

Enlightenment: A Cross Platform Window Manager and Toolkit

Dealing with Enlightenment portability issues in FreeBSD and elsewhere

Daniel Kolesa
Samsung Open Source Group
d.kolesa@samsung.com
@octaforge
FOSDEM 2015

State of the ecosystem

Where are we now?



Overview



- ▶ During the last few years - drastic change of ecosystem

- ▶ During the last few years - drastic change of ecosystem
- ▶ Graphics stack in Linux kernel

- ▶ During the last few years - drastic change of ecosystem
- ▶ Graphics stack in Linux kernel
- ▶ Systemd

- ▶ During the last few years - drastic change of ecosystem
- ▶ Graphics stack in Linux kernel
- ▶ Systemd
- ▶ High level components depending on low level stuff (libudev)

BSDs in the ecosystem



BSDs in the ecosystem



- ▶ Lagging behind

BSDs in the ecosystem



- ▶ Lagging behind
- ▶ Losing compatibility with Linux stuff

BSDs in the ecosystem



- ▶ Lagging behind
- ▶ Losing compatibility with Linux stuff
- ▶ Custom solutions needed

BSDs in the ecosystem



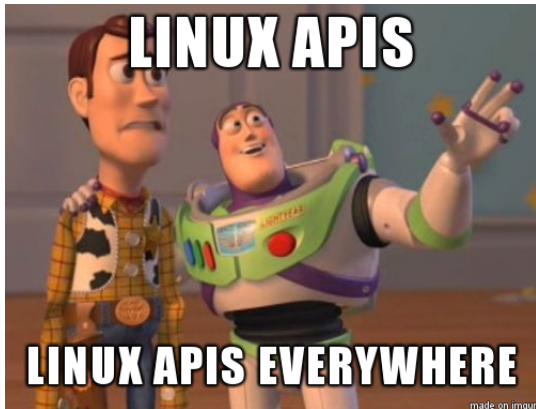
- ▶ Lagging behind
- ▶ Losing compatibility with Linux stuff
- ▶ Custom solutions needed
- ▶ (or wrapper shims)

BSDs in the ecosystem



- ▶ Lagging behind
- ▶ Losing compatibility with Linux stuff
- ▶ Custom solutions needed
- ▶ (or wrapper shims)
- ▶ (trying to avoid that)

General portability tips



Overview



- ▶ We have a very diverse ecosystem

- ▶ We have a very diverse ecosystem
- ▶ This includes a wide range of operating systems

- ▶ We have a very diverse ecosystem
- ▶ This includes a wide range of operating systems
- ▶ Not all operating system have the same features

- ▶ We have a very diverse ecosystem
- ▶ This includes a wide range of operating systems
- ▶ Not all operating system have the same features
- ▶ Writing portable software is painful, but very much worth it

- ▶ We have a very diverse ecosystem
- ▶ This includes a wide range of operating systems
- ▶ Not all operating system have the same features
- ▶ Writing portable software is painful, but very much worth it
- ▶ And the end result comes out cleaner

Don't write against a platform



Don't write against a platform



- ▶ A big mistake we've done in the EFL

Don't write against a platform



- ▶ A big mistake we've done in the EFL
- ▶ We wrote code against Linux

Don't write against a platform



- ▶ A big mistake we've done in the EFL
- ▶ We wrote code against Linux
- ▶ Every other platform is expected to implement the same APIs

Don't write against a platform



- ▶ A big mistake we've done in the EFL
- ▶ We wrote code against Linux
- ▶ Every other platform is expected to implement the same APIs
- ▶ Wrappers then implement API shims

Why is this wrong?



Why is this wrong?



- ▶ System specific APIs are often unnecessarily low level

Why is this wrong?



- ▶ System specific APIs are often unnecessarily low level
- ▶ Low level → difficult to write

Why is this wrong?



- ▶ System specific APIs are often unnecessarily low level
- ▶ Low level → difficult to write
- ▶ Low level → difficult to maintain

Why is this wrong?



- ▶ System specific APIs are often unnecessarily low level
- ▶ Low level → difficult to write
- ▶ Low level → difficult to maintain
- ▶ And a pain to port

Why is this wrong?



- ▶ System specific APIs are often unnecessarily low level
- ▶ Low level → difficult to write
- ▶ Low level → difficult to maintain
- ▶ And a pain to port
- ▶ Also, every time you do it, a kitten dies

Why is this wrong?



- ▶ System specific APIs are often unnecessarily low level
- ▶ Low level → difficult to write
- ▶ Low level → difficult to maintain
- ▶ And a pain to port
- ▶ Also, every time you do it, a kitten dies
- ▶ Too late to save them now

The right approach



The right approach



- ▶ Write general code

The right approach



- ▶ Write general code
- ▶ If you need any specific functionality, design a high level API for it

The right approach



- ▶ Write general code
- ▶ If you need any specific functionality, design a high level API for it
- ▶ Use this API from your code

The right approach



- ▶ Write general code
- ▶ If you need any specific functionality, design a high level API for it
- ▶ Use this API from your code
- ▶ Write OS specific backends implementing this API

The right approach



- ▶ Write general code
- ▶ If you need any specific functionality, design a high level API for it
- ▶ Use this API from your code
- ▶ Write OS specific backends implementing this API
- ▶ Abstracted, high level, easy to write, easy to maintain

KISS principle



KISS principle



- ▶ Plays an important role

KISS principle



- ▶ Plays an important role
- ▶ Keep your API simple and as general purpose as possible

KISS principle



- ▶ Plays an important role
- ▶ Keep your API simple and as general purpose as possible
- ▶ Don't implement very specific features

KISS principle



- ▶ Plays an important role
- ▶ Keep your API simple and as general purpose as possible
- ▶ Don't implement very specific features
- ▶ Instead always ask yourself a question:

- ▶ Plays an important role
- ▶ Keep your API simple and as general purpose as possible
- ▶ Don't implement very specific features
- ▶ Instead always ask yourself a question:
- ▶ **Can I generalize this? Can this be reused?**

Don't repeat yourself!



Don't repeat yourself!



- ▶ Write reusable code

Don't repeat yourself!



- ▶ Write reusable code
- ▶ And actually reuse it

Don't repeat yourself!



- ▶ Write reusable code
- ▶ And actually reuse it
- ▶ The worst thing you can do is copy paste a snippet in 10 places

Don't repeat yourself!



- ▶ Write reusable code
- ▶ And actually reuse it
- ▶ The worst thing you can do is copy paste a snippet in 10 places
- ▶ Any update will force you to update it in all 10 places

No internal dependencies



No internal dependencies



- ▶ Internal dependencies are bad

No internal dependencies



- ▶ Internal dependencies are bad
- ▶ They force you to maintain them

No internal dependencies



- ▶ Internal dependencies are bad
- ▶ They force you to maintain them
- ▶ They are not reusable even though they could be

No internal dependencies



- ▶ Internal dependencies are bad
- ▶ They force you to maintain them
- ▶ They are not reusable even though they could be
- ▶ They hinder portability

Do not lock yourself to a toolchain



Do not lock yourself to a toolchain



- ▶ Abusing compiler extensions might be tempting

Do not lock yourself to a toolchain



- ▶ Abusing compiler extensions might be tempting
- ▶ End result is often maintenance hell

Do not lock yourself to a toolchain



- ▶ Abusing compiler extensions might be tempting
- ▶ End result is often maintenance hell
- ▶ Porting such code to a new toolchain sucks

Domain specific languages are good



Domain specific languages are good



- ▶ DSLs allow you to reduce the amount of code

Domain specific languages are good



- ▶ DSLs allow you to reduce the amount of code
- ▶ They increase readability of your code by restricting it

Domain specific languages are good



- ▶ DSLs allow you to reduce the amount of code
- ▶ They increase readability of your code by restricting it
- ▶ They add extra safety

Domain specific languages are good



- ▶ DSLs allow you to reduce the amount of code
- ▶ They increase readability of your code by restricting it
- ▶ They add extra safety
- ▶ They are high level → easier to port

Portability is not only platforms



Portability is not only platforms



- ▶ Operating systems are the typical thing you imagine by portability

Portability is not only platforms



- ▶ Operating systems are the typical thing you imagine by portability
- ▶ But it also includes hardware architectures

Portability is not only platforms



- ▶ Operating systems are the typical thing you imagine by portability
- ▶ But it also includes hardware architectures
- ▶ And programming languages (bindings)

Portability is not only platforms



- ▶ Operating systems are the typical thing you imagine by portability
- ▶ But it also includes hardware architectures
- ▶ And programming languages (bindings)
- ▶ And rendering APIs

Portability is not only platforms



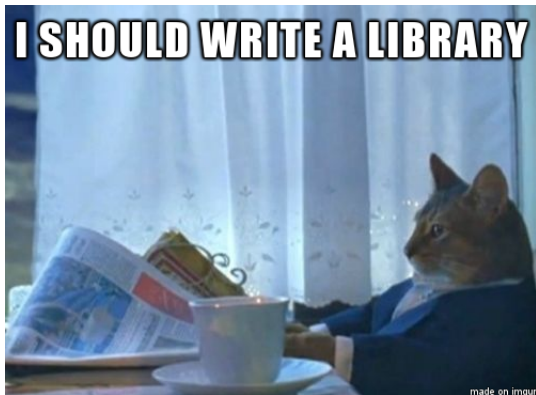
- ▶ Operating systems are the typical thing you imagine by portability
- ▶ But it also includes hardware architectures
- ▶ And programming languages (bindings)
- ▶ And rendering APIs
- ▶ Sound architectures

Portability is not only platforms



- ▶ Operating systems are the typical thing you imagine by portability
- ▶ But it also includes hardware architectures
- ▶ And programming languages (bindings)
- ▶ And rendering APIs
- ▶ Sound architectures
- ▶ And others

Enlightenment/EFL overview



What is Enlightenment?



What is Enlightenment?



- ▶ X11/Wayland desktop shell for Linux, the BSDs and others

What is Enlightenment?



- ▶ X11/Wayland desktop shell for Linux, the BSDs and others
- ▶ Playground for the EFL

What is Enlightenment?



- ▶ X11/Wayland desktop shell for Linux, the BSDs and others
- ▶ Playground for the EFL
- ▶ The prettiest window manager around

What is Enlightenment?



- ▶ X11/Wayland desktop shell for Linux, the BSDs and others
- ▶ Playground for the EFL
- ▶ The prettiest window manager around
- ▶ Crashy mess with portability issues

What is EFL?



What is EFL?



- ▶ Enlightenment Foundation Libraries

What is EFL?



- ▶ Enlightenment Foundation Libraries
- ▶ Masterpiece of engineering

What is EFL?



- ▶ Enlightenment Foundation Libraries
- ▶ Masterpiece of engineering
- ▶ A suite of libraries originally created for Enlightenment

What is EFL?



- ▶ Enlightenment Foundation Libraries
- ▶ Masterpiece of engineering
- ▶ A suite of libraries originally created for Enlightenment
- ▶ These days it is what you mean by Enlightenment

What does the EFL include?



What does the EFL include?



- ▶ Low level libraries (such as C data structures)

What does the EFL include?



- ▶ Low level libraries (such as C data structures)
- ▶ Convenience libraries (D-Bus interface library, physics engine wrapper and others)

What does the EFL include?



- ▶ Low level libraries (such as C data structures)
- ▶ Convenience libraries (D-Bus interface library, physics engine wrapper and others)
- ▶ Graphical libraries (canvas, UI toolkit and others)

What does the EFL include?



- ▶ Low level libraries (such as C data structures)
- ▶ Convenience libraries (D-Bus interface library, physics engine wrapper and others)
- ▶ Graphical libraries (canvas, UI toolkit and others)
- ▶ Some non-portable wrapper mess

EFL portability problems



Build system



- ▶ EFL uses GNU Autotools

Build system



- ▶ EFL uses GNU Autotools
- ▶ Autotools is a terrible monster that eats little children

- ▶ EFL uses GNU Autotools
- ▶ Autotools is a terrible monster that eats little children
- ▶ But it works acceptably on Unix-like systems

- ▶ EFL uses GNU Autotools
- ▶ Autotools is a terrible monster that eats little children
- ▶ But it works acceptably on Unix-like systems
- ▶ It's problematic on Windows

- ▶ EFL uses GNU Autotools
- ▶ Autotools is a terrible monster that eats little children
- ▶ But it works acceptably on Unix-like systems
- ▶ It's problematic on Windows
- ▶ And kind of on OS X

- ▶ EFL uses GNU Autotools
- ▶ Autotools is a terrible monster that eats little children
- ▶ But it works acceptably on Unix-like systems
- ▶ It's problematic on Windows
- ▶ And kind of on OS X
- ▶ No real alternatives

- ▶ EFL uses GNU Autotools
- ▶ Autotools is a terrible monster that eats little children
- ▶ But it works acceptably on Unix-like systems
- ▶ It's problematic on Windows
- ▶ And kind of on OS X
- ▶ No real alternatives
- ▶ Potential alternatives so far proved to be worse

- ▶ EFL uses GNU Autotools
- ▶ Autotools is a terrible monster that eats little children
- ▶ But it works acceptably on Unix-like systems
- ▶ It's problematic on Windows
- ▶ And kind of on OS X
- ▶ No real alternatives
- ▶ Potential alternatives so far proved to be worse
- ▶ Had to go with the lesser evil

Ecore main loop



Ecore main loop



- ▶ Works on all supported platforms

Ecore main loop



- ▶ Works on all supported platforms
- ▶ Can use epoll on Linux for better performance

Ecore main loop



- ▶ Works on all supported platforms
- ▶ Can use epoll on Linux for better performance
- ▶ Therefore we should also have kqueue support

- ▶ Works on all supported platforms
- ▶ Can use epoll on Linux for better performance
- ▶ Therefore we should also have kqueue support
- ▶ Cleanup is needed - move the epoll parts out of mainloop source

- ▶ Currently only supports PulseAudio (limited support for ALSA)

- ▶ Currently only supports PulseAudio (limited support for ALSA)
- ▶ PulseAudio works on the BSDs

- ▶ Currently only supports PulseAudio (limited support for ALSA)
- ▶ PulseAudio works on the BSDs
- ▶ Most people don't want it

- ▶ Currently only supports PulseAudio (limited support for ALSA)
- ▶ PulseAudio works on the BSDs
- ▶ Most people don't want it
- ▶ Solution - implement OSS support

- ▶ Currently Linux only

- ▶ Currently Linux only
- ▶ *BSD support would be relevant

- ▶ Currently Linux only
- ▶ *BSD support would be relevant
- ▶ Uses libinput and optionally systemd-login/logind (otherwise needs root)

- ▶ Currently Linux only
- ▶ *BSD support would be relevant
- ▶ Uses libinput and optionally systemd-login/logind (otherwise needs root)
- ▶ Solution for libinput - have to wait

- ▶ Currently Linux only
- ▶ *BSD support would be relevant
- ▶ Uses libinput and optionally systemd-login/logind (otherwise needs root)
- ▶ Solution for libinput - have to wait
- ▶ Solution for systemd-login/logind - perhaps ConsoleKit2?

- ▶ Currently Linux only
- ▶ *BSD support would be relevant
- ▶ Uses libinput and optionally systemd-login/logind (otherwise needs root)
- ▶ Solution for libinput - have to wait
- ▶ Solution for systemd-login/logind - perhaps ConsoleKit2?
- ▶ Or use the LoginKit shim

- ▶ Currently Linux only
- ▶ *BSD support would be relevant
- ▶ Uses libinput and optionally systemd-login/logind (otherwise needs root)
- ▶ Solution for libinput - have to wait
- ▶ Solution for systemd-login/logind - perhaps ConsoleKit2?
- ▶ Or use the LoginKit shim
- ▶ Depending on LoginKit feels messy

- ▶ Also Linux only right now

- ▶ Also Linux only right now
- ▶ Uses evdev

- ▶ Also Linux only right now
- ▶ Uses evdev
- ▶ Solution for evdev?

- ▶ Also Linux only right now
- ▶ Uses evdev
- ▶ Solution for evdev?
- ▶ GSoC 2014 implements evdev in FreeBSD, but not yet upstream

- ▶ Also Linux only right now
- ▶ Uses evdev
- ▶ Solution for evdev?
- ▶ GSoC 2014 implements evdev in FreeBSD, but not yet upstream
- ▶ Other BSDs? Have everyone implement evdev?

- ▶ Also Linux only right now
- ▶ Uses evdev
- ▶ Solution for evdev?
- ▶ GSoC 2014 implements evdev in FreeBSD, but not yet upstream
- ▶ Other BSDs? Have everyone implement evdev?
- ▶ Or split away the evdev stuff and write OS specific backends?

- ▶ Also Linux only right now
- ▶ Uses evdev
- ▶ Solution for evdev?
- ▶ GSoC 2014 implements evdev in FreeBSD, but not yet upstream
- ▶ Other BSDs? Have everyone implement evdev?
- ▶ Or split away the evdev stuff and write OS specific backends?
- ▶ Also needs libwayland - need to wait for Wayland ports

- ▶ Also Linux only right now
- ▶ Uses evdev
- ▶ Solution for evdev?
- ▶ GSoC 2014 implements evdev in FreeBSD, but not yet upstream
- ▶ Other BSDs? Have everyone implement evdev?
- ▶ Or split away the evdev stuff and write OS specific backends?
- ▶ Also needs libwayland - need to wait for Wayland ports
- ▶ Blocks on ecore_drm

- ▶ Udev wrapper library (+ libmount)

- ▶ Udev wrapper library (+ libmount)
- ▶ Bad idea

- ▶ Udev wrapper library (+ libmount)
- ▶ Bad idea
- ▶ Temporary solution - use (and potentially extend) libdevq?

- ▶ Udev wrapper library (+ libmount)
- ▶ Bad idea
- ▶ Temporary solution - use (and potentially extend) libdevq?
- ▶ Might not be possible

- ▶ Udev wrapper library (+ libmount)
- ▶ Bad idea
- ▶ Temporary solution - use (and potentially extend) libdevq?
- ▶ Might not be possible
- ▶ Current plan - deprecate Eeze

- ▶ Udev wrapper library (+ libmount)
- ▶ Bad idea
- ▶ Temporary solution - use (and potentially extend) libdevq?
- ▶ Might not be possible
- ▶ Current plan - deprecate Eeze
- ▶ Come up with a high level library instead

- ▶ Udev wrapper library (+ libmount)
- ▶ Bad idea
- ▶ Temporary solution - use (and potentially extend) libdevq?
- ▶ Might not be possible
- ▶ Current plan - deprecate Eeze
- ▶ Come up with a high level library instead
- ▶ Platform specific backends in the library (udev, devd/libdevq...)

Enlightenment portability problems



Overview



- ▶ EFL portability problems also affect Enlightenment

- ▶ EFL portability problems also affect Enlightenment
- ▶ No wayland support on *BSD

- ▶ EFL portability problems also affect Enlightenment
- ▶ No wayland support on *BSD
- ▶ No eeze on *BSD

- ▶ EFL portability problems also affect Enlightenment
- ▶ No wayland support on *BSD
- ▶ No eeze on *BSD
- ▶ And other problems

Ptrace



- ▶ Since a while ago, Enlightenment startup executable uses ptrace

- ▶ Since a while ago, Enlightenment startup executable uses ptrace
- ▶ Used to catch segfaults and display a window allowing a restart

- ▶ Since a while ago, Enlightenment startup executable uses ptrace
- ▶ Used to catch segfaults and display a window allowing a restart
- ▶ Replaces old unreliable way

- ▶ Since a while ago, Enlightenment startup executable uses ptrace
- ▶ Used to catch segfaults and display a window allowing a restart
- ▶ Replaces old unreliable way
- ▶ PT_GETSIGINFO is used - Linux specific extension

- ▶ Since a while ago, Enlightenment startup executable uses ptrace
- ▶ Used to catch segfaults and display a window allowing a restart
- ▶ Replaces old unreliable way
- ▶ PT_GETSIGINFO is used - Linux specific extension
- ▶ Therefore ptrace is not used on *BSD and a crash will go to tty

- ▶ Used to manage devices in Enlightenment

- ▶ Used to manage devices in Enlightenment
- ▶ No eeze → no device management

- ▶ Used to manage devices in Enlightenment
- ▶ No eeze → no device management
- ▶ Also used for backlight handling

- ▶ Used to manage devices in Enlightenment
- ▶ No eeze → no device management
- ▶ Also used for backlight handling
- ▶ Also used for temperature monitoring

- ▶ Used to manage devices in Enlightenment
- ▶ No eeze → no device management
- ▶ Also used for backlight handling
- ▶ Also used for temperature monitoring
- ▶ Solution: eeze replacement

Mixer



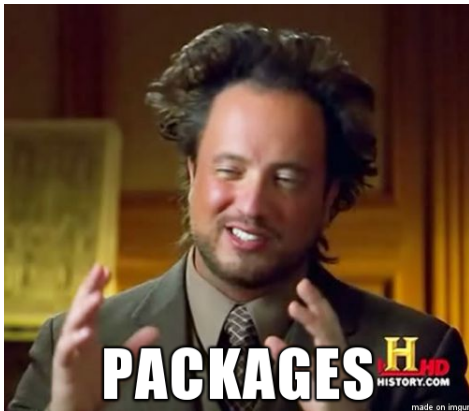
- ▶ Current mixer module only supports PulseAudio and ALSA

- ▶ Current mixer module only supports PulseAudio and ALSA
- ▶ Also causes high CPU loads on FreeBSD with Pulse

- ▶ Current mixer module only supports PulseAudio and ALSA
- ▶ Also causes high CPU loads on FreeBSD with Pulse
- ▶ New mixer in development

- ▶ Current mixer module only supports PulseAudio and ALSA
- ▶ Also causes high CPU loads on FreeBSD with Pulse
- ▶ New mixer in development
- ▶ OSS support needed in the new mixer

Other problems



Distribution



- ▶ FreeBSD ports provide EFL/E → good

- ▶ FreeBSD ports provide EFL/E → good
- ▶ Poor communication with upstream EFL and the other way around

- ▶ FreeBSD ports provide EFL/E → good
- ▶ Poor communication with upstream EFL and the other way around
- ▶ I'm the only bridge

- ▶ FreeBSD ports provide EFL/E → good
- ▶ Poor communication with upstream EFL and the other way around
- ▶ I'm the only bridge
- ▶ Relatively low interest (but there is some)

- ▶ FreeBSD ports provide EFL/E → good
- ▶ Poor communication with upstream EFL and the other way around
- ▶ I'm the only bridge
- ▶ Relatively low interest (but there is some)
- ▶ Situation getting better

Windows



The good



The good



- ▶ Evil library provides part of POSIX

The good



- ▶ Evil library provides part of POSIX
- ▶ Most components have Windows related code

The good



- ▶ Evil library provides part of POSIX
- ▶ Most components have Windows related code
- ▶ Native gdi/ddraw graphics backends

The good



- ▶ Evil library provides part of POSIX
- ▶ Most components have Windows related code
- ▶ Native gdi/ddraw graphics backends
- ▶ Overall decent code coverage

Build system



- ▶ Because of autotools, we can only support MinGW/MSYS environments

- ▶ Because of autotools, we can only support MinGW/MSYS environments
- ▶ Create Visual Studio project files?

- ▶ Because of autotools, we can only support MinGW/MSYS environments
- ▶ Create Visual Studio project files?
- ▶ Use CMake? Premake? ...

- ▶ Because of autotools, we can only support MinGW/MSYS environments
- ▶ Create Visual Studio project files?
- ▶ Use CMake? Premake? ...
- ▶ Neither of these solutions provide some of our used Autotools features

- ▶ Because of autotools, we can only support MinGW/MSYS environments
- ▶ Create Visual Studio project files?
- ▶ Use CMake? Premake? ...
- ▶ Neither of these solutions provide some of our used Autotools features
- ▶ No distcheck, no easy file pre-generation

- ▶ Because of autotools, we can only support MinGW/MSYS environments
- ▶ Create Visual Studio project files?
- ▶ Use CMake? Premake? ...
- ▶ Neither of these solutions provide some of our used Autotools features
- ▶ No distcheck, no easy file pre-generation
- ▶ Create build scripts to trigger from build system?

Availability



- ▶ The above → difficult to ship

- ▶ The above → difficult to ship
- ▶ No official Windows builds

- ▶ The above → difficult to ship
- ▶ No official Windows builds
- ▶ win-builds.org provides unofficial builds

Other issues



- ▶ Ecore audio support should be added

OS X



The good



The good



- ▶ Native Cocoa backend

The good



- ▶ Native Cocoa backend
- ▶ Unix-like guts → easy to cover

The good



- ▶ Native Cocoa backend
- ▶ Unix-like guts → easy to cover
- ▶ Some FreeBSD APIs present (kqueue)

Build system



- ▶ Similar issues as on Windows to a lesser degree

Build system



- ▶ Similar issues as on Windows to a lesser degree
- ▶ Standard shell tools are present

Build system



- ▶ Similar issues as on Windows to a lesser degree
- ▶ Standard shell tools are present
- ▶ XCode project files?

Availability



- ▶ No official or unofficial builds (as far as I know)

- ▶ No official or unofficial builds (as far as I know)
- ▶ You have to compile on your own

- ▶ No official or unofficial builds (as far as I know)
- ▶ You have to compile on your own
- ▶ Major lack of testing (no CI setup, very few developers)

Final summary



- ▶ Linux infra changes made an already difficult thing even more difficult

Final summary



- ▶ Linux infra changes made an already difficult thing even more difficult
- ▶ Code modularization and abstraction is needed

- ▶ Linux infra changes made an already difficult thing even more difficult
- ▶ Code modularization and abstraction is needed
- ▶ Build system might not be ideal, but it's the best we have

- ▶ Linux infra changes made an already difficult thing even more difficult
- ▶ Code modularization and abstraction is needed
- ▶ Build system might not be ideal, but it's the best we have
- ▶ Windows support is a little painful

- ▶ Linux infra changes made an already difficult thing even more difficult
- ▶ Code modularization and abstraction is needed
- ▶ Build system might not be ideal, but it's the best we have
- ▶ Windows support is a little painful
- ▶ Same goes for Mac

- ▶ Linux infra changes made an already difficult thing even more difficult
- ▶ Code modularization and abstraction is needed
- ▶ Build system might not be ideal, but it's the best we have
- ▶ Windows support is a little painful
- ▶ Same goes for Mac
- ▶ Improvements are coming :)

Thank you.

Daniel Kolesa
Samsung Open Source Group
d.kolesa@samsung.com
@octaforge
FOSDEM 2015