L'AUDIO (e il multimedia) IN LINUX

La guida non è ancora del tutto completa. La porterò avanti quando avrò tempo. Spero possa esservi utile in qualche modo, o che comunque possa essere per voi da spinta all'uso di Linux per i vostri progetti audio, o comunque da volano per effettuare voi stessi nuove ricerche. Un sincero saluto da Marco (Etnablog).

Ultimo aggiornamento 21 Maggio 2010

* * *

Riguardo l'audio su Linux in questo periodo c'è gran parlare: sono in molti a sostenere che vi sia troppa confusione nella gestione di audio e contenuti multimediali su questo sistema operativo.

Questo purtroppo confonde gli utenti, e soprattutto gli sviluppatori di software, che si trovano nel dubbio su quali librerie API sia meglio usare per gestire l'audio delle loro opere informatiche.

Premessa: è un argomento piuttosto complicato, ho tentato di sintetizzare ma i concetti sono veramente tanti e ritengo sia necessaria più di una sola lettura, lenta e attenta, per capire. Vi avviso inoltre che mi terrò largo, e con la scusante di parlare di audio, parlerò anche di multimedia e di concetti generali di Linux.

Mi rendo conto che scrivere di Linux, in un momento come questo, in cui il pinguino sta vivendo un periodo di cambiamenti così rapidi, radicali, è pressappoco un suicidio; probabilmente già domattina quanto avrò scritto saprà di vecchio, ma trovo stimolante scrivere, porta a chiarirmi le idee e appoggio l'idea della condivisione delle conoscenze, fondamento del software libero.

Come viene gestito l'audio in linux? A grandi linee, fra i prodotti che si spartiscono la torta della gestione dell'audio (e del multimedia in generale..) in linux oggi troviamo ALSA, OSS, ESD, aRts, JACK, LADSPA, GStreamer, NMM, XINE, Helix, Phonon, SDL, OpenAL, FMOD, PulseAudio.. In realtà ce ne sarebbero altri..

Non tutti sanno cosa sono, a che servono, e quali sono i loro limiti e pregi. PulseAudio si sta molto diffondendo ultimamente e sempre più distribuzioni linux lo scelgono come loro server audio "di serie". Ma sono in molti a domandarsi: "ma perchè PulseAudio?", "ma ne avevamo veramente bisogno?", senza contare che in molti si domandano persino cosa sia un server audio..

Cercherò di spiegarvi con la massima chiarezza che mi riesce, portandovi con me in un ideale cammino, che comincia dal momento in cui accendete il pc e la scheda audio viene rilevata, fino ad arrivare all'emissione del suono.

La logica che sta dietro questo processo io ritengo sia affascinante.

- Il riconoscimento dell'hardware, il Kernel
- La gestione dei devices audio
- I frameworks
- I soundservers
- La latenza dell'audio
- Linux e gestione di player multimediali

Il riconoscimento dell'hardware, il kernel

Il kernel è il cuore del sistema operativo GNU Linux, il "programma più importante". Quando il sistema viene acceso ("fase di boot"), questo viene caricato nella memoria RAM del vostro PC. E' il Kernel che interagisce con l'hardware del vostro PC, mediante i drivers, che sono forniti sotto forma di appositi "moduli del kernel", detti LKM ("Loadable Kernel Modules") o semplicemente KMOD ("Kernel Modules").

Stampante, tastiera, mouse, hard-disk, tutto ciò che collegate al vostro pc sono i "devices" che, per funzionare, necessitano che l'apposito "device driver" del kernel venga caricato.
I LKM si trovano nella cartella /lib/modules e sono files con estensione ".ko" (a partire dal kernel 2.6). Il comando "uname -r" mostra la versione del kernel attualmente in vostro uso.

I moduli possono anche essere non a codice aperto ("proprietari"), e come tali, essendo non noto il loro codice, possono potenzialmente minare la sicurezza del vostro sistema.

Grazie al comando lsmod potete vedere quali moduli sono stati caricati dal kernel ed il loro stato mentre invece quelli che in generale sono disponibili (ovvero proprio i files con estensione .ko ("Kernel object"), possono essere visualizzati con modprobe -l o semplicemente navigando nella cartella /lib/modules/`uname -r`

Per quello che riguarda l'audio, è "ALSA (Advanced Linux Sound Architecture) il componente del Kernel linux che fornisce oggi il device driver per le schede audio rimpiazzando il "vecchio" OSS. Come mai questo cambiamento? Cosa offriva ALSA che OSS non aveva? Andiamo un pò indietro nel tempo..

La storia dell'audio in linux cominciò con i VoxWare drivers, realizzati da Hannu Savolainen. Quando lo stesso autore fu assunto dalla 4Front, sulla base dei VoxWare, nel 1992, realizzò OSS (Open Sound System).

Di OSS vi era una versione Free (che venne aggiunta al kernel 2.24.x) ed una commerciale (a pagamento..), la cui differenza consisteva nelle utility di configurazione e riconoscimento dell'hardware.

Dopo alcuni anni, la mancanza in OSS di alcune importanti funzioni, tipo il mixing di piu fonti (ma non solo..), convinse Jaroslav Kysela (che era stufo dello scarso supporto di OSS per la sua scheda audio Gravis Ultrasound) a lavorare ad un nuovo progetto chiamato ALSA, che, dopo un periodo di testing, venne introdotto nel kernel 2.6 al posto di OSS per la fornitura del "device driver" per la schede audio.

ALSA era come concezione completamente diverso da OSS.. Ricordo che in OSS era possibile ascoltare qualcosa semplicemente facendo su shell "cat example.wav > /dev/dsp".
In OSS l'audio era insomma concepito come un file, addirittura vi si accedeva con chiamate tipo open(),read(),write(), come secondo quel vecchio detto che diceva che "in linux tutto è un file".. Per essere più chiari, l'accesso alle periferiche in generale ("devices") avveniva come se si stesse accedendo ad un file contenuto in una directory /dev.

Con ALSA questo non è più possibile e i "devices" sono oggetti di più alto livello. Si accede ai PCM devices mediante interfacce tipo "hw:0" o "plughw:0" (se ne riparlerà dopo).

ALSA, rispetto ad OSS, era capace di miscelare insieme più flussi audio contemporaneamente e di gestire più schede audio contemporaneamente.
L'utilizzo di OSS fu così dichiarato "deprecato" in linux, ma OSS non è morto, ed è attualmente usato da versioni di Unix, BSD, Solaris e in realtà continua a migliorare.. Non è piu quello di una volta. C'è gente che, tutt'oggi, sceglie di installare su Linux stesso OSS4 rispetto ad ALSA, ritenendo che abbia un supporto migliore per l'hardware, un sound mixer di (addirittura..) maggiore qualità (OSSv3 non poteva riprodurre più suoni in una volta, OSSv4 invece si), inoltre non è linux specifico come ALSA. Le opinioni sono contrastanti

al riguardo, ma l'ipotesi di disinstallare ALSA a favore di OSS4 rappresenta comunque una eventualità da prendere in considerazione, specie in presenza di seri problemi di supporto dell'hardware del vostro pc da parte di ALSA..

In caso di problemi con il vostro audio, prima di mollare del tutto ALSA, tenete in considerazione che:
- Diversi problemi di ALSA possono essere risolti mediante l'installazione di qualche apposita "ALSA plugin" e/o la personalizzazione ad hoc del suo relativo file di configurazione. Questo è soprattutto valido per vecchie schede audio.
- Altri problemi possono essere causati dalla presenza di un eventuale soundserver (come Pulseaudio) che si trova interposto tra voi ed ALSA e dalla buona configurazione di questo. I Soundservers sono sempre più comuni oggi nelle distribuzioni linux ed in generale si occupano di gestire il soundmixing al posto di ALSA, facendogli arrivare il segnale già bello e mixato pronto da riprodurre.. E' pertanto FONDAMENTALE che voi sappiate gestire il vostro soundserver (ne riparleremo dopo).

Scendiamo più nel dettaglio, cominciando più precisamente ad addentrarci in ALSA.

ALSA (Advanced Linux Sound Architecture) è il progetto che fornisce oggi al sistema linux la "comunicazione" con la scheda audio per l'audio e il MIDI. E' costituito da:
- Gli indispensabili driver integrati nel kernel (si parla di "ALSA sound drivers" infatti, sono moduli del kernel che di solito cominciano con prefissi tipo "snd-" o "snd_").
- Una serie di librerie API (le alsa-lib) che forniscono ai programmatori i "comandi" per riuscire ad usare gli ALSA drivers nel kernel.

Preparatevi a fare un piccolo tuffo nei meandri del kernel linux: diciamo che ogni scheda audio (o meglio, il suo "chipset") ha il suo modulo ALSA specifico nel kernel di Linux (ad esempio quello per la scheda sound blaster 16 bit è il "snd-sb16", tanto per dirne uno).
Volete sapere se i moduli relativi alla scheda audio sono stati caricati regolarmente? Fate lsmod | grep -i snd
Nel pc che sto usando adesso, l'output di quel comando è piuttosto lungo (tanti moduli..), cito solo le prime righe:

```
snd_emu10k1_synth 5156 0
snd_emux_synth 31695 1 snd_emu10k1_synth
snd_seq_virmidi 4213 1 snd_emux_synth
snd_usb_audio 75765 2
snd_seq_midi_emul 5492 1 snd_emux_synth
...[continua]
```

Cerchia mo di capirci qualcosa: I moduli del kernel sono salvati in hard disk come files (di solito cartella /lib/modules) e quando sono aggiunti nel kernel vengono copiati in memoria come parte del kernel (potete fare questo usando i comandi insmod e modprobe).

lsmod è un comando che mostra quali moduli sono caricati nel kernel (come dire che mostra in una maniera "carina" i files contenuti in /proc/modules). L'output di lsmod è una lista di moduli del kernel organizzata in 3 colonne:
- La prima ("module") è il nome del modulo.
- La seconda ("size") è la quantità di memoria che usa (ricordatevi che la cartella /proc in realtà è solo virtuale).
- La terza ("Used by") è il numero di quanti altri moduli del kernel stanno usando quel modulo e i nomi di questi, separati, se piu di uno, da virgole.

Per ciascuno di quei moduli che vedete in lista, potete ottenere ulteriori informazioni col comando modinfo seguito dal nome del modulo.

Tenete sempre a mente che al giorno d'oggi le schede audio possono essere oltre che per porta pci, anche per porta USB. La cosa non è da trascurare.. Rack e pedaliere di effetti per chitarra elettrica o basso, mixer digitali multi-ingresso, hanno sempre piu spesso la possibilità di essere usate per porta USB, e vengono a tutti gli effetti visti dal vostro pc come "schede audio USB"..

La vostra scheda audio è per porta pci? Provate a cercarla nella lista delle periferiche pci con: lspci | grep -i audio
Da me questo comando mostra:

```
$ lspci | grep -i audio
05:06.0 Multimedia audio controller: Creative Labs SB Live! EMU10k1 (rev 07)
```

Maggiori infomazioni si ottengono facendo anche: lspci -v o meglio ancora lspci -vv
Fra le varie informazioni che potete ottenere in più, vi è la possibilità di osservare la "latenza" che i devices pci hanno (valore tra 0 e 248). Se il sonoro che esce dalle vostre casse è ricco di "click" e "pop", potete provare ad impostare il "latency setting" della vostra scheda video verso un valore piu basso (se è scheda AGP, attenzionate il PCI-AGP bridge, se la scheda video è PCI-E allora questo discorso non vale..), a vantaggio di quello audio, che alzerete un pò (gli darete così una "priorità più alta").. Badate bene che si sta parlando di "PCI latency", che non ha nulla a che vedere con la "latenza" dell'audio, che è cosa del tutto diversa:
- La PCI latency è il tempo di "uso esclusivo" del bus PCI di cui un device può disporre prima di consegnarlo ad un altro device.
- La latenza dell'audio è il tempo di ritardo che impiega l'audio dall'input all'output e dovete modificare il "buffer size" dell'audio driver se volete ottenere un vantaggio in tal senso. Il buffer size va "bilanciato" tra un valore più alto (che aumenta il "lag", la latenza), e un valore più basso, che è più snello e reattivo. Se il suono risulta "metallico" e pieno di "crack" forse il buffer è troppo basso, e va impostato ad un valore più alto.
Faccio un esempio. Dal mio lspci ottengo che la mia scheda audio funziona al bus 05, device 06, funzione 0 (05:06.0) ed ha latenza 32:
[...]

```
05:06.0 Multimedia audio controller: Creative Labs SB Live! EMU10k1 (rev 07)
Subsystem: Creative Labs Device 8040
Control: I/O+ Mem- BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR+ FastB2B- DisINTx-
Status: Cap+ 66MHz- UDF- FastB2B+ ParErr- DEVSEL=medium >TAbort- SERR-
Latency: 32 (500ns min, 5000ns max)
Interrupt: pin A routed to IRQ 21
Region 0: I/O ports at e480 [size=32]
Capabilities:
Kernel driver in use: EMU10K1_Audigy
Kernel modules: snd-emu10k1
```

[...]

Voglio aumentare la sua PCI latency, quindi faccio:

```
$ sudo setpci -s 05:06.0 LATENCY_TIMER=40
```

40 è un numero esadecimale (in realtà è 64, quindi facendo così porterò la Latency a 64). Se volessi impostare il massimo, anzichè 40 dovrei scrivere "ff", che specifica 256 (ma viene arrotondato giu a 248, che è il massimo).
Generalmente 32 (esadecimale "20") è il valore di default un pò per tutte le periferiche e va bene.. Ma molti suggeriscono di impostare un valore di 128 (esadecimale "80") per la scheda video, al fine di ottenere migliori

prestazioni "video".
Penso che questo possa bastare riguardo il PCI, semplicemente tenete presente il discorso "latenza PCI", qualora vi venisse in mente di "overcloccare" il bus e doveste cominciare a sentire suoni sgradevoli..

Passiamo all'USB. La vostra scheda audio è su porta USB e volete vedere se viene correttamente rilevata? (tipo un mixer digitale USB) Fate: lsusb
Nel mio pc, lsusb mostra:

Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 002: ID 08bb:2904 Texas Instruments Japan PCM2904 Audio Codec
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 002: ID 0c45:607c Microdia CCD PC Camera (PC390A)
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

In pratica rileva la mia pedaliera d'effetti per basso, collegata con porta usb (PCM 2904 è il chip interno che monta, l'interfaccia USB).

Le ALSA library dispongono di diversi file di configurazione. Fra questi ricordiamo:

* /usr/share/alsa/alsa.conf
* /etc/asound.conf
* ~/.asoundrc
* /usr/share/alsa/{cards,pcm}/*.conf

Non è detto che tali file siano tutti presenti nel linux che voi usate. Nella mia Ubuntu ad esempio etc/asound.conf e persino ~/.asoundrc mancano del tutto.. E' che sarebbero i file di configurazione delle "ALSA plugins".

Il principale, quello di default, è alsa.conf. Si tratta di un file installato da libasound2 (che fornisce funzioni ad ALSA). Il fatto che si trovi in /usr/share e non in /etc, dove spesso si trovano i file di configurazione in linux, deve farvi pensare sul fatto che non dovreste modificare questo file, che tuttavia già da se, di solito è adeguato per la maggior parte delle installazioni. Sia alsa-utils che alsa-plugins "cercano" in questo file di configurazione (consideratelo un "file di informazioni"), che è quindi definibile come "condiviso" ("to share") fra piu pacchetti.

Una volta c'era il programma alsaconf per fare modifiche ad alsa.conf. Chi lo usava, rischiava a volte di combinare un casino, che poteva spesso risolversi eliminando la cartella /usr/share/alsa/, reinstallando le libasound2 ed eventualmente rieseguendo alsaconf. In seguito bisognava riavviare il sonoro con "sudo /etc/init.d/alsa-utils restart". Oggi alsaconf è stato rimosso dalle alsa-utils. Rendeva la vita piu facile a chi aveva schede audio isa o comunque non plug & play, ma oggi il riconoscimento dell'hardware funziona diversamente che in passato..

..Tuttavia, qualora doveste avere problemi nel riconoscimento hardware della vostra scheda audio, tenete in considerazione l'ipotesi di reinstallare il buon vecchio alsaconf (ipotesi estrema?..).

In generale, le possibilità di ALSA possono essere estese dalle "ALSA plugins", che permettono di disporre di funzionalità come il resampling, channel routing, sample format conversion e software controllo volume e altre. Ma in realtà se la vostra scheda audio sembra funzionare bene nel vostro sistema linux, difficilmente ve ne potrà mai importare qualcosa delle ALSA plugin..

Le ALSA plugins, per esercitare la loro funzione su ALSA, hanno bisogno dei loro files di configurazione tipo

/etc/asound.conf e ~/.asoundrc. Modificando tali files di configurazione si possono dare ad ALSA direttive personalizzate a vostro piacimento, o addirittura risolvere talora dei veri e propri problemi di compatibilità hardware specifici per determinate schede audio. In questo senso, le direttive da impartire sono precise, non inventate da voi, ma vengono reperite nella documentazione di ALSA, forum o wiky, e vanno prese e copiate "così come sono".

Se date un'occhiata all'interno di alsa.conf, che in fin dei conti non è altro che un file di testo, potreste notare che è lui stesso il fulcro, che richiama gli altri files di configurazione, tra cui /etc/asound.conf e ~/.asoundrc (..ed altri, date un'occhiata in alsa.conf per scoprire quali sono).

Le sintassi usate in questi files di configurazione sono complesse e non trattabili qui, vi basterà, giusto per comprendere un minimo, che "pcm" fa riferimento a "playback", "ctl" a "control". Non credo vi serva sapere altro al riguardo, se non che grazie a questo il server audio Pulseaudio può funzionare.

Come detto prima, per vedere che modulo del kernel riguardante l'audio è caricato, fate lsmod | grep -i snd

lsmod è un comando shell che mostra in modo formattato la lista dei moduli del kernel attualmente caricati, la stessa che in fin dei conti è ottenibile eseguendo cat /proc/modules

Fate "cat /proc/asound/cards" per verificare le schede audio installate. Possono anche essere più di una, vengono numerate (0, 1, 2... ecc..). Nel mio portatile, il contenuto di /proc/asound/cards è:

0 [Intel]: HDA-Intel - HDA Intel
HDA Intel at 0xf8400000 irq 22

Il che indicherebbe che ho una sola scheda audio, che è una "Intel" e lui la chiama "0". Ma una scheda audio è una realtà complessa.. Ci sono prese in ingresso (viste come devices in ingresso o "cattura"), prese in uscita (casse? Uscita Line-out?), e poi il midi e la sua sincronizzazione.. Alsa fa distinzione di tutti questi elementi vedendoli come devices diversi.

Grazie al comando "cat /proc/asound/devices" potete vedere, per ogni scheda audio, che devices vengono riconosciuti:
- acquisizione, viste come "capture",
- riproduzione viste come "playback",
- "control" per regolarne il volume, il "mixer" o settarne il muto. Vi sono schede audio, come la RME 9652 (una scheda audio professionale), che non hanno questo "mixer"
- "timer", che serve per il "timing", la sincronizzazione
- "sequencer", che serve per il midi (la "sincronizzazione" può essere importante per il corretto funzionamento del midi..)

Nel pc che sto usando adesso, il comando "cat /proc/asound/devices" mi restituisce:

2: : timer
3: : sequencer
4: [0- 6]: digital audio playback
5: [0- 6]: digital audio capture
6: [0- 1]: digital audio playback
7: [0- 0]: digital audio playback
8: [0- 0]: digital audio capture
9: [0- 1]: hardware dependent
10: [0- 0]: hardware dependent
11: [0] : control

I numeri dentro alle parentesi quadre ("[x- y]") sono indici relativi alla scheda audio (x) e del device (y) in quella scheda audio. Notate lo "0", è il numero cui lui associa l'unica scheda audio che ho nel portatile. La 0,0 e la 0,6 si capisce che sono duplex (riproduzione e cattura).

La mia scheda audio è stata "scomposta" da ALSA in multipli devices nelle sue componenti di playback, capture, control, timer, sequencer.
Nel mio caso, guardando l'output di cui sopra, si evince che ho 1 canale di "controllo" (per controllo volume, mic gain e altri settaggi), 3 PCM playback devices, 2 PCM capture devices, un MIDI sequencer, un timer, le "hardware dependent". Cosa sono queste "hardware dependent"? Potreste non avere voi, in pratica il modulo kernel snd-hda-intel (il "driver" della mia scheda audio) è specifico per il "chip" audio. Ma ogni produttore che costruisce una scheda audio mette magari nel suo prodotto, oltre al chip, qualcosa di diverso.. Considerate queste hardware dependent un "workaround", ovvero uno stratagemma, per risolvere i bug derivanti da queste differenze (abilitare l'amplificatore per fare funzionare correttamente lo speaker del mio pc portatile, settare correttamente il volume delle cuffie, disabilitare lo speaker quando si inseriscono le cuffie ecc.).

Per testare la riproduzione della mia scheda audio (serve un file PCM con estensione .wav) potrò eseguire il comando:

aplay -D plughw:0,0 file.wav

Tenete presente che /proc/asound/ e il suo contenuto esiste se il modulo nel kernel "snd" funziona correttamente. Può darsi che vi stiate domandando cosa sia questa cartella "proc".. beh attenzione perchè in realtà, sebbene voi la vediate come una cartella presente, come qualsiasi altra, nel vostro hard disk, in realtà è un vero e proprio filesystem montato su quella cartella, che di fatto nel vostro HD non c'è. Diciamo che, mediante opportuni comandi presenti in questa cartella "virtuale", riusciamo ad ottenere delle informazioni direttamente dal kernel. "/proc" e anche la cartella "/sys" hanno dimensione zero e di questo potete rendervi conto semplicemente eseguendo, da shell, il comando "ls -l".

A grandi linee come gestisce l'audio il nostro pc? Considerate l'impulso audio nella sua forma più generica (la tipica "onda"), questa nel nostro pc viaggia, per essere riprodotta ("playback"), tradotta in impulsi codificati in un formato detto "raw pcm" (pulse code modulation).

Il MIDI (Musical Instrument Digital Interface) invece è un discorso a parte e nulla a che vedere con il PCM. In sostanza la musica MIDI è prodotta da un "sequencer" (di solito un editor software, ad esempio il famoso programma "Cubase") sottoforma di "codici" standard (lo standard midi, per l'appunto) che definiscono segnali tipo "controllo voltaggio" e "on/off", che vengono letti, interpretati e riprodotti da un synthesizer (ovvero uno strumento capace di produrre suoni elettronicamente, che può essere hardware ma anche software).

Risultando il brano scomposto in codici, in ogni sua parte, i file MIDI sono leggerissimi ed estremamente maneggiabili (sono usatissimi nei karaoke, o dai musicisti per motivi di studio, in quanto gli strumenti possono essere esclusi uno ad uno, la tonalità del brano può essere modificata senza variarne il tempo o viceversa, o si può perfino ricavare immediatamente, mediante appositi programmi, delle tablature complete). Per conseguenza, uno stesso file midi, riprodotto da schede audio diverse o persino da software diversi (variando il soundfont) può sentirsi diversamente, in meglio o in peggio. Programma molto usato in Linux, per riprodurre file MIDI, è Timidity, che è un synthesizer software. Possiamo anche impostare in Timidity soundfonts .SF2 diversi (esempi Unison.SF2, di circa 28 megabytes, o PC51f.sf2, di circa 60 M), per ottenere sonorità migliori.

OSS accedeva al midi attraverso /dev/midi*
ALSA, come detto sopra, ha un accesso all'hardware audio completamente diverso. La lista delle periferiche MIDI è visibile con " aplaymidi -l ".
"aconnect" è l'utility che permette la connessione con le porte dell'ALSA sequencer.

aconnect -i serve a vedere le porte in ingresso, aconnect -o quelle in uscita ("output"), aconnect -l mostra lo stato della connessione.

L'ALSA sound driver è il nostro obiettivo.. E' con lui che, in un modo o nell'altro, dobbiamo cercare di comunicare dato che sarà grazie a lui che comunicheremo con la scheda audio del PC (che in questo modo riprodurrà il suono..).

Ma in che modo comunicare con L'ALSA driver (e quindi accedere alla scheda audio)? Vi sono diversi sistemi ("livelli di astrazione"):
- Usando le API in alsa-lib (talora dette libasound), che sono le librerie create dagli stessi sviluppatori di ALSA
- Usando OSS (OSS era usato prima di ALSA in linux, ma vi sono molti programmi che ancora lo usano)
- Usando un soundserver (..e adottandone le sue API), tipo PulseAudio, EsD, Jack ecc..

Sebbene OSS sia stato dichiarato "deprecato", vi sono ancora molti programmi/giochi che ancora lo usano. Audacity, Slab, MpegTV, XMMS, MPlayer, Xine, KMix, Skype, Doom, Quake sono solo alcune applicazioni che usano o possono usare ALSA (alcune applicazioni oggi danno addirittura la possibilità di scegliere, per il loro stesso funzionamento, tra OSS, ALSA o qualche Soundserver). Ebbene per non perdere la compatibilità con le applicazioni che non usano ALSA ma usano OSS, ALSA può usare due stratagemmi:

- Uno funziona a livello del kernel, installando degli appositi moduli (snd-pcm-oss, snd-mixer-oss, snd-seq-oss, rispettivamente per emulare le devices PCM, mixer, sequencer).

- l'altro è lo script aoss (si tratta di un "OSS wrapper") contenuto nel pacchetto alsa-oss (che contiene anche la libreria libaoss, libreria di compatibilità ALSA OSS). Se ad esempio state usando il lettore multimediale xmms (impostato per funzionare con output OSS mediante apposito plugin) basta fare da shell "aoss xmms" per farlo funzionare con ALSA.

Il primo sistema è semplice da implementare, ma non fa al caso nostro se ci interessa "mixare" l'audio, il secondo ha pure una sua piccola magagna, cioè che dà qualche problema se il programma OSS accede all'audio con fopen(), ma sembra che siano poche le applicazioni OSS a farne uso.

Nonostante quanto detto, c'è ancora chi preferisce programmare usando OSS, per varie ragioni, tra cui la maggiore facilità d'uso delle API e/o la loro migliore documentazione. C'è anche da dire che ALSA c'è solo in linux, ed è in questo senso più "ristretto", mentre OSS c'è anche in altri sistemi *nix, e chi lo usa si appoggia al fatto che tanto OSS su ALSA, in fin dei conti, funziona lo stesso.. ..In un modo o nell'altro.

Ma che strumenti, in generale, ci fornisce ALSA stessa per gestire i suoi ALSA drivers nel kernel? Sono diversi, e contenuti in diversi pacchetti (ALSA packages) fra i quali ricordiamo:

* Alsa-lib: detto anche "libasound". Fornisce librerie che permettono la comunicazione ("livello di astrazione") con l'ALSA driver che fa parte del kernel, le cui interfacce si trovano nella directory /dev. Diciamo che i programmi potrebbero comunicare direttamente con i driver nel kernel, ma le alsa-lib permettono di farlo in modo molto più semplice.
* Alsa-oss: contiene il wrapper aoss.
* Alsa-tools: contiene in se altri piccoli programmi, alcuni specifici per determinate schede audio.
* Alsa-utils: contiene un insieme di utilities varie, tipo aplay (un semplice player), alsamixer (un mixer, con un interfaccia "tipo testo"), amixer (è un mixer per controllare ALSA da "riga di comando"), aplaymidi (per riprodurre files midi) ed altri.
* Alsa-plugins: è un insieme di "plugins". Per esempio alsa-plugins-pulseaudio fa da anello di collegamento tra ALSA e PulseAudio, Alsa-plugins-jack è anello di collegamento tra ALSA e JACK.

La gestione dei devices audio su Linux

Come sappiamo "i devices", in generale, in linux, per essere usati, vengono "montati" in delle cartelle.

In sostanza, la cartella /dev è usata in linux per conservare, come files, i "device nodes" che fanno riferimento a certi devices del sistema (devices che ci sono, ma anche devices che nel nostro sistema non ci sono..). In pratica programmi, giochi, per interfacciarsi ai devices sfruttano questi nodes.

ALSA monta i devices audio nella cartella /dev/snd/ (invece OSS le monta in /dev/dsp) ed è proprio questa la cartella che le applicazioni usano per eseguire funzioni audio (registrare, riprodurre, modificare i volumi). Diamo uno sguardo dentro la cartella /dev/snd/, potreste vedere diversi devices, riconoscibili come diversi file "pcm", "control", "timer", "seq", dipende dal vostro hardware. Per esempio il /dev/snd/pcmC0D6c che io vedo tra i miei devices, indica la scheda audio 0, device 6, "c" sta per "capture", come ad esempio un "microfono" ("p" invece starebbe per "playback", quindi si parlerebbe di un device di "riproduzione").

Come fa ALSA a cacciare dentro quella cartella /dev/snd/ i propri devices?

I Kernels dal 2.6 controllano un filesystem chiamato sysfs, montato sulla cartella "/sys", che contiene informazioni sui devices attualmente collegati al sistema. In generale per vedere i file system montati nel vostro sistema, digitate su shell "mount" (il che è equivalente a scrivere "cat /etc/mtab").

udev, il "device manager" del kernel 2.6 (rimpiazza il vecchio devfs), sfrutta proprio le informazioni contenute nel sysfs per creare una directory /dev "dinamica".
udev sta in esecuzione in sottofondo ("demone udevd") e usa queste informazioni date direttamente dal kernel (gli "uevents") per creare in tempo reale i "device nodes" corrispondenti all'hardware attualmente in vostro uso proprio nella cartella "/dev" .
Gli uevents inviati dal kernel sono gestiti sulla base di specifiche "regole", che possono così portare dal caricamento di specifici moduli del kernel, la creazione dei device nodes, l'assegnazione costante dei permessi per un device node ad un owner o a un gruppo, fino all'esecuzione di un programma che gestisca il device.
Tali regole si trovano in /etc/udev/rules.d/ sotto forma di files con estensione ".rules". Potete impostare voi stessi, ad esempio, una regola che assegni ad una determinata scheda sempre lo stesso valore hwplug:x,y .
Considerate inoltre che udev si crea persino un proprio database dei device collegati nel sistema.
Se date un'occhiata nella cartella /sys, vi rendete conto di come sia ben strutturata, bene suddividendo i devices in cartelle e persino sottocartelle e files, che definiscono così delle "gerarchie ad albero". Strumento di gestione di udev è il comando udevadm (una volta "udevinfo"). Il comando "udevadm monitor" mette il vostro pc in attesa, provate in questa fase ad inserire una periferica usb (come una memorietta usb, un mouse o altro) e vedete se viene riconosciuta.

E una volta creato il device node che succede?.. Beh Il riconoscimento dell'hardware in linux non si ferma qui.. Quando ad esempio inserite una memorietta usb nel vostro pc, il kernel la intercetta mediante udev (che ne crea il "node" in /dev), e poi comunica la cosa, mediante D-Bus, a Devicekit (una volta invece la comunicava ad "HAL").

D-bus (e le sue API) è un protocollo IPC (interprocess communication) che permette il passaggio di informazioni tra applicazioni (mediante le librerie libdbus). Tale passaggio di informazioni può avvenire direttamente tra le applicazioni ("peer to peer") o anche attraverso il "D-bus daemon". Fra le applicazioni che ne fanno uso ricordiamo gnome-mount (contiene programmi per il montaggio, smontaggio e l'espulsione di dispositivi di archiviazione. La sua configurazione è modificabile da gconf-editor).

D-bus rimpiazza:
- Bonobo (che era basato sull'architettura di CORBA, uno standard sviluppato da OMG) su Gnome

- DCOP usato in KDE 3 (in sostanza Bonobo, CORBA, DCOP servivano/servono a consentire a software diversi di interoperare e condividere informazioni fra loro anche se si tratta di applicazioni sviluppate con linguaggi di programmazione diversi).

D-bus è fra i progetti di freedesktop.org (un progetto che lavore per risolvere i problemi di incompatibilità fra diverse piattaforme) quindi diciamo che è "cross-desktop" che, tradotto in soldoni, significa che in un certo senso e da questo punto di vista D-bus ha messo daccordo KDE e GNOME, e il che non è un male..

D-bus oggi svolge un ruolo chiave persino nella gestione del filesystem con gvfs.
L'accesso i/o ai files è oggi gestito da GNOME mediante gvfs, che rimpiazza il vecchio gnome-vfs.
GVFS opera mettendo in esecuzione un singolo demone principale (gvfsd) che tiene traccia dei mount GVFS attuali. Ogni mount è eseguito in un demone separato (anche se alcuni mount condividono un processo demone, la maggior parte non lo fa).
Le applicazioni controllano gvfs mediante la libreria "gio" (libgio, parte di glib) che contiene le API (in pratica il "livello di astrazione").
L'interazione coi demoni vari avviene per lo piu mediante D-Bus.
GIO fornisce funzioni per il monitoraggio dei file, per l'IO asincrono e per il completamento dei nomi file.

HAL (Hardware Abstraction Layer), ha rappresentato una tappa evolutiva importantissima: col suo demone hald, in sostanza manteneva un database dei devices connessi al pc in tempo reale (per vederlo bastava installate il pacchetto gnome-device-manager, che aggiungeva l'apposito strumento "device manager" nel menu applicazioni di Gnome. Ll'elenco dei devices si modificava in tempo reale inserendo e disinserendo dispositivi nelle porte USB). HAL forniva inoltre delle API che le applicazioni potevano usare per rilevare i devices, monitorarli, eseguire operazioni su di loro se PolicyKit (sistema di autenticazione) lo consentiva. I processi HAL in esecuzione potevano essere visti col comando ps -eaf | grep hal
Vi domanderete perchè parlare di come vengono gestite le periferiche usb in una guida che parla di audio..
Beh il fatto è che al giorno d'oggi, sempre più spesso, si fa ricorso ad accessori audio o persino vere e proprie schede, che fanno uso di porta usb.

Il mio cat /proc/asound/devices del pc da cui sto scrivendo adesso è:

2: : timer
3: : sequencer
4: [0- 6]: digital audio playback
5: [0- 6]: digital audio capture
6: [0- 0]: digital audio playback
7: [0- 0]: digital audio capture
8: [0] : control

Subito dopo avere inserito il mio Mixer 8 uscite per porta USB, diventa:

2: : timer
3: : sequencer
4: [0- 6]: digital audio playback
5: [0- 6]: digital audio capture
6: [0- 0]: digital audio playback
7: [0- 0]: digital audio capture
8: [0] : control
9: [1- 0]: digital audio playback
10: [1- 0]: digital audio capture
11: [1] : control

Inoltre dopo l'inserzione del mixer nuovi devices compaiono nella cartella /dev/snd, a testimonianza del fatto che il mio mixer viene immediatamente visto come scheda audio.

HAL, progetto molto importante, col tempo si è parecchio ingrandito perchè è necessario che le sue capacità di "leggere" l'hardware siano elevate ed accurate (si tratta di un computer che può essere sospeso? puo ibernare? ACPI?), e che le sue API siano efficienti. Alla fine si è reso necessario dividerlo in un insieme di progetti detto Devkit, che sta pian piano prendendo il posto del sempre più grosso monolitico HAL.

DeviceKit numera i devices, li "identifica" e assegna in classificazioni tipo "audio players, cameras" e altro, emette segnali se queste sono aggiunte o rimosse dal pc.

Beh in realtà DeviceKit stesso, neanche il tempo di entrare in scena, ha subito un mare di cambiamenti al punto tale che è difficile ricostruirne la storia.. Esisteva un demone DeviceKit, presto sostituito dalle librerie libudev e libgudev per reperire le informazioni di udev sui devices e sugli eventi (scavalcando D-bus?). In verità quello che sta succedendo è che si sta cercando la massima integrazione possibile tra udev e DeviceKit, prova ne è il fatto che cominciano a condividere insieme librerie fondamentali. Ne vedremo delle belle, ma forse è ancora troppo presto per parlarne, e per il momento vi è una forte confusione tra HAL e DeviceKit, che ancora parzialmente coesistono insieme.

Nel contesto di DeviceKit troviamo:
- upower (ex "DeviceKit-Power"): si occupa del power management del pc (e quindi anche della gestione della batteria di un pc portatile).
- udisks (ex "DeviceKit-disks"): serve a tenere traccia dei devices block ("storage devices", tipo hard-disks e memoriette usb). GNOME vi interagisce mediante un frontend grafico chiamato palimpsest (programma utile a gestire la creazione di filesystems, partizionamenti, encryption, raid e altro) contenuto nel pacchetto gnome-disk-utility. Nello stesso pacchetto vi è anche una estensione di nautilus chiamata nautilus-gdu, che serve a Nautilus per sfruttare proprio DeviceKit-disks.
- udev-extra: fornisce delle regole di funzionamento a udev in modo tale che lo stesso deviceKit possa funzionare. Anche HAL poteva essere "regolato" nel suo funzionamento, creando delle regole in degli appositi files XML con estensione .fdi .

I frameworks in Linux

In pratica l'ALSA sound driver accetta i flussi audio che noi gli mandiamo solo in formato non compresso raw PCM.
L'ALSA rispetto a OSS può accettare più flussi "raw PCM" contemporaneamente e miscelarli insieme.

In pratica, tutti i vari formati multimediali più o meno compressi come MP3, OGG, WAV, ACC, WMA, RMA, etc. perchè possano essere riprodotti, è necessario che vengano tradotti in raw PCM ed è questo che fanno i riproduttori multimediali che usiamo, quando ascoltiamo musica (mp3, wav, WMA ecc..) o guardiamo un video (mpg, avi, ecc..).

Sicchè uno sviluppatore di un riproduttore multimediale si perde nel mare dei migliaia di formati audio/video che la sua opera dovrebbe gestire.. A venirgli in aiuto subentrano i cosiddetti "multimedia framework", tipo Gstreamer, NMM, Xine, Helix, e anche il vecchio aRTS di KDE 3. Lo sviluppatore adotta quindi le API di Gstreamer, ad esempio, e si preoccuperà lui di decodificare il formato video dell'utente..

Gli sviluppatori di GNOME consigliano di usare il framework Gstreamer al momento.. Altra strada hanno scelto gli sviluppatori di KDE 4.4. Hanno infatti tolto il vecchio soundserver aRTS (che era anche un audio framework, tra l'altro) dichiarando che "non era la soluzione ideale", in quanto mancante di diverse importanti funzioni , e l'hanno sostituito con Phonon, che un soundserver NON è, ma è un ulteriore livello di astrazione. In sostanza Phonon ha le sue proprie API per gestire i formati multimediali, scritte nel modo più semplice possibile, ed è Phonon stesso a preoccuparsi di dialogare con i frameworks vari (Xine-lib, che sembra essere il "piu scelto", Gstreamer, libvlc, libsmplayer, persino Directshow di Windows, QuickTime di Apple o altro), essendo in grado di cambiare il framework usato addirittura al volo.. In molti sono favorevoli a questo approccio, in quanto si libera lo sviluppatore della preoccupazione di scegliere un framework, gli si fornisce delle API semplici da gestire e per di più multipiattaforma. Ma in molti altri non sono daccordo all'uso di tale sistema adducendo le motivazioni più varie. Il futuro vedrà chi ha ragione..

In linea di massima chi usa GTK+ tenderebbe ad usare GST (GStreamer) o libcanberra.
Chi usa Qt o QtLibs tenderebbe ad usare Phonon.

Per ora è francamente troppo presto per parlarne, e la lotta Phonon Vs Gstreamer Vs altri Framework, lotta che di striscio o in pieno investe anche la solita lotta fra toolkit GTK+ Vs Qt Vs Qt/KDELibs, almeno per il momento si aggiunge alle centinaia di lotte che si infiammano all'interno della comunità del pinguino, che creano un mare di divisione, lotte, confusione, ma anche libertà di scelta..

LADSPA (Linux Audio Developers Simple Plugin Architecture): per quanto riguarda LADSPA, ritengo necessario fare una piccola premessa. Di questi tempi, nello sviluppo di un qualsiasi sotware, si va sempre di più alla ricerca della "modularità", ovvero della possibilità di aggiungere, ad un programma già esistente, nuove funzionalità semplicemente mediante dei "plugin".

Questo succede in linux per moltissimi programmi. Chiunque può creare e distribuire nuovi plugin, e contribuire così attivamente allo sviluppo.

Considerate LADSPA come un sistema per realizzare plugin per manipolare audio, per tutti quei programmi che accettano i plugin realizzati usando le API di LADSPA.

I plug-in scritti usando LADSPA possono essere facilmente integrati in altri programmi.

Sono stati realizzati con LADSPA effetti audio strabilianti: delay, flanger, equalizzatori, riverberi, pitch shifter (servono a modificare la tonalità di un brano), e numerosissimi altri.

L'idea non è del tutto nuova a dire il vero, un altro sistema simile sono i plugin VST (non free, usati ad esempio, su sistemi Windows, da programmi come l'ottimo Cubase).

DSSI (Disposable Soft Synth Interface) invece è un linguaggio per scrivere plugins che viene definito "LADSPA per gli strumenti", come una sorta di "estensione di LADSPA". DSSI può addirittura funzionare con strumenti VSTi mediante il wrapper plugin dsst-vst. DSSI è supportato da sofware come Rosegarden, ma in linea di massima non ha conosciuto una grande diffusione.

In questo periodo si sente sempre più parlare di un nuovo standard per la creazione di plugins audio, nato per "superare le limitazioni di LADSPA" e DSSI, detto LV2 (Ladspa Version 2), buono sia per effetti che per strumenti e sembra essere molto promettente.

Audacity, Ardour, l'applicazione "jack-rack" (jackd invia l'audio a jack-rack, e questo glielo restituisce in tempo reale rimodulato dai plugin LADSPA che gli avete installato. Jack-rack è come una sorta di "rack" per effetti LADSPA..), il sequencer Rosegarden sono programmi capaci di usare i plugin LADSPA, aprendo a possibilità enormi.

I plugin LADSPA sono di solito contenuti in "/usr/lib/ladspa/", e sono file con estensione ".so".

GStreamer, non è un soundserver ma una piattaforma di sviluppo, una serie di API scritte in linguaggio C. In sostanza i contenuti multimediali vengono spesso distribuiti in formato compresso secondo determinati algoritmi (codecs), che per essere letti devono essere prima "tradotti in un linguaggio comprensibile" (raw PCM). Alla base di GStreamer sta il concetto di Pipeline (tubatura). In sostanza la "tubatura" consiste di tre parti:

1. Source: lettura della sorgente (ad esempio un file mp3) attraverso le API di GStreamer.
2. Decoder filter: GStreamer decodifica il sorgente in formato PCM. A questo punto il sorgente è perfettamente comprensibile e quindi manipolabile.
3. Sink: GStreamer fornisce le API per decodificare e anche per riprodurre la sorgente. L'ultimo elemento della catena, quindi, è praticamente la riproduzione.

Per quel che riguarda l'audio, GStreamer vi dà la possibilità di scegliere il vostro intermediario con l'ALSA driver. In altre parole voi eseguite un mp3 con un programma che usa GStreamer ("source", ad esempio Rhythmbox), GStreamer lo decodifica e lo manda a.... A chi volete voi! Eseguite da terminale "gstreamer-properties" e scegliete come completare la vostra "pipeline".. Scegliete se questo segnale decodificato deve andare ad alsa-libs (pipeline "alsasink"), a OSS ("osssink"), a PulseAudio ("pulsesink"). GStreamer e la struttura a "pipeline" consumano parecchie risorse della CPU.

In sostanza il funzionamento di GStreamer è legato ai plugin installati (ancora una volta ricorre il concetto di "plugin"). Se volessimo potremmo persino usare ESound, basta soltanto procurarsi il plugin "esdsink" che si trova in qualche pacchetto della vostra distribuzione (per ubuntu Jaunty è gstreamer0.10-esd).

Anche i "codecs" ovvero gli algoritmi usati per comprimere i contenuti multimediali, vengono decodificati da Plugin.

I plugin per i codec di GStreamer sono raggruppati, in base al livello decrescente di "qualità" del loro sviluppo, in 3 gruppi: Good, Bad, Ugly. In bad e Ugly si trovano gli algoritmi per la decompressione di codecs proprietari il cui sviluppo è spesso complicato da problematiche persino legali.

Esempi di applicazioni che usano GStreamer sono Amarok (audio player), Banshee (audio player), Elisa (media center), Exaile (audio player), Pitivi (video editor), Rhythmbox, Serpentine (audio cd recorder), Sound Converter, Sound Juicer (cd ripper), Totem (movie player) e altre.

In realtà GStreamer non è il solo decoding engine in Linux, sistema "gia pronto", in linux, per decodificare codec. "libavcodec" sono librerie per encoding/decoding video e audio usate da software quali FFmpeg (un convertitore di formato video, che rientra in realtà nell'ambito di un progetto più vasto, che include diversi programmi per audio-video, compresi player e convertitori), Avidemux, MPlayer, VLC, Xine. Ricordiamo inoltre le "xine-libs", che sono le librerie di decoding di "xine", ma anche di programmi come "Totem" e "Kaffeine". Alcuni programmi possono usare anche più di un sistema di decodifica, ad esempio "Totem" può usare le xine-libs ma anche GStreamer.

SDL (Simple Directmedia Layer): si tratta di librerie multipiattaforma utili per creare finestre, svolgere operazioni grafiche complesse, gestire audio, cd-rom, video, trattare contenuti multimediali (non solo "audio" quindi..). Le librerie SDL sono usate ad es. da software quali DOSBox, Bochs (SDL plugin per la sua GUI), MPlayer, ScummVM, VLC con SDL plugin ed altri.

OpenAL (Open Audio Library, multipiattaforma), sono delle API per gestire audio in 3D (e quindi multi-speakers). Le librerie OpenAL sono usate in videogiochi come Doom3, Quake3, Unreal tournament 2003, Prey.

FMOD (prodotto commerciale, multipiattaforma) sono librerie audio (con API). Sono usate in videogiochi come Call of Duty 4, Far Cry, Second Life. Il loro uso è gratuito per prodotti non commerciali, altrimenti è un prodotto che si paga (e costa..).

Questo è il punto, alcune applicazioni usano le API ALSA, altre quelle aRts, altre le API di JACK, alcune gestiscono l'audio da se, ma solo se l'audio passa tutto da un server solo, può essere realmente controllabile. Pensate in che situazione si trova uno sviluppatore di un software, che deve prima di tutto scegliere come gestire l'audio della sua applicazione e adottarne le API. Allora quindi scegliere le API più performanti? Quelle in più fervente sviluppo? Quelle apparentemente più supportate (oggi..)? O quelle più "multipiattaforma"?..

I soundservers

"sound servers" (o "sound daemons"), sono ad esempio Pulseaudio, ESounD, NAS, aRTS, JACK.

In sostanza noi passiamo l'audio al sound server, e sarà lui poi a preoccuparsi della comunicazione col driver ALSA nel kernel.

Perchè mettere fra il noi e L'ALSA driver un soundserver?

Il soundserver si preoccupa di gestire i diversi flussi audio, passandoli miscelati all'ALSA driver, magari modificati secondo preferenze da noi impostate.

Ci permette quindi di gestire l'audio in maniera molto flessibile. Per fare questo i soundservers funzionano come programmi in "background" (daemons, "demoni"). In realtà dividere un programma in un "demone" che funziona in background ed in un'interfaccia che lo gestisca ("GUI"), è un approccio sempre più spesso usato nei progetti liberi. Progetti come network manager e dbus possono essere usati da chiunque senza installare gnome. Tale libertà assicura una larga accettazione ed attrae verso lo sviluppo del progetto. dcop necessitava di kdelibs e qt. Il demone, in quanto progetto separato, non ha dipendenze su GNOME o KDE. La GUI invece sarà sviluppata in base all'ambiente desktop scelto.

I soundservers sono apparsi indispensabili con OSS, che non poteva gestire più flussi contemporaneamente. Oggi i soundservers permettono spesso, oltre ad una gestione "centralizzata" dell'audio, anche numerose altre possibilità come l'uso di più schede audio contemporaneamente e persino l'invio dello stesso flusso audio alla rete (e altre funzioni..).

JACK è un audio server (un "demone" ovvero un programma eseguito in background. Di solito i demoni hanno dei nomi che terminano con la lettera "d") multipiattaforma (Linux, Windows, OS X e altre), ha le sue API (funzioni usate dagli sviluppatori di software nei propri programmi, per usare JACK) ed è ritenuto ottimo per l'editing e il mixing.

E' considerato piuttosto performante (bassa latenza, ovvero basso ritardo tra azione ed effetto. Considerate che una latenza di appena 30 ms rende difficile il lavoro ad un musicista in quanto il suono verrebbe percepito come "eco").

qjackctl è la GUI (Graphical user interface) più usata per controllare il demone Jackd.
jack-rack permette di applicare dei filtri audio LADSPA
jackdmp è una versione di Jack per sistemi multiprocessore.
jackeq è un equalizzatore.
patchage: mostra il grafico delle connessioni Jack correnti.

JACK è usato da software come Ardour (registratore multitraccia), Hydrogen (batteria elettronica), Audacity, altre (vedi di LADSPA più avanti).

Jackd è il JACK daemon. Possiamo usare jackd in riga di comando passandogli dei parametri, e comunicargli ad esempio di riprodurre usando il driver ALSA (esempio: jackd -d alsa -d hw:0 -r 44100).

aRts (analog realtime synthesizer) era un server audio usato da KDE. Il demone era chiamato "artsd". Si appoggia a OSS o ALSA. KDE 4 ha rimpiazzato Phonon al posto di aRts. Phonon non è un server audio.

Enlightened Sound Daemon (ESD o EsounD), sound server di Enlightened ("window manager"), adesso del tutto rimpiazzato in GNOME da:

- libcanberra (per gli "event sounds" ovvero i suoni di sistema degli ambienti desktop, come GNOME. libcanberra può usare ALSA, PulseAudio, OSS, GStreamer)
- GStreamer/PulseAudio per tutto il resto

Pulseaudio è oggi il sound server standard in diverse distribuzioni linux, è considerato un "tentativo di sostituire ESounD", rispetto a cui ha tempi di latenza minori ed una migliore architettura. Pulseaudio può addirittura mandare l'audio a un altro computer collegato in rete, che monti anch'esso Pulseaudio.

Bisogna considerare che perchè PulseAudio possa interfacciarsi ad altre infrastrutture come GStreamer, ALSA, JACK, bisogna installare dei pacchetti separati, spesso "plugin" dei vari programmi che intendono usare PulseAudio.
xmms-pulse è il plugin di XMMS (è un programma music player) perchè questo usi PulseAudio.
libao-pulse è il plugin per le librerie cross-plattform libao
gst-pulse è il plugin per GStreamer.
pulseaudio-esound-compat: diciamo che questo è uno "script di compatibilità". In sostanza permette di rimpiazzare il pacchetto ESD "esound" creando un link /usr/bin/esd che è un link simbolico a PulseAudio. Sicchè tutte le applicazioni che vorrebbero usare ESD, in realtà vengono "fregate" e usano PulseAudio.

Con PulseAudio potete addirittura fare in modo che solo gli utenti autorizzati possano avere accesso all'audio del pc; usa infatti dei "gruppi":
- pulse (che è per uso interno di PulseAudio, non vi interessa, gli utenti non devono prenderne parte)
- pulse-access (quando in "system mode", il server audio accetta solo le connessioni dei membri al gruppo)
- pulse-rt (per avere accesso all'audio "real time", ma bisogna avere dei kernel "particolari" detti real-time.)

Se il vostro username non fa parte di quei gruppi (per vedere i gruppi di un utente in linux fate su shell "groups username") ma sentite lo stesso l'audio, questo di certo sarà perchè il server è stato fatto partire "localmente" (per verificare ciò fate in shell "ps aux | grep pulse" e verificate se pulseaudio è stato fatto partire dal vostro utente (lo identificate con il codice nella prima colonna, il vostro lo vedete scrivendo su shell "echo $UID" oppure semplicemente "id -u").

In sostanza PulseAudio passa l'audio di tutte le applicazioni che usano le API di Pulseaudio all'ALSA driver. Bisogna fare in modo però che anche le applicazioni che NON usano Pulseaudio, passino comunque da Pulseaudio, in modo che questo possa miscelare il tutto e passarlo all'ALSA driver. Per questo c'è un apposito ALSA pulseaudio plug in. In sostanza prima che i suoni raggiungono l'ALSA driver, l'alsa plugin li dirotta verso Pulseaudio, che può quindi mandarli all'ALSA driver.

In sostanza, fra i file di configurazione di ALSA, dovreste trovare qualche riga di codice in cui è specificato che Pulseaudio viene dichiarato periferica di riproduzione ("pcm") e controllo ("ctl") di default per ALSA. Leggete quindi /usr/share/alsa/alsa.conf, lì dovrebbe trovarsi una lista di altri file di configurazione di ALSA, che vengono infatti letti proprio a partire da alsa.conf, che è il principale. Fra tali file potreste trovare .asoundrc, /etc/asound.conf, /usr/share/alsa/pulse.conf che è appunto il file dove più probabilmente si trova la modifica in questione (o un'altra sottolista di file in cui guardare!).

La modifica consiste di queste semplici righe:

```
pcm.!default {
type pulse
}

ctl.!default {
type pulse
}
```

```
pcm.pulse {
type pulse
}

ctl.pulse {
type pulse
}
```
Le prime righe impostano "pulse" come periferiche pcm e ctl di default. Le altre righe creano una unità virtuale chiamata "pulse" per pcm (riproduzione) e ctl (controllo).

Se effettuate modifiche ricordatevi di fare ripartire nuovamente le ALSA utils (sudo /etc/init.d/alsa-utils restart).

Due software sono attualmente in sviluppo per gestire PulseAudio:

* pavucontrol (PulseAudio Volume Control): vi dà la possibilità di regolare addirittura il volume di ogni singola applicazione che in quel momento sta riproducendo (o registrando?) qualcosa.. Vi consiglio vivamente di installarlo..

* Paprefs (PulseAudio Preferences). Paprefs configura solo i servers locali e richiede "module-gconf" installato, che serve a rendere alcune opzioni modificabili da GConf (sistema utilizzato da gnome per memorizzare le impostazioni di configurazione, i cui cambiamenti sono controllati dal demone gconfd).

* Gli altri software prodotti per Pulseaudio sono da considerarsi obsoleti:
* Paman (Pulseaudio manager)
* Pavumeter (Pulseaudio Volume Meter)
* Padevchooser (Pulseaudio device chooser)

Gnome stesso fornisce uno strumento di base che si chiama gnome-volume-control (contenuto nel pacchetto gnome-media), che già in se integra un pò tutto quello che serve in fatto di volumi e regolazioni. Purtroppo però, almeno al momento, risulta ancora indispensabile installare strumenti come pavucontrol per una gestione completa dei flussi in entrata/uscita e le applicazioni che vi interagiscono..

Molte applicazioni KDE usano aRts. si può settare aRts ad usare ESD (ESD ha un apposito plugin per usare PulseAudio..).

Amarok, XMMS ed altri media players permettono di scegliere il "motore" da usare tramite le loro "preferenze" (Quindi PulseAudio).

Audacity, basato sulle librerie cross-platform wxWidgets (per la sua interfaccia), usa le API del soundserver multipiattaforma "PortAudio" (Windows, Unix, Macintosh e altre), che alla fine usa ALSA (e PulseAudio controlla ALSA, grazie al "giochetto" sul suo "asound.conf"). Audacity può supportare il server audio Jackd. Fino ad un pò di tempo fa Audacity aveva qualche problema a funzionare con Pulseaudio, che qualcuno risolveva usando, da riga di comando, il comando padsp ("Pulseaudio OSS wrapper", in pratica è un comando per ridirezionare il suono OSS a Pulseaudio. padsp per intenderci è tipo "aoss", che ridireziona invece verso ALSA). Altri risolvevano il problema sospendendo del tutto Pulseaudio usando "pasuspender".

Oggi, personalmente non riscontro più alcun problema ad usare audacity con Pulseaudio, data l'evoluzione di entrambi, suppongo..

Sono in molti ad usare Audacity per registrare l'audio dei filmati di youtube (video musicali soprattutto), telefonate o altro, in mp3 (grazie a pavucontrol, che permette di impostare la registrazione a "monitor

interno".

Vorrei aggiungere qualche parola su Audacity. Sono in molti a dire che sia un software limitato, non assolutamente a paragonabile ad altri famosi audioeditors.

Personalmente ritengo che Audacity dalla versione 1.3.12 in poi, installato e con le sue estensioni abilitate, ovvero le librerie LAME (che servono alla manipolazione di file MP3) e le FFmpeg (che consentono ad Audacity di manipolare al volo formati audio persino proprietari, come i .wma, sebbene lo stesso sito di Audacity dica che non si possa fare..bah!), sia un software estremamente potente e dotato, a differenza di altri più blasonati prodotti, di una interfaccia intuitiva ed accattivante.

Inoltre Audacity può persino sfruttare plugin LADSPA, VST e Nyquist (sono tutti dei linguaggi per creare dei programmi atti a manipolare audio) estendendo le sue possibilità. Non nego che, usandolo, a volte Audacity possa bloccarsi (capita..) ma grazie all'autosalvataggio del progetto, che Audacity opera in automatico di continuo, non ho mai perso un secondo del mio lavoro anche a seguito di un blocco. Devo dire che questa 1.3.12 per ora non mi si è mai bloccata.

La latenza dell'audio

Come misurare la LATENZA del vostro sistema (inteso come hardware e software audio che usate)? Settate il sistema per registrare dal microfono. Registrate qualcosa, preferibilmente qualcosa di "cadenzato", come il suono di un metronomo (ottimo generare una "click track", è fra gli strumenti di Audacity). A questo punto fate in modo che il sistema mandi in riproduzione quanto registrato prima e contemporaneamente "registri" se stesso. Osservate quindi lo sfasamento dei due canali registrati (servitevi degli audio meters in basso, nell'interfaccia di Audacity). A questo punto cercate la "correzione latenza" fra le preferenze di Audacity e impostatevi il valore da voi determinato prima.

Quanto seriamente avete intenzione di usare Linux per produrre audio? Un vantaggio in termini di latenza lo dà l'installazione di un kernel rt (realtime).
CONFIG_PREEMPT_RT

SEZIONE DA DA COMPLETARE..

Linux e gestione di players multimediali

Creative Zen usa il protocollo proprietario MTP (Media Transfer Protocol).

In realtà si tratta di un set di estensioni di un altro protocollo gia esistente detto PTP (Picture Transfer Protocol). Questo protocollo è usato per il trasferimento di files multimediali (coi loro metadati) da pc a player attraverso USB.

Rispetto a tutti i player multimediali che vengono gestiti dal pc come fossero delle semplici memorie di massa USB, i player che usano questo protocollo sono piu sicuri in caso di disgiunzione accidentale dal pc. Questo perchè, a differenza delle normali memorie di massa USB, il file system di queste unità viene controllato dal dispositivo stesso e non dal pc, sebbene il sistema operativo cerchi di non farci accorgere di questa sostanziale differenza.

Linux riesce a gestire l'MTP tramite le libmtp (usate da programmi come Amarok, Banshee, Rhythmbox) e libgphoto2 (usate ad esempio da gphoto2, un programma da riga di comando, ed F-Spot).

Nautilus gestisce le MTP tramite gphoto2.

La verità è che sinceramente trovo ancora un macello la gestione di questo protocollo su Linux.

Volete copiare un file dal vostro Linux dritto sul vostro Creative Zen?.. Ovviamente la prima cosa che fate è prenderlo da Nautilus e spostarlo dentro il Creative Zen, cosa infruttuosa dato che il file viene copiato ma non viene caricato nel database che il dispositivo si crea delle canzoni (e quindi la canzone "non viene vista" dal dispositivo quando lo staccate dal vostro pc..).

Potete allora usare Gnomad2, che però, insieme ad Rhythmbox e altri, danno la possibilità di inserire canzoni nel vostro dispositivo ma non in comode ed ordinate sottocartelle.. di conseguenza vi ritroverete presto 6-7 giga di mp3(o anche di più..) cacciati e ammassati tutti dentro un'unica enorme cartella "Music"..

Alle fine ho installato Virtualbox e ci ho virtualizzato dentro Windows 7. Ecco come ho fatto:

- Ho installato Virtualbox versione PUEL dal sito del produttore (non installate la versione OSE contenuta nei repository della vostra distribuzione, non ha il supporto USB).
- Ho creato una macchina virtuale con 1 giga di RAM (il mio pc ha 2 Gb di ram)
- Ho installato Windows 7
- Ho installato le VBoxGuestAdditions (non c'è bisogno di installare il software del Creative Zen sul Windows 7 virtualizzato, funziona perfettamente senza..).
- Ho aggiornato il driver AC97 per Windows 7 con Windows Update (in modo di potere disporre dell'audio e del midi)
- Ho messo in condivisione il mio Desktop di Linux con Virtualbox e l'ho chiamato "shared" (vedi opzioni di Virtualbox).
- Poi da CMD su Windows 7 ho eseguito il comando **net use x: vboxsvrshared**
- Và che è una bellezza..

Soluzione discutibile quanto volete ma che in quanto ad efficienza spacca..

Spero che Nautilus, strumento CARDINE di Gnome, cominci presto ad integrare una affidabile gestione dell'MTP..