Configurar Gnu/Linux para producción musical

Cualquier distribución de Gnu-Linux puede configurarse para trabajar con audio en tiempo real (con “tiempo real” me refiero a acceder a los procesos relacionados con el audio sin latencia o tiempo de retardo), en Microsoft Windows se precisa de controladores especiales para aprovechar el proceso DSP de las placas de sonido profesionales, estos driver se denominan ASIO (Audio Stream Input/Output). En Gnu/Linux existe la ventaja de que no se precisan drivers ni controladores especiales sino que el sistema se configura para este propósito, desde el corazón mismo que se denomina kernel, permitiendo así mayor control y rendimiento y no dependiendo de que los fabricantes de las placas de sonido, nos provean un driver ASIO.
Voy a tratar de hacer un resumen de como configurar un sistema GNU/Linux para trabajar con audio, basándome en las distribuciones más conocidas, igualmente existen distribuciones armadas para esto, pero siempre es bueno saber como se hace.

1-En distribuciones que derivan de Debian como Ubuntu, GNU/Linux Mint, etc; el kenel real time lo encontramos en los repositorios oficiales de software, así que abriendo una terminal o consola y escribiendo:

sudo aptitude install linux-image-2.6.XX.X-rt

(El comando sudo es para poder acceder a los permisos de administrador y hacer cambios de sistema)

En Debian o distribuciones más seguras con el comando su, nos solicita directamente la contraseña de administrador, y podemos operar con estos permisos, una vez introducida la clave.

También se puede instalar utilizando el gestor de paquetes visual que tenga la distribución GNU-Linux, por ejemplo el Gestor de paquetes Synaptic.

2-Luego lo más importante es editar el archivo /etc/security/limits.conf (para lo que se puede usar como superusuario el editor de textos gedit, nano, mcedit o cualquier otro disponible), otra vez en la terminal escribimos:

sudo gedit /etc/security/limits.conf y agregamos al final estas líneas:

@audio - rtprio 99
@audio - nice -10
@audio - memlock 4000000

Guardamos el archivo y reiniciamos la máquina.

Explicación de lo anterior: El valor rtprio es la máxima prioridad con la que un usuario del grupo audio puede ejecutar una tarea. El valor memlock es la máxima cantidad de memoria que un miembro del grupo audio puede bloquear para una tarea siendo ejecutada en tiempo real. Debería ser inferior a la máxima cantidad de memoria física; siendo una recomendación común establecerla a la mitad exacta. El valor nice es el mínimo con el que una tarea puede ser ejecutada; se trata de la predisposición de una tarea a liberar tiempo de CPU.

Luego podemos instalar y configurar jack y qjack:

sudo aptitude install qjack jackd

#y luego configurar correctamente jack, ejecutando qjackctl y activando en setup (o configuración) estas opciones:

tiempo real (activado)
no bloquear memoria (activado)
modo tolerante (activado)
forzar 16bit (activado si querés trabajar en 16 bits)

#y agregás los valores:
Prioridad=70
Cuadros/Períodos=128 (con esto conseguís 5.8 ms de latencia podes usar 256 también si querés liberar más el cpu consiguiendo 11.4 ms de latencia -recomendado en pcs mas antiguas-)
Frecuencia de muestreo=44100
Períodos/buffer=2
Máximos puertos=128
Límite de tiempo= 5000 (ms)
Interface= la placa que uses o default
Suavizado=ninguno
Audio=duplex
Dispositivo de entrada=(default)
Dispositivo de salida=(default)
canales=0
latencias=0

Algunos ejemplos de Software Libre Musical:

Ardour: El clon Pro Tools, super rápido-estable y muy profesional
Rosegarden: Secuencer Midi-Audio.
NtEd: Editor de Partituras.
Qsynth: Sampler de Sound Fonts.
Hydrogen: Máquina de ritmos
Audacity: Editor de audio multipista.
ZynAddSubFX: Sinte virtual, con varios tipos de síntesis.
Qjackctl: Interface gráfica para jackd (servidor de audio en tiempo real)

conecta todo que te nombro y lo gestiona.
Plugins Caps y Tap: Muy buenos plugins LADPSA para usar con cualquier secuencer o editor.
Lmms: Clon de Fruity Loops.
Seq24: Secuenciador midi, basado en loops.

Establecer prioridades RT, o sea aprovechar el kernel real time:

En esta parte de la explicación me “baso” en el artículo de Florian Paul Schmidt (muy completo el que quiera lo puede buscar en la web) pero agregando todo lo que yo pude experimentar.

El paso final es establecer unas cuantas prioridades de tareas. El parche para tiempo real debería haber creado varias tareas IRQ, así como una tarea hrtimer. Pueden visualizarse con el comando:

 # ps -e

nos muestra lo siguiente (solo copio las tareas que nos pueden llegar a interesar para audio ya que son muchas, para ahorrar espacio):

PID	TTY	TIME	CMD
1	?	00:00:01	init
2	?	00:00:00	kthreadd
3	?	00:00:00	migration/0
4	?	00:00:00	posixcputmr/0
6	**?**	**00:01:11**	**sirq-timer/0**
12	**?**	**00:00:00**	**sirq-hrtimer/0**
53	?	00:00:00	IRQ-9
295	?	00:00:25	IRQ-12
296	?	00:00:00	IRQ-1
303	?	00:00:00	IRQ-8

616	?	00:00:00	IRQ-16
628	?	00:00:00	IRQ-19
645	?	00:00:00	IRQ-18
665	?	00:00:00	IRQ-23
692	?	00:00:00	IRQ-6
752	?	00:00:02	IRQ-14
753	?	00:00:05	IRQ-15
1435	?	00:00:00	IRQ-7
1493	?	00:01:14	IRQ-17
1872	?	00:00:08	IRQ-20
2480	?	00:00:00	IRQ-4
2483	?	00:00:00	IRQ-3

Nos interesan las tareas: sirq-timer/0, sirq-hrtimer/0 y la IRQ de nuestra placa de sonido que tenemos que ubicar cual es (ya voy a explicar como ubicarla). En un sistema core dual debe haber dos de cada tarea, una por cada CPU. Yo no pude comprobarlo porque no tengo core dual.
Vamos a tomar nota del PID de cada una: 6 (sirq-timer) y 12 (sirq-hrtimer).

Luego vamos a buscar el pid de la tarea IRQ en la que reside la tarjeta de sonido. (En el caso de que la tarjeta de sonido sea usb, correr jackcontrol con esta tarjeta, y en una consola ver con el comando “top” que dirección IRQ levanta).
Vemos la salida de:

 # cat /proc/interrupts

debería revelar esto:

	CPU0		
0:	200	IO-APIC-edge	timer
1:	14890	IO-APIC-edge	i8042
3:	2	IO-APIC-edge	
4:	1	IO-APIC-edge	
6:	7	IO-APIC-edge	floppy
7:	0	IO-APIC-edge	parport0
8:	2	IO-APIC-edge	rtc0
9:	0	IO-APIC-fasteoi	acpi
12:	529675	IO-APIC-edge	i8042
14:	88412	IO-APIC-edge	ide0
15:	587044	IO-APIC-edge	ide1
16:	0	IO-APIC-fasteoi	uhci_hcd:usb1
17:	**1685387**	**IO-APIC-fasteoi**	**ICE1712**
18:	0	IO-APIC-fasteoi	uhci_hcd:usb3
19:	0	IO-APIC-fasteoi	uhci_hcd:usb2
20:	19611	IO-APIC-fasteoi	eth0
23:	229710	IO-APIC-fasteoi	ehci_hcd:usb4
NMI:	0	Non-maskable interrupts	
LOC:	7295599	Local timer	

		interrupts
RES:	0	Rescheduling interrupts
CAL:	0	function call interrupts
TLB:	0	TLB shootdowns
TRM:	0	Thermal event interrupts
SPU:	0	Spurious interrupts
ERR:	0	
MIS:	0	

Entonces según esta información:

17: 1660453 IO-APIC-fasteoi ICE1712

el IRQ-17 es donde está mi placa de sonido (en mi caso me doy cuenta porque ICE1712 es el chipset de las m-audio, que es mi placa)

Para obtener el PID, uso de nuevo el comando:

 # ps -e

si se fijan al principio del documento (cuando mostré el resultado de #ps -e) era el PID 1493

Ahora vamos a utilizar el comando chrt para establecer prioridades real time. En Debian, este comando forma parte del paquete schedutils. No importa el orden en el que se establecen las prioridades.
Establecemos nuevas prioridades para cada tarea:

En mi caso:

 # chrt -f -p 99 6
 # chrt -f -p 99 12
 # chrt -f -p 99 1493

Esto debe establecer una prioridad de tiempo real a las tareas identificadas como 6, 12 y 1493. Una prioridad tan alta como 99 podría no ser en realidad

necesaria e incluso un tanto insegura.
Hay que probar con valores (por lo menos de 50 a 99) hasta encontrar los adecuados.

Luego y usamos el comando:

 # top

para ver si cambiaron las prioridades (antes ejecuté jackd para poner en uso la placa de sonido para que aparezca IRQ-17 entre los procesos) me devuelve:

PID	USER	PR	NI	VIRT	RES	SHR	S	% CPU	% MEM	TIME+	COMMAND
2747	gabriel	20	0	22588	2052	1716	S	11.6	0.1	0:02.91	jackd
2718	gabriel	20	0	37700	13m	9.9m	S	11.3	0.9	0:09.77	gnome-terminal
2741	gabriel	20	0	44080	11m	9.8m	S	10.0	0.8	0:02.98	qjackctl
2577	root	20	0	97.8m	11m	4788	S	6.6	0.8	0:07.85	Xorg
1493	**root**	**RT**	**-5**	**0**	**0**	**0**	**S**	**3.3**	**0.0**	**0:01.19**	**IRQ-17**
2713	gabriel	20	0	30472	11m	7524	S	1.7	0.7	0:01.77	pcmanfm
295	root	-51	-5	0	0	0	S	0.7	0.0	0:00.70	IRQ-12
2711	gabriel	20	0	20560	10m	7168	S	0.7	0.7	0:04.87	lxpanel
2739	gabriel	20	0	2428	1136	900	R	0.3	0.1	0:00.41	top
1	root	20	0	2132	748	652	S	0.0	0.0	0:01.38	init
2	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	posixcputmr/0

5	root	-51	-5	0	0	0	S	0.0	0.0	0:00.00	sirq-high/0
6	**root**	**RT**	**-5**	**0**	**0**	**0**	**S**	**0.0**	**0.0**	**0:00.33**	**sirq-timer/0**
7	root	-51	-5	0	0	0	S	0.0	0.0	0:00.00	sirq-net-tx/0
8	root	-51	-5	0	0	0	S	0.0	0.0	0:00.00	sirq-net-rx/0

en este caso no aparece la línea:

12	**root**	**RT**	**-5**	**0**	**0**	**0**	**S**	**0.0**	**0.0**	**0:00.01**	**sirq-hrtimer/0**

pero se debe a que top esta en movimiento todo el tiempo (pero les aseguro que está).

Como pueden ver en la sección PR se puede ver la prioridad RT a la que están corriendo las tareas sirq-timer/0, sirq-hrtimer/0 y IRQ-17.

En realidad todo lo que hicimos hasta ahora sirve para probar el funcionamiento de las prioridades. Estas prioridades serán restablecidas después de cada reinicio, así que deberá crearse un script de inicio sencillo, para recuperarlas.
Dado que los pids de sirq-timer/0 y sirq-hrtimer/0 no van a cambiar, pero el pid de la placa de sonido sí, vamos a usar "pgrep" para que busque la tarea IRQ-17, a continuación describo como tiene que ser el contenido del script y donde debemos copiarlo, (yo voy a usar el editor de textos nano pero ustedes pueden usar el que quieran) :

 # nano realtime.sh
copiamos lo que sigue a continuación adentro:

```
#!/bin/bash
#el 70 es la prioridad y el 6 el pid del sirq-timer
chrt -f -p 70 6
#el 12 es el pid de sirq-hrtimer
chrt -f -p 70 12
#IRQ-17 es el IRQ de la placa de sonido
chrt -f -p 70 `pgrep IRQ-17`
```

presionamos ctrl-o para salvar y luego <enter>

Luego le damos permisos de ejecución con:

 # chmod +x realtime.sh

Y lo copiamos a /etc/init.d :

 # cp realtime.sh /etc/init.d/

Y creamos un enlace simbólico que copiamos a /etc/rc5.d ; /etc/rc4.d ; /etc/rc3.d y a /etc/rc2.d

 # ln -s /etc/init.d/realtime.sh /etc/rc5.d/S99realtime
 # cp /etc/rc5.d/S99realtime /etc/rc4.d/
 # cp /etc/rc5.d/S99realtime /etc/rc3.d/
 # cp /etc/rc5.d/S99realtime /etc/rc2.d/

Para comprobar que todo funciona una vez reiniciado el sistema usar el comando **top** de nuevo.

Nota sobre APIC: En placas bases con APIC y kernels donde APIC está habilitado, pueden tenerse dificultades al establecer prioridades en las IRQ. Simplemente iníciese la máquina con la opción noapic en grub para el kernel utilizado, y compruébese si desaparecen los problemas.

Gabriel Nicolás González Ferreira
(Correcciones de Esteban Segreto)