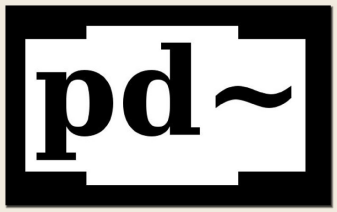


kit de conexion de audio

la piedra angular de
la produccion de audio
con software libre

Misiones
Software Libre

charla presentada en el
4to festival misionero
de software libre
21 de mayo 2011
obera - misiones

Qué es JACK?

- JACK es un sistema para manejar en tiempo real audio de baja latencia y MIDI.
- Multiplataforma: GNU/Linux, Solaris, FreeBSD, OS X, y MS Windows.
- Puede conectar distintas aplicaciones a un dispositivo de audio a la vez que les permite compartir audio y MIDI entre ellas.
- Las aplicaciones cliente puden correr procesos independientes o fucionar dentro del servidor JACK como un plugin.
- Permite distribuir el procesamiento de audio en redes LAN y WAN.
- JACK fue diseñado desde cero para el trabajo profesional de audio y su diseño se enfoca en dos areas clave: ejecución sincronizada de todos los clientes, y operación con bajas latencias.

Entendiendo a JACK

•La Interfaz de Programación de Aplicaciones (API) de JACK
Definición de estructuras de datos, protocolos y funciones utilizadas por programas que utilicen JACK.

•Una implementación de la API
Programas, incluyendo un servidor, y bibliotecas utilizadas por clientes. Hay dos implementaciones actualmente: JACK1 y JACK2

•Una instancia de JACK y una variedad de aplicaciones "JACK-aware" o "Jackified" .
Normalmente un conjunto de un demonio-servidor, la GUI de control, y cero o más aplicaciones que usan **JACK**, como **Ardour** (DAW), **Hydrogen** (drum machine), **Rosegarden** (secuenciador MIDI), **ZynAddSubFX** (sintetizador de software), **Pure data** (lenguage de programación para multimedia) y **Blender** (suite de creación de contenidos 3D) por citar algunos.

Diseño e implementación

Extractos de la presentación de Paul Davis (Linux Audio Systems).

LAD Conference, ZKM Karlsruhe, 2003

Porqué?

Mayo 2001

LAD discute la necesidad de intercambiar datos sincronozados de baja latencia y alto ancho de banda entre aplicaciones.

- Editor/secuenciador multitrack
- Sintetizador Modular por software
- Drum machine
- MIDI pattern sequencer

Otra motivación

- Muchos plugins LADSPA requieren GUIs complicadas y dedicadas.

Enfoques tradicionales

Los usuarios de Ms Windows y MacOS se enfrentaban con los mismos problemas

Que Hicieron?

- Escribir aplicaciones host monoliticas, desarrollar varias APIs para plugins, y correr muchas cosas como plugins (VSTi, Dxi, etc)
- Las APIs de los plugins requieren que los plugins sean objetos compartidos que se ejecutan dentro del contexto del host.

Qué más?

- ReWire (Propellerheads) y DirectConnect (Digidesign) aparecen como alternativas.

Objetivos de Diseño

- Jack debería permitir streaming de datos de baja latencia y alto ancho de banda entre aplicaciones independientes.
- Jack debería soportar cualquier tipo de streaming de datos, no sólo audio.
- En un setup de JACK, habrá un servidor y uno o más plugins.
- Las aplicaciones conectadas via Jack pueden tener sus propias interfaces gráficas. Jack no se limitará a ciertas bibliotecas o toolkits gráficos.
- Jack debería proveer sincronzación completa y precisa. O sea, ejecución sincronizada de todos los plugins clientes.

Objetivos de Diseño II

- Un cliente de Jack puede consumir o producir múltiples streams de datos.
- El API de Jack debería ser especificado en ANSI C. No hay restricciones en cómo se implementan los clientes y servidores.
- Debería ser posible conectar aplicaciones en ejecución.
- Debería ser posible añadir y quitar clientes Jack mientras el servidor está corriendo.

Latencia

Experimentada como la demora entre una acción y su efecto, por ejemplo:

- Presionar una teclar, oir una nota
- Mover el mouse, cambiar el volumen

Latencias de más de 3ms hacen dificil el trabajo de músicos e ingenieros de sonido. (Distancias típicas a un parlante generan 3ms de latencia)

Latencia II

Latencia de entrada

- Los datos llegan a la placa de sonidos
- Espera en buffer hasta la próxima interrución
- IRQ
- A disposición del CPU

Latencia de salida

- Datos listos para la placa de sonido
- El driver bloquea/retiene hasta la proxima interrupción
- IRQ
- El CPU puede entregar los datos

pero...

- Otros datos pueden ya estar en cola

Latencia III

En síntesis

- Latencia total de salida: tamaño del buffer
- Latencia de entrada: intervalo de interrupción
- En otras palabras: podemos reaccionar a las entradas dentro de 1 interrupción, y entregar audio dentro de 2 interrupciones.

El PROBLEMA CON X WINDOW

- X es un sistema de abstracción de bajo nivel.
- Otras capas de soft llamadas "toolkits" permiten escribir aplicaciones más sencillamente (GTK+, Qt, FLTK, etc)
- Es muy dificil utilizar 2 toolkit en el mismo proceso.
- Si un plugin utiliza el Toolkit A y el host el Toolkit B las cosas se complican

Entonces Jack permite correr plugins en otro proceso.

Beneficios:

- Aisla al host de posibles errores en los plugins
- Los desarrolladores pueden elegir sus Toolkits para la GUI
- Pero, esto trae un overhead cuando corremos plugins.

UN NUEVO DISEÑO

Objetivos simplificados

- Correr clientes internos como cualquier API de objetos compartidos
- Correr clientes externos también (en otro proceso)

JACK como un API de audio

La intención de Jack no era proveer una nueva API sino introducir un nuevo modelo de programación

Como API JACK provee muchos beneficios por su alto nivel de abstracción

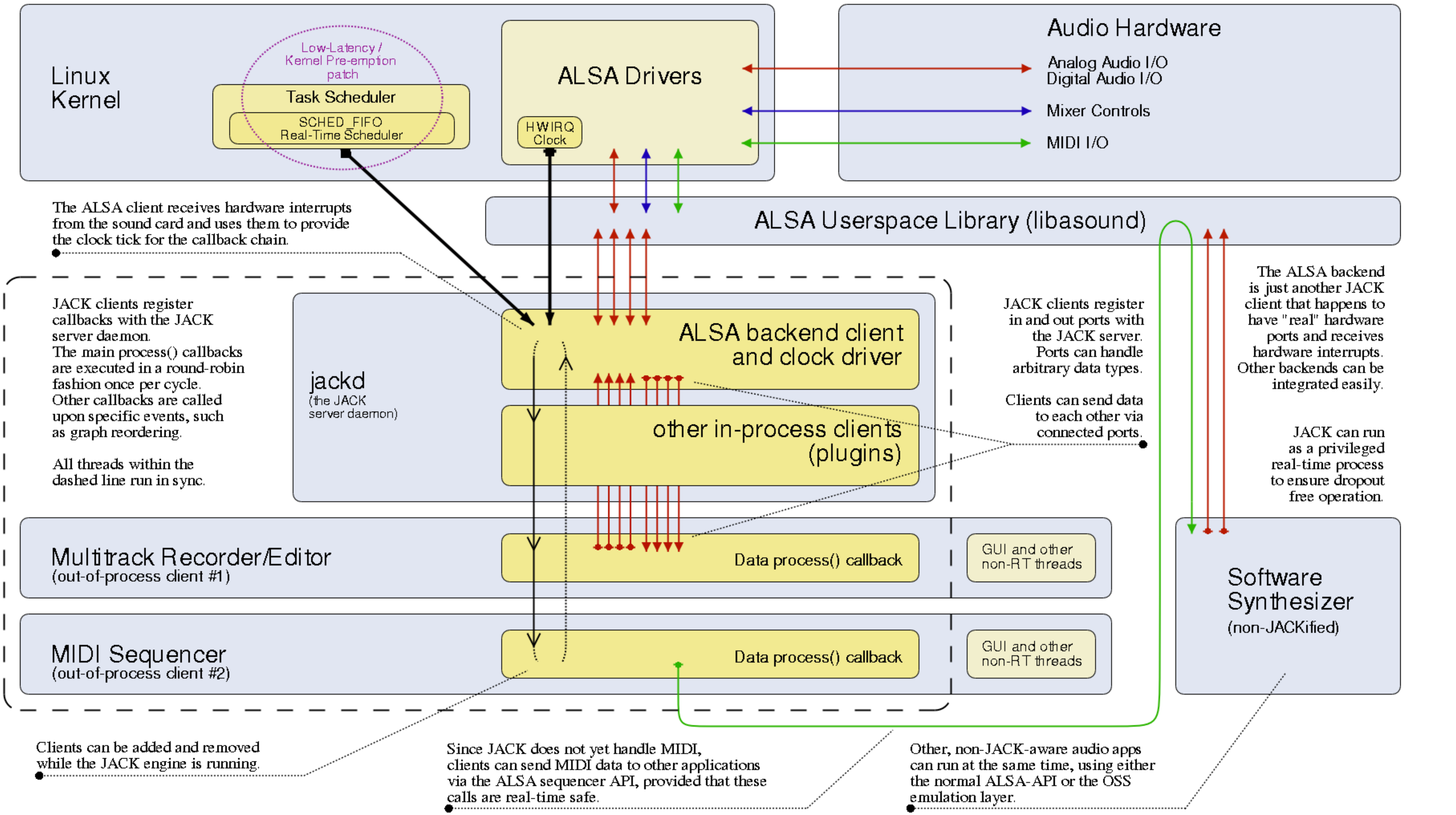
- No hay configuración de hardware
- No hay configuración de software (driver)
- No hay negociación de formatos (todos los canales son mono, 32 bit float)
- No hay un loop principal (JACK lo maneja)

Por otro lado:

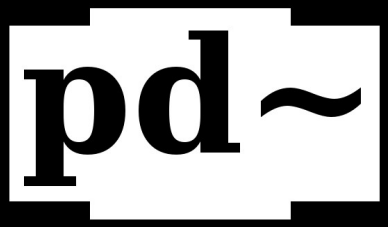
- Requiere capacidades de programación multi-threaded (lock-free)
- Necesita un sistema cliente/servidor
- El modelo "pull" es más complicado en ciertas aplicaciones

The JACK Audio Connection Kit

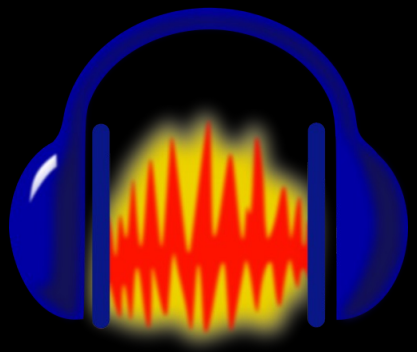
Hooking up audio applications in real-time and sample-synchronized



Algunas aplicaciones que utilizan JACK



Rosegarden



Más info:

- http://jackaudio.org/
- http://es.wikipedia.org/wiki/JACK_Audio_Connection_Kit
- http://www.musix.org.ar/wiki/index.php?title=JACK-Es
- http://lac.linuxaudio.org/2003/zkm/slides/paul_davis-jack/title.html
- http://www.tomshardware.com/reviews/audio-production-software-linux-ubuntu,2860.html
- http://createdigitalmusic.com/2009/08/linux-music-workflow-switching-from-mac-os-x-to-ubuntu-with-kim-cascone/

Contacto

Matías Morawicki

Ing. en Electrónica

Sysadmin GNU/Linux

Tecnología Multimedia

matias@tecnosoul.com.ar

www.tecnosoul.com.ar

Puerto Rico, Misiones