EDE
Light Desktop Environment
Description And Best Practices

Sanel Zukan
karijes@users.sourceforge.net
Introduction

EDE = Equinox Desktop Environment

Desktop environment like:

- KDE
- GNOME
- XFCE
Introduction

- Fast
- Familiar look and behaviour
- Simple
- Small
- Suitable for older computers
Look and feel

- Already known look and behaviour
- Common desktop elements
- Not trying to reinvent the wheel
- Almost no documentation to start using it
GUI Toolkit

FLTK = Fast Light ToolKit

http://www.fltk.org
GUI Toolkit

- Used in movie industry (Titanic, King Kong, etc.)
- ONLY GUI toolkit
- Small core and very portable
- Small library dependency
- Fast startup time
GUI Toolkit

ldd on

- gtk+ 2.10 (gtk-demo): 33
- Qt 3.3 (qtconfig): 36
- Qt 4.4 (qtconfig): 31
- FLTK (fluid): 22
GUI Toolkit

EDE 1.x used our FLTK fork eFLTK (extended FLTK)
GUI Toolkit

edelib:
- Additional widgets
- freedesktop.org stuff
- D-Bus
- Common things like:
  - linked list
  - strings
  - functions (scandir, strlcpy, etc.)
GUI Toolkit

edelib:

- NO STL
- NO Boost
- NO whatever-today-is-popular-in-C++-world
GUI Toolkit

std::list vs. edelib::list

- SMALLER executable size
- FASTER compile time
- STABLE sort() algorithm
Executable size

```g++ -O2```
Executable size

`g++ -Os`
Compilation time in sec.

- `edelib::list`
- `std::list`
GUI Toolkit

edelib:

- And NO exceptions
No exceptions?#@!!

- Hard to use correctly
- Code becomes hard to follow
- Hard to deduce what are exceptional cases and what not
- Binary size increases
“g++ -fno-exceptions” possible with FLTK and edelib
Desktops memory usage comparison (quick)
XFCE 4.4.1

Total Memory
- Total Free Memory 80%
- Used Physical Memory 19%
- 1.61 GB free

Physical Memory
- Free Physical Memory 59%
- Disk Cache 20%
- Application Data 15%
- 591.39 MB free

Swap Space
- Free Swap 100%
- 1.03 GB free
<table>
<thead>
<tr>
<th></th>
<th>Total Memory</th>
<th>Physical Memory</th>
<th>Swap Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Free Memory</td>
<td>79%</td>
<td>58%</td>
<td>100%</td>
</tr>
<tr>
<td>Used Physical Memory</td>
<td>20%</td>
<td>Disk Cache 21%</td>
<td>Free Swap 100%</td>
</tr>
<tr>
<td>Application Data</td>
<td>14%</td>
<td>Application Data 14%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.61 GB free</td>
<td>588.46 MB free</td>
<td>1.03 GB free</td>
</tr>
</tbody>
</table>
EDE 2.0b

Total Memory
- Total Free Memory 81%
- Used Physical Memory 18%
- 1.64 GB free

Physical Memory
- Free Physical Memory 62%
- Disk Cache 18%
- Application Data 13%
- 624.37 MB free

Swap Space
- Free Swap 100%
- 1.03 GB free
Compilation time counted too!
Build tools

• *make* unusable without autoconf and automake

• Have you every tried to write advanced *make* scripts???

• *make* replaced with jam
  
  (http://www.perforce.com/jam)
Build tool

• jam benefits:
  • Sane syntax
  • No make recursion problem: targets are scanned in one pass
  • Allow you to write some library and reuse code
  • FAST!!!
jam sample

rule EdeProgram
{
    if ! $(EDELIBLIB) {
        Echo "EDELIBLIB not defined; $(1) will not be built" ;
        return ;
    }

    MakeProgramPrivate $(1) : $(2)
    : $(EDELIB_GUI_LIB) $(EDELIBLIB) $(FLTKLIB) $(STDLIB)
    : $(GLOBALFLAGS) $(EDELIBINCLUDE) $(FLTKINCLUDE) ;

    if $(3) != "noinstall" {
        InstallEdeProgram $(1) ;
    }
}

# calling
EdeProgram demo : demo-file1.cpp demo-file2.cpp ;
freedesktop.org (fd.o) standards conformance
Implemented fd.o specs

- XSETTINGS
- Base Directory specification
- Desktop Entry
- Icon theme
- Menu
- Window manager hints (not full)
- Autostart
Pending fd.o specs

- File trash
- Sound themes
- Thumbnailing (when/if gets completed ;))
Coding?
#include <FL/Fl.H>
#include <FL/Fl_Window.H>

int main(int argc, char **argv) {
    Fl_Window *win = new Fl_Window(100, 100, "Sample window");
    win->show(argc, argv);
    return Fl::run();
}
g++ foo.cpp -o foo `fltk-config --cflags --ldflags`
Simple window in edelib

```cpp
#include <FL/Fl.H>
#include <edelib/Window.h>

EDELIB_NS_USING_AS(Window, MyWindow)

int main(int argc, char **argv) {
    MyWindow *win = new MyWindow(100, 100, "Sample window");
    win->show(argc, argv);
    return Fl::run();
}
```
g++ foo.cpp -o foo `pkg-config edelib --cflags --libs` `fltk-config --cflags --ldflags`
Another example

```c
#include <FL/Fl.H>
#include <FL/Fl_Button.H>
#include <edelib/Window.h>
#include <edelib/IconTheme.h>

EDELIB_NS_USING_AS(Window, MyWindow)
EDELIB_NS_USING(IconTheme)
EDELIB_NS_USING(ICON_SIZE_SMALL)

static void close_cb(Fl_Widget*, void *ww) {
    MyWindow *win = (MyWindow*) ww;
    win->hide();
}

int main(int argc, char **argv) {
    MyWindow *win = new MyWindow(300, 300, "My Window");
    win->begin();
    Fl_Button *b = new Fl_Button(10, 10, 90, 25, "Click me");
    b->callback(close_cb, win);
    IconTheme::set(b, "some-icon", ICON_SIZE_SMALL);
    win->end();
    win->show(argc, argv);
    return Fl::run();
}
```
http://equinox-project.org