTTON(togglebutton));
  print_active("Radiobutton1",
     GTK_TOGGLE_BUTTON(radiobutton1));
  print_active("Radiobutton2",
     GTK_TOGGLE_BUTTON(radiobutton2));
  printf("\n");
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *button;
  GtkWidget *vbox;

  gtk_init (&argc, &argv);
  window =
     gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_window_set_default_size(GTK_WINDOW(window),
     200, 200);
  g_signal_connect ( GTK_OBJECT (window), “destroy”,
     GTK_SIGNAL_FUNC ( closeApp), NULL);
```

GtkButton Widget

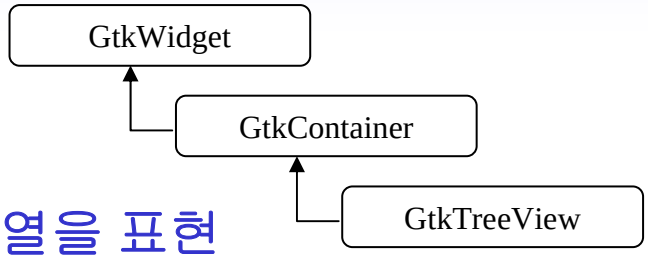
GtkButton 예제 프로그램

```
button = gtk_button_new_with_label("Ok");
checkbutton = gtk_check_button_new();
togglebutton =
    gtk_toggle_button_new_with_label(“Toggle”);
radiobutton1 = gtk_radio_button_new(NULL);
radiobutton2 = gtk_radio_button_new_from_widget
    (GTK_RADIO_BUTTON(radiobutton1));

vbox = gtk_vbox_new (TRUE, 4);
add_widget_with_label ( GTK_CONTAINER(vbox),
    “CheckButton:”, checkbutton);
add_widget_with_label (GTK_CONTAINER(vbox),
    “ToggleButton:”, togglebutton);
add_widget_with_label (GTK_CONTAINER(vbox), “Radio
    1:”, radiobutton1);
add_widget_with_label (GTK_CONTAINER(vbox), “Radio
    2:”, radiobutton2);
add_widget_with_label (GTK_CONTAINER(vbox),
    “Button:”, button);

g_signal_connect(GTK_OBJECT(button), “clicked”,
        GTK_SIGNAL_FUNC(button_clicked), NULL);

gtk_container_add(GTK_CONTAINER(window), vbox);
gtk_widget_show_all(window);
gtk_main ();

return 0;
}
```

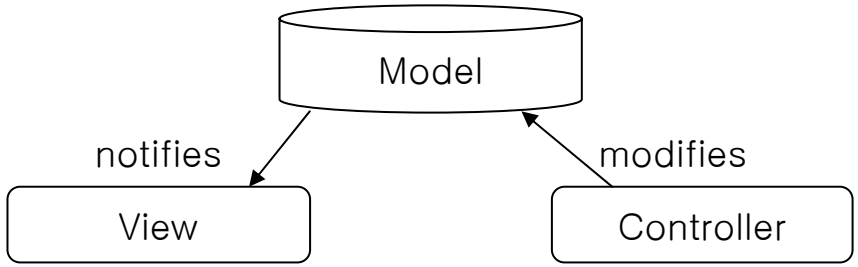


GtkTreeView Widget

- 데이터의 리스트 뷰나 트리 뷰를 만듦
- 기본 구성 요소
  - **GtkTreeView :** 트리 뷰와 리스트 뷰
  - **GtkTreeViewColumn :** 리스트나 트리의 열을 표현
  - **GtkCellRenderer :** 그리기 셀을 제어
  - **GtkTreeModel :** 트리와 리스트 데이터를 표현



- **MVC(Model/View/Controller)** 패턴으로 제작



  - 복수의 뷰 또는 컨트롤러 수행이 가능
    - 예) 문서편집기에서 한 뷰는 문서 윤곽 표시, 한 뷰는 전체 문서 출력
  - 불필요한 중복 없이 다른 뷰가 동시에 데이터 렌더링 가능
  - 웹 프로그래밍에서 많이 사용

GtkTreeView Widget

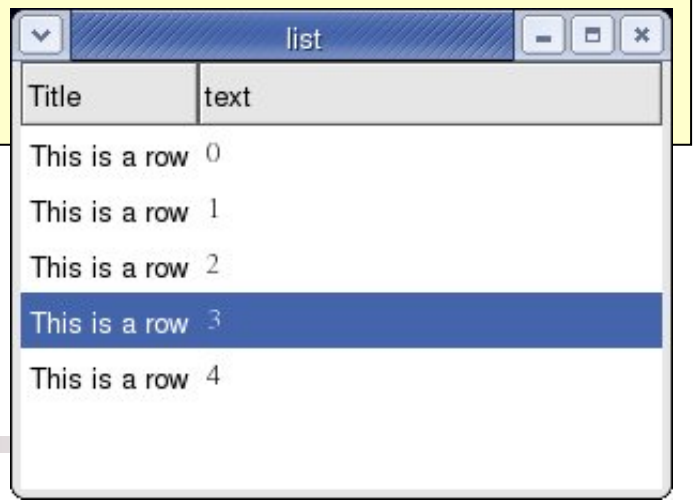
- 모델 구성요소
  - **GtkTreeStore** – 계층 구조의 다중 수준 데이터 저장
  - **GtkListStore** – 계층이 없는 데이터 저장
  - 생성
    - **GtkWidget * gtk_tree_store_new (gint,);**
  - 사용 함수
    - **void gtk_tree_store_append (GtkTreeStore *tree_store, GtkTreeIter *iter, GtkTreeIter *parent);**
      - 새로운 행을 tree_store 에 추가하고 GtkIter 획득
    - **void gtk_tree_store_set (GtkTreeStore *store, GtkTreeIter *iter, …);**
      - 트리 모델의 특정 행에 자료 설정
- **GtkTreeView :** 트리 뷰와 리스트 뷰
  - 생성
    - **GtkWidget * gtk_tree_view_new_with_model (GTK_TREE_MODEL(store));**
  - 사용 함수
    - **void gtk_tree_view_insert_column_with_attributes (GtkTreeView *tree_view, gint position, const gchar *title, GtkCellRenderer *cell, ...);**
      - 트리 뷰에 새로운 열을 추가
- **GtkTreeViewColumn :** 리스트나 트리의 열을 표현
- **GtkCellRenderer :** 그리기 셀을 제어
  - 텍스트 셀**(GtkCellRendererText)**, 픽스맵 그래픽 셀**(GtkCellRendererPixBuf)**, 토글 버튼 셀**(GtkCellRendererToggle)** 의 하위 클래스 포함
  - 생성
    - **GtkCellRenderer * gtk_cell_renderer_[text|pixbuf|toggle]_new (void);**

GtkTreeView Widget

List 예제 프로그램

```
#include <gtk/gtk.h>
/* This enum is useful to label the columns */
enum
{
  FIRST_COLUMN,
  SECOND_COLUMN,
  N_COLUMNS
};

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkListStore *store;
  GtkWidget *view;
  GtkTreeIter iter;
  GtkCellRenderer *renderer;
  int i;

  gtk_init(&argc, &argv);
  window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_window_set_default_size ( GTK_WINDOW(window),
      300, 200);
  g_signal_connect ( GTK_OBJECT (window), “destroy”,
      GTK_SIGNAL_FUNC ( closeApp), NULL);
```

```
  store = gtk_list_store_new (N_COLUMNS,
      G_TYPE_STRING, G_TYPE_INT);
  for (i = 0; i < 5; i++) {
    gtk_list_store_append (store, &iter);
    gtk_list_store_set (store, &iter,
                  FIRST_COLUMN, "This is a row",
                  SECOND_COLUMN, i, -1);
  }
  view = gtk_tree_view_new_with_model
      (GTK_TREE_MODEL (store));
  renderer = gtk_cell_renderer_text_new ();
  gtk_tree_view_insert_column_with_attributes
      (GTK_TREE_VIEW(view), 0, "Title", renderer, “text”,
      FIRST_COLUMN, NULL);
  gtk_tree_view_insert_column_with_attributes
      “GTK_TREE_VIEW(view), 1, "text", renderer, “text”,
      SECOND_COLUMN, NULL);
  gtk_container_add(GTK_CONTAINER(window), view);
  gtk_widget_show_all(window);
  gtk_main ();

  return 0;
}
```



GtkTreeView Widget

Tree 예제 프로그램

```
#include <gtk/gtk.h>
/* This enum is useful to label the columns */
enum {
  COLUMN_TITLE,
  COLUMN_ARTIST,
  COLUMN_CATALOGUE,
  N_COLUMNS
};

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkTreeStore *store;
  GtkWidget *view;
  GtkTreeIter parent_iter, child_iter;
  GtkCellRenderer *renderer;

  gtk_init(&argc, &argv);
  window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_window_set_default_size ( GTK_WINDOW(window),
     300, 200);
```

```
g_signal_connect (GTK_OBJECT (window), “destroy”,
    GTK_SIGNAL_FUNC ( closeApp), NULL);

 /* Create the tree store, passing in the number and
    type of each column*/
store = gtk_tree_store_new (N_COLUMNS,
    G_TYPE_STRING,  G_TYPE_STRING,
    G_TYPE_STRING);

/*  Create a new top-level row and acquire the row
    iterator */
gtk_tree_store_append (store, &parent_iter, NULL);

/* Add some data to the row */
gtk_tree_store_set (store, &parent_iter,
    COLUMN_TITLE, "Dark Side of the Moon",
            COLUMN_ARTIST, "Pink Floyd",
            COLUMN_CATALOGUE, "B000024D4P",-1);
/* Next create a child row, acquiring the child row
    iterator */
gtk_tree_store_append (store, &child_iter,
    &parent_iter);

/* Set the child row data */
gtk_tree_store_set (store, &child_iter,
    COLUMN_TITLE, "Speak to Me", -1);

view = gtk_tree_view_new_with_model
    (GTK_TREE_MODEL (store));
```

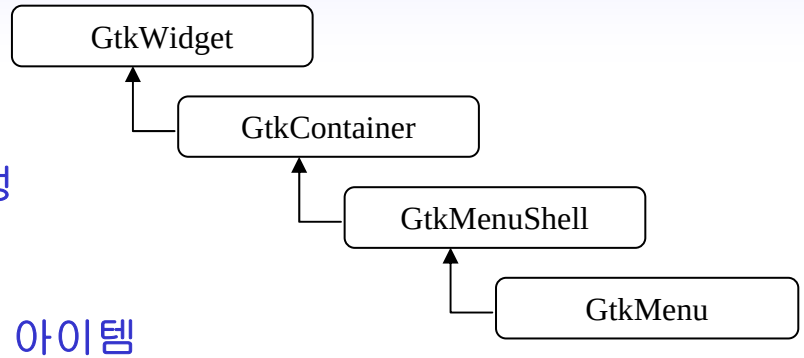
GtkTreeView Widget

Tree 예제 프로그램

```
  /* Finally we set the column datasource and titles */
  renderer = gtk_cell_renderer_text_new ();
  gtk_tree_view_insert_column_with_attributes
     (GTK_TREE_VIEW(view), COLUMN_TITLE, "Title",
     renderer, "text", COLUMN_TITLE, NULL);
  gtk_tree_view_insert_column_with_attributes
     (GTK_TREE_VIEW(view), COLUMN_ARTIST, "Artist",
     renderer, "text", COLUMN_ARTIST, NULL);
  gtk_tree_view_insert_column_with_attributes
     (GTK_TREE_VIEW(view),
     COLUMN_CATALOGUE, "Catalogue", renderer, "text",
     COLUMN_CATALOGUE, NULL);

  gtk_container_add(GTK_CONTAINER(window), view);
  gtk_widget_show_all(window);
  gtk_main ();

  return 0;
}
```

GtkMenu Widget

- 메뉴를 만듦
- 프로그래밍 방법
  - 직접 메뉴와 메뉴 아이템 각각을 생성
  - **GtkItemFactory** 를 사용하여 쉽게 생성
    - **GNOME** 메뉴 생성과 비슷



- 메뉴 관련 위짓 종류
  - **GtkMenuItem** – 사용자가 선택할 메뉴 아이템
  - **GtkMenu** – 메뉴 아이템들을 포함하는 컨테이너인 메뉴
  - **GtkMenuBar** – 개별 메뉴들을 포함하는 컨테이너인 메뉴바
- 생성
  - **GtkWidget *gtk_menu_bar_new(void);**
  - **GtkWidget *gtk_menu_new(void);**
  - **GtkWidget *gtk_menu_item_new(void);**
  - **GtkWidget *gtk_menu_item_new_with_label(const char *label);**
- 사용 함수
  - **void gtk_menu_item_set_submenu(GtkMenuItem *menu_item, GtkWidget *submenu);**
    - 메뉴 아이템에 서브메뉴를 설정
  - **void gtk_menu_bar_append(GtkMenuBar *menu_bar, GtkWidget *menu_item);**
    - 메뉴바에 메뉴 아이템을 추가 **(**또는 **gtk_container_add()** 도 가능**)**

GtkMenu Widget

GtkMenu 예제 프로그램

```
#include <gtk/gtk.h>
#include <gtk/gtk.h>
#include <stdio.h>

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

/* We define a callback for a menu item */
void item_clicked(GtkWidget *widget, gpointer
     user_data)
{
  printf("Item Clicked!\n");
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *menubar, *toplevelitem;
  GtkWidget *menu, *item1, *item2;
  GtkWidget *submenu, *subitem;
  GtkWidget *vbox;

  gtk_init(&argc, &argv);

  window =  gtk_window_new
     (GTK_WINDOW_TOPLEVEL);
```

```
gtk_window_set_title (GTK_WINDOW(window), "Gtk
    Menus");
gtk_window_set_position (GTK_WINDOW(window),
    GTK_WIN_POS_CENTER);
gtk_window_set_default_size (GTK_WINDOW
    (window), 300, 200);
gtk_signal_connect ( GTK_OBJECT (window),
    “destroy”, GTK_SIGNAL_FUNC (closeApp), NULL);

vbox = gtk_vbox_new ( FALSE, 0);
menubar = gtk_menu_bar_new ();
gtk_box_pack_start (GTK_BOX (vbox), menubar,
    FALSE, FALSE, 0);

toplevelitem = gtk_menu_item_new_with_mnemonic
    ("Toplevel item");
gtk_container_add (GTK_CONTAINER (menubar),
    toplevelitem);

menu = gtk_menu_new ();
gtk_menu_item_set_submenu (GTK_MENU_ITEM
    (toplevelitem), menu);

item1 = gtk_menu_item_new_with_mnemonic
    ("Menu item 1");
gtk_container_add (GTK_CONTAINER (menu),
    item1);
item2 = gtk_menu_item_new_with_mnemonic
    ("Menu item 2");
gtk_container_add (GTK_CONTAINER (menu),
    item2);
```

GtkMenu Widget

GtkMenu 예제 프로그램

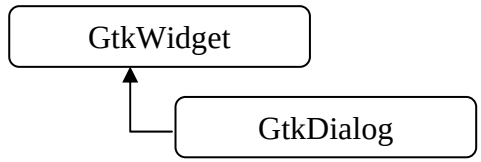
```
  submenu = gtk_menu_new ();
  gtk_menu_item_set_submenu (GTK_MENU_ITEM
      (item2), submenu);

  subitem = gtk_menu_item_new_with_mnemonic
      ("Submenu item");
  gtk_container_add (GTK_CONTAINER (submenu),
      subitem);

  g_signal_connect (GTK_OBJECT (subitem), "activate",
                GTK_SIGNAL_FUNC(item_clicked), NULL);

  gtk_container_add (GTK_CONTAINER(window), vbox);

  gtk_widget_show_all(window);
  gtk_main ();

  return 0;
}
```

대화 상자 (Dialog)

Dialog

- 사용자와 대화하고 중요한 이벤트를 사용자에게 알리는 임시 윈도우
- 대화 상자의 종류
  - 모달 대화 상자 – 사용자 응답을 받기 전에 모든 작동을 차단
  - 넌모달 대화 상자 – 대화 상자를 다른 윈도우처럼 처리

GtkDialog Widget



- 윈도우 영역의 분리
  - 위짓 내용을 위한 영역
  - 내용 아래에 있는 버튼 영역
- 생성
  - **GtkWidget * gtk_dialog_new_with_buttons (const gchar *title, GtkWindow *parent, GtkDialogFlags flags, const gchar *first_button_text, …);**
    - Parent : 응용 프로그램의 주 윈도우
    - Flags ; 대화 상자의 속성의 조합을 결정, GTK_DIALOG_MODAL, GTK_DIALOG_DESTROY_WITH_PARENT, GTK_DIALOG_NO_SEPARATOR 등

모달 대화 상자

- 사용자 응답을 받기 전에 모든 작동을 차단
  - 심각한 문제나 에러나 경고 메시지를 보고할 때 유용
- 사용 방법
  - **GTK_DIALOG_MODAL** 플래그를 설정하고 **gtk_widget_show**
  - **gtk_dialog_run** 은 대화 상자가 처리될 때까지 프로그램 실행 정지
    - 사용자가 버튼을 누르면 어떤 버튼을 눌렀는지에 대한 정수**(int)** 값 반환

```
typedef enum {
  GTK_RESPONSE_NONE = -1,
  GTK_RESPONSE_REJECT = -2,
  GTK_RESPONSE_ACCEPT = -3,
  GTK_RESPONSE_DELETE_EVENT = -4,
  GTK_RESPONSE_OK     = -5,
  GTK_RESPONSE_CANCEL = -6,
  GTK_RESPONSE_CLOSE  = -7,
  GTK_RESPONSE_YES    = -8,
  GTK_RESPONSE_NO     = -9,
  GTK_RESPONSE_APPLY  = -10,
  GTK_RESPONSE_HELP   = -11
} GtkResponseType;
```

```
GtkWidget *dialog = gtk_dialog_new..
int result = gtk_dialog_run(GTK_DIALOG(dialog));
switch (result) {
case GTK_RESPONSE_ACCEPT:
    delete_file();
    break;
case GTK_RESPONSE_REJECT:
    do_nothing();
    break;
default:
    break;
}
gtk_widget_destroy(dialog);
```

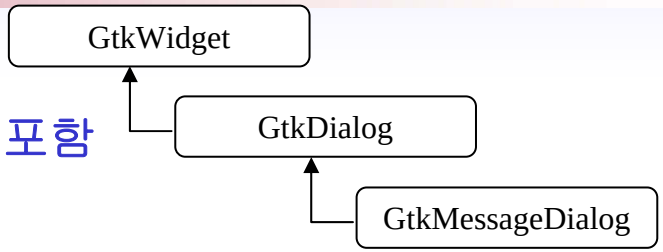
넌모달 대화 상자

- 대화 상자를 다른 윈도우처럼 처리
  - 사용자가 즉시 응답할 필요가 없음
  - 대화 상자를 여러 번 여는 상황 고려해야 함
- 사용 방법
  - **gtk_dialog_run** 대신 콜백 함수를 **GtkDialog** 응답 신호에 연결
    - 사용자가 버튼을 누르면 어떤 버튼을 눌렀는지에 대한 정수**(int)** 값 반환

```
void dialog_clicked (GtkWidget *dialog, gint
result, gpointer user_data)
{
    switch (result) {
    case GTK_RESPONSE_ACCEPT:
        delete_file();
        break;
    case GTK_RESPONSE_REJECT:
        do_nothing();
        break;
    default:
       break;
    }
    gtk_widget_destroy(dialog);
}
```

```
int main()
{
...
GtkWidget *dialog = gtk_dialog_new...

g_signal_connect (GTK_OBJECT(dialog), “response”,
GTK_SIGNAL_FUNC (dialog_clicked), user_data));
...
gtk_widget_show(dialog);
...
}
```

GtkMessageDialog



- 매우 간단한 대화 상자
  - 아이콘과 제목, 구성할 수 있는 버튼 포함
- 생성
  - **GtkWidget * gtk_message_dialog_new (GtkWindow *parent, GtkDialogFlags flags, GtkMessageType type, GtkbuttonsType buttons, const gchar *message_format, …);**
    - **type:** 메시지 대화상자의 형식, **GTK_MESSAGE_INFO, GTK_MESSAGE_WARNING, GTK_MESSAGE_QUESTION, GTK_MESSAGE_ERROR** 등
    - **buttons:** 버튼 종류

GtkButtonsType	설명	GtkButtonsType	설명
GTK_BUTTONS_OK	확인 버튼	GTK_BUTTONS_YES_NO	예/아니오 버튼
GTK_BUTTONS_CLOSE	닫기 버튼	GTK_BUTTONS_OK_CANCEL	확인/취소 버튼
GTK_BUTTONS_CANCEL	취소 버튼	GTK_BUTTONS_NONE	버튼 없음

    - **message_format:** 대화 상자의 텍스트, **printf** 사용과 비슷하게 사용 가능

GtkMessageDialog Widget

GtkMessageDialog 예제 프로그램

```
#include <gtk/gtk.h>
#include <stdio.h>

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

int main (int argc, char *argv[])
{
  GtkWidget *window;
  GtkWidget *dialog;
  char filename[15] = "important.txt";
  gint result;

  gtk_init(&argc, &argv);
  window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_window_set_default_size ( GTK_WINDOW(window),
      200, 80);
  g_signal_connect ( GTK_OBJECT (window), "destroy",
                  GTK_SIGNAL_FUNC ( closeApp), NULL);

  dialog = gtk_message_dialog_new
      (GTK_WINDOW(window),
              GTK_DIALOG_DESTROY_WITH_PARENT,
              GTK_MESSAGE_QUESTION,
              GTK_BUTTONS_YES_NO,
             "Are you sure you wish to delete %s?",
              filename);
result = gtk_dialog_run (GTK_DIALOG (dialog));
  gtk_widget_destroy(dialog);
  switch (result) {
  case GTK_RESPONSE_YES:
    printf("you selected Yes.\n");
    break;
  case GTK_RESPONSE_NO:
    printf("you selected No.\n");
    break;
  default:
    printf("you didn't selected.\n");
  }
  gtk_widget_show(window);
  gtk_main ();

  return 0;
}
```

GNOME Widget

- **GNOME Widget**
  - **GTK+** 는 기본적인 위젯만 제공하므로, 데스크톱에 유용한 기능(파일 선택 창, 도움말 창 등) 부족
  - **GNOME** 위젯은 **GTK+** 위젯을 확장하여 더욱 사용하기 쉽게 함
- **GNOME** 라이브러리
  - **GNOME** 위젯 포함
  - **libgnome-2.0, libgnomeui-2.0**
- **GNOME** 프로그래밍 순서
  - 환경 초기화 – **gnome_program_init**
    - **GnomeProgram * gnome_program_init (const char *app_id, const char *app_version, const GnomeModuleInfo *module_info, int argc, char **argv, const char *first_property_name, …);**
  - **GnomeApp** 위젯 생성
  - …

GNOME Widget

▣ 기본 GNOME 응용 프로그램

```
#include <gnome.h>

int main( int argc, char *argv[] )
{
    GtkWidget *app;

    gnome_program_init ("gnome1", "1.0", LIBGNOMEUI_MODULE, argc,
      argv, NULL);
    app = gnome_app_new ("gnome1", "The Window Title");
    gtk_widget_show (app);
    gtk_main ();

    return 0;
}
```

◈ GNOME 프로그램 컴파일

```
$ gcc -Wall -g gnome1.c -o gnome1 `pkg-config --cflags –libs libgnome-2.0
libgnomeui-2.0`
```

GNOME Menu

▣ 드롭다운 메뉴는 GnomeUIInfo 구조체의 배열로 표현

◈ GnomeUIInfo 구조체

```
Typedef struct {
    GnomeUIInfoType type;
    gchar const *label;
    gchar const *hint;
    gpointer moreinfo;
    gpointer user_data;
    gpointer unused_data;
    GnomeUIPixmapType pixmap_type;
    gconstpointer pixmap_info
    guint accelerator_key;
    GdkModifierType ac_mode;
    GtkWidget *widget;
} GnomeUIInfo;
```

```
typedef enum {
        GNOME_APP_UI_ENDOFINFO,
        GNOME_APP_UI_ITEM,
        GNOME_APP_UI_TOGGLEITEM,
        GNOME_APP_UI_RADIOITEMS,
        GNOME_APP_UI_SUBTREE,
        GNOME_APP_UI_SEPARATOR,
        GNOME_APP_UI_HELP,
        GNOME_APP_UI_BUILDER_DATA,
        GNOME_APP_UI_ITEM_CONFIGURABLE,
        GNOME_APP_UI_SUBTREE_STOCK,
        GNOME_APP_UI_INCLUDE
} GnomeUIInfoType;
```

- **type:** 메뉴 항목의 형식
- **user_data** – 콜백 함수에 전달된 임의의 포인터
- **pixmap_type, pixmap_info** – 비트맵 아이콘 추가
- **accelerator_key, ac_mode** – 키보드 단축키 정의

GNOME Menu

GnomeUIInfo 구조체에서 매크로 사용하면 편리

개별적인 메뉴 항목에 대한 매크로

```
#define <libgnomeui/libgnomeui.h>

#define     GNOMEUIINFO_MENU_OPEN_ITEM          (cb, data)
#define     GNOMEUIINFO_MENU_SAVE_ITEM          (cb, data)
#define     GNOMEUIINFO_MENU_SAVE_AS_ITEM       (cb, data)
#define     GNOMEUIINFO_MENU_PRINT_ITEM         (cb, data)
#define     GNOMEUIINFO_MENU_PRINT_SETUP_ITEM   (cb, data)
#define     GNOMEUIINFO_MENU_CLOSE_ITEM         (cb, data)
#define     GNOMEUIINFO_MENU_QUIT_ITEM          (cb, data)
#define     GNOMEUIINFO_MENU_CUT_ITEM           (cb, data)
....
```

최상위 수준 메뉴 항목에 대한 매크로

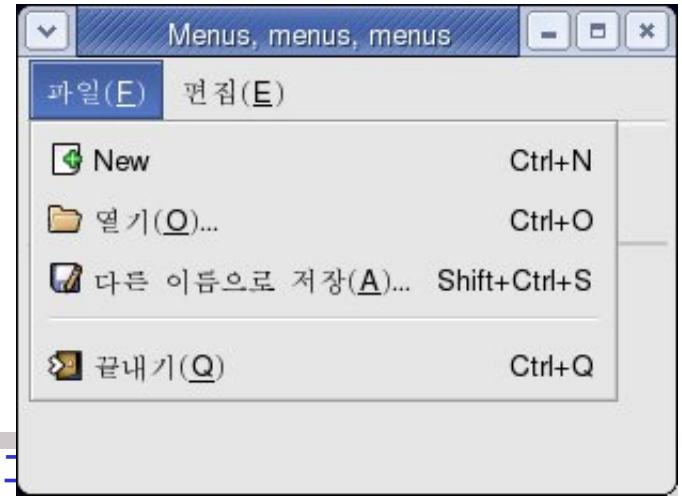
```
#define     GNOMEUIINFO_MENU_FILE_TREE          (tree)
#define     GNOMEUIINFO_MENU_EDIT_TREE          (tree)
#define     GNOMEUIINFO_MENU_VIEW_TREE          (tree)
#define     GNOMEUIINFO_MENU_SETTINGS_TREE      (tree)
#define     GNOMEUIINFO_MENU_FILES_TREE         (tree)
#define     GNOMEUIINFO_MENU_WINDOWS_TREE       (tree)
#define     GNOMEUIINFO_MENU_HELP_TREE          (tree)
#define     GNOMEUIINFO_MENU_GAME_TREE          (tree)
```

GNOME Menu

GNOME메뉴 예제 프로그램

```
#include <gnome.h>

void closeApp ( GtkWidget *window, gpointer data)
{
  gtk_main_quit();
}

/* The macros provide stock icons and keyboard
     shortcuts */
static GnomeUIInfo filemenu[] = {
  GNOMEUIINFO_MENU_NEW_ITEM ("New", "Menu
     Hint", NULL, NULL ),
  GNOMEUIINFO_MENU_OPEN_ITEM (NULL, NULL),
  GNOMEUIINFO_MENU_SAVE_AS_ITEM (NULL, NULL),
  GNOMEUIINFO_SEPARATOR,
  GNOMEUIINFO_MENU_EXIT_ITEM (NULL, NULL),
  GNOMEUIINFO_END
};

static GnomeUIInfo editmenu[] = {
  GNOMEUIINFO_MENU_FIND_ITEM (NULL, NULL),
  GNOMEUIINFO_END
};

static GnomeUIInfo menubar[] = {
  GNOMEUIINFO_MENU_FILE_TREE (filemenu),
  GNOMEUIINFO_MENU_EDIT_TREE (editmenu),
  GNOMEUIINFO_END
};
```

```
int main (int argc, char *argv[])
{
  GtkWidget *app;

  gnome_program_init ("gnome1", "0.1",
     LIBGNOMEUI_MODULE, argc, argv,
     GNOME_PARAM_NONE);
  app = gnome_app_new("gnome1", "Menus, menus,
     menus");
  gtk_window_set_default_size ( GTK_WINDOW(app),
     300, 200);
  gtk_signal_connect ( GTK_OBJECT (app), "destroy",
              GTK_SIGNAL_FUNC ( closeApp), NULL);

  gnome_app_create_menus ( GNOME_APP(app),
     menubar);
  gtk_widget_show(app);

  gtk_main();
  return 0;
}
```



Glade

- **Glade**
  - **GTK+** 와 **GNOME** 라이브러리를 사용하는 **GUI** 를 생성하는 도구
  - 현재 **Glade-2 (2.x** 버전) 사용
- **Glade** 를 사용한 **GTK+** 프로그램 개발 단계
  - **Glade** 를 사용하여 인터페이스 제작
  - **Glade** 를 사용하여 소스코드 빌드
  - 소스 코드를 필요에 알맞게 편집
  - 프로젝트 파일들을 컴파일

Glade

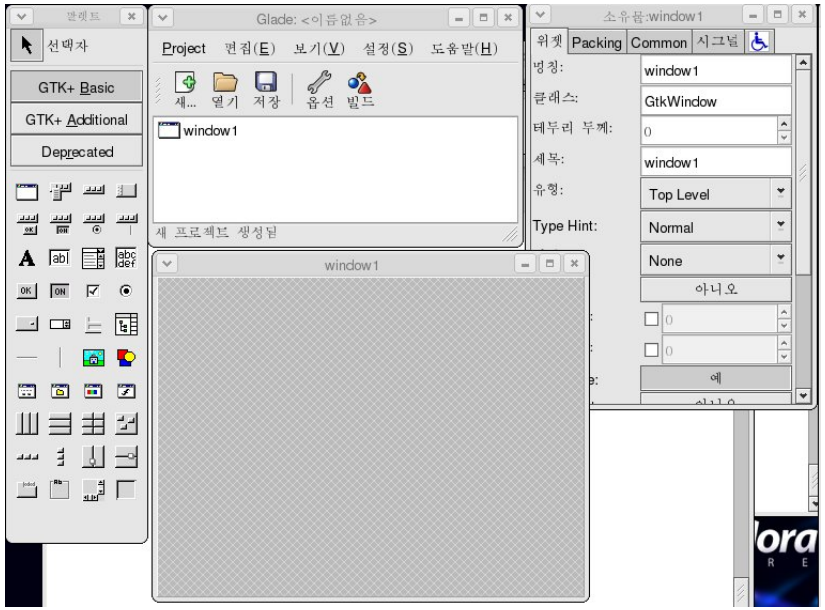
- 구성
  - **Main Window**
    - 기본적인 주요 기능을 포함하는 최상위 윈도우
    - 메뉴(파일,편집,보기,도움말), 툴바(열기, 저장, 옵션, 코드 빌드), 최상위 윈도우 목록 창
  - **Palette**
    - 다양한 **GTK+, GNOME** 위짓 선택 창
  - **Properties Editer**
    - 선택한 위짓에 대한 속성 설정, 시그널에 대한 콜백 설정
  - **Widget Tree**
    - 특정 최상위 윈도우에 대한 하위 트리에 속한 위짓들 목록 표시
  - **Menu Editor**
    - 메뉴바에서 선택 가능하며 메뉴를 자유롭게 편집하는 윈도우
- 코드 생성
  - **Autoconf, automake** 를 위한 여러 파일들 **(Makefile.am, autogen.sh, configure.in, macros/, po/, acconfig.h …)**
  - 소스 파일들
    - 주요 루틴 수행 파일 **(main.c)** – 한 번 생성 후 덮어쓰지 않음
    - 인터페이스 생성 파일 **(interface,h, interface.c)**
    - 신호 처리 및 콜백 함수 **(callback.h, callbacks.c)** – 한 번 생성 후 덮어쓰지 않음
    - 지원 파일 **(support.c)**

Glade 사용 예제

- **Glade** 를 이용한 프로그래밍 예제
  - **Glade-2** 실행

```
$ glade-2
```

  - 새 프로젝트→**New GTK+** 프로젝트 선택
  - 팔레트에서 **Window(**윈도우**)** 선택
    - **"window 1"** 주 윈도우 생성 확인



Glade 사용 예제

▣ Glade 를 이용한 프로그래밍 예제 계속

◈ 인터페이스 생성

➢ 팔레트에서 **Vertical Box(세로박스)** 선택한 후, 주 윈도우에서 클릭

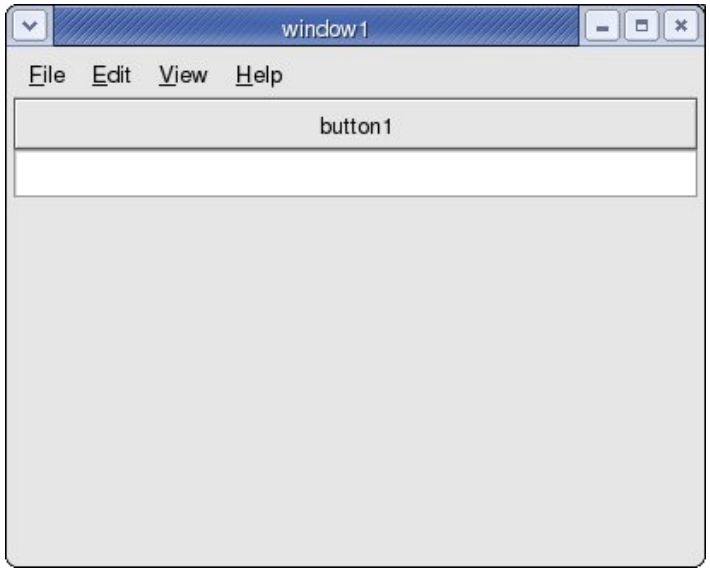
– 주 윈도우 창이 세로로 세 등분되는 것 확인

➢ 기타 위짓 추가

– Menubar(메뉴바) → 주 윈도우 최상위 등분 영역

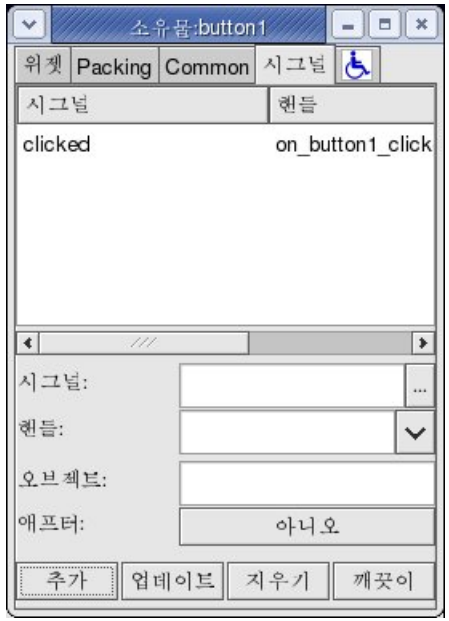
– Button(버튼) → 주 윈도우 둘째 등분 영역

– Text Entry(텍스트 엔트리) → 주 윈도우 최하위 등분 영역



Glade 사용 예제

- **Glade** 를 이용한 프로그래밍 예제 계속
  - **Signal/Callback** 연결
    - **"button1"** 버튼을 선택한 후 **Properties** 창에서 **Signal(**시그널**)** 탭 선택
    - "시그널:" 라벨 옆의 **"…"** 버튼 선택 후, 팝업 창에서 **"clicked"** 선택 후 확인
    - **Add(**추가**)** 버튼 클릭



  - 저장 - **Save(**저장**)** 후 옵션 창에서 디렉토리 설정 후 디폴트 옵션으로 저장
  - 소스코드 빌드 - **Build(**빌드**)** 버튼을 클릭
  - 끝내기 - **Project(**프로젝트**)** 메뉴에서 **Quit(**끝내기**)** 버튼 클릭

Glade 사용 예제

Glade 를 이용한 프로그래밍 예제 계속

소스 파일 수정 : interface.c

```
...
g_signal_connect ((gpointer) button1, "clicked", G_CALLBACK
(on_button1_clicked), entry1));
...
```

소스 파일 수정 : callbacks.c

```
...
void
on____________q_1_activate            (GtkMenuItem     *menuitem,
                                       gpointer         user_data)
{
      gtk_main_quit();
}
...
void
on_button1_clicked                    (GtkButton       *button,
                                       gpointer         user_data)
{
      const char *hello = "Hello, World!";

      gtk_entry_set_text (GTK_ENTRY ((GtkWidget *) user_data), hello);
}
```

Glade 사용 예제

▣ Glade 를 이용한 프로그래밍 예제 계속

◈ Makefile 생성 및 make

```
$ cd ~userid/Projects/project1
$ ./autogen.sh
$ make
```

◈ 실행

```
$ cd src
$ ./project1
```

Qt 프로그래밍

Qt

- 멀티플랫폼 **GUI** 응용을 위한 **C++ GUI Toolkit**
  - 객체 지향 프로그래밍 구현 – 시그널과 슬롯 등 제공
  - 풍부한 위젯**(Widget)** 제공
  - 다양한 플랫폼 지원 – **Windows, Mac OS X, Linux, Solaris,HP-UX**
  - **Unicode** 와 **i18n** 등의 다양한 언어 지원
- 유용한 도구 제공
  - **Qt Designer(GUI** 빌더**) , Qt Linguist, qmake** 등
- 대표적인 리눅스 데스크탑 환경인 **KDE** 의 기반
  - **Qt/X11** 사용
- 노르웨이의 **Trolltech** 사에서 개발
- 다양한 배포판으로 제공됨
  - **Qt Enterprise Edition and Qt Professional Edition**
  - **Qt Free Edition**
  - **Qt Embedded Edition**

Qt/Embedded 와 Qtopia

- 임베디드 **GUI** 와 응용 개발을 위한 **C++ GUI Toolkit**
  - 임베디드 리눅스와 함께 다양한 프로세서 상에서 실행
  - **X** 윈도우 시스템을 사용하지 않고 프레임버퍼에 직접 접근
  - 표준 **Qt API** 를 사용하므로 윈도우와 유닉스 상에서도 개발 가능
- **PDA** 등의 임베디드 응용 환경인 **Qtopia** 의 기반
  - 모바일 기기를 위한 윈도우 환경과 응용 슈트
    - **Qt/Embedded** 기반의 윈도우 매니저 **(Window Manager)**
    - **MS-Windows**처럼 윈도우 상의 아이콘을 클릭하여 **Qt/Embedded** 응용 프로그램을 실행할 수 있는 환경
    - 효율적인 메모리 사용 – **6~8MB**
    - 오픈 소스, 전용성, 통합성 제공
  - **Qtopia** 제공 응용 프로그램들
    - 개인 정보 관리**(Personal Information Management)**, 인터넷 컨텐츠, **Entertainment,** 다중 플랫폼 **OS**와의 동기화**(synchronization)** 소프트웨어 등
    - **Address Book, Calculator, Calendar, To do List, text editor, MPEG and MP3 players, international city clock, asteroid game, patience, mine hunt, File Manager, Go, etc.**

Qt/Embedded 적용 사례

- **Sharp Zaurus PDA**

The Sharp Zaurus PDA using Qt/Embedded

- **Motorola Smart Phones**

Qt 의 구조

▣ 메모리 효율성

◈ **X Windows** 를 전혀 이용하지 않고 커널 프레임버퍼에 직접 접근

Qt Application			
Qt API			
Qt/Windows	Qt/X11	Qt/Macintosh	Qt/Embedded
GDI	Xlib	Carbon	Embedded Linux
M$-Windows	Unix/Linux	Mac OS X	

◈ 가속 기능 최적화로
속도가 느린 단점 보완

◈ 라이브러리 크기 조정 가능

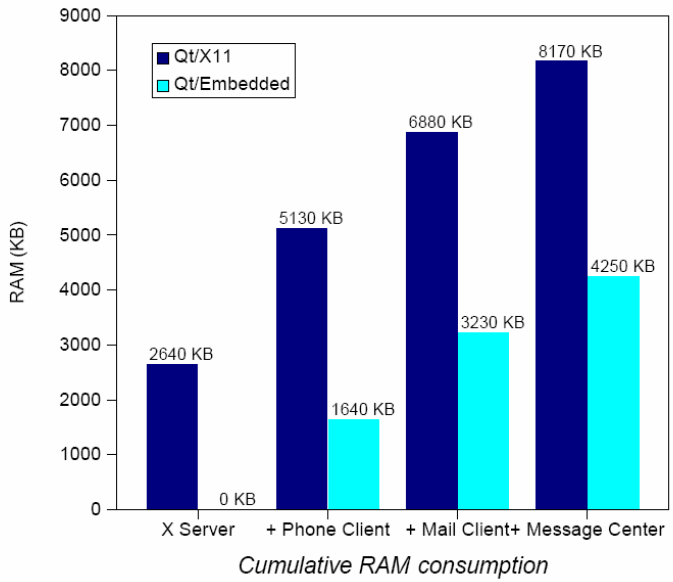


Figure 1. Memory comparison between Qt/X11 and Qt/Embedded for Ericsson's screen phone

Qt 의 구조

- 윈도우 시스템
  - 서버 프로세스와 나머지 클라이언트들로 구성
  - 서버 프로세스
    - 입력 메소드(IME) 제공하거나 클라이언트 시동 사용자 인터페이스 제공
  - 클라이언트
    - 서버와 공유 메모리를 통해 통신
    - 서버 프로세스가 개입하지 않고 프레임버퍼에 직접 씀
  - **QCOP** 채널을 통해 메시지 교환
  - **Qprocess** 의 비동기 **IPC** 메커니즘 제공
  - 다양한 폰트 포맷 지원
    - **TTF(TrueType Fonts), PostScript Type1, BDF(Bitmap Distribution Format), QPF(Qt Pre-rendered Fonts)**

Signal 과 Slot

- **Signal** 과 **Slot** 매커니즘으로 객체 간 통신
  - C++ 로 구현된 빠르고, 유연하고 완전한 객체 지향 구조
  - 기존의 콜백 함수 이용 방식의 단점 개선
    - 콜백 함수 이용 방식은 특정 이벤트와 처리 코드 포인트를 연결
    - 콜백 함수는 클래스 독립적인 구조에 적합하지 않음
    - 또한, 매개변수에 대한 타입체크가 없음
  - 사용 방법
    - 미리 시그널을 처리할 함수(슬롯)을 만들고 **connect()** 함수를 이용하여 시그널과 슬롯을 연결
    - 이벤트가 발생하면 발생 위젯이 시그널을 만들고 해당 슬롯에 의해 처리
    - 매개변수에 대한 타입 체크, 즉 **Type safe**
    - 예) **Quit** 버튼의 **clicked()** 시그널과 **quit()** 슬롯을 연결하여 사용자가 **Quit** 버튼을 클릭하면 응용 프로그램이 종료
    - ➔ **connect (button, SIGNAL(clicked()), qApp, SLOT(quit()));**

Signal 과 Slot

Signal 과 Slot

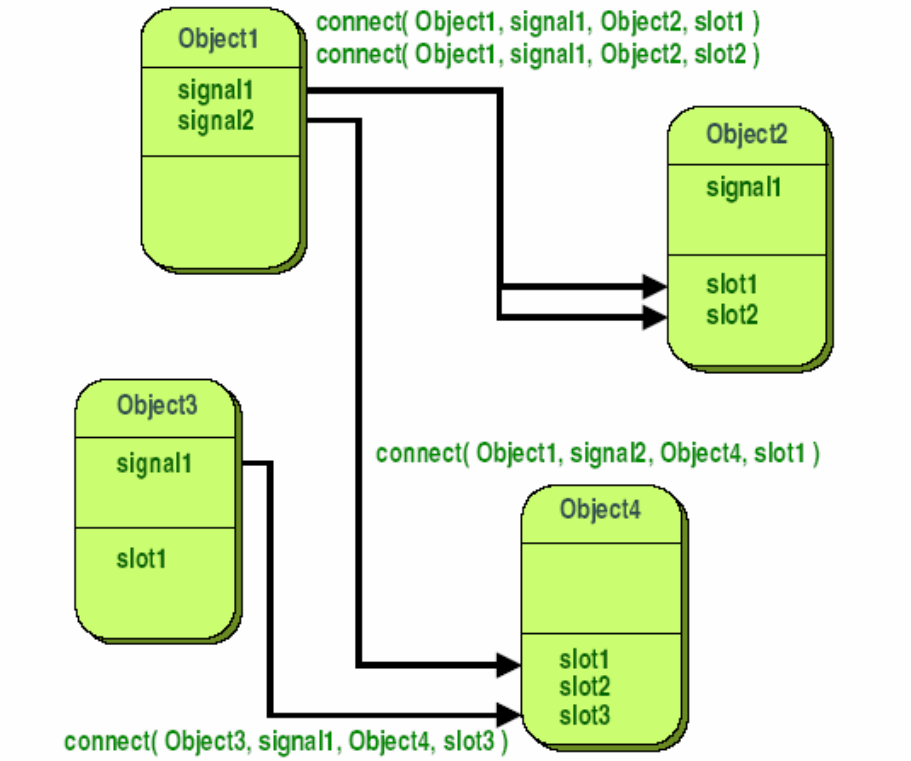


Figure 4. An abstract view of some signals and slots connections

Signal 과 Slot

▣ 간단한 Signal 과 Slot 예제

```
class BankAccount : public QObject
    {
    Q_OBJECT
public:
    BankAccount() { curBalance = 0; }
    int balance() const { return curBalance; }
public slots:
    void setBalance( int newBalance );
signals:
    void balanceChanged( int newBalance );
private:
    int curBalance;
};
```

```
void BankAccount::setBalance( int newBalance )
{
    if ( newBalance != curBalance ) {
        curBalance = newBalance;
        emit balanceChanged( curBalance );
    }
}
```

balanceChanged() 시그널은 따로 구현하지 않음

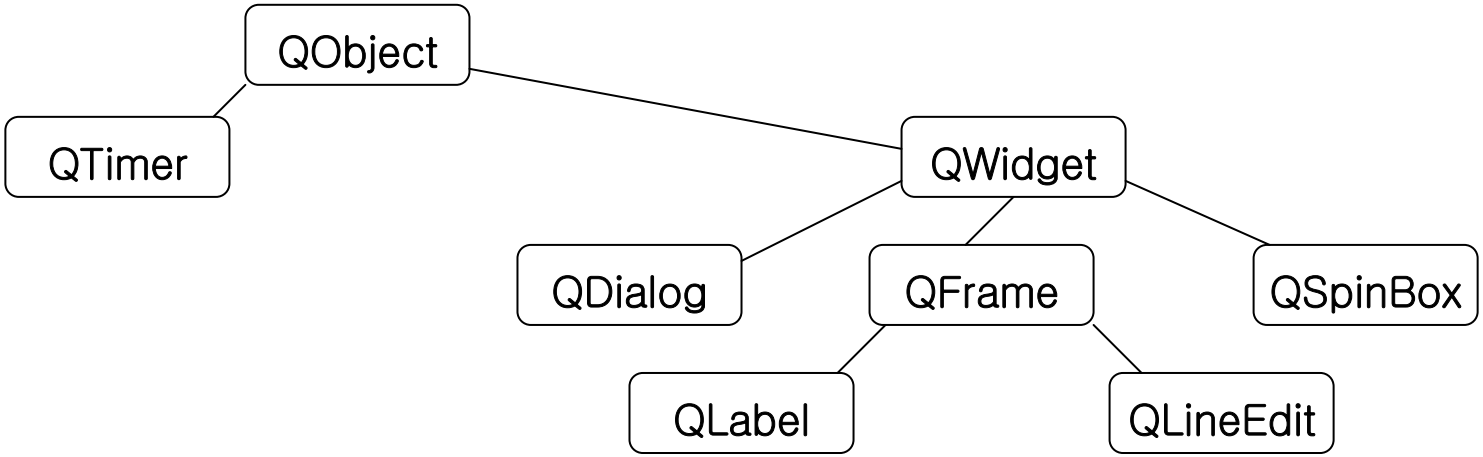
balancChanged() 시그널이 발생하여 매개변수 curBanalce 를 해당 슬롯에 전달

Signal 과 Slot

- **Meta Object Compiler (MOC)**
  - **Signal 과 slot 매커니즘을 지원**
    - **signal and slot 관련 코드를 C++ preprocessor 와 moc 가 표준 C++ 코드로 구현**
  - 응용의 헤더 파일을 읽고 **signal** 과 **slot** 을 지원하는 코드를 생성
  - **Qt** 의 **translation** 메커니즘, 속성 시스템, 실행 시간 타입 정보 지원

Widgets

- Qt 는 버튼, 스크롤바 등의 풍부한 위젯들 제공
  - 위젯들은 **QWidget** 의 인스턴스들



  - 위젯의 특성
    - 부모 위젯은 많은 수의 자식 위젯들을 포함
    - 자식 위젯들은 부모 위젯 영역 안에 표시
    - 부모 위젯이 없는 위젯은 **top-level widget** – 타이틀바와 프레임 포함
    - 자식 위젯 위치는 수동 또는 **layout manager** 에 의해 자동으로 설정
    - 위젯이 숨겨지거나 삭제되면 자식 위젯들도 같은 영향을 받음

Qt Programming

Hello Qt 예제

```
#include <qapplication.h>
#include <qlabel.h>

int main(int argc, char **argv)
{
  QApplication app(argc,argv);
  QLabel *hello = new QLabel( "<font color=blue>Hello"
                              " <i>Qt World!</i></font>", 0 );
  app.setMainWidget(window);
  hello->show();
  return app.exec();
}
```

Qt Programming

기본 Qt 프로그램의 개발 단계

- 클래스 정의
  - 인터페이스와 신호/슬롯 관계를 고려하여 생성자, 메소드 등 정의
- 인터페이스 제작
  - 위젯 생성 및 속성 설정
  - 위젯 계층 정의
- 신호와 슬롯 작성
  - 슬롯 루틴을 작성하여 신호와 연결
- Main 함수 작성
  - 주요 윈도우의 인스턴스를 만든 다음 윈도우를 화면에 표시
- 프로그램 컴파일 및 실행
  - 메타 객체 컴파일러(moc) 를 이용하여 전처리
  - C++ 컴파일러(g++ 등) 으로 컴파일

Qt Programming

- **ButtonWindow** 예제 프로그램
  - 주 윈도우에 버튼을 표시하고 클릭하면 메시지 출력
- 클래스 정의
  - 클래스 선언을 입력하고 **ButtonWindow.h** 로 저장

```
#include <qmainwindow.h>


class ButtonWindow : public QMainWindow
{
  Q_OBJECT

  public:
    ButtonWindow(QWidget *parent = 0, const char *name = 0);
    virtual ~ButtonWindow();

  private slots:
    void Clicked();

};
```

- **QMainWindow** 로부터 확장한 클래스(**ButtonWindow**) 정의
- 생성자(**ButtonWindow()**)와 소멸자(**~ButtonWindow()**)를 정의
- 버튼을 눌러 **clicked** 신호가 발생하면 연결할 슬롯 함수(**Clicked()**) 정의

Qt Programming

▣ 인터페이스 제작

◈ 클래스를 구현하며 **ButtonWindow.cpp** 파일로 저장

◈ 헤더파일 포함 및 선언부 작성

```
#include "ButtonWindow.moc"
#include <qpushbutton.h>
#include <qapplication.h>
#include <iostream>
```

◈ 생성자 작성

➢ 윈도우 타이틀을 설정하고, 버튼 만들고, **clicked** 신호를 슬롯과 연결

```
ButtonWindow::ButtonWindow(QWidget *parent, const char *name) :
    QMainWindow(parent, name)
{
 this->setCaption("This is the window Title");
 QPushButton *button = new QPushButton("Click Me!", this, "stuff");
 button->setGeometry(50,30,70,20);
 connect (button, SIGNAL(clicked()), this, SLOT(Clicked()));
}
```

➢ **setCaption()** 함수는 윈도우 타이틀을 설정

➢ **connect ()** 함수를 사용하여 시그널과 슬롯을 연결

◈ 소멸자 작성

➢ **Qt** 는 위젯 소멸을 자동으로 관리하므로, 소멸자는 비어 있음

```
ButtonWindow::~ButtonWindow()
{
}
```

Qt Programming

◈ 슬롯 함수 구현

➢ 터미널에 메시지 출력

```
void ButtonWindow::Clicked(void)
{
  std::cout << "clicked!\n";
}
```

▣ Main 함수 작성

◈ Qt 응용 프로그램 초기화

◈ 주 윈도우 (ButtonWindow)의 인스턴스 생성

◈ 응용 프로그램의 주 윈도우로 설정

◈ 윈도우를 화면에 표시

```
int main(int argc, char **argv)
{
  QApplication app(argc,argv);
  ButtonWindow *window = new ButtonWindow();

  app.setMainWidget(window);
  window->show();
  return app.exec();
}
```

Qt Programming

▣ Qt 프로그램 컴파일

◈ 메타 객체 컴파일러(moc) 를 이용하여 전처리 실행

➢ **ButtonWindow.h** 파일에 적용하여 **ButtonWindow.moc** 파일 생성

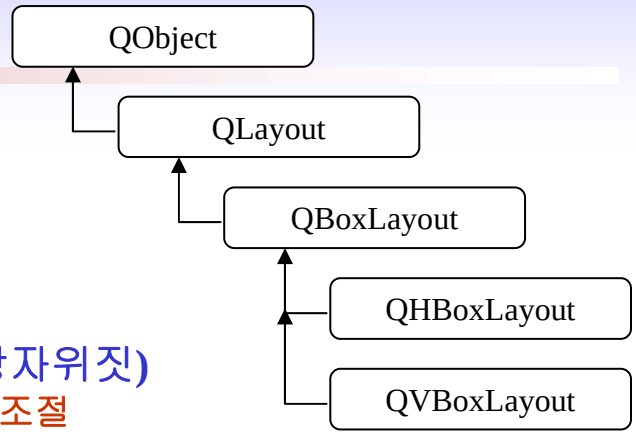
```
$ moc ButtonWindow.h –o ButtonWindow.moc
```

◈ ButtonWindow.cpp 프로그램 컴파일

```
$ g++ -o button ButtonWindow.cpp –I$QTDIR/include –L$QTDIR/lib
    –lqt-mt
```

◈ 실행

Qt Layout 객체



- 위짓 레이아웃을 정밀하게 제어할 수 있는 객체
- 종류
  - **QLayout** 클래스 – **Qobject** 로부터 파생
    - 자동으로 크기를 조절
    - 위짓이 아니므로 **QVBoxLayout** 등을 만들 수 없다.
  - **QBox** 위짓 **(QHBox:**수평상자위짓**, QVBox:**수직상자위짓**)**
    - **QWidget::resizeEvent()** 를 호출하여 수동으로 크기 조절
- 사용 방법
  - **QVBoxLayout** 과 **QHBoxLayout** 와 같은 수직/수평방향의 컨테이너 객체 사용
- 생성
  - **QBoxLayout::QBoxLayout (QWidget *parent, int margin, int spacing, const char *name);**
  - **QBoxLayout::QBoxLayout (QLayout *parentLayout, int spacing, const char *name);**
  - **QBoxLayout::QBoxLayout (int spacing, const char *name);**
    - ***margin :*** **TRUE** 이면 포함 위짓이 모두 같은 공간 차지
    - ***spacing :*** 위짓 간의 간격을 픽셀 단위로 지정
- 사용 함수
  - **QBoxLayout:addWidget (QWidget *widget, int stretch = 0, int alignment = 0);**
    - **QHBox** 의 왼쪽과 **QVBox**의 아래쪽에 위짓 추가
  - **QBoxLayout::addLayout (Qlayout *layout, int stretch = 0);**
    - **QHBox** 의 오른쪽과 **QVBox**의 위쪽에 위짓

QBoxLayout 클래스

QBoxLayout 예제 프로그램

```
#include <qmainwindow.h>

class LayoutWindow : public QMainWindow
{
  Q_OBJECT

  public:
    LayoutWindow(QWidget *parent = 0, const char
      *name = 0);
    virtual ~LayoutWindow();

};
```

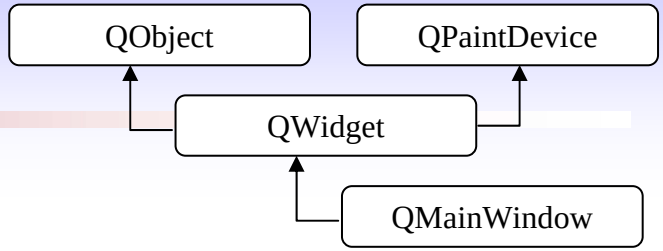
```
#include <qapplication.h>
#include <qlabel.h>
#include <qlayout.h>

#include "LayoutWindow.moc"

LayoutWindow::LayoutWindow(QWidget *parent, const
      char *name) : QMainWindow(parent, name)
{
  this->setCaption("Layouts");
  QWidget *widget = new QWidget(this);
  setCentralWidget(widget);

  QHBoxLayout *horizontal = new
     QHBoxLayout(widget);
  QVBoxLayout *vertical = new QVBoxLayout();
  QLabel* label1 = new QLabel("Top",
     centralWidget(), "textLabel1" );
  QLabel* label2 = new QLabel("Bottom", widget, "Label
     2");
  QLabel* label3 = new QLabel("Right",widget, "Label
     3");

  vertical->addWidget(label1);
  vertical->addWidget(label2);
  horizontal->addLayout(vertical);
  horizontal->addWidget(label3);
  resize( 150, 100 );
}
```

QBoxLayout 클래스

QBoxLayout 예제 프로그램 실행 결과

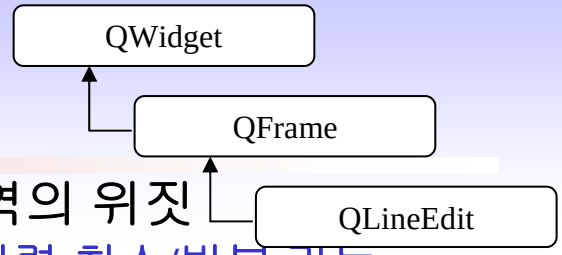
```
LayoutWindow::~LayoutWindow()
{
}

int main(int argc, char **argv)
{
  QApplication app(argc,argv);
  LayoutWindow *window = new LayoutWindow();

  app.setMainWidget(window);
  window->show();

  return app.exec();
}
```

QMainWindow Widget



- 모든 Qt 응용 프로그램의 주 프레임워크 제공
  - 위젯의 부모 클래스로서 동작 가능
  - 메뉴바, 툴바, 위젯 도킹 영역 (중앙), 상태바 영역으로 구성 가능
- 생성
  - **QMainWindow::QMainWindow (QWidget * parent = 0, Qt::WFlags flags = 0);**
- 사용 함수
  - **void void QMainWindow::addDockWidget (Qt::DockWidgetArea area, QDockWidget * dockwidget);**
  - **void QMainWindow::addToolBar (Qt::ToolBarArea area, QToolBar * toolbar);**
  - **void QMainWindow::setCentralWidget (QWidget * widget);**
  - **void QMainWindow::setCorner (Qt::Corner corner, Qt::DockWidgetArea area);**
  - **void QMainWindow::setIconSize (const QSize & iconSize);**
  - **void QMainWindow::setMenuBar (QMenuBar * menuBar);**
  - **void QMainWindow::setStatusBar (QStatusBar * statusbar);**
  - **bool QMainWindow::restoreState (const QByteArray & state, int version = 0);**
  - **QByteArray saveState (int version = 0) const ;**

QLineEdit Widget



- 사용자가 키보드로 입력할 수 있는 단일-라인 영역의 위짓
  - 편집 기능 포함하므로 텍스트 부분 선택, 자르기, 입력 취소/반복 가능
- 생성
  - **QLineEdit::QLineEdit (Qwidget *parent, const char * name = 0);**
  - **QLineEdit::QLineEdit (const QString &contents, Qwidget *parent, const char * name = 0);**
  - **QLineEdit::QLineEdit (const QString &contents, const QString &inputMask, Qwidget *parent, const char * name = 0);**
- 사용 함수
  - **void QLineEdit::setInputMask (const Qstring &inputMask);**
  - **void QLineEdit::insert (const Qstring &newText);**
  - **bool QLineEdit::isModified (void);**
  - **void QLineEdit::setMaxLength (int length);**
  - **void QLineEdit::setReadOnly (bool read);**
  - **void QLineEdit::setText (const QString &test);**
  - **QString QLineEdit::text (void);**
  - **void set QLineEdit::EchoMode (EchoMode mode);**
    - 텍스트가 위짓에 나타나는 방식 결정, **QLineEdit::Normal, QLineEdit::Password, QLineEdit::NoEcho(**아무것도 표시않음**)** 가능

QLineEdit Widget

QLineEdit 예제 프로그램

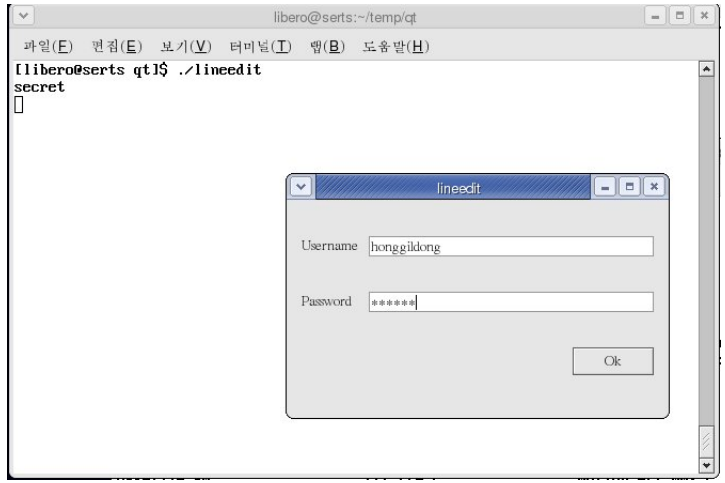
```
#include <qmainwindow.h>
#include <qlineedit.h>
#include <qstring.h>

class LineEdit : public QMainWindow
{
  Q_OBJECT

  public:
    LineEdit(QWidget *parent = 0, const char *name =
      0);
    QLineEdit *username_entry;
    QLineEdit *password_entry;

  private slots:
    void Clicked();
};
```

```
#include "LineEdit.moc"
#include <qpushbutton.h>
#include <qapplication.h>
#include <qlabel.h>
#include <qlayout.h>
#include <qstring.h>
#include <iostream>

LineEdit::LineEdit(QWidget *parent, const char
      *name) : QMainWindow(parent, name)
{
  QWidget *widget = new QWidget(this);
  setCentralWidget(widget);

  QGridLayout *grid = new QGridLayout(widget,3,2,10,
      10,"grid");

  username_entry = new
      QLineEdit( widget, "username_entry");

  password_entry = new
      QLineEdit( widget, "password_entry");
  password_entry->setEchoMode(QLineEdit::Password);

  grid->addWidget(new QLabel("Username",
      widget, "userlabel"), 0, 0, 0);
  grid->addWidget(new QLabel("Password",
      widget, "passwordlabel"), 1, 0, 0);

  grid->addWidget(username_entry, 0,1, 0);
  grid->addWidget(password_entry, 1,1, 0);
```

QLineEdit Widget

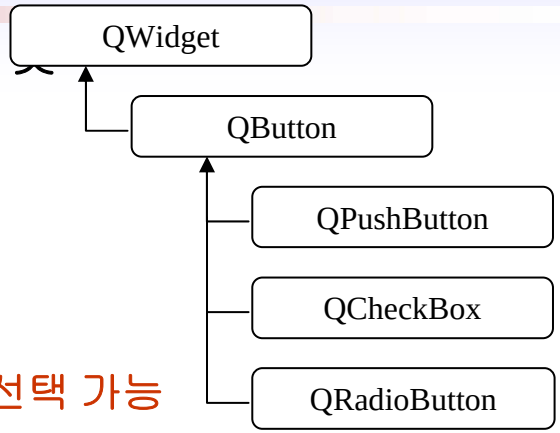
QLineEdit 예제 프로그램

```
  QPushButton *button = new QPushButton ("Ok",
      widget, "button");
  grid->addWidget(button, 2,1,Qt::AlignRight);

  resize( 350, 200 );

  connect (button, SIGNAL(clicked()), this,
      SLOT(Clicked()));
}

void LineEdit::Clicked(void)
{
   std::cout << password_entry->text() << "\n";
}

int main(int argc, char **argv)
{
  QApplication app(argc,argv);
  LineEdit *window = new LineEdit();

  app.setMainWidget(window);
  window->show();

  return app.exec();
}
```



Qt Button Widget

- 사용자가 클릭하여 특정 동작을 유발하는 위젯
- 종류 – 기본 클래스는 **QButton**
  - 명령(command) 버튼 - **QPushButton**
  - 상태(state) 버튼
    - 사용자가 상태 값으로 선택 가능
    - **QCheckBox :** 체크가 토글됨
    - **QRadioButton :** 그룹으로 묶여 그 중 하나만 선택 가능



- **QButton** 클래스
  - 버튼의 설정/해제 상태를 질의/전환, 버튼의 텍스트나 픽스맵 설정 가능
  - 사용 함수
    - **virtual void QButton::setText (const QString &);**
    - **virtual void QButton::setPixmap (const QPixmap &);**
    - **bool QButton::isToggleButton () const;**
    - **virtual void QButton::setDown (bool);**
    - **bool QButton::isDown () const;**
    - **bool QButton::isOn () const;**
    - **enum QButton::ToggleState (Off, NoChange, On);**
    - **ToggleState QButton::state () const;**
      - 토글버튼의 상태 획득

Qt Button Widget

- **QPusbButton Widget**
  - 표준 일반 버튼이며 텍스트와 픽스맵 아이콘 가능
  - 활성화되었을 때 **"clicked"** 신호 발생
  - 생성
    - **QPushButton::QPushButton (QWidget *parent, const char *name = 0);**
    - **QPushButton::QPushButton (const QString &text, QWidget *parent, const char *name = 0);**
    - **QPushButton::QPushButton (const QIconSet &icon, const QString &text, QWidget *parent, const char *name = 0);**
  - 사용 함수
    - **void QPushButton::setToggleButton (bool);**
- **QCheckBox Widget**
  - 설정/해제될 수 있는 상태를 가진 버튼이며, 텍스트가 옆에 있는 체크 박스 형태 가능
  - "변경되지 않음" 을 의미하는 세 번째 중간 상태 가능
  - 생성
    - **QCheckBox::QCheckBox (QWidget *parent, const char *name = 0);**
    - **QCheckBox::QCheckBox (const QString &text, QWidget *parent, const char *name = 0);**
  - 사용 함수
    - **void QCheckBox::isChecked ();**
    - **void QCheckBox::setTristate (bool y = TRUE);**
    - **Bool QCheckBox::isTristate ();**

Qt Button Widget

- **QRadioButton Widget**
  - 같은 형식의 버튼을 여러 개를 묶어서 하나만을 선택
  - **RadioButton** 그룹은 **QButtonGroup** 클래스를 사용
  - 생성
    - **QRadioButton::QRadioButton (QWidget *parent, const char *name = 0);**
    - **QRadioButton::QRadioButton (const QString &text, QWidget *parent, const char *name = 0);**
  - 사용 함수
    - **bool QRadioButton::isChecked ();**
- **QButtonGroup** 클래스
  - 생성
    - **QButtonGroup::QButtonGroup (QWidget *parent, const char *name = 0);**
    - **QButtonGroup::QButtonGroup (const QString &title, QWidget *parent, const char *name = 0);**
  - 사용 함수
    - **int QButtonGroup::insert (QButton *button, int id = -1);**
    - **void QButtonGroup::remove (QButton *button)**
    - **Int QButtonGroup::id (QButton *button) const;**
    - **Int QButtonGroup::count () const;**
    - **Int QButtonGroup::selectedId () const;**

Qt Button Widget

Qt Button 예제 프로그램

```
#include <qmainwindow.h>
#include <qcheckbox.h>
#include <qbutton.h>
#include <qradiobutton.h>

class Buttons : public QMainWindow
{
 Q_OBJECT

 public:
  Buttons(QWidget *parent = 0, const char *name =
    0);

 private:
  void PrintActive(QButton *button);
  QCheckBox *checkbox;
  QRadioButton *radiobutton1, *radiobutton2;

 private slots:
  void Clicked();

};
```

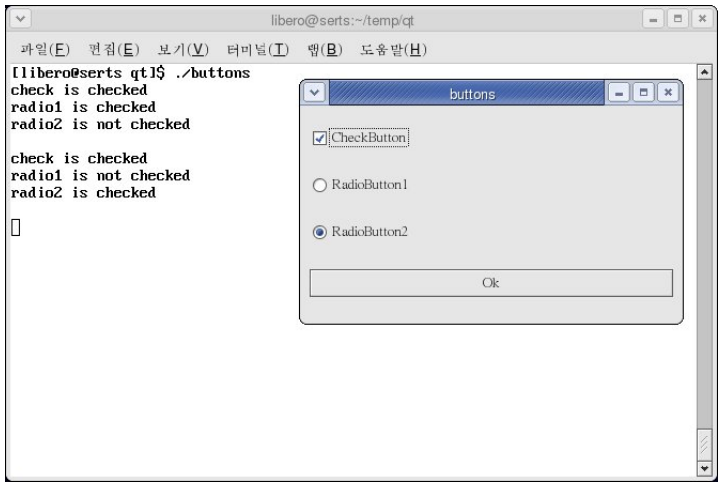
```
#include "Buttons.moc"
#include <qbuttongroup.h>
#include <qpushbutton.h>
#include <qapplication.h>
#include <qlabel.h>
#include <qlayout.h>
#include <iostream>

Buttons::Buttons(QWidget *parent, const char
    *name) : QMainWindow(parent, name)
{
 QWidget *widget = new QWidget(this);
 setCentralWidget(widget);

 QVBoxLayout *vbox = new QVBoxLayout(widget,5,
    10,"vbox");

 checkbox = new QCheckBox("CheckButton",
    widget, "check");
 vbox->addWidget(checkbox);
 QButtonGroup *buttongroup = new QButtonGroup(0);

 radiobutton1 = new QRadioButton("RadioButton1",
    widget, "radio1");
 buttongroup->insert(radiobutton1);
 vbox->addWidget(radiobutton1);

 radiobutton2 = new QRadioButton("RadioButton2",
    widget, "radio2");
 buttongroup->insert(radiobutton2);
 vbox->addWidget(radiobutton2);
```

Qt Button Widget

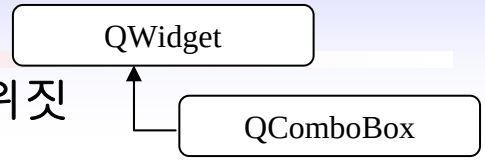
Qt Button 예제 프로그램

```
  QPushButton *button = new QPushButton ("Ok",
     widget, "button");
  vbox->addWidget(button);

  resize( 350, 200 );

  connect (button, SIGNAL(clicked()), this,
     SLOT(Clicked()));
}

void Buttons::PrintActive(QButton *button)
{
  if (button->isOn())
    std::cout << button->name() << " is checked\n";
  else
    std::cout << button->name() << " is not
      checked\n";
}

void Buttons::Clicked(void)
{
   PrintActive(checkbox);
   PrintActive(radiobutton1);
   PrintActive(radiobutton2);
   std::cout << "\n";
}
```

```
int main(int argc, char **argv)
{
  QApplication app(argc,argv);
  Buttons *window = new Buttons();

  app.setMainWidget(window);
  window->show();

  return app.exec();
}
```



QComboBox Widget



- 제한이 없는 개수의 항목 중에서 하나를 선택할 수 있는 위젯
  - **QLineEdit, QPushButton,** 드롭다운 메뉴의 기능 혼합
- 생성
  - **QComboBox::QComboBox (QWidget *parent = 0, const char *name = 0);**
  - **QComboBox::QComboBox (bool readwrite, QWidget *parent = 0, const char *name = 0);**
- 사용 함수
  - **int QComboBox::count();**
  - **void QComboBox::insertStringList (const QStringList &list, int index = -1);**
  - **void QComboBox::insertStrList (const QStringList &list, int index = -1);**
  - **void QComboBox::insertStrList (const QStringList *list, int index = -1);**
  - **void QComboBox::insertStrList (const char **strings, int numStrings = -1, int index = -1);**
  - **void QComboBox::insertItem (const QString &t, int index = -1);**
  - **void QComboBox::removeItem (int index);**
  - **virtual void QComboBox::setCurrentItem (int index);**
  - **QString QComboBox::currentItem (int index);**
  - **QString QComboBox::currentText ();**
  - **virtual void QComboBox::setCurrentText (const QString &);**
  - **void QComboBox::setEditable (bool);**

QComboBox Widget

QComboBox 예제 프로그램

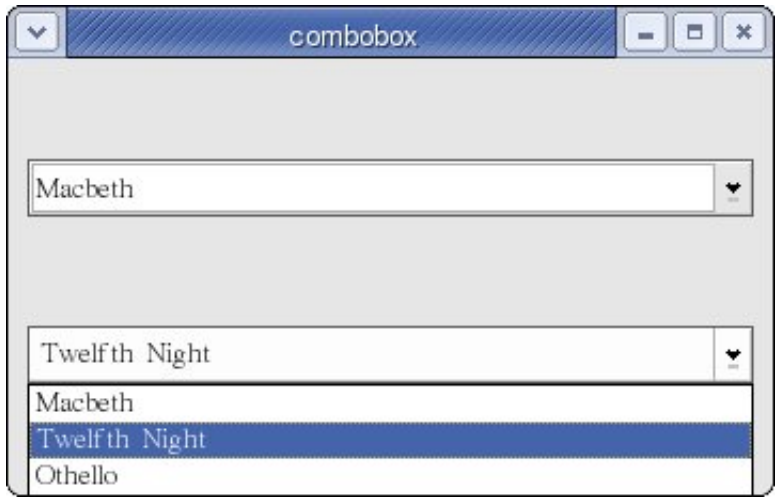
```
#include <qmainwindow.h>
#include <qcombobox.h>

class ComboBox : public QMainWindow
{
  Q_OBJECT

  public:
    ComboBox(QWidget *parent = 0, const char *name
      = 0);

  private slots:
    void Changed(const QString& s);

};
```

```
#include "ComboBox.moc"

#include <qlayout.h>
#include <iostream>

ComboBox::ComboBox(QWidget *parent, const char
    *name) : QMainWindow(parent, name)
{
  QWidget *widget = new QWidget(this);
  setCentralWidget(widget);

  QVBoxLayout *vbox = new QVBoxLayout(widget,5,
     10,"vbox");

  QComboBox *editablecombo = new QComboBox(TRUE,
     widget, "Editable");
  vbox->addWidget(editablecombo);
  QComboBox *readonlycombo = new
     QComboBox(FALSE, widget, "readonly");
  vbox->addWidget(readonlycombo);

  static const char* items[] = { "Macbeth", "Twelfth
     Night", "Othello", 0 };
  editablecombo->insertStrList (items);
  readonlycombo->insertStrList (items);

  connect (editablecombo, SIGNAL(textChanged(const
     QString&)),
         this, SLOT(Changed(const QString&)));
  resize( 350, 200 );

}
```

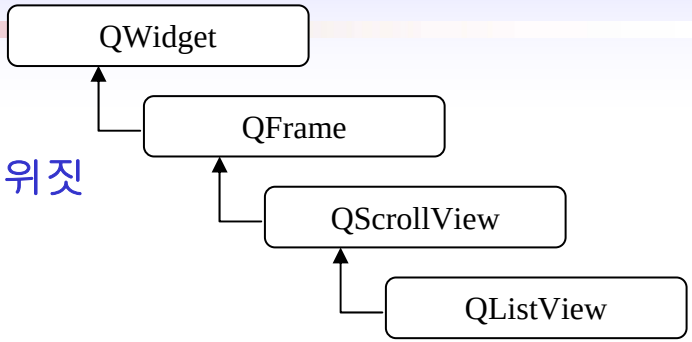
QComboBox Widget

QComboBox 예제 프로그램

```
void ComboBox::Changed(const QString& s)
{
  std::cout << s << "\n";
}

int main(int argc, char **argv)
{
  QApplication app(argc,argv);
  ComboBox *window = new ComboBox();

  app.setMainWidget(window);
  window->show();

  return app.exec();
}
```



QListView Widget



- 데이터의 리스트 뷰나 트리 뷰를 만듦
- 기본 구성 요소
  - **QListView :** 트리 뷰와 리스트 뷰를 지원하는 위짓
  - **QListViewItem :** 리스트나 트리의 행을 표현
- **QListView** 위짓
  - 생성
    - **QListView::QListView (QWidget *parent = 0, const char *name = 0);**
  - 사용 함수
    - **int QListView::addColumn();**
    - **void QListView::setRootIsDecorated (da);**
- **QListViewItem** 클래스
  - 생성
    - **QListViewItem::QListViewItem (QWidget *parent = 0, const char *name = 0);**
  - 사용 함수
    - **Virtual void QListViewItem::insertItem (QListViewItem * newChild);**
    - **Virtual void QListViewItem::setText (int column, const Qstring &test);**
    - **Virtual Qstring QListViewItem::text (int column) const;**
    - **QListViewItem * QListViewItem::firstChild () const;**
    - **QListViewItem * QListViewItem::nextSibling () const;**
    - **QListViewItem * QListViewItem::parent () const;**
    - **QListViewItem * QListViewItem::itemAbove ();**
    - **QListViewItem * QListViewItem::itemBelow ();**

QListView Widget

Tree 예제 프로그램

```
#include <qmainwindow.h>
#include <qlistview.h>


class ListView : public QMainWindow
{
  Q_OBJECT

  public:
    ListView(QWidget *parent = 0, const char *name = 0);

  private:
    QListView *listview;

  private slots:

};
```

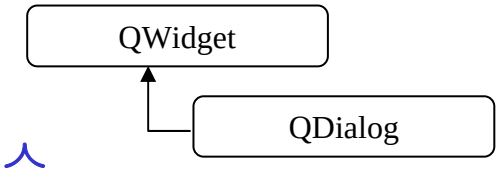

```
#include "ListView.moc"

ListView::ListView(QWidget *parent, const char
      *name) : QMainWindow(parent, name)
{
  listview = new QListView(this, "listview1");

  listview->addColumn("Artist");
  listview->addColumn("Title");
  listview->addColumn("Catalogue");
  listview->setRootIsDecorated(TRUE);

  QListViewItem *toplevel = new
      QListViewItem(listview, "Avril Lavigne", "Let
      Go", "AVCD01");

  new QListViewItem(toplevel, "Complicated");
  new QListViewItem(toplevel, "Sk8er Boi");

  setCentralWidget(listview);
}

int main(int argc, char **argv)
{
  QApplication app(argc,argv);
  ListView *window = new ListView();

  app.setMainWidget(window);
  window->show();

  return app.exec();
}
```

대화 상자 (Dialog)

Dialog

- 사용자와 대화하고 중요한 이벤트를 사용자에게 알리는 임시 윈도우
- 대화 상자의 종류
  - 모달 대화 상자 – 사용자 응답을 받기 전에 모든 작동을 차단
  - 모드리스 대화 상자 – 대화 상자를 다른 윈도우처럼 처리
  - 세미모달 대화 상자 – 고유의 이벤트 반복문을 가지지 않는 모달 대화 상자

QDialog Widget



- **Qt** 의 기본 대화 상자 클래스
  - 모달/넌모달 대화 상자를 위한 **exec/show** 메소드 제공
  - 필수 **Layout** 을 가지며, 몇 가지 신호와 슬롯 포함
  - **Accept** 와 **reject** 의 슬롯을 사용하여 대화 상자의 결과 반환
- 생성
  - 보통 **Qdialog** 을 상속하여 전용 대화 상자를 위한 클래스를 마들고 대화 상자 인터페이스를 만들기 위해 위짓 추가

모달 대화 상자

- 사용자 응답을 받기 전에 모든 작동을 차단
  - 심각한 문제나 에러나 경고 메시지를 보고할 때 유용
- 사용 방법
  - **Exec** 호출을 통해 대화상자를 활성화하고
  - **Exec** 를 통해 대화 상자가 처리될 때까지 프로그램 실행 정지
    - 사용자가 누른 버튼에 해당하는 슬롯에 따라 **Qdialog:Accepted** 또는 **Qdialog::Rejected** 값 반환

```
MyDialog *dialog = new MyDialog(this, “MyDialog”);
if (dialog->exec() == QDialog::Accepted) {
    // 사용자가 “Ok” 를 클릭함
    doSomething();
} else {
    // 사용자가 “Cancel”을 클릭하였거나 대화상자가 종료
    doSomethingElse();
}
delete dialog;
```

년모달 대화 상자

- 대화 상자를 다른 윈도우처럼 처리
  - 사용자가 즉시 응답할 필요가 없음
  - 부모 윈도우 위에 위치하며, 작업 표시줄 항목을 공유하고, **accept** 또는 **reject** 슬롯이 호출될 때 자동으로 대화상자가 숨겨짐
- 사용 방법
  - **show()** 를 호출
  - 버튼에 해당하는 슬롯을 작성하여 응답 신호에 연결
    - 사용자가 버튼을 누르면 어떤 버튼을 눌렀는지에 대한 정수**(int)** 값 반환

```
MyDialog *dialog = new MyDialog(this, “MyDialog”);
dialog->show();
...
MyDialog::MyDialog(Qwidget *parent, const char *name) : QDialog
(parent, name)
{
...
connect (ok_button, SIGNAL(clicked()), this, SLOT(OKClicked()));
connect (cancel_button, SIGNAL(clicked()), this, SLOT(CancelClicked()));
}
MyDialog::OKClicked()
{ ...}
MyDialog::CancelClicked()
{...}
```

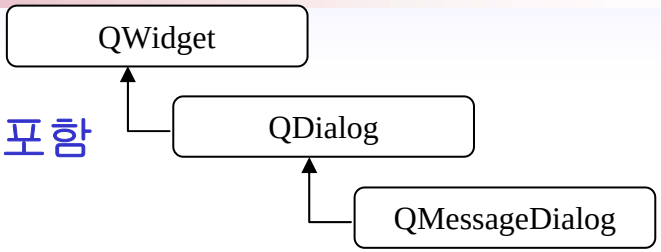
세미모달 대화 상자

- 고유의 이벤트 반복문을 가지지 않는 모달 대화 상자
  - 제어를 응용 프로그램에 반환할 수 있지만 다른 대화상자에 대한 입력은 계속 차단
  - 사용자의 의지에 따라 취소 가능해야 하며, 중요한 작업의 진행 상황을 보여주고자 할 때 유용
- 사용 방법
  - **Qdialog** 생성자의 모달 플래그를 설정하고 **show()** 를 호출
  - 주기적으로 **QApplication::processEvent** 를 호출하여 대화상자 갱신
    - 파일 선택, 텍스트 입력, 진행율 표시기, 메시지 상자와 같은 작업에 유용

```
MySMDialog *dialog = new MySMDialog(this, “semimodal”);
dialog->show();
While (processing) {
    doSomeProcessing();
    app->processEvents();
    if (dialog->wasCanceled()) break;
...
}
MySMDialog::MySMDialog(Qwidget *parent, const char *name) :
Qdialog (parent, name, TRUE)
{
...
}
```

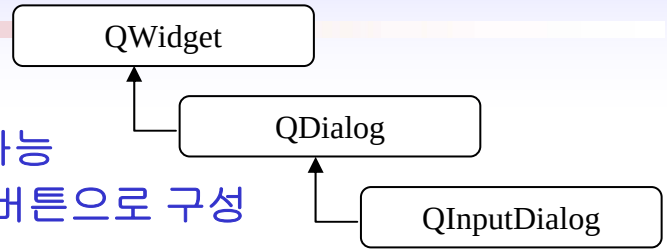
QMessageBox

- 매우 간단한 대화 상자
  - 아이콘과 제목, 구성할 수 있는 버튼 포함
- 정적 메소드
  - **int information (QWidget *parent, const QSting &caption, const QString &text, int button0, int button1=0, int button2=0);**
  - **int warning (QWidget *parent, const QString &caption, const QString &text, int button0, int button1, int button2=0);**
  - **int critical (Qwidget *parent, const QString &caption, const QString &text, int button0, int button1, int button2=0);**
- **buttons:** 버튼 종류



버튼 종류	설명	버튼 종류	설명
QMessageBox::Ok	확인 버튼	QMessageBox::Abort	중지 버튼
QMessageBox::Cancel	취소 버튼	QMessageBox::Retry	재시도 버튼
QMessageBox::Yes	예 버튼	QMessageBox::Ignore	무시 버튼
QMessageBox::No	아니오 버튼		

QInputDialog



- 사용자로부터 값을 입력받는 대화 상자
  - 텍스트, 드롭다운 리스트, 정수/부동소수점 값 가능
  - **QLineEdit/QComboBox 위짓과 OK 및 Cancel 버튼으로 구성**
- 정적 메소드
  - **QString getText (const QSting &caption, const QString &label, QLineEdit::Echomode mode=QLineEdit::Normal, const QString&text=QString::null, bool *ok=0, QWidget *parent = 0, const char *name=0);**
    - **QLineEdit 사용, EchoMode 설정 가능, 어떤 버튼을 눌렀는지는 ok 로 확인(Ok 버튼은 TRUE 반환)**
  - **QString getItem (const QSting &caption, const QString &label, const QStringList &list, int current=0, bool editable=TRUE, bool *ok=0, QWidget *parent = 0, const char *name=0);**
    - **QComboBox 형태로 옵션 목록 제공**
  - **QString getInteger (const QSting &caption, const QString &label, int num=0, int from = -2147483647, int to = 2147483647, int step=1, bool *ok=0, Qwidget *parent = 0, const char *name=0);**
  - **QString getDouble (const QSting &caption, const QString &label, double from = -2147483647, double to = 2147483647, int decimals=1, bool *ok=0, Qwidget *parent = 0, const char *name=0);**

Dialog Widget

Dialog 예제 프로그램

```
#ifndef MYDIALOG_H
#define MYDIALOG_H

#include <qdialog.h>

class MyDialog : public QDialog
{
   Q_OBJECT

   public:
      MyDialog(QWidget *parent = 0, const char
     *name = 0);
};

#endif
```

```
#include "MyDialog.moc"
#include <qlayout.h>
#include <qlabel.h>
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qmessagebox.h>
#include <qinputdialog.h>
#include <qstring.h>

MyDialog::MyDialog(QWidget *parent, const char
     *name) : QDialog(parent, name)
{
 QVBoxLayout *vbox = new QVBoxLayout(this);
 QHBoxLayout *hbox1 = new QHBoxLayout();
 QHBoxLayout *hbox2 = new QHBoxLayout();
 QPushButton *ok_button = new QPushButton ("Ok",
    this, "button1");
 QPushButton *cancel_button = new QPushButton
    ("Cancel", this, "button2");

 vbox->addLayout(hbox1);
 vbox->addLayout(hbox2);
 hbox1->addWidget(new QLabel("Enter your name",
    this));
 hbox1->addWidget(new QLineEdit(this, 0));
 hbox2->addWidget(ok_button);
 hbox2->addWidget(cancel_button);

 connect (ok_button, SIGNAL(clicked()), this,
    SLOT(accept()));
 connect (cancel_button, SIGNAL(clicked()), this,
    SLOT(reject()));
}
```

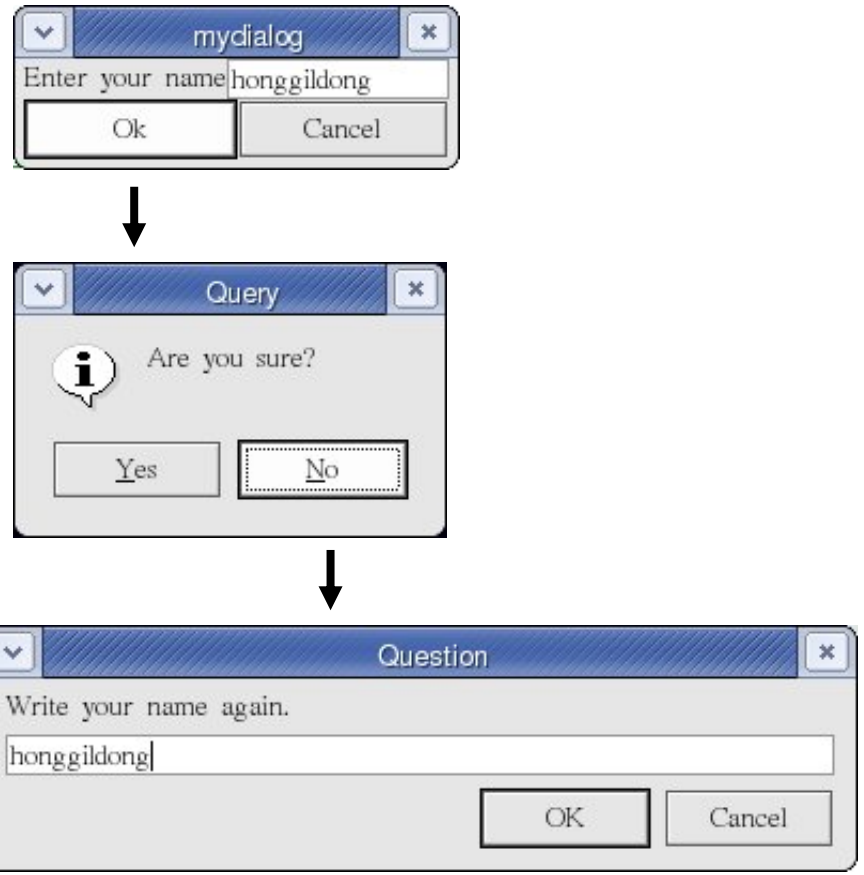


```
int main(int argc, char **argv)
```

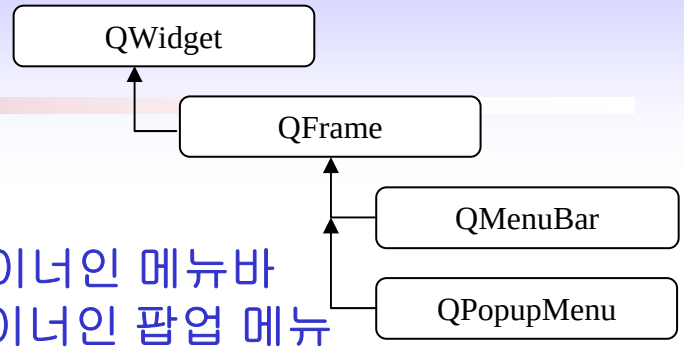
Dialog Widget

Dialog 예제 프로그램

```
int main(int argc, char **argv)
{
  QApplication app(argc,argv);
  MyDialog *dialog = new MyDialog(0, "mydialog");

  if (dialog->exec() == QDialog::Accepted) {
    int result = QMessageBox::information
      (dialog, "Query", "Are you sure?",
               QMessageBox::Yes | QMessageBox::Default,
               QMessageBox::No | QMessageBox::Escape);
    switch (result) {
    case QMessageBox::Yes:
       break;
    case QMessageBox::No:
       bool retry;
       QString text = QInputDialog::getText ("Question",
                          "Write your name again.",
                          QLineEdit::Normal,
                          QString::null, &retry,
                          dialog, "input");
       break;
    }
  }
  delete dialog;

  return 0;
}
```



Qt Menu Widget



- 메뉴를 만듦
- 메뉴 관련 위젯 종류
  - **QMenuBar** – 개별 메뉴들을 포함하는 컨테이너인 메뉴바
  - **QPopupMenu** – 아이템들을 포함하는 컨테이너인 팝업 메뉴
- 생성
  - **QPopupMenu (QWidget * parent = 0, const char * name = 0);**
  - **QMenuBar (QWidget * parent = 0, const char * name = 0);**
- 사용 함수
  - **int insertItem (const QString & text, const QObject * receiver, const char * member, const QKeySequence & accel = 0, int id = -1, int index = -1);**
  - **int insertItem (const QPixmap & pixmap, const QObject * receiver, const char * member, const QKeySequence & accel = 0, int id = -1, int index = -1);**
  - **int insertSeparator (int index = -1);**
  - **bool isItemEnabled (int id) const;**
  - **void setItemEnabled (int id, bool enable);**
  - **bool isItemVisible (int id) const;**
  - **void setItemVisible (int id, bool visible);**

Qt Menu Widget

Qt Menu 예제 프로그램

```
#ifndef MENU_H
#define MENU_H
#include <qwidget.h>
#include <qmenubar.h>
#include <qlabel.h>

class MenuTest : public QWidget
{
   Q_OBJECT

public:
   MenuTest(QWidget *parent=0, const char
     *name=0);
public slots:
   void open();
   void news();
   void save();
   void undo();
   void redo();
   void about();
   void printer();
   void printerSetup();

protected:
   void    resizeEvent( QResizeEvent * );

signals:
   void    explain( const QString& );

private:
   void contextMenuEvent ( QContextMenuEvent * );
   QMenuBar *menu;
   QLabel   *label;
};
#endif // MENU_H
```

```
#include "Menu.moc"
#include <qcursor.h>
#include <qapplication.h>
#include <qmessagebox.h>

/* XPM */
static const char * new_xpm[] = {
"16 16 3 1",
"      c None",
".     c #000000000000",
"X     c #FFFFFFFFFFFF",
"                ",
"   ......       ",
"   .XXX.X.      ",
"   .XXX.XX.     ",
"   .XXX.XXX.    ",
"   .XXX.....    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   .XXXXXXX.    ",
"   ........     ",
"                ",
"                "};
```

```
MenuTest::MenuTest( QWidget *parent, const char
      *name )
    : QWidget( parent, name )
{
```

Qt Menu Widget

Qt Menu 예제 프로그램

```
MenuTest::MenuTest( QWidget *parent, const char
    *name ) : QWidget( parent, name )
{
   QPixmap p1( new_xpm );
   QPopupMenu *print = new QPopupMenu( this );
   Q_CHECK_PTR( print );
   print->insertTearOffHandle();
   print->insertItem( "&Print to printer", this,
     SLOT(printer()) );
   print->insertSeparator();
   print->insertItem( "Printer &Setup", this,
     SLOT(printerSetup()) );

   QPopupMenu *file = new QPopupMenu( this );
   Q_CHECK_PTR( file );
   file->insertItem( p1, "&New", this, SLOT(news()),
     CTRL+Key_N );
   file->insertItem( "&Open",  this, SLOT(open()),
     CTRL+Key_O );
   file->insertItem( "&Save", this, SLOT(save()),
     CTRL+Key_S );
   file->insertSeparator();
   file->insertItem( "&Print", print, CTRL+Key_P );
   file->insertSeparator();
   file->insertItem( "E&xit",  qApp, SLOT(quit()),
     CTRL+Key_Q );

   QPopupMenu *edit = new QPopupMenu( this );
   Q_CHECK_PTR( edit );
   int undoID = edit->insertItem( "&Undo", this,
     SLOT(undo()) );
   int redoID = edit->insertItem( "&Redo", this,
     SLOT(redo()) );
   edit->setItemEnabled( undoID, FALSE );
   edit->setItemEnabled( redoID, FALSE );

   QPopupMenu *help = new QPopupMenu( this );
   Q_CHECK_PTR( help );
   help->insertItem( "&About", this, SLOT(about()),
     CTRL+Key_H );

   menu = new QMenuBar( this );
   Q_CHECK_PTR( menu );
   menu->insertItem( "&File", file );
   menu->insertItem( "&Edit", edit );
   menu->insertSeparator();
   menu->insertItem( "&Help", help );
   menu->setSeparator(QMenuBar::InWindowsStyle );

   label = new QLabel( this );
   Q_CHECK_PTR( label );
   label->setGeometry( 20, 20, 20, 40 );
   label->setFrameStyle(QFrame::Box|QFrame::Raised );
   label->setLineWidth( 1 );
   label->setAlignment( AlignCenter );

   connect( this,  SIGNAL(explain(const QString&)),
          label, SLOT(setText(const QString&)) );
   setMaximumSize( 300, 200 );
   setFocusPolicy( QWidget::ClickFocus );
}
```

Qt Menu Widget

Qt Menu 예제 프로그램

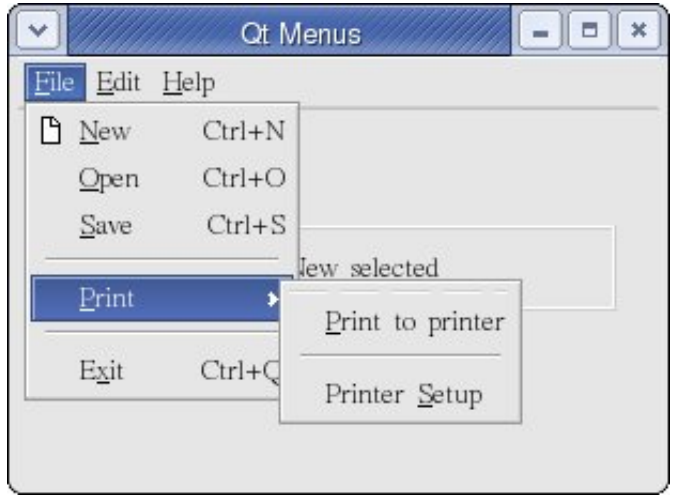
```
Void MenuTest::contextMenuEvent
     ( QContextMenuEvent *)
{
   QPopupMenu *contextMenu=new QPopupMenu(this);
   Q_CHECK_PTR( contextMenu );
   QLabel *caption = new QLabel( "<font
     color=darkblue><u><b>"
       "Context Menu</b></u></font>", this );
   caption->setAlignment( Qt::AlignCenter );
   contextMenu->insertItem( caption );
   contextMenu->insertItem( "&New",  this,
     SLOT(news()), CTRL+Key_N );
   contextMenu->insertItem( "&Open...", this,
     SLOT(open()), CTRL+Key_O );
   contextMenu->insertItem( "&Save", this,
     SLOT(save()), CTRL+Key_S );
   QPopupMenu *submenu = new QPopupMenu( this );
   Q_CHECK_PTR( submenu );
   submenu->insertItem( "&Print to printer", this,
     SLOT(printer()) );
   contextMenu->insertItem( "&Print", submenu );
   contextMenu->exec( QCursor::pos() );
   delete contextMenu;
}

void MenuTest::open()
{
   emit explain( "File/Open selected" );
}
```

```
void MenuTest::news()
{
   emit explain( "File/New selected" );
}

void MenuTest::save()
{
   emit explain( "File/Save selected" );
}

void MenuTest::undo()
{
   emit explain( "Edit/Undo selected" );
}

void MenuTest::redo()
{
   emit explain( "Edit/Redo selected" );
}

void MenuTest::about()
{
   QMessageBox::about( this, "Qt Menu Test",
            "This example demonstrates simple use of
     Qt menus.\n");
}

void MenuTest::printer()
{
   emit explain( "File/Printer/Print selected" );
}

void MenuTest::printerSetup()
{
   emit explain( "File/Printer/Printer Setup selected" );
}
```

Qt Menu Widget

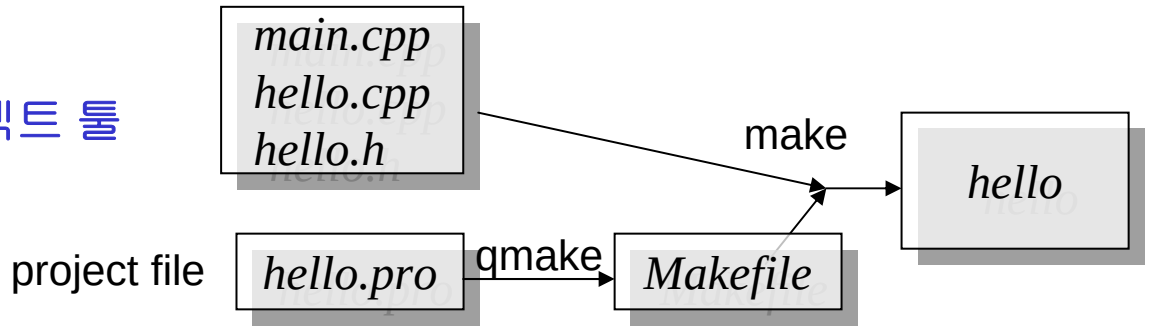
Qt Menu 예제 프로그램

```
void MenuTest::resizeEvent( QResizeEvent * )
{
    label->setGeometry( 20, rect().center().y()-20, width()-40,
      40 );
}

int main( int argc, char ** argv )
{
    QApplication a( argc, argv );
    MenuTest m;
    m.setCaption("Qt Menus");
    a.setMainWidget( &m );
    m.show();
    return a.exec();
}
```



Qt 개발 환경

- **Qt Designer**
  - Qt 와 KDE 라이브러리를 사용하는 GUI 를 생성하는 도구
- **Kdevelop**
  - 텍스트 에디터, 디버거 등을 포함한 C/C++ 통합 개발 환경
- **Uic (Qt User Interface Compiler)**
  - Qt Designer 에 의해 만들어진 .ui 파일을 이용하여 소스 파일(.h, .cpp) 생성 도구
- **Qmake**
  - Qt 소프트웨어 프로젝트 툴



- **Qt Designer/uic/qmake 를 사용한 Qt 프로그램 개발 단계**
  - Designer 를 사용하여 인터페이스 제작
  - uic 를 사용하여 소스코드 빌드
  - 소스 코드를 필요에 알맞게 편집
  - 프로젝트 파일을 작성하여 qmake 를 사용하여 Makefile 생성
  - Make 로 소스 파일 컴파일 및 실행파일 생성

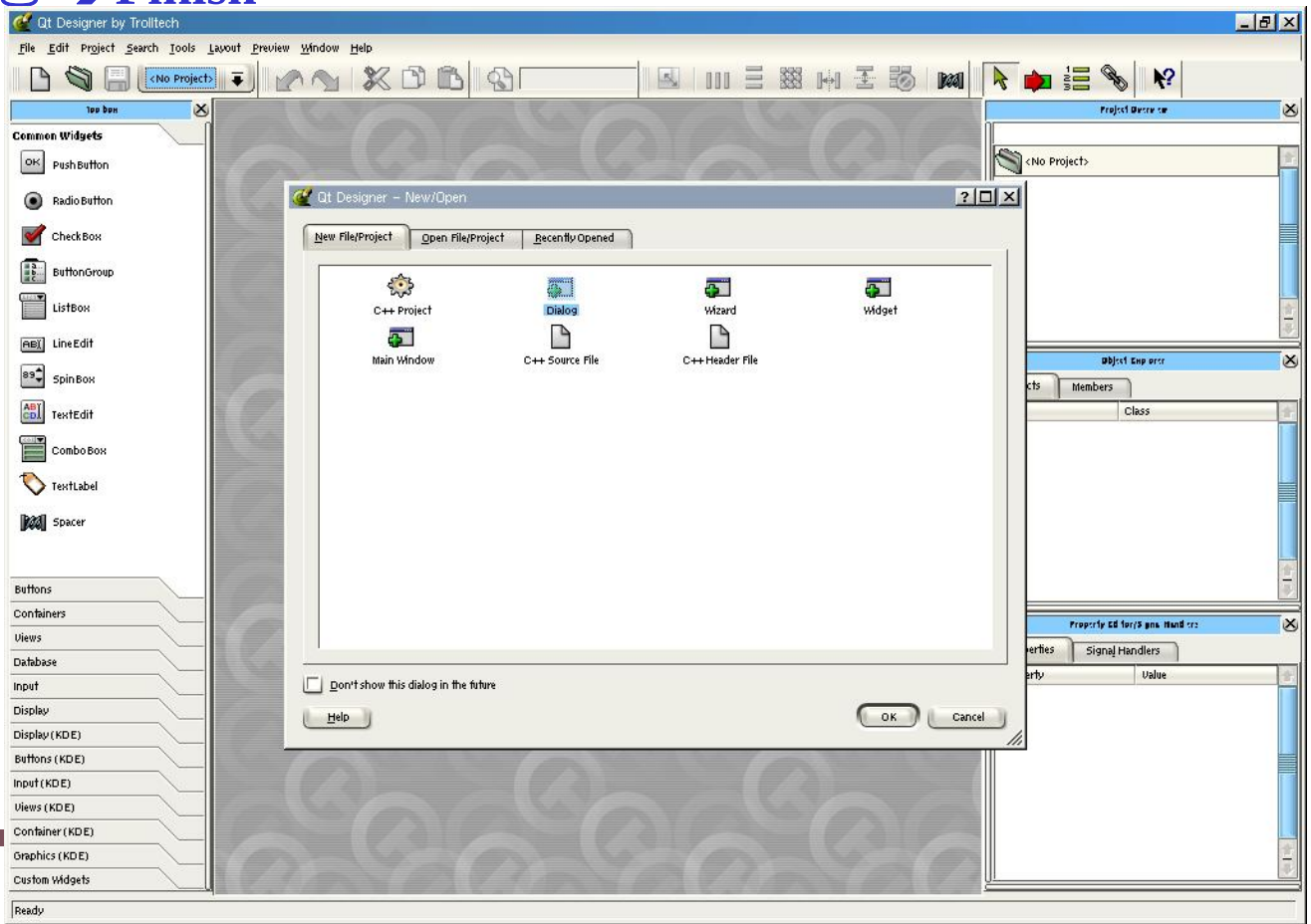
Qt Programming 예제

Qt Designer 를 이용한 프로그래밍 예제

- Qt Designer 실행

```
$ designer
```

- MainWindow 선택 ➔Next 선택➔New/Open/Save 를 오른쪽 창으로 이동 ➔Finish



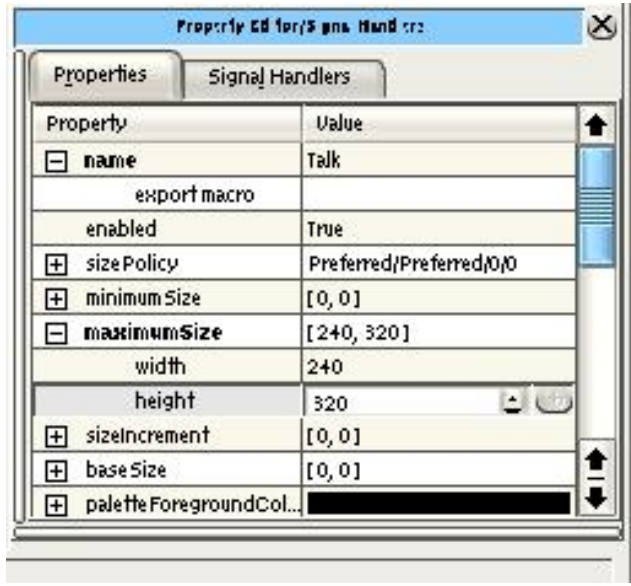
Qt Programming 예제

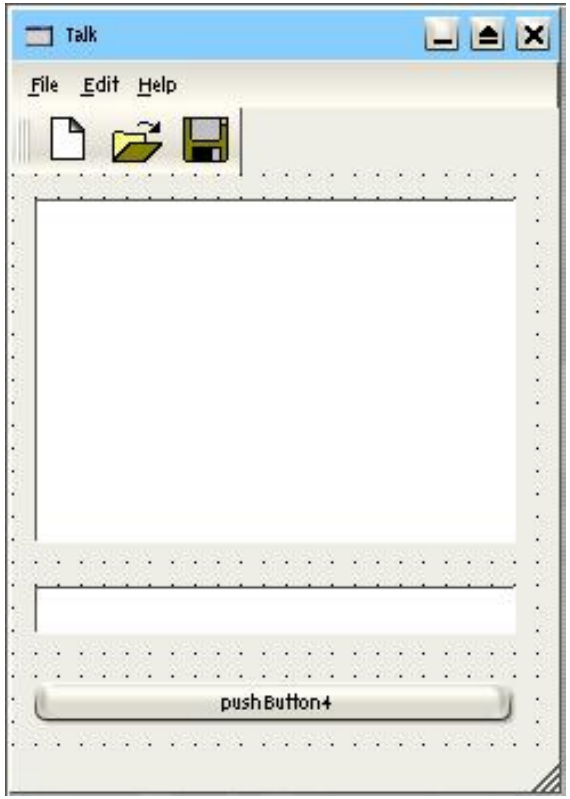
Qt Designer 를 이용한 프로그래밍 예제 계속

Property Edit 창에서 속성 값 확인 및 변경

- Name&Caption: Talk, maximumSize: (240,320)

위젯 추가: TextEdit, LineEdit, PushButton



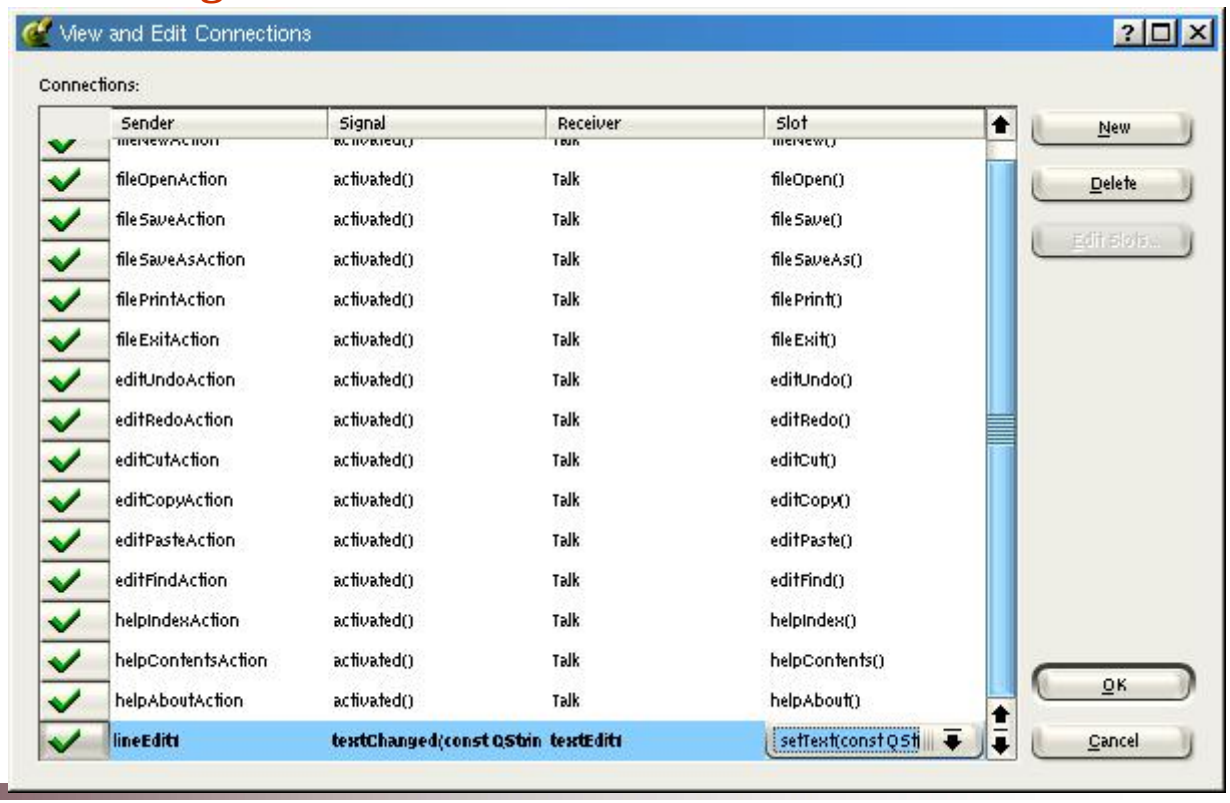


Qt Programming 예제

Qt Designer 를 이용한 프로그래밍 예제 계속

Signal/Slot 연결

- Connect Signal/Slots 항목 선택
- LineEdit 위젯을 클릭한 후 TextEdit 위젯까지 드래그
- Signal 은 textChanged, slot 은 setText 로 선택



저장

- talk.ui

Qt Programming 예제

Qt Designer 를 이용한 프로그래밍 예제 계속

프로젝트 파일 제작 : talk.pro

```
TEMPLATE = app
CONFIG = qt warn_on release
SOURCES = main.cpp
INTERFACES = talk.ui
TARGET = talk
```

소스 파일 제작 : main.cpp

```
#include <qapplication.h>
#include "talk.h"

int main (int argc, char **argv)
{
        QApplication a(argc, argv);
        Talk *mw = new Talk;
        mw->show();
        a.connect(&a, SIGNAL(lastWindowClosed()), &a, SLOT(quit()) );

        return a.exec();
}
```

Qt Programming 예제

Qt Designer 를 이용한 프로그래밍 예제 계속

사용자 인터페이스 소스 생성 : talk.h, talk.cpp

```
$ uic talk.ui > talk.h
$ uic –impl talk.h talk.ui > talk.cpp
```

Makefile 생성 및 make

```
$ qmake talk.pro –o Makefile
$ make
```

Qt Programming 예제

▣ Qt Designer 를 이용한 프로그래밍 예제 계속

◈ 실행

```
$ ./talk
```

GUI Programming 실습

- 간단한 **GTK+ GUI** 채팅 프로그램 작성, 설치 및 연결 테스트
- 간단한 **Qt GUI** 채팅 프로그램 작성, 설치 및 연결 테스트