There is no “Linux” Platform

André Weber
Henry Rocha
Thomas Queiroz

O que é uma Plataforma?

Para desenvolvedores

- Sistema operacional
- SDK e ferramentas integradas ao SO
- Documentação, exemplos e tutoriais
- Loja de aplicativos

Para designers

- Sistema operacional
- Ferramentas de design integradas ao SO
- Linguagem de design visual
- Guia de interface de usuários

Para usuários

- Sistema operacional
- Loja de aplicativos integrada ao SO
- Suporte ao usuário

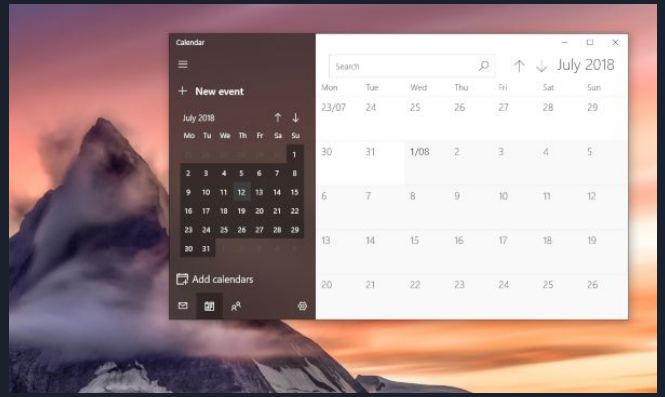
Resumindo

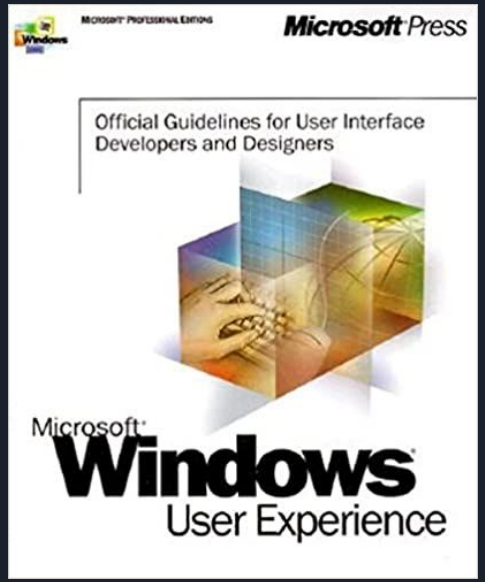
- Sistema operacional
- Ecossistema para desenvolvedores
- Linguagem de design
- Loja de aplicativos

Plataformas mais conhecidas

Windows

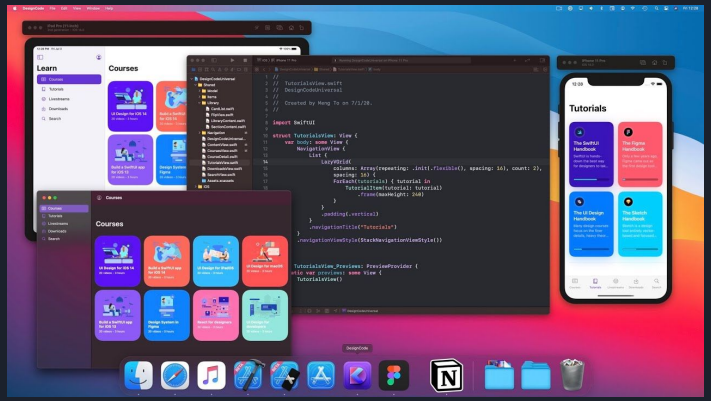
- Windows 10/11
- .NET e WinForms
- Fluent Design System
- Windows Store

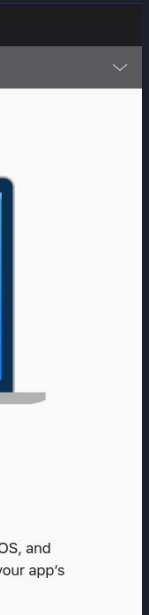




macOS / IOS

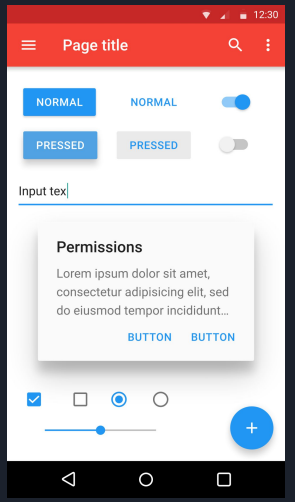
- macOS Monterey
- XCode & Swift
- Guia de interface de usuários
  - SwiftUI
- Apple Store

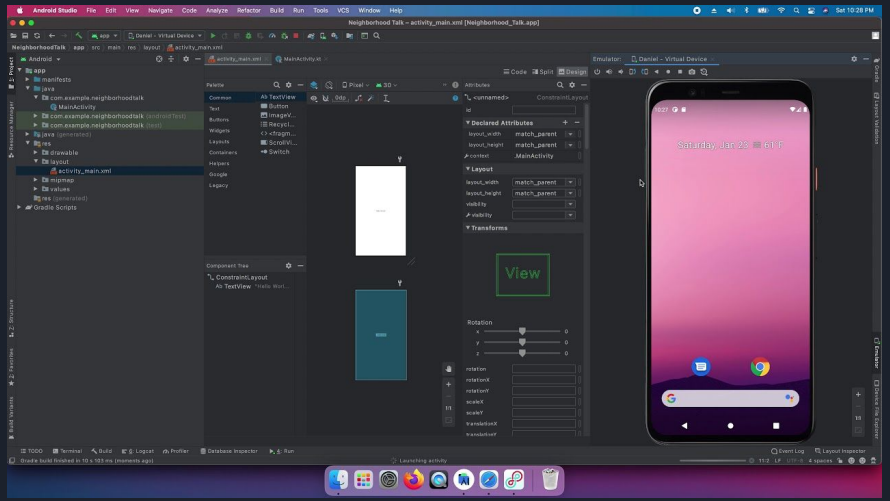




Android

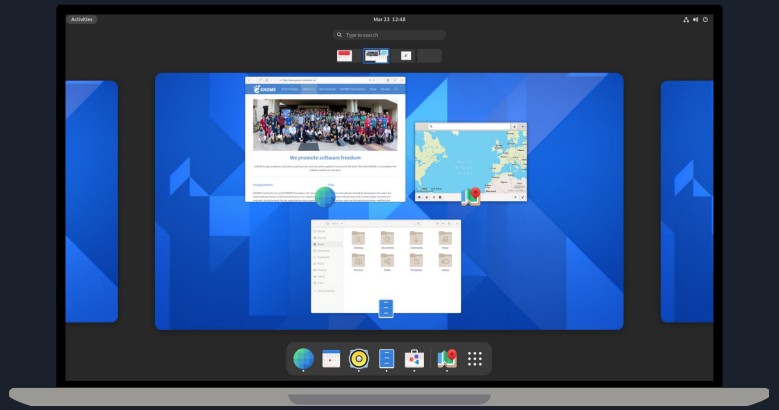
- Windows, macOS e Linux → Android
- Android Studio e Kotlin
- Material Design
- Google Play Store





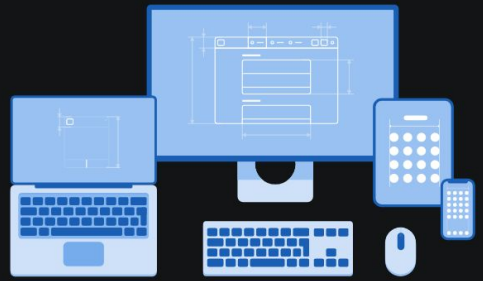
Linux

- Kernel
- X11 e Wayland
- GTK
  - Ubuntu
  - Elementary
  - GNOME
- Qt
  - KDE



Linux

GNOME Human Interface Guidelines



GNOME Human Interface Guidelines

The GNOME Human Interface Guidelines are the primary source of design documentation for those creating software with the GNOME development platform. They are primarily intended for application designers and developers, but are relevant to anyone wanting to familiarize themselves with GNOME UX.

Platform Definition

The HIG is intended to be used in conjunction with recent versions of the GNOME platform, as provided by the GNOME Flatpak SDK. It has been written to be used in conjunction with GTK 4 and the associated libadwaita library. However, much of the HIG is also relevant to applications that are using GTK 3 and libhandy.

Human Interface Guidelines

Search...

- Introduction
- Design Philosophy
- User Workflow
- Desktop Integration

WIDGETS

- Creating Layouts
- Providing Feedback
- Selection Controls
- Avoid Widgets That Do Nothing
- Interactive Widgets
- Containers

REFERENCE

- Iconography
- Text

- HIG Style Guide

Powered By GitBook

Introduction

Create a beautifully consistent experience on the elementary OS desktop

These guidelines are for interface designers, graphic artists, and software developers working on apps for elementary OS. They not only define specific design elements and principles, but also help instill a philosophy that enables you to decide when it is appropriate to deviate from the guidelines.

Adhering to the suggestions contained here provides many benefits:

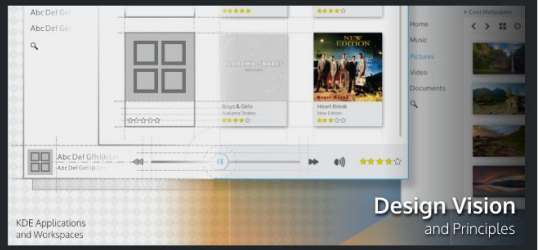
- Users will **learn to use your app faster**, because it shares common elements that they are already familiar with.
- Users will **accomplish tasks more quickly**, because you will have a straight-forward interface design that isn't confusing or difficult.
- Your app will appear **native to the desktop**, and share the same elegant look as default applications.
- Your app will be **easier to document**, because an expected behavior does not require explanation.
- The **amount of support you will have to provide will decrease**—including the number of issues filed—for the reasons above.

To help you achieve these goals, these guidelines cover basic interface elements, how to use them and put them together effectively, and how to make your app integrate well with the desktop. The most important thing to remember is that **following these guidelines makes it easier to design a new**

KDE Human Interface Guidelines

KDE Human Interface Guidelines

The KDE Human Interface Guidelines (HIG) offer designers and developers a set of recommendations for producing beautiful, usable, and consistent user interfaces for convergent desktop and mobile applications and workspace widgets. Their aim is to improve the experience for users by making application and widget interfaces more consistent and hence more intuitive and learnable.



Simple by default, powerful when needed.

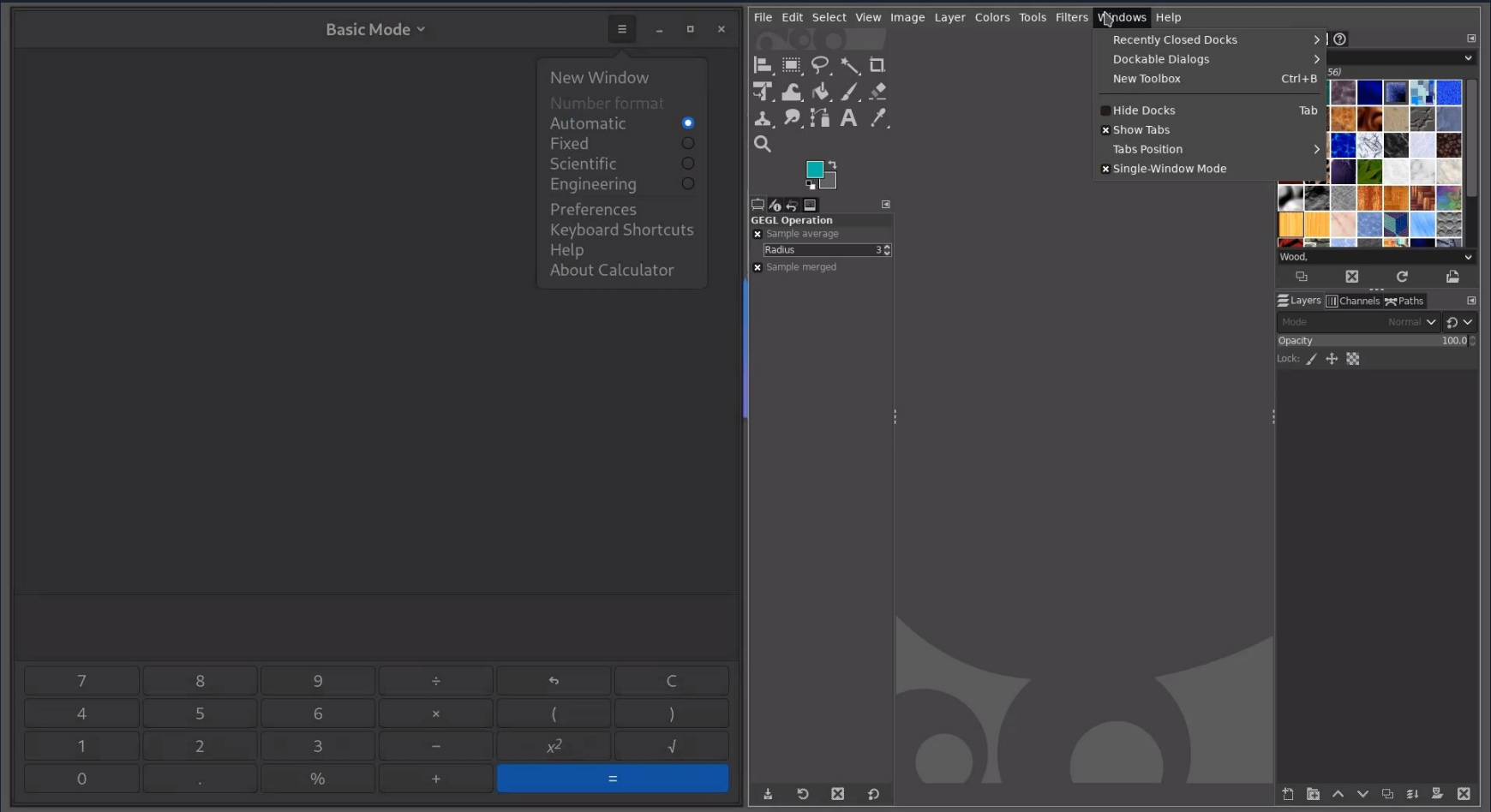
Design Vision

KDE's design vision focuses on two attributes of KDE software that connect its future to its history:

Simple by default...

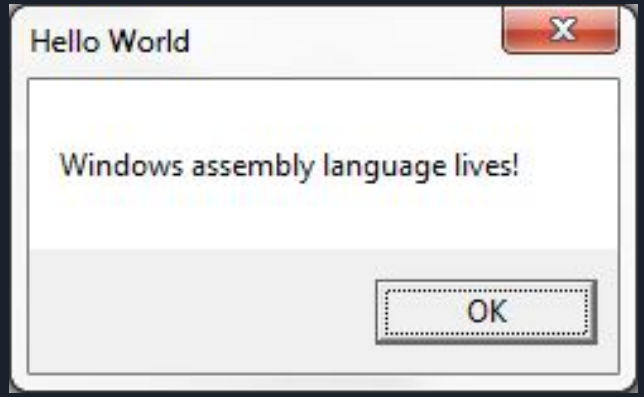
Simple and inviting. KDE software is pleasant to experience and easy to use.

- **Make it easy to focus on what matters** — Remove or minimize elements not crucial to the primary or main task. Use spacing to keep things organized. Use


Basic Mode
New Window
Number format
Automatic
Fixed
Scientific
Engineering
Preferences
Keyboard Shortcuts
Help
About Calculator
7 8 9 ÷ C
4 5 6 × ( )
1 2 3 − x² √
0 . % + =
File Edit Select View Image Layer Colors Tools Filters Windows Help
Recently Closed Docks
Dockable Dialogs
New Toolbox Ctrl+B
Hide Docks Tab
Show Tabs
Tabs Position
Single-Window Mode
GEGL Operation
Sample average
Radius 3
Sample merged
Wood,
Layers Channels Paths
Mode Normal
Opacity 100.0
Lock:

Criando uma interface gráfica no Windows

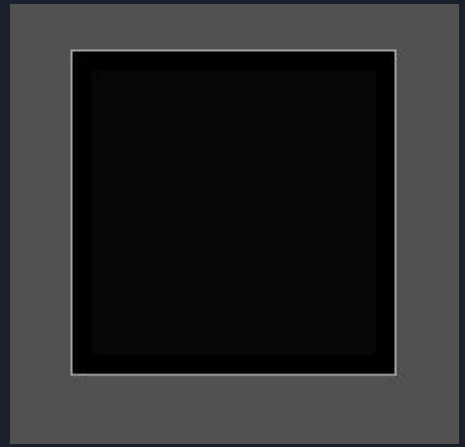
```
.686
.model flat, stdcall

EXTERN MessageBoxA@16 : proc
EXTERN ExitProcess@4 : proc

.const
msgText db 'Windows assembly language lives!', 0
msgCaption db 'Hello World', 0

.code
Main:
push 0
push offset msgCaption
push offset msgText
push 0
call MessageBoxA@16
push eax
call ExitProcess@4

End Main
```



https://www.bigmessowires.com/2015/10/06/assembly-language-windows-programming/

Criando uma interface gráfica no Linux

```c
#include <unistd.h> /* pause() */

#include <xcb/xcb.h>

int main() {
    /* Open the connection to the X server */
    xcb_connection_t *connection = xcb_connect(NULL, NULL);

    /* Get the first screen */
    const xcb_setup_t *setup = xcb_get_setup(connection);
    xcb_screen_iterator_t iter = xcb_setup_roots_iterator(setup);
    xcb_screen_t *screen = iter.data;

    /* Create the window */
    xcb_window_t window = xcb_generate_id(connection);
    xcb_create_window(connection,                    /* Connection          */
                      XCB_COPY_FROM_PARENT,          /* depth (same as root)*/
                      window,                        /* window Id           */
                      screen->root,                  /* parent window       */
                      0, 0,                          /* x, y                */
                      150, 150,                      /* width, height       */
                      10,                            /* border_width        */
                      XCB_WINDOW_CLASS_INPUT_OUTPUT, /* class               */
                      screen->root_visual,           /* visual              */
                      0, NULL);                      /* masks, not used yet */

    /* Map the window on the screen */
    xcb_map_window(connection, window);

    /* Make sure commands are sent before we pause so that the window gets shown */
    xcb_flush(connection);

    pause(); /* hold client until Ctrl-C */

    xcb_disconnect(connection);

    return 0;
}
```



O problema

Debate acerca da definição de plataforma trazida pelo GNOME

A FAVOR

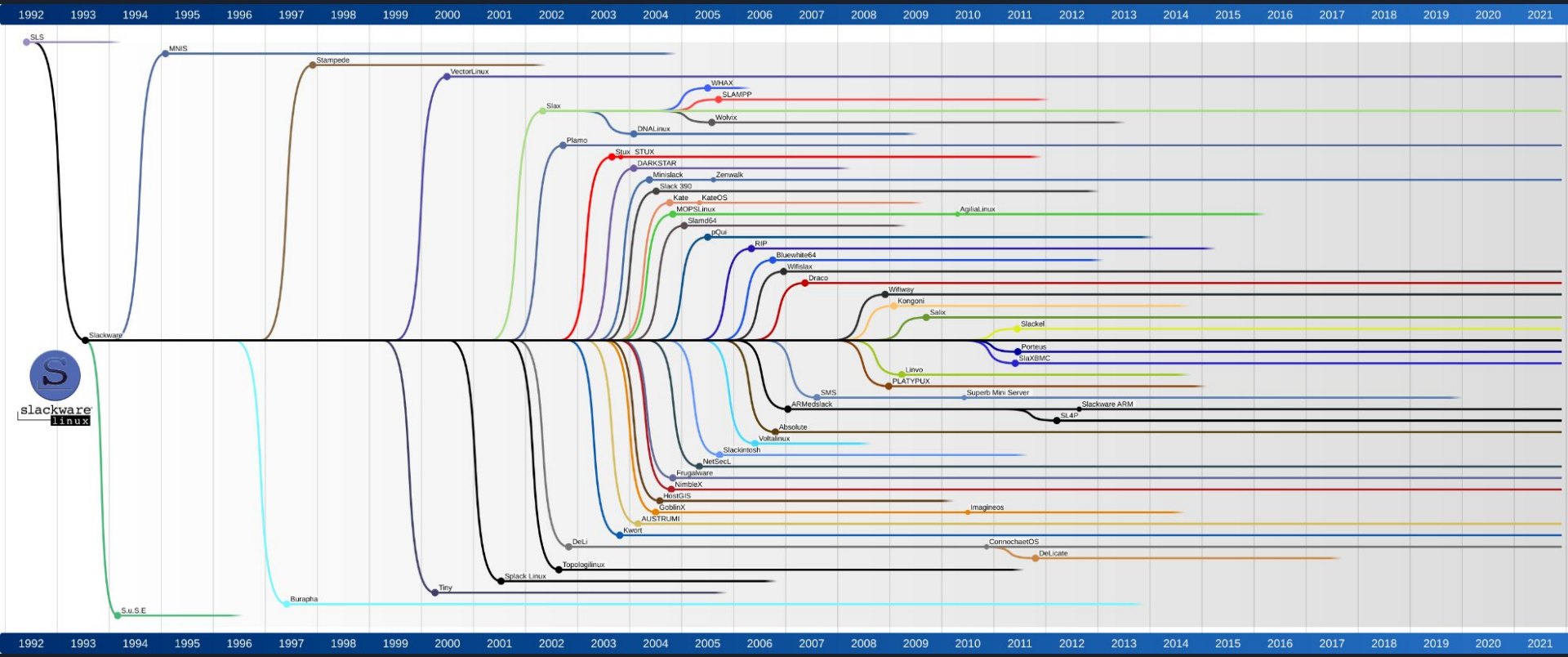
- Pouca acessibilidade aos gerenciadores de pacotes por CLI para usuários comuns.
- Pouca usabilidade no método de instalação de recursos e no manuseio dos múltiplos repositórios.
- O download livre de recursos na internet pode ser prejudicial ao usuário comum.
- Definição de plataforma depende do período atual.

CONTRA

- Sistemas de distribuição, como apt e dnf, são suficientemente integrados.
- Download de recursos era feito inteiramente na internet sem necessidade de uma store.
- Linux não foi desenvolvido para ser um sistema doméstico para usuário comum.

https://blogs.gnome.org/tbernard/2019/12/04/there-is-no-linux-platform-1/
https://blogs.gnome.org/tbernard/2020/03/25/there-is-no-linux-platform-2/

Distribuições



https://github.com/FabioLolix/LinuxTimeline

Distribuindo software

.tar.gz

.deb

.tar.zst

.tar.xz

.rpm

Distribuindo software

AppImage

FLATPAK

snapcraft

Perspectiva do Futuro

- Várias distribuições ou unificação delas.
- Muito trabalho pela frente.
- Compreensão do problema.

O "fim" das plataformas

Electron

PWA

Tauri

Flutter