Adding VST Support to Linux Audio Applications

Paul Davis

Linux Audio Systems

(based on work by Kjetil Mattheussen, Torben Hohn and Paul Davis)

What is VST?

- Technology from Steinberg
- An audio/MIDI plugin API
- Support for win32/x86, MacOS9, MacOS X
- Plugins roughly divided into a DSP core and an optional customized GUI

The Problem

- VST plugins for Windows are packaged in a win32/x86 file format
- VST plugins for Windows make win32 API calls
- Linux does not (natively) support either of these.

The Solution

- Wine
- WINdows Emulator?
- Wine Is Not an Emulator?
- Wine provides 95%+ of the win32 API by translating it into native linux library and OS calls.
- Traps systems calls, exceptions
- 1,000,000+ lines of code, +25%/month

The Problem, Part II

- Wine is intended for running win32 programs
- We have win32 plugins
- We want to use them in a native Linux application
- We are not alone: Mono, game people using DirectX, others.

The Kjetil is not Black

- Kjetil Mattheussen did the first open source implementation of VST support
- Created a win32 application
- Compiled it with winebuild
- Result: a native Linux app that can load win32 .dll's (VST plugins)
- Connect to other Linux programs with a client/server protocol

VSTserver Features

- Isolates VST and win32 code from host
- Plugin can crash without affecting the host
- Plugin cannot access host data structures
- Sweet!
- But ...

Problems with VSTserver

- 2 context switches to run a plugin's process() callback
- Doesn't scale
- Imagine a session with 2-4 plugins on most of 24 tracks.

A Different Approach

- Run VST plugins in the same process as the host
- Must be able to load and run win32 code
- Don't use wine(1) or winebuild
- Mono's shared wine hack

Mono and setjmp

```
WinMain () {
    longjmp (jmpbuf, 1);
}
```

```
if (setjmp (jmpbuf) == 0) {
    wine_init (“run me”);
    /*NOTREACHED*/
}
```

```
... now have a win32-ready thread...
```

The Problem, Part III

- The Mono hack doesn't support threads
- All audio apps require threads
- Therefore ...

Win32 Thread Ugliness

- All thread systems need some way to identify the current thread
- win32/x86 uses the %fs register
- for comparison, NPTL and later linuxthreads use %gs
- Contents point to a block of information allocated from the processor Local Descriptor Table (LDT)
- Any thread running win32 must have %fs point to a valid LDT entry

Ugly? Or Just Windows?

- Wine takes over thread management
- Wine starts before the application
- With a native Linux application, there are threads created using the native thread API
- Wine doesn't know about these threads, thus %fs is not set up
- `wine_adopt_thread ();`

How We Do It

- Allocate a special proxy thread when wine is started, using win32 thread API
- Thread waits for notifications of new linux threads
- Tells Wine about them, passes back %fs information
- Linux thread has %fs set, can now execute win32 code.

The GUI

- Most VST plugins come with their own custom editor
- Native win32 or vstgui calls – its all the same to us
- Need an entire win32 event loop running the host.
- Window management ...

Window Management 1

- the win32-created window has no WM “decoration”
- the win32 thread is an Xwindow, but no X toolkit knows about it
- linux apps can't manage it as if it was a normal toolkit window
- we need to “adopt” the window
- XReparentWindow

Window Management 2

- we wait for the window to be created
- use whatever wrapper around XReparentWindow a toolkit provides
- for GTK+, GtkSocket, gtk_socket_add_id(), gtk_socket_steal()
- ideally, use XEMBED extension
- not supported by wine

Window Management 3

- after reparenting, when the window is moved, the win32 GUI layer doesn't know
- coordinates of mouse events are wrong
- have to forward the correct kind of XEvent to the win32 layer
- in GTK+, catch “configure” events, synthesize a new XConfigure event, send to the XWindow underlying the win32 window.

Writing a Host

- you need to:
- (1) supply an “audioMaster” callback to handle requests from the plugin
- (2) probably augment the FST data structures with something application specific (think LADSPA)
- (3) take care of window management (configure events, etc.)

Plugin Discovery

- Suprise – its really slow (loading multiple win32 .dll files)
- “fstconfig” builds caches of basic VST plugin info in foo.fst
- fst_get_info (dllname) will get (and if necessary re-build) info for a .dll
- much, much, much faster
- LRDF?

The API

```
fst_init ();
fst_load ();
fst_unload ();
fst_instantiate ();
fst_close ();
fst_run_editor ();
fst_destroy_editor ();
fst_get_info ();
```

Success Stories

- Kontakt, Battery from Native Instruments
- Crystal
- Lots and lots of small, interesting and useful plugins

Problems

- where to begin?

Without Whom

- Torben Hohn
- Kjetil Mattheussen
- The Mono crew for getting us started
- Alexandre Julliard, Mike Hearns, Mike McCormack and Chris Morgan of the Wine project for advice and guidance.
- Smartelectronix for open source plugins.
- Steinberg. Hmm. Yes.