Composition musicale et Temps Réel

Emmanuel Saracco

Dans l'article Composition musicale sous GNU/Linux (Linux+ DVD 4/2008) vous aviez vu les possibilités de GNU/Linux en matière de MAO. Mais pour composer sans être confronté à des problèmes de désynchronisation et autres événements gênants qui viendraient troubler votre créativité et le rendu final, il est primordial de permettre aux logiciels, et en particulier au serveur de sons, de fonctionner en Temps Réel. Pour ce faire il faut tout d'abord revoir la configuration du noyau. Ensuite il faut configurer le système. Il s'agit-là d'un domaine qui peut devenir complexe car il dépend de nombreux facteurs, comme la version de Linux, les logiciels, les bibliothèques, votre matériel etc. Mais il est possible d'arriver à un résultat satisfaisant !

linux@software.com.pl

Qu'est-ce que le Temps Réel ? Sans entrer dans le détail, il existe deux types de mise en oeuvre du Temps Réel. Le premier type est ce qu'on appelle un *Temps Réel dur*. Il s'agit d'un système strictement déterministe dans lequel le temps de latence entre l'arrivée d'une demande et son traitement doit être garanti et donc ne jamais être dépassé.

Ce type de Temps Réel se rencontre principalement sur des systèmes critiques (pilote automatique, système de freinage, centrale nucléaire etc.). Le second type est appelé *Temps Réel mou*.

Cela signifie qu'on peut *garantir* une latence maximum pour la prise en compte des événements, mais aussi que cette latence peut dans certains cas être dépassée (lorsque la charge est trop élevée par exemple) sans que cela soit catastrophique pour le bon fonctionnement du système.

C'est de ce type de Temps Réel dont nous avons besoin pour la MAO. Cela va permettre aux applications de traiter de manière quasiment synchrone pour l'oreille humaine les événements relatifs au traitement du son.

Configuration du noyau

Par défaut le noyau des distributions GNU/Linux standard n'est pas adapté à la MAO. Il faut le personnaliser. Cela nécessite donc que vous récupériez l'archive et l'installiez sous */usr/src/*. Listing 1 montre les opérations à effectuer pas à pas.

Dans tous les exemples les numéros de version ne sont là qu'à titre indicatif. Adaptez-les aux évolutions des logiciels.

L'exécution de `make menuconfig` vous permettra de configurer le noyau. Si vous avez utilisé le gestionnaire de paquets de votre distribution pour obtenir le code source alors il y a de grandes chances pour que la configuration standard soit d'emblée correcte. Pour ceux

Cet article explique...

Dans cet article, vous découvrirez ce qu'on entend par Temps Réel, et ce qu'il faut faire pour obtenir des performances raisonnables dans le but de composer en toute tranquillité sous GNU/Linux.

qui ont suivi les instructions ci-dessus et partent d'un noyau vanilla, vous devrez dans un premier temps l'adapter à votre système. Mais vous ne verrez ici que la configuration des options directement liées au Temps Réel.

Il faut savoir que des parties du patch Temps Réel d'Ingo Molnar sont régulièrement intégrées au noyau, ce qui permet à certains (dont l'auteur) de se contenter d'un noyau standard. Je vous conseille donc dans un premier temps de faire des tests sans ce patch. Et si vraiment vous avez besoin d'une latence plus faible ou plus stable, alors patchez !

Une fois la configuration terminée (cette partie est expliquée plus loin), que vous utilisiez le patch Temps Réel ou non il faudra compiler et installer le noyau. La procédure est simple :

```
$ make
$ make modules_install
$ su -c "make install"
```

Lisitng 1. Installation noyau

```
$ cd /usr/src/
$ wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.24.tar.bz2
$ tar jxvf linux-2.6.24.tar.bz2
$ ln -s  linux-2.6.24 linux
$ cd linux/
$ make menuconfig
```

Lisitng 2. Configuration noyau vanilla

```
Processor type and features --->
    Preemption Model (Preemptible Kernel (Low-Latency Desktop)) --->
    [*] Preempt The Big Kernel Lock
    Timer frequency (1000 HZ) --->

  Device Drivers  --->
    Character devices  --->
      <M> Enhanced Real Time Clock Support
      <M> Generic /dev/rtc emulation
      [*]   Extended RTC operation

    <M> Real Time Clock  --->
      [*]   /sys/class/rtc/rtcN (sysfs)
      [*]   /proc/driver/rtc (procfs for rtc0)
      [*]   /dev/rtcN (character devices)
      <M>   PC-style 'CMOS'

    Sound  --->
      <M> Sound card support
        Advanced Linux Sound Architecture  --->
          <M> Advanced Linux Sound Architecture
            <M>   Sequencer support
            <M>     Sequencer dummy client
            <M>   OSS Mixer API
            <M>   OSS PCM (digital audio) API
            <M>   RTC Timer support
            [*]     Use RTC as default sequencer timer
          Generic devices  --->
            <M> Virtual MIDI soundcard
          PCI devices  --->
            <M> RME Hammerfall DSP Audio

  Security options --->
    [*] Enable different security models
    <M>   Default Linux Capabilities
```

Éditez ensuite la configuration de votre gestionnaire d'amorçage afin d'y ajouter la prise en compte du nouveau noyau. N'oubliez pas d'adapter les répertoires et les périphériques dans les deux exemples suivants avant de les utiliser !

Si vous utilisez *GRUB*, ouvrez le fichier */boot/grub/menu.lst* et ajoutez-y le contenu de :

```
title Noyau 2.6.24 MAO
root (hd0,5)
kernel /boot/vmlinuz-2.6.24 root=/
      dev/sda6 ro
savedefault
```

Si vous utilisez *Lilo*, ouvrez le fichier */etc/lilo.conf* et ajoutez-y :

```
image = /boot/vmlinuz-2.6.24
label = MAO
root = /dev/sda6
read-only
```

N'oubliez pas de lancer la commande lilo pour que la modification soit prise en compte.

À partir d'un noyau standard

Vous ne trouverez dans le Listing 2 que les options relatives au Temps Réel et au son. Certaines ne sont pas nécessairement adaptées à votre configuration. À vous de les adapter. C'est le cas par exemple du choix du module RME Hammerfall DSP Audio dans la section PCI devices qui correspond à ma carte son. Pour le reste, basez-vous sur cette configuration et effectuez des tests. Il faut être pragmatique dans ce domaine...

Avec le patch Realtime Preemption

Il s'agit d'un patch activement maintenu par Ingo Molnar. Il permet d'améliorer les capacités Temps Réel du noyau. Voici les étapes à suivre pour l'installer :

Ce qu'il faut savoir...

Les connaissances de base de GNU/Linux et de l'administration système sont nécessaires à la compréhension de cet article. Il vous faudra également connaître les rudiments de l'utilisation de `make` et de la configuration/installation d'un noyau. Des précautions, qui ne seront pas détaillées ici, doivent être prises lors de certaines opérations décrites dans cet article.

```
$ cd /usr/src/
$ wget http://www.kernel.org/pub/
      linux/kernel/projects/rt/
      patch-2.6.24-rt1.bz2
$ cd linux/
$ bzcat ../patch-2.6.24-rt1.bz2 |
patch -p1
```

Voilà votre noyau patché et prêt à être configuré. Si l'on reprend la configuration d'un noyau standard décrite précédemment, voici ce qui est spécifique à ce patch :

```
General setup --->
    [*] Enable concurrent radix tree
          operations
    [*] Enabled optimistic locking
Processor type and features --->
    [*] Enable priority boosting of
          RCU read-side critical
sections
```

Il faut également bien prendre garde à ce que les options suivantes soient décochées (Listing 3). Votre noyau est prêt à être compilé et testé!

La grenouille et le bœuf...

La fiabilité du Temps Réel au niveau du noyau est une chose, le temps de réponse du matériel en est une autre. Rien ne sert de vouloir descendre en dessous de 3 ms si votre carte son ne suit pas. Le principal étant que, dans des limites acceptables pour votre matériel, votre latence soit la plus stable possible.

Lisitng 3. Configuration noyau avec patch realtime-preemption

```
Processor type and features --->
    [ ] Enable tracing for RCU - currently stats in debugfs

  Device Drivers  --->
    Character devices  --->
      [ ] Real Time Clock Histogram Support
      < > Parallel Port Based Latency Measurement Device

  Kernel hacking  --->
    [ ] Wakeup latency timing
    [ ] Non-preemptible critical section latency timing
    [ ] Interrupts-off critical section latency timing
```

Lisitng 4. Installation de JACK

```
$ cd /usr/src/
$ wget http://jackaudio.org/downloads/jack-audio-connection-kit
      -0.109.0.tar.gz
$ tar zxvf jack-audio-connection-kit-0.109.0.tar.gz
$ cd jack-audio-connection-kit-0.109.0/
$ ./configure -prefix=/usr --enable-mmx --enable-sse --enable-dynsimd --
      enable-optimize --enable-resize --enable-timestamps -enable-posix-
      shm-with-gnu-ld
$ make
$ su -c "make install"
```

Lisitng 5. Installation de QjackCtl

```
$ cd /usr/src/
$ wget http://switch.dl.sourceforge.net/sourceforge/qjackctl/qjackctl-
      0.3.2.tar.gz
$ tar zxvf qjackctl-0.3.2.tar.gz
$ cd qjackctl-0.3.2/
$ ./configure -prefix=/usr
$ make
$ su -c "make install"
```

Configuration du système

Maintenant que vous avez correctement configuré le noyau il faut appliquer quelques optimisations, mais surtout, il est nécessaire de donner les droits nécessaires aux logiciels qui doivent profiter pleinement du Temps Réel.

Optimisations

Dans ce domaine il ne faut rien sous-estimer. C'est à force de petites optimisations par-ci par-là qu'on arrive à quelque chose de sensible.

Il ne s'agit que de quelques pistes... N'hésitez pas à pousser plus avant la recherche d'optimisations pour votre système ; la fluidité de vos applications ne peut qu'en être améliorée !

RTC et HPET

Dans le prolongement de la configuration noyau, il faut augmenter la fréquence maximale de l'horloge Temps Réel (RTC pour *Real Time Clock*) en ajoutant la ligne `dev.rtc.max-user-freq = 1024` au fichier */etc/sysctl.conf*.

Faites en de même pour le HPET (*High Precision Event Timer*, futur remplaçant de la RTC), si votre machine en est pourvue et que vous l'avez intégré au noyau, en ajoutant la ligne `dev.hpet.max-user-freq = 1024`. Par défaut ces valeurs sont à 64, ce qui est trop peu. Vous pouvez même essayer de les pousser jusqu'à 4096 si le coeur vous en dit.

Une fois ces modifications effectuée, faites un `sysctl -p` pour que les nouvelles valeurs soit prises en compte.

Disque

Vous pouvez chercher à optimiser les entrées/sorties de vos disques durs à l'aide d'utilitaires comme *hdparm* (pour de l'IDE), *blktool* ou *sdparm* (pour du *SCSI* ou du *SATA*).

Reportez-vous aux pages de manuel pour la syntaxe spécifique de ces deux outils. De même, il est possible de modifier les options de montage des partitions sur lesquelles les logiciels écrivent et lisent vos fichiers de travail en modifiant */etc/fstab* pour y ajouter l'option noatime.

Cette opération devrait donner une ligne ressemblant à `/dev/sda7 /mao ext3 defaults,errors=remount-ro,noatime 0 1`. Une fois la modification faite il faut soit remonter la partition en tapant `mount -o remount,rw /dev/sda7`, soit redémarrer pour que l'option soit prise en compte.

PCI

Souvent, les priorités par défaut données aux interruptions PCI ne sont pas optimisées pour un usage MAO. Pour lister les bus PCI et les périphériques associés, utilisez lspci. Pour connaître la priorité donnée à chacun, utilisez l'argument `-v` de cette même commande. La sortie ressemble à :

```
[...]
04:01.0 Multimedia audio controller:
      Xilinx Corporation RME
      Hammerfall DSP (rev 98)
   Flags: bus master, medium devsel,
         latency 255, IRQ 17
   Memory at dcdf0000 (32-bit, non-
         prefetchable) [size=64K]
[...]
```

On a le numéro du bus, et le nom du périphérique associé sur la première ligne. La priorité de la latence est indiquée ici sur la seconde ligne : `latency 255`. Attention, il ne s'agit pas d'un temps de latence maximum mais bien d'une *priorité*. Plus cette valeur est élevée et plus la priorité sera haute.

Pour la modifier il faut utiliser la commande setpci. Par exemple `setpci -v -s "*:*" latency_timer=b0` donnera une priorité de 176 à tous les périphériques sur tous les bus, et `setpci -v -s 04:01.0 latency_timer=ff` donnera la priorité maximum au périphérique d'un bus particulier.

La valeur à passer à l'argument `-s` dépend évidemment de votre propre configuration PCI (affichée via *lspci*). La valeur passée à l'argument `latency_timer` est de type hexadécimal.

B0 correspond donc à 176 et *FF* à 255 (la priorité maximum). Vous pouvez partir sur ces valeurs et optimiser par la suite. Il est également tout à fait possible de donner une priorité de 0 aux périphériques les moins importants.

Gestion des droits

Bien souvent, cette notion de gestion des droits associée au Temps Réel est confuse pour les nouveaux arrivants. Certains installent le module *realtime-lsm* en pensant qu'il s'agit du *patch* Temps Réel, d'autres n'installent que le patch Temps Réel sans prendre en compte les droits...

La confusion vient en grande partie du fait qu'il existe un module *realtime-lsm*, désormais obsolète mais tout de même traité ici pour être exhaustif, dont le nom a semé la confusion.

Les choses sont pourtant claires : l'accès au Temps Réel n'étant autorisé qu'à l'utilisateur *root* il faut déléguer son pouvoir dans certaines conditions aux utilisateurs dont les applications en ont besoin.

Le module realtime-lsm

Il s'agit d'un module écrit par Torben Hohn et Jack O'Quin pour permettre à un groupe particulier d'utilisateurs ou bien à tous les groupes d'utiliser les fonctionnalités Temps Réel du noyau Linux. Ce module a été utilisé pendant longtemps, mais à présent il est considéré comme obsolète.

Néanmoins, si vous utilisez une version un peu ancienne de Linux ou bien si votre bibliothèque *PAM* ne gère pas encore la manipulations des *rlimits*, cette méthode est pour vous. Contrairement aux deux autres (décrites plus loin) elle nécessite les fichiers sources du noyau.

Pour commencer, il vous faut récupérer le source. Vous pouvez soit utiliser le *patch* noyau, soit récupérer une archive contenant le module afin de le compiler à part.

C'est cette seconde méthode qui sera expliquée ici. Dans les deux cas il faudra que vous ayez les fichiers sources du noyau installés et configurés, et que vous ayez redémarré sur le nouveau noyau. Puis, entrez :

```
$ cd /usr/src/
$ wget http://ovh.dl.sourceforge.net/
```

Modification du noyau à la volée

Pour faire des tests sur la fréquence de l'horloge *Temps Réel* et du HPET, inutile de passer systématiquement par `sysctl`. Comme pour beaucoup de paramètres de configuration du noyau, vous pouvez modifier les valeurs à la volée en écrivant directement sous */proc* avec un `echo 4096 > /proc/sys/dev/rtc/max-user-freq` ou un `echo 4096 > /proc/sys/dev/hpet/max-user-freq`.

Musique et sécurité ne font pas toujours bon ménage

Tous les réglages indiqués ici sont à revoir en fonction de vos besoins et surtout de votre politique de sécurité. Il est bien connu qu'une machine dédiée à la MAO est souvent loin du respect strict des impératifs en matière de sécurité informatique !

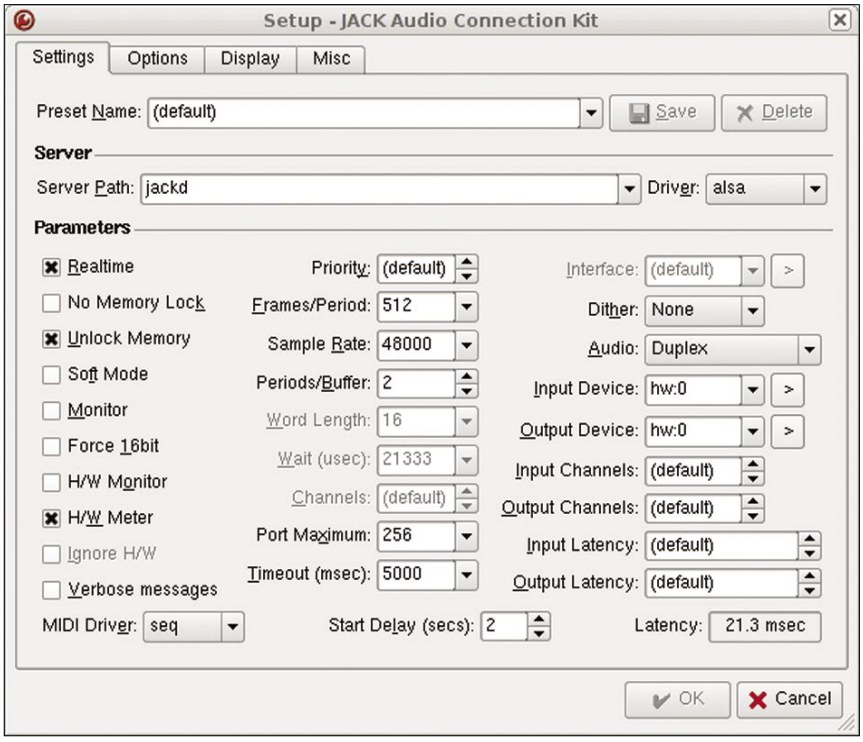

Figure 1. Configuration du noyau

```
      sourceforge/realtime-lsm/
      realtime-lsm-0.8.7.tar.gz
$ tar zxvf realtime-lsm-0.8.7.tar.gz
$ cd realtime-lsm-0.8.7/
$ make
$ su -c "make install"
```

Ensuite, pour charger le module vous pouvez utiliser `modprobe realtime any=1 gid=29 mlock=0`. Cela signifie que tous les programmes (*any*) lancés par des membres du groupe (*gid*) *audio* (*29*) peuvent accéder au mode Temps Réel sans restriction quant à l'utilisation de la mémoire (*mlock*).

Pour automatiser le chargement de ce module au démarrage du système avec les bonnes options cela dépend de votre distribution GNU/Linux. Elle utilise certainement un fichier */etc/modprobe.conf* ou sur les systèmes plus récents un */etc/modprobe.d/*.

Il faudra ajouter une ligne ressemblant à `options realtime any=1 gid=29 mlock=0`.

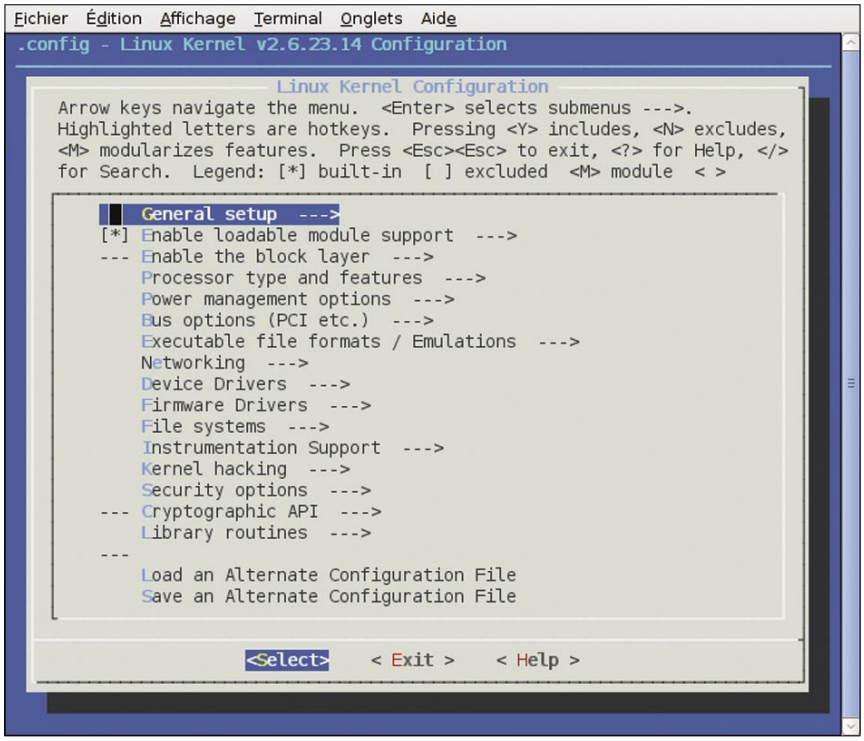

Figure 2. Fenêtre de configuration JACK de QJackCtl

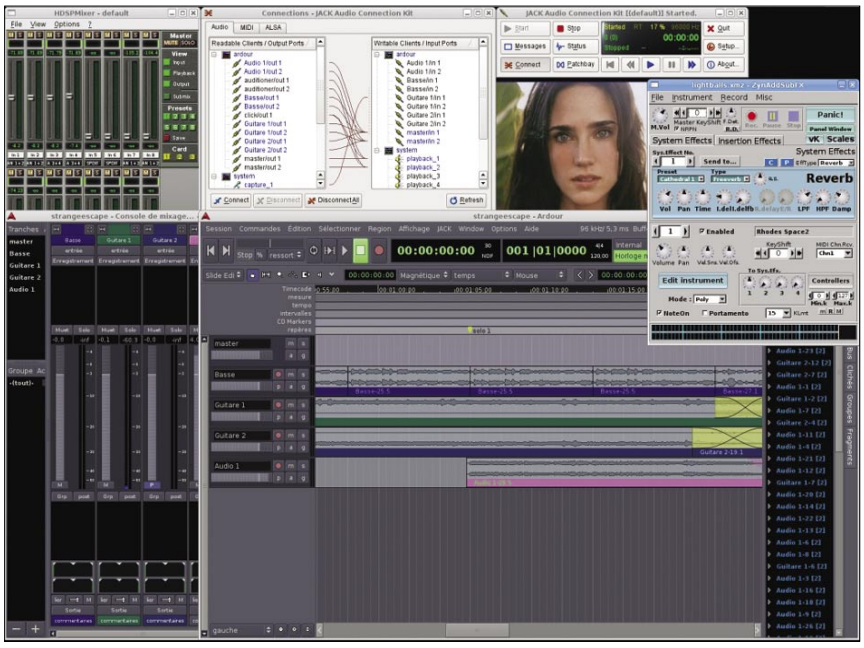

Figure 3. Bureau MAO

L'utilitaire set_rlimits

Il s'agit d'un utilitaire écrit par Jonathan Withe afin de pouvoir manipuler directement les rlimits du noyau.

À vrai dire cet outil n'est utile, comme c'est le cas pour le module *realtime-lsm*, que si votre version de *PAM* est trop ancienne et ne sait pas le faire elle-même. Certains diront que le passage par *PAM* provoque un léger ralentissement. Si vous êtes de ceux-là, utilisez `set_rlimits` !

Cet outil est disponible sur le site de son concepteur et évolue peu. Les opérations à effectuer pour le récupérer et l'installer sont les suivantes :

```
$ cd /usr/src/
$ wget http://www.physics.adelaide.
      edu.au/~jwoithe/set_rlimits-
      1.2.0.tgz
$ tar zxvf set_rlimits-1.2.0.tgz
$ cd set_rlimits-1.2.0/
```

Ensuite, éditez le fichier *Makefile* et changez la ligne `PREFIX=/usr/local` en `PREFIX=/usr` si vous voulez que l'application s'installe avec les autres applications du système sous */usr*. Puis, tapez :

```
$ make clean
$ make
$ su -c "make install"
```

À présent, vous pouvez modifier la configuration dans le fichier */etc/set_rlimits.conf*. Il vous faudra ajouter une ligne par application à autoriser. Si vous voulez autoriser *QjackCtl* par exemple pour tous les membres du groupe audio, ajoutez la ligne `@audio /usr/bin/qjackctl nice=-10 rtprio=100`.

Ensuite, à chaque fois il vous faudra lancer les applications ajoutées dans la configuration en utilisant `set_rlimits -r`. Pour lancer *QjackCtl* cela donnera donc `set_rlimits -r qjackctl`. La page de manuel est très bien faite. N'hésitez pas à vous y reporter.

La bibliothèque PAM

Il s'agit à mon avis de la solution la plus élégante actuellement. Pendant longtemps les deux solutions précédentes ont été privilégiées (surtout celle du *realtime-lsm*), mais à présent que la majeure partie des distributions GNU/Linux proposent une version de la bibliothèque *PAM* capable de manipuler les rlimits du noyau, pourquoi s'en priver ?

La configuration est simple et elle ressemble à celle de `set_rlimits` tout en

étant plus complète. Il faut éditer le fichier */etc/security/limits.conf*. Pour autoriser par exemple tous les membres du groupe audio, ajoutez-y les lignes suivantes :

```
@audio - rtprio 100
@audio - nice -10
@audio - memlock unlimited
```

Un autre intérêt de *PAM* par rapport à `set_rlimits` est que la gestion se fait de manière complètement transparente pour l'utilisateur. Vous lancez simplement le programme, et *PAM* se charge du reste.

Le serveur de sons

Comme vous l'aviez vu dans l'article *Composition musicale sous GNU/Linux*, le serveur de son est une brique importante qui permet de relier entre eux les logiciels de production et de capture des sons. C'est ce que fait le serveur *JACK* (*Jack Audio Connection Kit*), et c'est pourquoi il doit pouvoir utiliser les capacités Temps Réel du noyau pour router les événements en temps voulu.

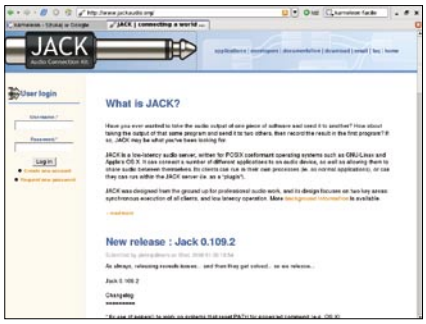

Figure 4. Site de Jack

Pour récupérer et installer *JACK* lancez les commandes du Listing 4.

Le frontal graphique le plus abouti pour simplifier l'utilisation de *JACK* est *QjackCtl*. Cet outil permet le routage graphique des sons et dispose d'une console pour conduire en toute simplicité les applications compatibles avec *JACK*. Vous allez voir dans ce chapitre comment configurer *JACK* via l'interface de *QjackCtl*.

Pour récupérer et installer *QjackCtl* rapportez-vous au Listing 5.

Ensuite, la configuration de base est simple. Les réglages en place sur la copie d'écran de la fenêtre de configuration de *QjackCtl* devraient correspondre à la plupart des cartes. C'est à vous de les affiner en fonctions des spécifications de la vôtre. En fonction de votre carte, il faudra essayer de baisser la valeur de l'option *Frames/Period* ou d'augmenter la valeur de *Sample Rate*.

Certaines cartes ne supportent qu'un échantillonnage de 48000 Hz alors que d'autres peuvent aller jusqu'à 192000 Hz. Il en va de même avec la latence maximale.

Si vous avez plusieurs cartes sons installées sur votre système, utilisez les options *Input Device* et *Output Device* pour sélectionner la carte qui servira à la capture et celle qui servira au rendu (*playback*) des sons. Pour ne faire que de la capture ou que du rendu vous pouvez optimiser les traitements en adaptant l'option *Audio*.

Testez différents réglages ; tout en sachant que la chose à éviter par dessus tout est ce qu'on appelle les *xruns*. Un xrun se produit lorsque le traitement d'une information a dépassé la latence maximale indiquée en bas à droite de la fenêtre de configuration. Il est probable que vous en aurez de temps en temps (Temps Réel *mou* oblige) ; mais si vous en avez régulièrement vous commencerez à entendre des craquements durant l'écoute, puis le traitement des sons finira par se dégrader complètement.

Si trop de *xruns* se produisent, augmentez la valeur *Frames/Period*, baissez la fréquence de l'échantillonnage ou l'option *Periods/Buffer*. Si les xruns continuent, vérifiez bien votre configuration *Temps Réel* et les options de compilation de *JACK*. Évitez d'avoir trop de programmes en cours d'exécution autres que les logiciels dont vous vous servez pour composer. Et si vous ne vous en sortez toujours pas, changez de matériel !

Conclusion

Comme vous avez pu le constater par votre propre expérience, adapter un système pour faire de la MAO n'est pas du tout un repos agréable. Ceci-dit, malgré pas mal de tâtonnements et d'efforts on arrive toujours à un résultat acceptable. Et si d'aventure ce travail préparatoire vous rebutait vous pourriez encore vous orienter vers une distribution spécialisée, comme *64 Studio*, *Unbuntu Studio* ou encore *Planet CCRMA*.

L'intérêt est évident de prime abord. Mais il ne faut pas se faire trop d'illusions : que ce soit pour un problème relatif à votre matériel, ou bien pour tester la toute dernière version de développement d'un logiciel dont vous ne pouvez ou ne pourrez plus vous passer, viendra un moment où vous devrez mettre le nez sous le capot ; donc autant savoir ce qui s'y trame.

Quelle que soit la solution vers laquelle vous vous orientez, n'oubliez pas de garder toujours à l'esprit qu'en MAO, comme dans beaucoup d'autres domaines, il ne faut pas hésiter à tester et tester encore des configurations, du matériel et des logiciels...

Et un beau jour, tout s'éclaire !

À propos de l'auteur

Emmanuel Saracoo travaille depuis plus de six ans en tant que de chef de projet informatique pour la SSLL Easter-eggs. Il compose activement avec des outils libres sous GNU/Linux depuis plus de trois ans et développe en parallèle plusieurs logiciels libres, dont gURLChecker, gospy-applet, wbmclamav, wbmtranslator et phpRemoteShell.
Courriel : *esaracco@free.fr*
Site : *http://esaracco.free.fr/*

Sur le réseau

- Site de JACK : *http://jackaudio.org/,*
- Site de QjackCtl : *http://qjackctl.sourceforge.net/,*
- Site du noyau Linux : *http://www.kernel.org/,*
- Patch Temps Réel : *http://www.kernel.org/pub/linux/kernel/projects/rt/,*
- Module realtime-lsm : *http://sourceforge.net/projects/realtime-lsm/,*
- *Utilitaire set_rlimits : http://www.physics.adelaide.edu.au/~jwoithe/,*
- Distribution Planet CCRMA : *http://ccrma.stanford.edu/planetccrma/software/,*
- Distribution 64 Studio : *http://www.64studio.com/,*
- Distribution Ubuntu Studio : *http://ubuntustudio.org/,*
- Liste de diffusions Linux Audio User List : *http://music.columbia.edu/mailman/listinfo/linux-audio-user/*