

# GUI プログラミング入門



西尾 太

Momonga Project

**Momonga Linux 4**

# Do You Know Mo?

- コミュニティで作る Linux ディストリビューション
  - rpm を利用しています
- 開発者が使いたい Linux
  - なので、パッケージが豊富
  - だけど、偏ったパッケージング
- とりあえず、新しもの好き
  - 最新アプリをとりあえず突っ込む
- だけど安定指向



Momonga Linux 4

# ■ Momonga Linux 4 の特徴

- Compiz Fusion 採用
  - 日本語化は Momonga の成果が採用
- Web アプリ開発ツールが豊富
  - Ruby on Rails、TurboGears、Catalyst 等
- 日本語環境でも安定



Momonga Linux 4

# ■ Compiz Fusion のデモ

- 3D デスクトップ
- デスクトップの切り替え
- アプリケーションの切り替え
- 炎のエフェクト
- 雨のエフェクト
- 雪のエフェクト
- 下のアプリケーションを表示する
- 最大化・アイコン化



Momonga Linux 4

# Momonga Linux を使うには

- ダウンロードして DVD-R や CD-R に焼く
  - ダウンロード先はチラシを参照してください
  - インストール用・Windows 用・Live CD 等
- インストールして楽しむ
- Microsoft Window 上で楽しむ
  - coMomonga でお楽しみください
- Live CD ( DVD ) で楽しむ



Momonga Linux 4

# Momonga で何やってるの？

- GNOME のパッケージング
  - デスクトップ環境
  - 自分が使いたい
- Mono のパッケージング
  - .NET 開発環境
  - 使いたいアプリが C#
- Autotools のパッケージング
  - autoconf
  - automake
  - libtool



Momonga Linux 4

# GNOME とは？

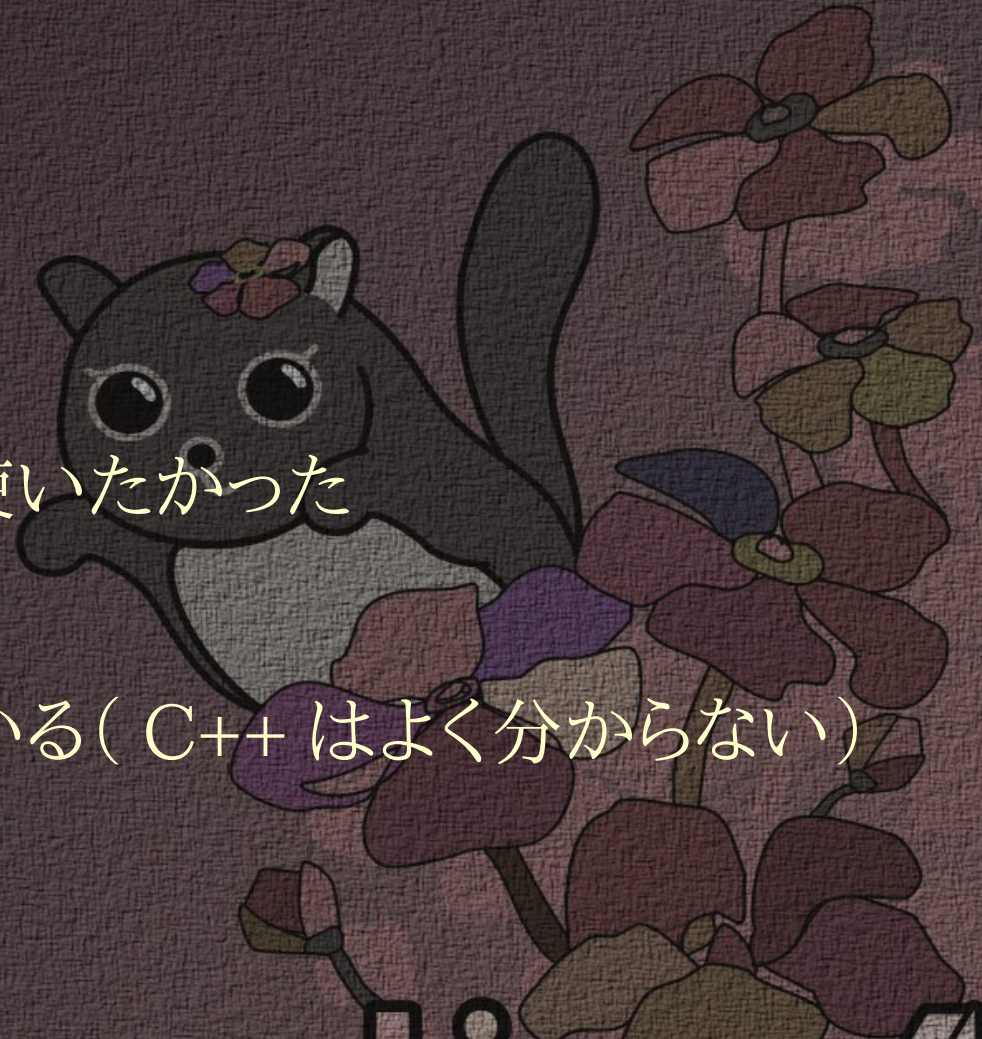
- GNU Network Object Model Environment
  - GUI の G ではない
- Linux の代表的なデスクトップ環境
  - KDE や Xfce4 も有名
- freedesktop 準拠
  - アプリケーション間の通信
  - メニュー
  - etc.



Momonga Linux 4

# なぜ GNOME ?

- GUI 操作が楽
  - CLI は面倒
- GNU だから
  - GNU の開発環境を使ったかった
- gtk+ だから
  - C 言語で開発されている (C++ はよく分からない)



Momonga Linux 4

# Linux の魅力

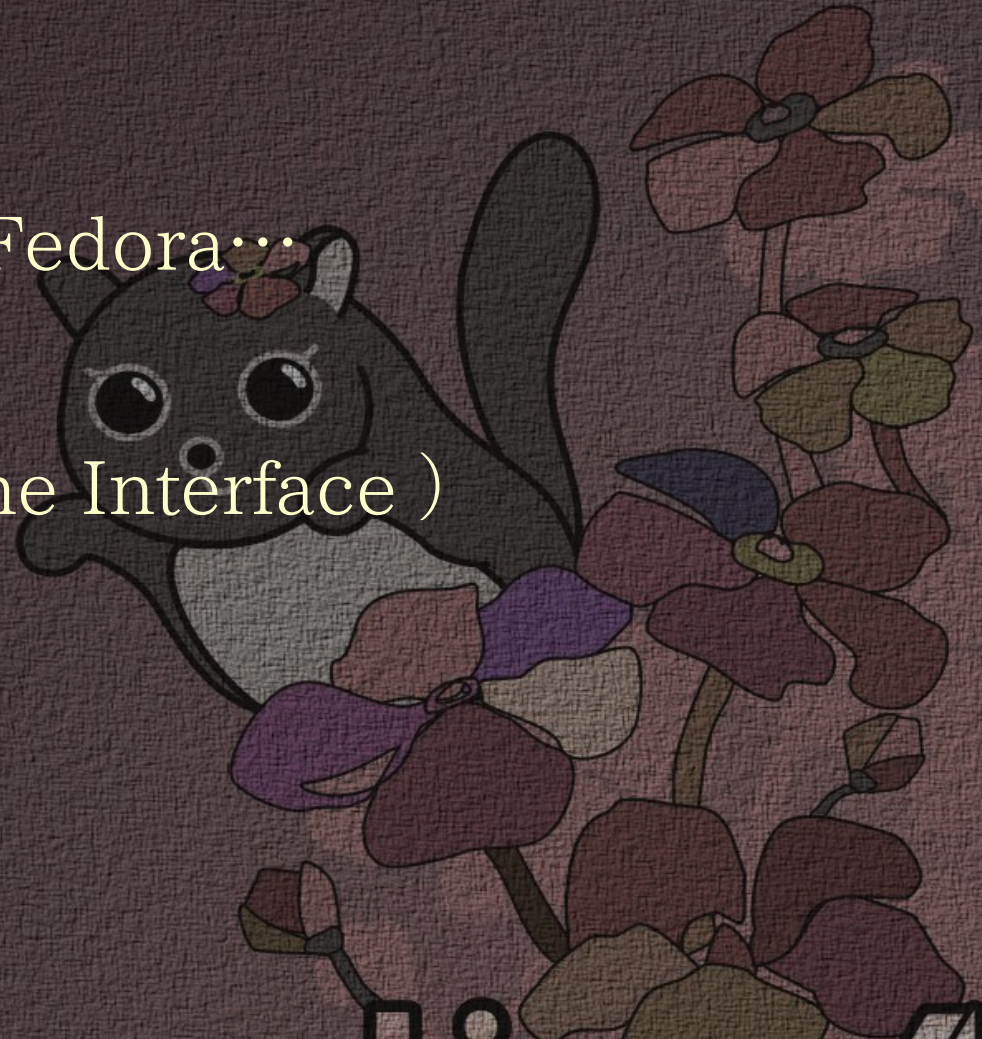
- 普段使うのは困らない
  - Web ブラウズ
  - メール
  - 音楽・動画鑑賞
  - 写真編集
- 中まで分かる
  - オープンソース
  - 修正も可能



# Momonga Linux 4

# Linux って大変そう

- 選ぶのが大変
  - Debian Slackware Fedora...
- コマンドが大変
  - CLI ( Command Line Interface )
- 英語が大変
  - マニュアルが英語



# Momonga Linux 4

# ☐ プログラミングは楽しい

- ハードウェアを好きにあやつれる
  - kernel 開発などがお勧め
- 勝手に計算してほしい
  - CLI プログラムがお勧め



Momonga Linux 4

## ■ コンピュータを使うだけなら

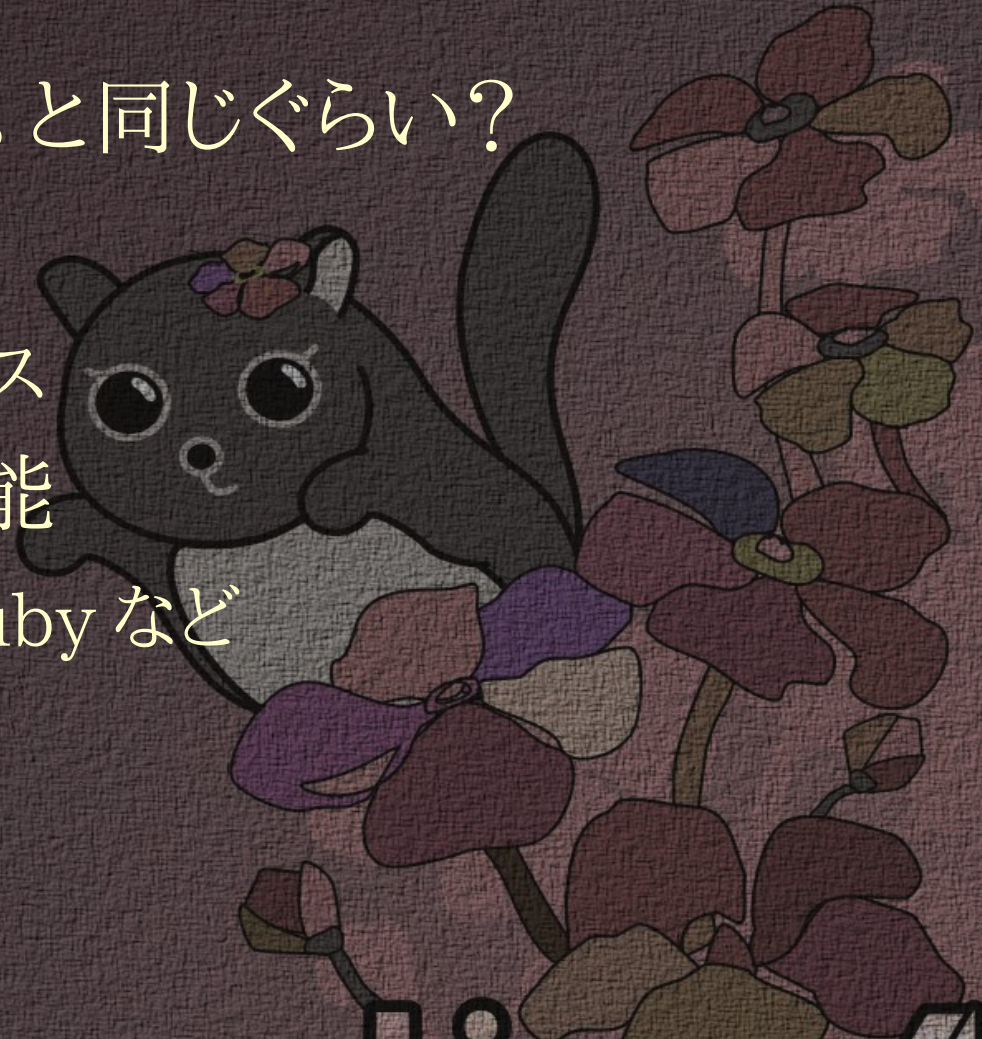
- CLI のプログラミングが楽
  - でも、最近使ってもらえない
- CLI のプログラムに GUI ラッパー
  - コマンドを生成する GUI アプリ
  - バックエンドでは CLI のプログラムが動作



Momonga Linux 4

# GUI プログラミングって大変？

- Microsoft Windows と同じぐらい？
- ライブラリが豊富
  - ほとんどオープンソース
- スクリプト言語でも可能
  - bash perl python ruby など



Momonga Linux 4

# GUI プログラムの基本

- すべての Window に対し
  - 初期化(いろいろ)
    - イベント定義
  - 無限ループ(イベント待ち)
  - 終了処理(終了イベント)
- 最近は、Widget を使う



Momonga Linux 4

# GUI の Hello

- Xlib → Win32 SDK
- Xt gtk+ → MFC の C 版
- gtkmm → MFC
- gtk-sharp → C#
- mono-basic → VB.Net



Momonga Linux 4

```
#include <X11/Xlib.h>
```

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

```
{  
    Display *d;  
    Window w;  
    GC gc;  
    Font f;  
    XEvent e;  
    int exit = 1;  
  
    d = XOpenDisplay (NULL);  
    w = XCreateSimpleWindow (d, DefaultRootWindow (d), 0, 0, 225, 71, 1, BlackPixel (d, 0), WhitePixel (d, 0));  
    XSelectInput (d, w, ButtonPressMask | ExposureMask);  
    XMapWindow (d, w);  
  
    gc = XCreateGC (d, w, 0, 0);  
    f = XLoadFont (d, "-*-times-*-*-*-*100-*-*-*-*-*-*");  
    XSetFont (d, gc, f);  
    XSetForeground (d, gc, BlackPixel (d, 0));  
    XSetBackground (d, gc, WhitePixel (d, 0));  
  
    XDrawString (d, w, gc, 4, 70, "Hello", 5);  
  
    XFlush (d);  
  
    while (exit)  
    {  
        XNextEvent (d, &e);  
        switch (e.type)  
        {  
            case ButtonPress:  
                puts ("Hello");  
                exit = 0;  
                break;  
            case Expose:  
                XDrawString (d, w, gc, 4, 70, "Hello", 5);  
                break;  
        }  
    }  
    return 0;  
}
```

# Xlib



# Momonga Linux 4

Xt

```
#include <X11/Intrinsic.h>
#include <X11/Shell.h>
#include <X11/Xaw/Command.h>
#include <stdlib.h>
#include <X11/StringDefs.h>
```

```
static
void Quit (Widget w, XtPointer client_data, XtPointer call_data)
{
    puts ("Hello");
    exit (0);
}
```

```
int
main (int argc, char* argv[])
{
    Widget w, quit;

    w = XtInitialize ("Hello", "Hello", NULL, 0, &argc, argv);
    quit = XtCreateManagedWidget ("Hello", commandWidgetClass, w, NULL, 0);
    XtAddCallback (quit, XtNcallback, Quit, 0);
    XtRealizeWidget (w);
    XtMainLoop ();
}
```



Momonga Linux 4

gtk+

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

static void
hello (GtkWidget *widget, gpointer data)
{
    g_print ("Hello\n");
    gtk_main_quit ();
}

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

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

    button = gtk_button_new_with_label ("Hello");
    g_signal_connect (G_OBJECT (button), "clicked", G_CALLBACK (hello), NULL);

    gtk_container_add (GTK_CONTAINER (window), button);

    gtk_widget_show (button);
    gtk_widget_show (window);

    gtk_main ();

    return 0;
}
```



Momonga Linux 4

```
#include <iostream>
#include <gtkmm.h>
```

```
class HelloWorld : public Gtk::Window {
public:
    HelloWorld();
    virtual ~HelloWorld();
protected:
    Gtk::Button m_button;
    virtual void on_button_clicked();
};
```

```
HelloWorld::HelloWorld() : m_button("Hello")
{
    m_button.signal_clicked().connect(sigc::mem_fun(*this, &HelloWorld::on_button_clicked));
    add(m_button);
    m_button.show();
}
```

```
HelloWorld::~~HelloWorld() {}
```

```
void HelloWorld::on_button_clicked()
{
    std::cout << "Hello" << std::endl;
    Gtk::Main::quit();
}
```

```
int main(int argc, char *argv[])
{
    Gtk::Main kit(argc, argv);
    HelloWorld helloworld;
    Gtk::Main::run(helloworld);
    return 0;
}
```

# gtkmm



# Momonga Linux 4

```
using System;
using Gtk;
```

# gtk-sharp

```
class MainClass {
    public static void Main (string[] args) {
        Application.Init ();

        Window w = new Window ("");
        Button b = new Button ("Hello");

        b.Clicked += new EventHandler (Button_Clicked);

        w.Add (b);
        w.ShowAll ();
        Application.Run ();
    }

    static void Button_Clicked (object o, EventArgs args) {
        System.Console.WriteLine ("Hello");
        Application.Quit ();
    }
}
```



# Momonga Linux 4

# mono-basic

```
Public Class MyForm
  Inherits System.Windows.Forms.Form

  Public Sub New()
    MyBase.New()
    InitializeComponent()
  End Sub

  Private components As System.ComponentModel.IContainer

  Friend WithEvents Button As System.Windows.Forms.Button
  <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.Button = New System.Windows.Forms.Button
    Me.Button.Text = "Hello"
    Me.Controls.Add(Me.Button)
  End Sub

  Private Sub Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button.Click
    Console.WriteLine ("Hello")
  End
End Sub
End Class
```



**Momonga Linux 4**

スクリプトでも OK

- Tcl/Tk
  - tcl というスクリプト + tk というツールキット
- Zenity
  - Shell Script + GNOME の GUI ツール



Momonga Linux 4

# Tcl/Tk

```
#!/usr/bin/wish
```

```
button .button -text "Hello" -command {puts Hello; exit}  
pack .button
```



# Momonga Linux 4

# Zenity

```
#!/bin/sh
```

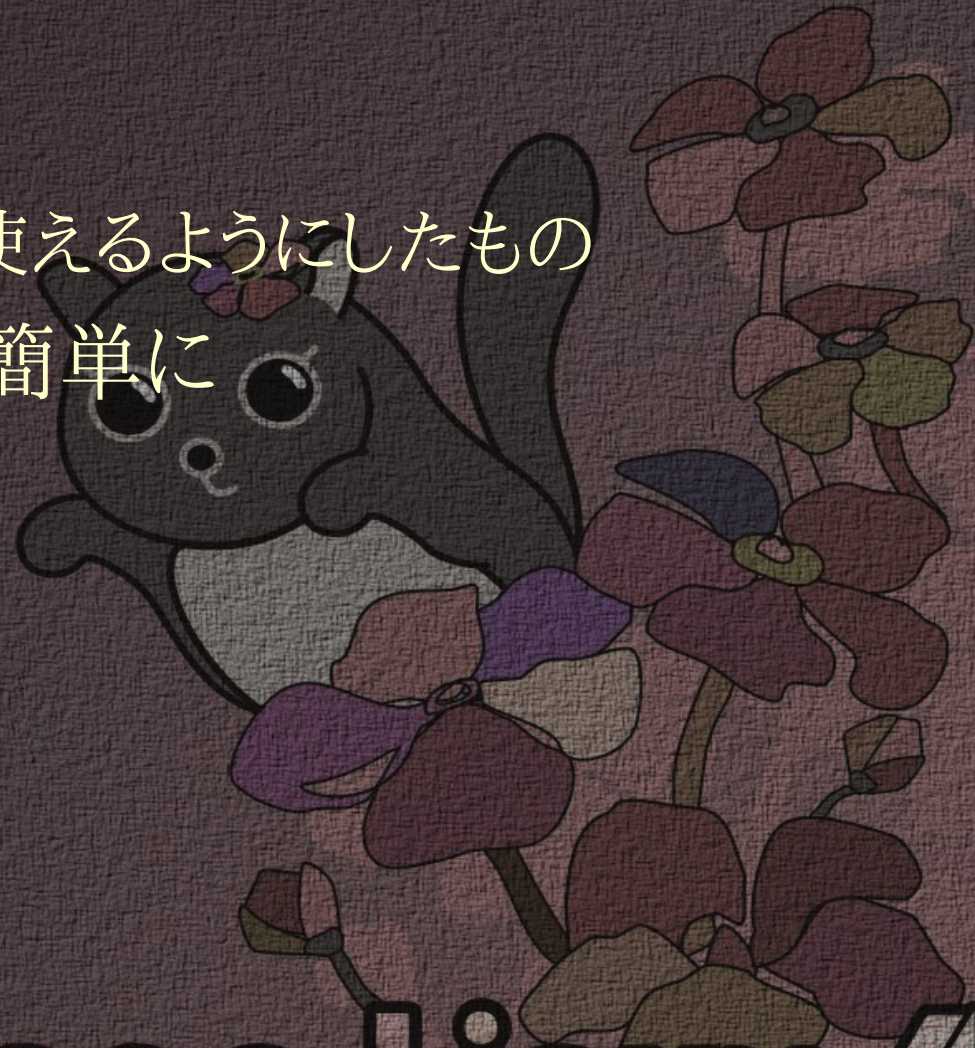
```
zenity --info --text "Hello"  
echo "Hello"
```



# Momonga Linux 4

# Python

- pygtk で簡単に
  - gtk+ を python から使えるようにしたもの
- glade を使ってさらに簡単に
  - GUI Builder
  - ファイルは XML 形式



Momonga Linux 4

# Python

```
#!/usr/bin/env python
import gtk

class hello:
    def __init__(self):
        self.window = gtk.Window()
        self.window.set_title("Hello")
        self.button = gtk.Button("Hello")
        self.button.connect("clicked", self.button_clicked)
        self.window.add(self.button)
        self.window.show_all()
    def button_clicked(self, data):
        print "Hello"
        gtk.main_quit()

if __name__ == "__main__":
    app = hello()
    gtk.main()
```



Momonga Linux 4

# glade

- Momonga Linux には glade-2 と glade-3 が存在
  - glade-2 はソース生成機能付き
  - glade-3 は…新しいだけ
- glade-2 と glade-3 のデモ



Momonga Linux 4

# glade

```
#!/usr/bin/env python
import gtk
import gtk.glade

class hello:
    def __init__(self):
        self.xml = gtk.glade.XML("hello_pygtk_glade.glade")
        self.xml.get_widget("window").set_title("Helo")
        dict = {"on_button_clicked":self.on_button_clicked}
        self.xml.signal_autoconnect(dict)
        self.xml.get_widget("window").show_all()
    def on_button_clicked(self, data):
        print "Hello"
        gtk.main_quit()

if __name__ == "__main__":
    app = hello()
    gtk.main()
```



**Momonga Linux 4**

# プログラムを作ってみたい

- オープンソースでは、他に作っている人がいる
  - 興味があるプロジェクトに参加してみてください
- 誰も作っていないらしい
  - 楽しくプログラミングしてください
  - そして、成果を公開してみてください



Momonga Linux 4

# ソースコードを配布する

- ビルド時の依存性を解消する
- 国際化対応
- メニューに登録する
- デフォルトの設定を記述する



Momonga Linux 4

# ビルド時の依存性を解消する

- `configure.ac` を書く
  - `autoconf`
- `Makefile.am` を書く
  - `automake`
- ライブラリの場合
  - `libtool`
  - `pkgconfig`



Momonga Linux 4

# 国際化対応

- gettext
  - メッセージの翻訳
- intltool
  - gettext を簡単に使えるようにするツール群



Momonga Linux 4

# メニューやデフォルトの設定

- メニューファイル: hoge.desktop
  - desktop-file-utils
  - アイコンも登録
- デフォルトの設定: hoge.schemas
  - Gconf



Momonga Linux 4

# 面倒だと思ったら

- とりあえず公開
  - 興味を持ってくれた人が作業してくれるかも
  - 興味を持ってくれた人が教えてくれるかも
- 世界中に仲間がいる
  - 中には、変な人もいますが
  - ネット社会は楽しい人とだけ仲良く



Momonga Linux 4

<http://www.momonga-linux.org/>

終わり



Momonga Linux 4