

Simple example of building a GUI using GTK

To build a GUI in a Linux environment using C with **GTK** (a popular toolkit for creating graphical user interfaces), we need to set up GTK, write a C program to create and manage GUI elements, and handle user interactions. The program flow for creating a simple GTK-based GUI is given and it provides a minimal example that creates a window with a button that prints a message when clicked.

Program Flow for Building a GUI with C and GTK

1. Set up the environment:

- Install GTK (e.g., `libgtk-3-dev` on Debian/Ubuntu).
- Include GTK headers (`gtk/gtk.h`) in your C program.
- Link the GTK library during compilation using `pkg-config`.

2. Initialize GTK:

- Call `gtk_init()` to initialize the GTK library and process command-line arguments.

3. Create the main window:

- Use `gtk_window_new()` to create a top-level window.
- Set window properties (e.g., title, size) using functions like `gtk_window_set_title()` and `gtk_window_set_default_size()`.

4. Add widgets:

- Create widgets (e.g., buttons, labels) using functions like `gtk_button_new_with_label()`.
- Add widgets to the window or a container (e.g., `gtk_box_new()` for layout).
- Use `gtk_container_add()` or `gtk_box_pack_start()` to organize widgets.

5. Connect signals:

- Attach event handlers (callbacks) to widgets for user interactions (e.g., button clicks) using `g_signal_connect()`.
- Define callback functions to handle events (e.g., printing a message or closing the window).

6. Show widgets and start the main loop:

- Call `gtk_widget_show_all()` to display the window and its contents.
- Start the GTK main loop with `gtk_main()` to process events and keep the GUI responsive.

7. Handle cleanup:

- Ensure the program exits cleanly when the window is closed (e.g., connect the window's destroy signal to `gtk_main_quit()`).
- Free resources if necessary (though GTK often handles this automatically).

Simple Example: Window with a Button

This C program creates a GTK window containing a button. Clicking the button prints "Button clicked!" to the console, and closing the window exits the program.

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

// Callback function for button click
static void on_button_clicked(GtkButton *button, gpointer user_data) {
    g_print("Button clicked!\n");
}

// Callback function for window close
static void on_window_destroy(GtkWidget *widget, gpointer user_data) {
    gtk_main_quit();
}

int main(int argc, char *argv[]) {
    // Step 1: Initialize GTK
    gtk_init(&argc, &argv);

    // Step 2: Create the main window
    GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(window), "Simple GTK Example");
    gtk_window_set_default_size(GTK_WINDOW(window), 300, 200);

    // Step 3: Create a button
    GtkWidget *button = gtk_button_new_with_label("Click Me");

    // Step 4: Add button to the window
    gtk_container_add(GTK_CONTAINER(window), button);

    // Step 5: Connect signals
    g_signal_connect(button, "clicked", G_CALLBACK(on_button_clicked), NULL);
    g_signal_connect(window, "destroy", G_CALLBACK(on_window_destroy),
    NULL);

    // Step 6: Show all widgets and start the main loop
    gtk_widget_show_all(window);
    gtk_main();

    return 0;
}
```

How to Compile and Run

1. **Install GTK:** On Debian/Ubuntu, install the GTK development library:

- ```
sudo apt update
```
2. `sudo apt install libgtk-3-dev`

3. **Compile the C program:** Use `pkg-config` to include GTK flags and libraries:

```
gcc -o gtk_example gtk_example.c `pkg-config --cflags --libs
gtk+-3.0`
```

The `pkg-config` command provides the necessary compiler and linker flags for GTK.

4. **Run the program:**

```
./gtk_example
```

This opens a window with a button. Clicking the button prints "Button clicked!" to the terminal, and closing the window exits the program.