Guide pratique du Plug-and-Play

Adaptation française du Plug-and-Play HOWTO

David S. Lawyer

<dave CHEZ lafn POINT org>

Guillaume Lelarge

<gleu CHEZ wanadoo POINT fr>

Traduction française

Jean-Philippe Guérard

<fevrier CHEZ tigreraye POINT org>

Préparation de la publication de la v.f.

Version : 1.14.fr.0.9

2006-03-05

Historique des versions		
Version 1.14.fr.0.9	2006-03-07	GL, JPG
Version française mise à jour mais non relue. Prise en compte des corrections suggérées par Jean-Philippe Guérard et par Bernard Adrian.		
Version 1.14	2006-02-??	DSL
Version 1.13.fr.0.9	2005-08-22	GL, JPG
Version française non relue. Prise en compte des corrections suggérées par Black Myst.		
Version 1.13	2005-08-??	DSL
Version 1.12	2005-03-??	DSL
Version 1.11.fr.0.9	2004-11-20	GL, JPG
Version 1.11	2004-11-??	DSL
Version 1.10.fr.0.9	2004-09-11	GL, JPG
Version 1.10	2004-08-??	DSL
Version 1.09	2004-08-??	DSL
Version 1.08.fr.0.9	2004-07-22	GL
Version 1.08	2004-06-??	DSL
Version 1.07.fr.0.9	2003-09-27	GL
Version 1.07	2002-08-??	DSL
Version 1.06.fr.0.9	2003-05-08	GL
Version 1.06	2002-09-??	DSL

Résumé

Le Plug-and-Play (ou PnP) est un système de détection et de configuration automatique du matériel PC. Ce guide pratique présente les différentes ressources gérées par le Plug-and-Play, telles que les adresses, les interruptions, et cætera. Ce guide couvre le bus PCI (qui a été conçu dès le départ pour le Plug-and-Play) et l'utilisation du Plug-and-Play sur l'ancien bus ISA. Si le Plug-and-Play faisait son travail correctement, vous n'auriez pas besoin de ce guide pratique. Dans le cas contraire, il vous expliquera le Plug-and-Play et vous aidera à détecter votre matériel et à lui affecter des ressources (configuration de bas niveau). Il ne couvre pas le UPnP (*Universal Plug and Play*).

Table des matières

Introduction

Droits d’utilisation, avertissements et remerciements

Copyright

> **Important**
>
> Le texte ci-dessous est la licence de ce document. Ce texte fait foi. Il est composé de la licence (en anglais) du document original, suivi de la licence (en français) de sa traduction.

Please freely copy and distribute (sell or give away) this document in any format. Send any corrections and comments to the document maintainer. You may create a derivative work and distribute it provided that you:

- If it's not a translation: Email a copy of your derivative work (in a format LDP accepts) to the author(s) and maintainer (could be the same person). If you don't get a response then email the LDP (Linux Documentation Project): <submit CHEZ en POINT tldp POINT org>.
- License the derivative work in the spirit of this license or use GPL. Include a copyright notice and at least a pointer to the license used.
- Give due credit to previous authors and major contributors.

If you're considering making a derived work other than a translation, it's requested that you discuss your plans with the current maintainer.

Droits d'utilisation

Vous pouvez copier et distribuer librement (vendre ou donner) ce document quel que soit son format. Envoyez toute correction ou tout commentaire à l'auteur du document. Vous pouvez créer et distribuer une version dérivée à condition que :

- vous envoyez au gestionnaire et à l'auteur (s'il ne s'agit pas de la même personne) une copie de votre travail par courrier électronique (s'il ne s'agit pas d'une traduction) dans un format que le LDP accepte. Si vous n'obtenez pas de réponse, alors envoyez votre message au LDP (Linux Documentation Project) : <submit CHEZ en POINT tldp POINT org> ;
- la licence utilisée pour votre travail soit dans le même esprit que cette licence ou utilise la licence GPL. Incluez les informations de droit d'utilisation ou au moins un pointeur vers la licence utilisée ;
- vous donnez crédit aux auteurs précédents et aux contributeurs majeurs.

Si vous pensez faire un travail dérivé autre qu'une traduction, vous devez en discuter avec le gestionnaire actuel.

Avertissements

Bien que je n'aie pas essayé de vous tromper intentionnellement, il est probable que des erreurs restent dans ce document. Faites-le moi savoir si vous en trouvez. Comme il s'agit d'une documentation libre, il est évident que je ne peux pas être tenu responsable des erreurs contenues dans ce texte.

Marques déposées

Tout nom de marque (avec une majuscule initiale tel que MS Windows) sera supposé être une marque déposée. Elles appartiennent à leur propriétaires respectifs.

Remerciements

- Daniel Scott a relu ce document en mars 2000 et y a trouvé de nombreuses erreurs ;
- Pete Barrett a donné une astuce pour empêcher Windows de réinitialiser les IRQ PCI ;
- Août 2004 : Ross Boylan a trouvé des erreurs et a indiqué un manque de clarté dans la partie où on indique au BIOS qu'il s'agit d'un système d'exploitation PnP.

Plans futurs : vous pouvez aider

Merci de m'indiquer toute erreur, que ce soit dans les faits, les opinions, la logique, l'orthographe, la grammaire, la clarté, les liens, et cætera. Mais tout d'abord, si la date du document est vieille de plus de quelques mois, vérifiez que vous possédez bien la dernière version. Envoyez-moi toute information que vous trouvez intéressante pour ce document.

Je n'ai pas encore étudié le code utilisé par les différents pilotes Linux et par le noyau pour implémenter le Plug-and-Play. Mais, j'ai commencé à jeter un ½il, notamment aux commentaires. Donc, ce guide pratique est encore incomplet. Il devrait expliquer plus en profondeur le « hot swapping », le « hot-plug » et le nouveau logiciel PnP du noyau 2.6. Et il ne couvre pas le firewire. Il comporte certainement quelques erreurs (faites-moi savoir où je me trompe). Dans ce guide pratique, j'ai utilisé deux points d'interrogation (??) pour signaler que je n'ai pas vraiment la réponse.

Nouvelles versions de ce guide pratique

De nouvelles versions du guide pratique Plug-and-Play devraient apparaître fréquemment et seront disponibles à la navigation et en téléchargement sur les sites miroirs du LDP. Pour une liste des sites miroirs, voir http://www.tldp.org/mirrors.html. Différents formats sont disponibles. Si vous voulez seulement vérifier rapidement la date de la dernière version, regardez sur http://www.traduc.org/docs/howto/lecture/Plug-and-Play-HOWTO.html. La version que vous lisez actuellement est la traduction française de la v1.14, de mars 2006.

Nouveautés des dernières versions

Pour un historique des révisions complet, voir le fichier source (dans son format DocBook SGML) sur http://cvsview.tldp.org/index.cgi/LDP/howto/linuxdoc/Text-Terminal-HOWTO.sgml en cliquant sur text ou voir le fichier source (dans son format linuxdoc) sur http://www.ibiblio.org/pub/linux/docs/HOWTO/other-formats/sgml/Plug-and-Play-HOWTO.sgml.gz.

- v1.14, Février 2006 : Révision de « Comment Linux gère-t'il le PnP » ; LPC devait être configuré par le BIOS. Balance des IRQ. Linux peut trouver des pilotes pour les périphériques détectés.
- v1.13, Juillet 2005 : Conflit d'IRQ. Plus grande clarté dans les descriptions des ressources. /proc/bus. Espace de configuration PCI accédé via l'espace d'adressage des entrées/sorties. Plus d'outils de détection de matériel. Message d'erreur *Can't allocate region*.
- v1.12, Mars 2005 : /dev/eth0 n'existe plus. Informations modifiées dans /sys et /proc depuis le noyau 2.6. L'espace d'adressage de la configuration PCI est « géographique ». scanpci pourrait trouver un périphérique que lspci ne voit pas. Le noyau peut affecter des adresses au démarrage.

- v1.11, Novembre 2004 : bus LPC. Erreur de format (texte dans le contenu).
- v1.10, Août 2004 : sysfs pas encore aussi utile que clamé, informations sur les cartes obsolètes déplacées dans la section « Historique ».
- v1.09, Août 2004 : Modifié suivant le courrier électronique de Ross Boylan (ross CHEZ biostat POINT ucsf POINT edu). Plus de clarté dans l'indication fournie au BIOS (système d'exploitation PnP). `/proc/isapnp` couvert. Erreurs corrigées. sysfs dans le noyau 2.6.
- v1.08, Juin 2004 : Modification de tout, excepté les annexes, pour plus de clarté et pour corriger quelques indications obsolètes suite à l'amélioration des capacités PnP de Linux. Ajout d'une petite information sur le « hot-plug » et sur l'USB.

Introduction générale. Avez-vous besoin de ce guide pratique ?

Le Plug-and-Play (PnP) est un système détectant automatiquement les périphériques tels que les disques, les cartes son, les cartes réseau, les modems, et cætera. Il trouve tous les périphériques du bus PCI et tous les périphériques supportant PnP sur l'ancien bus ISA. Avant PnP, beaucoup de périphériques étaient automatiquement recherchés en utilisant des méthodes non PnP mais n'étaient pas trouvés quelque fois. PnP fournit un moyen de trouver tous les périphériques supportant PnP. Il réalise aussi une configuration de bas niveau de ces périphériques. Les périphériques non PnP (et les périphériques PnP ayant été correctement configurés avec PnP) peuvent souvent être détectés par des méthodes n'utilisant pas PnP. Le bus PCI a été conçu pour le PnP alors que le vieux bus ISA ne l'a pas été à ses origines mais son support lui a été ajouté plus tard. Donc, souvent, PnP est utilisé pour signifier uniquement PnP pour l'ancien bus ISA. Par exemple, lorsque vous apercevez au démarrage des messages d'isapnp du style : « Plug & Play device », cela signifie seulement qu'il y a un périphérique ISA PnP. Dans ce guide pratique, nous traitons à la fois le PnP du bus ISA et du bus PCI.

Avec le temps, le noyau Linux améliore son support de PnP. À la fin du 20è siècle, on pouvait dire que Linux n'était pas un système d'exploitation PnP. Mais il est dit qu'avec la version 2.6 du noyau, Linux est maintenant totalement compatible avec PnP (en supposant que le noyau est construit avec le support de PnP). Bien que le système PnP n'est pas centralisé comme cela peut l'être avec MS Windows (et son registre), le PnP Linux décentralisé semble fonctionner correctement.

Linux conserve la trace des affectations de ressources demandées par les pilotes de périphériques et refuse toute requête s'il estime que cela causerait un conflit. Le noyau fournit aussi des programmes que les pilotes de périphériques peuvent appeler pour effectuer leurs opérations de plug-and-play. Le noyau lit aussi tous les registres de configuration de tous les périphériques PnP et maintient leur tables que les pilotes de périphériques peuvent consulter. Cette table aide les pilotes à trouver leur matériel. Le noyau 2.6 fournit aussi un meilleur support pour le « hot-plug ».

Le BIOS de votre PC travaillera certainement un peu pour le plug-and-play. Donc, si tout fonctionne en harmonie avec PnP, vous pouvez utiliser votre ordinateur sans avoir besoin de tout connaître sur le plug-and-play. Mais, si certains périphériques supportés par Linux ne fonctionnent pas (parce qu'ils n'ont pas été repérés par Linux ou parce qu'ils ont été mal configurés), vous aurez besoin de lire au moins une partie de ce guide pratique. Vous en apprendrez sur le PnP mais aussi sur la façon dont la communication prend place dans un ordinateur. Si vous avez un ordinateur moderne avec un bus PCI mais sans bus ISA, vous pouvez passer les parties sur le bus ISA.

Si vous avez des problèmes avec un périphérique, regardez les messages affichés lors du démarrage (remontez la liste avec **Shift**-**PageUp**). Si cela n'affiche pas aussi les messages du lancement, à partir du BIOS, utilisez la touche Pause (voir la section intitulée « Messages de démarrage » [p 34]).

Vérifiez que vous utilisez le bon pilote pour un périphérique et que ce pilote est trouvé et utilisé. Si le pilote est un module, tapez **insmod** (en tant qu'utilisateur privilégié) pour voir s'il est chargé (et utilisé). Si ce n'est pas un module, alors il doit être intégré au noyau.

Ce guide pratique ne couvre pas le problème de la recherche et de l'installation de pilotes de périphériques. Peut-être qu'il devrait. Un des problèmes possibles est qu'une carte (ou un autre périphérique physique) peut ne pas dire le type de composants qu'elle utilise. Le nom du pilote correspond souvent au nom de la puce et non pas au nom de la marque. Une façon de commencer la vérification du pilote est de regarder s'il en est question dans la documentation du noyau, dans un autre guide pratique ou plus généralement sur Internet. Attention : de telles documentations peuvent ne plus être à jour.

Les ordinateurs disposant de bus PCI (et sans bus ISA) ont significativement réduit le nombre de points posant problèmes. Concernant le bus ISA et le manque de support au niveau noyau pour l'ISA PnP (avant le noyau 2.4), beaucoup de choses posaient problèmes. Rappelez-vous que, parfois les problèmes semblant être relatifs à PnP sont dûs en réalité à un matériel défectueux ou à un matériel non conforme aux spécifications PnP.

Ce que PnP doit faire : allouer des « ressources bus »

En quoi consiste le Plug-and-Play (PnP) ?

Si vous ne comprenez pas cette section, lisez Périphériques matériels et la communication avec ces derniers [p 9] .

En simplifiant à l'extrême, Plug-and-Play indique aux pilotes de périphériques où trouver les différents matériels (périphériques) tels que modems, cartes réseau, cartes son, et cætera. La tâche du Plug-and-Play est de faire correspondre les périphériques physiques avec les logiciels (pilotes de périphériques) qui les font fonctionner, et d'établir des canaux de communication entre chaque périphérique physique et son pilote. Pour ce faire, PnP alloue les « ressources bus » suivantes aux matériels : adresses d'entrée/sortie, plages mémoire, IRQ, canaux DMA (uniquement pour les bus LPC et ISA). Ces quatre dernières sont parfois appelées des ressources de premier ordre ou simplement des ressources. PnP maintient un enregistrement de ce qu'il fait et autorise l'accès à ces informations aux pilotes de périphériques. Si vous ne comprenez pas ce que sont ces quatre ressources bus, lisez les sous-sections suivantes de ce guide pratique : Adresses d'entrée/sortie, IRQ, Canaux DMA, Régions mémoire. Un article de la Linux Gazette parle de trois des ressources bus. Il est disponible sur Introduction aux IRQ, DMA et adresses de base (NdT : une traduction française est disponible sur traduc.org). Une fois que ces ressources bus ont été assignées (et si le bon pilote est installé), le pilote actuel et ses « fichiers » du répertoire `/dev` sont prêt à être utilisés.

Cette méthode d'affectation PnP des ressources bus est parfois appelée « configuration » mais il s'agit seulement d'une configuration bas-niveau. Le répertoire `/etc` comprend beaucoup de fichiers de configuration mais un grand nombre d'entre eux ne concernent pas la configuration de PnP. Donc, la grande majorité des configurations de périphériques physiques n'a rien à voir avec PnP ou les ressources bus. Par exemple, l'initialisation d'un modem par une phrase d'initialisation ou la configuration de sa vitesse ne concerne pas PnP. Donc lorsque nous parlons de PnP, la configuration ne concerne qu'un seul type de configuration. Alors que d'autres documentations (telles que celles pour MS Windows) appellent les ressources bus « ressources », j'ai utilisé le terme de ressources bus au lieu du simple « ressources » pour les distinguer des nombreux autres types de ressources.

PnP est un long processus réalisé par différents logiciels et matériels. Si seul un programme gérait PnP sur Linux, cela serait simple. Mais, avec Linux, chaque pilote de périphérique fait son propre PnP en utilisant le logiciel fourni par le noyau. Le matériel du BIOS du PC utilise PnP au démarrage du PC. Et il y a bien plus que cela.

Périphériques matériels et la communication avec ces derniers

Un ordinateur est composé d'un processeur (CPU) réalisant les opérations et de mémoire vive (RAM) pour stocker les programmes et les données (en accès rapide). De plus, il existe de nombreux périphériques tels que différents types de disques, une carte vidéo, un clavier, des périphériques réseaux, des cartes modem, des périphériques sons, le bus USB, les ports séries et parallèles, et cætera. Dans les anciens temps, la plupart des périphériques étaient des cartes insérées dans des emplacements du PC. Aujourd'hui, beaucoup de périphériques, qui étaient auparavant des cartes, sont maintenant compris dans les composants intégrés à la carte-mère. On trouve aussi une alimentation apportant l'électricité, différents bus sur la carte mère pour connecter les périphériques au CPU et une boîte pour contenir le tout.

Les cartes se connectant sur la carte mère peuvent contenir plus d'un périphérique. Les cartes mémoires sont quelque fois considérées comme des périphériques mais ils ne sont pas Plug-and-Play si on suit le sens utilisé dans ce guide pratique.

Pour que l'ordinateur fonctionne bien, chaque périphérique doit être sous le contrôle de son « pilote ». Il s'agit d'un logiciel faisant partie du système d'exploitation, pouvant être chargé en tant que module et exécuté à partir du CPU. Les pilotes de périphériques sont associés à des « fichiers spéciaux » rangés dans le répertoire `/dev`, bien qu'ils ne soient pas vraiment des fichiers. Ils ont des noms tels que hda3 (troisième partition du disque a), ttyS1 (deuxième port série), eth0 (première carte Ethernet), et cætera.

Le périphérique eth0 est celui de la carte ethernet (carte réseau). Auparavant, il s'agissait de /dev/eth0 mais c'est maintenant un périphérique virtuel du noyau. Ce que eth0 référence dépend du type de carte ethernet que vous avez. Si le pilote est un module, cette affectation se trouve probablement dans une table interne du noyau mais pourrait aussi se trouver dans `/etc/modules.conf` (appelé « alias »). Par exemple, si vous disposez d'une carte ethernet utilisant le composant « tulip », vous devez placer la ligne « alias eth0 tulip » dans `/etc/modules.conf` pour que, lorsque votre PC cherche eth0, il trouve le pilote tulip. Néanmoins, les noyaux modernes peuvent généralement trouver le bon module du périphérique. De cette façon, vous n'aurez pas à le spécifier vous-même.

Pour contrôler un périphérique, le CPU (sous le contrôle du pilote de périphérique) envoie des commandes et des données du périphérique. Il reçoit aussi l'état et des données des différents périphériques. Pour cela, chaque pilote doit connaître l'adresse du périphérique qu'il doit contrôler. Connaître cette adresse revient à mettre en place un canal de communication, même si le « canal » physique se trouve être le bus de données interne au PC, bus partagé avec beaucoup d'autres périphériques.

Ce canal de communication est en fait un peu plus complexe que ce qui est décrit ci-dessus. Une « adresse » est en fait une plage d'adresses, ce qui fait que, parfois, le mot « plage » est utilisé à la place du mot « adresse ». Il peut exister plus d'une plage (sans recoupement) pour un seul périphérique. Il existe aussi un autre canal connu sous le nom d'interruptions permettant au périphérique d’envoyer une requête de « demande d’aide » urgente à leur pilote.

Adresses

Le bus PCI a trois plages d’adressage : les entrées/sorties, la mémoire principale (mémoire d’entrées/sorties) et la configuration. L’ancien bus ISA ne dispose pas de cette dernière. Seules les entrées/sorties et la mémoire principale sont utilisées pour les entrées/sorties du périphérique. Les adresses de configuration sont fixes et ne peuvent pas être modifiées donc elles n’ont pas besoin d’être affectées. Pour plus d’informations, voir Espace d’adressage de la configuration PCI [p ??] .

Quand le CPU veut accéder à un périphérique, il place l’adresse du périphérique sur un bus important de l’ordinateur (pour le PCI : le bus d’adresse/données). Tous les types d’adresses (tels que les entrées/sorties et la mémoire principale) partagent le même bus sur le PC. Mais la présence ou l’absence de voltage sur certains fils dédiés sur le bus du PC indique l’espace occupée par une adresse : entrée/sortie, mémoire principale (voir Espaces d’adressage [p 11]) ou la configuration (seulement PCI). Ceci est un peu trop simplifié car indiquer à un périphérique PCI qu’il s’agit d’une adresse de configuration est réellement plus complexe que la description ci-dessus. Voir Espace d’adressage pour la configuration PCI [p ??] pour plus d’informations. Voir Détails des adresses [p 45] pour plus d’informations sur l’adressage en général.

Les adresses d’un périphérique sont stockées dans les registres du matériel physique. Elles peuvent être changées par logiciel et elles peuvent être désactivées pour que le périphérique ne soit plus adressable, sauf en ce qui concerne les adresses de configuration du PCI qui ne peuvent être ni modifiées ni désactivées.

Adresses d’entrées/sorties (principes relatifs à d’autres ressources)

Les périphériques étaient originellement situés dans la plage d’adresses des entrées/sorties mais, aujourd’hui, ils peuvent utiliser la plage en mémoire principale. Une adresse d’entrée/sortie est quelque fois simplement appelée « I/O », « IO », « i/o » ou « io ». Les termes « port I/O » ou « plage d’entrées/sorties » sont aussi utilisés. Ne confondez pas ces ports d’entrées/sorties avec la mémoire d’entrées/sorties située en mémoire principale. Il existe deux étapes principales pour allouer des adresses I/O (ou d’autres ressources bus telles que les interruptions sur le bus ISA) :

- Initialiser l’adresse I/O, et cætera. dans le matériel (dans un de ses registres),
- Laisser son pilote de périphérique reconnaître l’adresse I/O, et cætera.

Souvent, le pilote du périphérique fait les deux (en quelque sorte). Le pilote de périphérique n’a pas réellement besoin d’initialiser une adresse d’entrée/sortie s’il découvre que l’adresse a déjà été initialisée (peut-être par le BIOS) et qu’il souhaite accepter cette adresse. Une fois que le pilote a soit trouvé une adresse précédemment configurée soit configuré l’adresse lui-même, alors il sait évidemment de quelle adresse il s’agit, donc il n’est pas nécessaire de lui fournir cette information -- il la connaît déjà.

Le processus en deux étapes ci-dessus (1. configurer l’adresse au niveau matériel. 2. mettre au courant le pilote.) ressemble aux deux parties d’un problème pour trouver le numéro de la maison d’une personne dans la rue. Quelqu’un doit installer un numéro sur l’entrée de la maison pour qu’elle puisse être trouvée, puis les personnes qui souhaitent aller à cette adresse doivent obtenir (et conserver) ce numéro pour qu’elles puissent trouver la maison. En informatique, le matériel du périphérique doit d’abord obtenir son adresse, qu’il placera dans un registre matériel spécial (ce qui revient à conserver le numéro dans notre exemple), puis le pilote de périphérique doit obtenir cette adresse (écrire ce numéro dans son carnet d'adresses). Les deux doivent être réalisés, soit automatiquement par le logiciel soit en saisissant manuellement des données dans des fichiers de configuration. Des problèmes pourraient survenir si seulement un des deux est fait correctement.

Pour la configuration manuelle de PnP, des personnes font l'erreur de ne faire qu'une de ces deux étapes et se demandent ensuite pourquoi l'ordinateur ne peut pas trouver le périphérique. Par exemple, ils utilisent **setserial** pour associer une adresse au port série sans réaliser que ceci ne fait que donner une adresse au pilote. Cela n'enregistre pas cette adresse au niveau du port série physique. Si vous avez dit une bêtise au port série, alors vous avez un problème. Une autre façon de parler au pilote est de donner l'adresse comme option au module du noyau (pilote de périphérique). Si ce que vous dites est faux, il peut y avoir des problèmes. Un pilote intelligent pourrait détecter comment le matériel est réellement configuré et rejeter les mauvaises informations fournies par l'option (ou au moins afficher un message d'erreur).

Un pré-requis évident est que le périphérique physique (comme une carte) doit connaître son adresse avant le pilote du périphérique. Comme les pilotes de périphérique se lancent souvent tout de suite après le démarrage de l'ordinateur, ils peuvent tenter d'accéder à une carte (pour vérifier sa présence, et cætera) avant que l'adresse ne soit enregistrée au niveau de la carte par le programme de configuration PnP. Et vous verrez un message d'erreur indiquant l'absence de la carte même si elle est bien dans le PC (mais n'a pas encore obtenue son adresse).

Ce qui a été dit dans les derniers paragraphes concernant les adresses I/O s'applique de la même manière à la majorité des autres ressources bus : la section intitulée « Plages mémoire » [p 11] , la section intitulée « IRQ - un aperçu » [p 12] et la section intitulée « DMA (accès direct à la mémoire) ou maîtrise du bus » [p 13] . Les trois prochaines sections expliqueront ce qu'ils sont. La seule exception est que les interruptions du bus PCI ne sont pas données par les registres d'une carte mais elles sont plutôt déroutées vers les IRQ par un composant de la carte mère. Ensuite, l'IRQ utilisée par cette carte PCI est inscrite dans un registre de la carte dans un but unique d'informer.

Pour voir quelles adresses d'entrées/sorties sont utilisées sur votre PC, regardez dans le fichier `/proc/ioports`.

Plages mémoire

Beaucoup de périphériques disposent d'une plage mémoire en mémoire principale. C'est quelquefois appelé « mémoire partagée » ou « mémoire d'entrées/sorties ». Cette mémoire est située physiquement dans le périphérique physique mais l'ordinateur y accède comme s'il accédait à des composants mémoire. En parlant de ressources bus, c'est souvent simplement appelé « mémoire », « mem », voire « iomem ». En plus de l'utilisation de cette « mémoire », un tel périphérique peut aussi utiliser une plage mémoire conventionnelle d'entrées/sorties. Pour connaître la mémoire utilisée sur votre ordinateur, cherchez dans `/proc/iomem`. Ce « fichier » inclut la mémoire utilisée par les composants mémoire habituels de la RAM, donc il affiche l'allocation mémoire en général et pas seulement l'allocation iomem. Si vous apercevez un numéro étrange au lieu d'un nom, c'est probablement le numéro d'un périphérique PCI, ce que vous pouvez vérifier en exécutant « lspci ».

Lorsque vous insérez une carte utilisant iomem, vous êtes aussi en train d'insérer un module mémoire pour la mémoire principale. Une adresse haute est sélectionnée pour lui par PnP de façon à ce que cela ne rentre pas en conflit avec les modules de la mémoire principale (composants). Cette mémoire peut être de la mémoire morte (ROM ou Read Only Memory) ou de la mémoire partagée. Cette dernière est partagée entre le périphérique et le CPU (ayant lancé le pilote du périphérique) de la même façon que la plage d'adresses d'entrées/sorties est partagée entre le périphérique et le CPU. Cette mémoire partagée sert en tant que moyen de « transfert » de données entre le périphérique et la mémoire principale. C'est de l'entrée/sortie mais ce n'est pas fait dans la plage d'adresses des entrées/sorties. La carte et le pilote doivent connaître la plage d'adresses.

La ROM (*Read-Only Memory*, soit mémoire en lecture seule) est un genre différent d'iomem. C'est plutôt un programme (parfois un pilote de périphérique), utilisé avec ce périphérique. Cela peut aussi être un code d'initialisation malgré que le pilote soit encore nécessaire. Avec un peu de chance, il fonctionnera aussi sous Linux, et pas seulement sous MS Windows. Elle peut être copiée en mémoire principale pour fonctionner plus rapidement. Mais dans ce cas, elle n'est plus « en lecture seule ».

IRQ - un aperçu

Après avoir lu ceci, vous pouvez lire la section intitulée « Détails sur les interruptions » [p 48] pour bien plus de détails. Ce qui suit est volontairement simplifié : en plus des adresses, il existe aussi un numéro d'interruption à gérer (tel que l'IRQ 5). Cela s'appelle un numéro d'IRQ (Interrupt ReQuest, ou demande d'interruption), ou plus simplement une « irq ». Nous avons déjà mentionné ci-dessus que le pilote de périphérique doit connaître l'adresse d'une carte pour être capable de communiquer avec elle.

Mais qu'en est-il de la communication en sens inverse ? Supposez que le périphérique ait besoin de dire quelque chose à son pilote immédiatement. Par exemple, le périphérique peut recevoir un grand nombre d'octets destinés à la mémoire principale et le tampon utilisé pour stocker ces octets est pratiquement plein. Du coup, le périphérique a besoin de demander à son pilote de récupérer ces octets avant que le tampon ne se voit dépassé par le flot continu d'octets. Un autre exemple serait de signaler au pilote que le périphérique a terminé d'envoyer un ensemble d'octets et qu'il attend maintenant de nouveaux octets à envoyer.

Comment le périphérique peut-il envoyer rapidement un signal à son pilote ? Il peut ne pas être capable d'utiliser le bus de données principal car il y a des chances qu'il soit déjà utilisé. Au lieu de cela, il place un voltage sur un fil d'interruption dédié (aussi appelé ligne ou trace) qui est souvent réservé pour ce seul périphérique. Ce signal est appelé une demande d'interruption (IRQ) ou plus simplement une « interruption ». Il existe l'équivalent de 16 (ou 24, et cætera.) fils de ce type dans un PC et chaque fil amène (indirectement) à un certain pilote de périphérique. Chaque fil a un numéro d'IRQ unique. Le périphérique doit placer son interruption sur le bon fil. Le fil sur lequel le périphérique envoie ces demandes d'aide est déterminé par le numéro d'IRQ enregistré dans le périphérique. Ce même numéro d'IRQ doit être connu par le pilote du périphérique pour que celui-ci sache quelle interruption écouter.

Une fois que le pilote reçoit l'interruption du périphérique, il doit trouver pourquoi cette interruption a été générée et agir de manière appropriée pour régler le problème. Sur le bus ISA, chaque périphérique a habituellement besoin de son propre numéro unique d'IRQ. Pour le bus PCI et dans d'autres cas spéciaux, le partage d'IRQ est autorisé (deux périphériques PCI, ou plus, pourraient avoir le même numéro d'IRQ). De même, pour le PCI, chaque périphérique PCI a un fil fixe *PCI Interrupt*. Mais un composant de routage programmable fait correspondre les fils PCI aux interruptions de type ISA. Voir la section intitulée « Détails sur les interruptions » [p 48] pour savoir comment cela fonctionne.

DMA (accès direct à la mémoire) ou maîtrise du bus

L'accès direct à la mémoire (DMA, acronyme de *Direct Memory Access*) est ce qui permet à un périphérique de prendre la main sur le bus principal de l'ordinateur et de transférer directement des octets vers la mémoire principale ou vers d'autres périphériques. Normalement, le processeur s'occupe des transferts d'un périphérique vers la mémoire principale par un processus en deux étapes :

- lire un ensemble d'octets à partir de la page mémoire d'entrées/sorties et les stocker dans le CPU lui-même ;
- écrire ces octets dans la mémoire principale.

Avec le DMA, il s'agit d'un processus en une seule étape consistant en l'envoi des octets directement du périphérique à la mémoire. Les périphériques doivent disposer de capacités DMA intégrées et, du coup, tous les périphériques ne peuvent pas faire de DMA. Alors que le DMA est en cours, le processeur ne peut pas faire grand chose car le bus principal est en cours d'utilisation par le transfert DMA.

Le bus LPC (et l'ancien bus ISA) peuvent faire du DMA alors que les bus PCI et LPC peuvent faire à peu près la même chose acquérir la maîtrise du bus. LPC et ISA ont besoin des ressources DMA (canaux). Sur le bus PCI, la maîtrise du bus est souvent appelé DMA. La maîtrise du bus permet aux périphériques de devenir temporairement les maîtres du bus et de transférer des octets un peu comme lorsque le maître du bus était le processeur. Il n'utilise aucun numéro de canal car l'organisation du bus PCI est telle que le matériel PCI sait quel périphérique est actuellement le maître du bus et quel périphérique réclame à devenir le maître du bus. Du coup, il n'y a pas d'allocation de ressources pour les canaux DMA pour le bus PCI et aucune ressource de canaux DMA n'existe pour ce bus. Le bus LPC est supposé être configuré par le BIOS pour que les utilisateurs n'aient pas à se soucier des canaux DMA.

Canaux DMA (non pas pour le bus PCI)

C'est seulement pour l'ancien bus ISA et le bus LPC. Quand un périphérique veut faire du DMA, il lance une requête de DMA en utilisant les fils dédiés à cela, un peu comme une requête d'interruption. En fait, le DMA aurait pû être géré en utilisant des interruptions mais cela aurait introduit des délais, donc il est plus rapide de faire cela en ayant un type spécial d'interruption connu en tant que requête DMA. Comme les interruptions, les demandes DMA sont numérotées pour identifier le périphérique lançant la requête. Ce nombre est appelé un canal DMA. Comme les transferts DMA utilisant tous le bus principal (et qu'un seul peut être lancé à la fois), ils utilisent tous les même canal pour le flot de données mais le numéro de « canal DMA » sert à identifier qui utilise le canal. Les registres matériels existent sur la carte mère, qui enregistre l'état actuel de chaque canal. Du coup, pour lancer une requête DMA, le périphérique doit connaître son numéro de canal DMA stocké dans un registre spécial du périphérique physique.

« Ressources » du périphérique et du pilote

Donc, les pilotes de périphériques doivent être « attachés » d'une façon quelconque au matériel qu'ils contrôlent. Ceci se fait en allouant des ressources bus (I/O, mémoire, IRQ, DMA) au périphérique physique et en laissant le pilote le découvrir. Par exemple, un port série utilise seulement deux ressources : une IRQ et une adresse d'entrées/sorties. Ces deux valeurs doivent être fournies au pilote et au périphérique physique. Le pilote (et son périphérique) dispose d'un nom dans le répertoire /dev (tel que ttyS1). L'adresse et le numéro IRQ sont stockés par le périphérique physique dans ses registres de configuration sur sa carte (ou dans un composant de la carte mère). Les vieux matériels (dans les années 1990) utilisaient des interrupteurs (ou des cavaliers) pour configurer physiquement l'IRQ et l'adresse au niveau du matériel. Ce paramétrage restera fixe tant qu'une personne n'enlevera pas le boîtier pour déplacer les cavaliers.

Mais, dans le cas de PnP (pas de cavaliers), les données du registre de configuration sont habituellement perdues lorsque le PC est éteint, donc les données des ressources bus doivent être fournies à chaque périphérique à chaque allumage du PC.

Les ressources sont limitées

L'ordinateur idéal

L'architecture du PC n'apporte qu'un nombre limité de ressources : IRQ, canaux DMA, adresses d'entrées/sorties et de plages mémoires. S'il n'existait qu'un nombre limité de périphériques et qu'ils aient tous des ressources bus standardisées (telles que des adresses d'entrées/sorties et des numéros d'IRQ uniques), il n'y aurait aucun souci pour attacher un pilote à son périphérique. Chaque périphérique devrait avoir un nombre fixe de ressources qui n'entreraient pas en conflit avec tout autre périphérique sur votre ordinateur. Deux périphériques ne devraient pas avoir les mêmes adresses, il n'y aurait pas de conflit d'IRQ sur le bus ISA, et cætera. Chaque pilote devrait être développé avec des adresses, des IRQ, et cætera, uniques codées en dur dans le programme. La vie serait simple.

Une autre façon d'empêcher les conflits d'adresses serait d'avoir un numéro d'emplacement, inclus dans l'adresse, pour chaque carte. Du coup, il n'y aurait plus de conflits d'adresses entre deux cartes différentes (car elles sont dans des emplacements différents). La conception des cartes ne permettrait pas les conflits d'adresses entre les différentes fonctions de la carte. Il en ressort que l'espace d'adressage (utilisé pour la demande et l'affectation de ressources) le fait réellement. Mais cela n'est pas pris en compte pour les adresses d'entrées/sorties et pour les régions mémoire. Partager des IRQ comme sur le bus PCI évite aussi des conflits mais peut poser d'autres problèmes.

L'ordinateur réel

Mais l'architecture du PC a des problèmes de conflit. L'augmentation du nombre de périphériques (incluant les multiples périphériques de même type) a tendance à augmenter les conflits potentiels. En même temps, l'introduction du bus PCI, où deux périphériques ou plus peuvent partager la même interruption et l'introduction d'interruptions supplémentaires, a tendance à réduire les conflits. Le résultat global, dû au passage au PCI, a été une réduction des conflits car les ressources les plus faibles sont les IRQ. Néanmoins, même sur le bus PCI, c'est un peu plus efficace pour éviter le partage des IRQ. Dans certains cas où les interruptions arrivent en une succession rapide et doivent être traitées rapidement (comme en audio), le partage peut causer des dégradations dans les performances. Donc, il n'est pas bon d'affecter tous les périphériques PCI au même IRQ, l'affectation doit être partagée. Néanmoins, certaines personnes trouvent que tous les périphériques PCI sont sur la même IRQ.

Donc, les périphériques ont besoin d'avoir de la flexibilité de façon à ce qu'ils puissent être initialisés avec n'importe quelle adresse, IRQ, et cætera. C'est nécessaire pour éviter tout conflit et arriver à un point d'équilibre. Mais quelques IRQ et adresses sont pratiquement des standards, comme ceux de l'horloge et du clavier. Ils n'ont pas besoin d'une telle flexibilité.

En plus du problème de conflit lors de l'allocation des ressources bus, une indication erronée en indiquant au pilote de périphérique quelles sont les ressources bus peut causer un autre problème. Cela a plus de chances d'arriver dans le cas de la configuration manuelle où l'utilisateur saisit les ressources utilisées dans un fichier de configuration sur le disque dur. Ceci fonctionne généralement bien quand les ressources sont initialisées avec des cavaliers sur les cartes (en supposant que l'utilisateur sache comment elles étaient initialisées et n'a fait aucune faute en saisissant ces données dans les fichiers de configuration). Mais, avec des ressources configurées par un logiciel PnP, elles ne seront pas toujours identiques et cela pourrait poser problème pour toute configuration manuelle.

L'allocation de ressources bus, lorsqu'elle est faite correctement, établit des canaux de communications sans conflit entre le matériel physique et le pilote associé. Par exemple, si une certaine plage mémoire d'entrées/sorties (ressource) est allouée à la fois au pilote de périphérique et au matériel, alors cela a établi une communication sur une voie à sens unique entre eux. Le pilote peut envoyer des commandes et des informations au périphérique. C'est donc un peu plus qu'une voie à sens unique car le pilote peut obtenir des informations du périphérique en lisant ces registres. Mais le périphérique ne peut pas initier une communication de cette façon. Pour initier une communication, le périphérique a besoin d'une IRQ pour qu'il puisse envoyer une interruption à son pilote. Ceci crée un canal de communication à double-sens où le périphérique physique et son pilote peuvent initier une communication.

Deuxième introduction au Plug-and-Play (PnP)

Introduction à PnP

Le terme Plug-and-Play (PnP) a différentes significations. Au sens général, il s'agit de la configuration automatique lorsqu'une personne connecte un périphérique et que celui-ci se configure lui-même. Le sens utilisé dans ce livre est tout autre : PnP signifie la configuration des ressources bus pour PnP (les configurer au niveau des périphériques physiques) et la communication de cette information aux pilotes de périphériques. Dans le cas de Linux, il s'agit souvent d'un pilote déterminant la façon dont le bus a initialisé les ressources bus et, si nécessaire, le pilote donnant une commande pour changer (réinitialiser) les ressources bus. Souvent, « PnP » ne signifie que PnP sur le bus ISA donc le message d'isapnp, « No Plug and Play device found », signifie simplement qu'aucun périphérique ISA PnP n'a été trouvé. Les spécifications du PCI standard (inventé avant l'utilisation du terme « PnP ») apportent l'équivalent de PnP au bus PCI.

PnP fait correspondre les périphériques avec leur pilote et spécifie leurs canaux de communication (en allouant des ressources bus). Il communique électroniquement avec les registres de configuration situés à l'intérieur des périphériques physiques en utilisant un protocole standardisé. Sur le bus ISA et avant le Plug-and-Play, les ressources bus étaient simplement initialisées au niveau matériel avec des interrupteurs de différentes sortes. Quelquefois, les ressources bus pouvaient être configurées sur le matériel par un programme (généralement écrit seulement pour un système MS) à partir d'une disquette fournie avec le matériel. Cela ressemblait à du PnP mais aucun protocole standardisé n'était utilisé donc il ne s'agissait pas de PnP. Pour Linux avant le PnP, les ressources bus étaient affectées aux pilotes logiciels par des fichiers de configuration (ou autre chose de ce genre) ou en parcourant le bus aux adresses où il s'attendait à trouver le périphérique. Mais ces méthodes sont toujours utilisées de nos jours pour permettre à Linux d'utiliser du matériel non PnP. Et, quelque fois, ces vieilles méthodes sont toujours utilisées de nos jours sur du matériel PnP (après que le BIOS ait affecté des ressources aux matériels par les méthodes PnP).

Le bus PCI ressemblait au PnP dès le début mais il n'a pas été appelé PnP à ce moment (et ne l'est encore pas).

Comment fonctionne le PnP (explication simplifiée)

Voici comment PnP devrait fonctionner. L'hypothétique programme de configuration PnP trouve tous les périphériques PnP et demande à chacun les ressources bus dont il a besoin. Ensuite, il vérifie quelles sont les ressources bus qu'il peut affecter (IRQ, et cætera). Bien sûr, s'il a réservé des ressources bus utilisées pour des périphériques non PnP (s'il les connaît), il ne les donne pas. Il utilise certains critères (ne faisant pas partie des spécifications PnP) pour donner les ressources bus de façon à ce qu'il n'y ait pas de conflit et de façon à ce que tous les périphériques disposent de ce qu'ils ont demandé (si possible). Il indique ensuite indirectement à chaque périphérique physique les ressources bus qui lui ont été assignées et les périphériques se configurent eux-mêmes pour n'utiliser que ces ressources-là. Enfin, les pilotes de périphériques trouvent d'une manière ou d'une autre quelles ressources sont utilisées par leur périphérique et sont donc capables de communiquer avec les périphériques qu'ils contrôlent.

Par exemple, prenons une carte ayant besoin d'une IRQ (d'un numéro d'IRQ) et de 1 Mo de mémoire partagée. Le programme PnP lit cette requête des registres de configuration de la carte. Il lui affecte l'IRQ 5 et 1 Mo d'espace mémoire, commençant à partir de l'adresse 0xe9000000. Le programme PnP lit aussi les informations d'identification de la carte, indiquant ainsi le type de périphérique, son numéro d'identifiant, et cætera. Ensuite, il informe, directement ou indirectement, le pilote de périphérique convenable pour ce matériel, de ce qu'il a fait. Si le pilote lui-même gère le PnP, alors il n'est pas nécessaire de trouver un pilote pour le périphérique (car le pilote est déjà en cours d'exécution). Sinon, le bon pilote de périphérique doit être trouvé et la configuration du périphérique doit lui être communiquée.

Ce n'est pas toujours aussi simple car la carte (ou la table de routage pour le PCI) pourrait indiquer qu'elle ne peut utiliser que certains numéros d'IRQ ou que les 1 Mo de mémoire doivent résider dans un certain espace d'adresses. Les détails sont différents pour les bus PCI et ISA, cette dernière étant la plus complexe.

Une façon habituellement utilisée pour allouer des ressources est de commencer avec un périphérique et de lui allouer des ressources bus. Puis de faire la même chose avec le périphérique suivant, et cætera. Ensuite, si finalement tous les périphériques obtiennent les ressources allouées sans conflit, alors tout va bien. Mais si allouer une ressource requise créait un conflit, alors il serait nécessaire de revenir en arrière et d'essayer de faire quelques changements dans les allocations précédentes de façon à conserver la ressource bus nécessaire. Ceci s'appelle le balancement. Linux ne le fait pas mais MS Windows en est capable dans certains cas. Pour Linux, tout ceci est fait par le BIOS, le noyau ou les pilotes de périphériques. Avec Linux, le pilote du périphérique n'obtient pas son allocation finale de ressources tant que le pilote n'est pas lancé, donc une façon d'éviter les conflits est de ne pas lancer un périphérique qui pourrait causer un conflit. Néanmoins, le BIOS alloue fréquemment des ressources au périphérique physique avant que Linux ne soit complètement démarré et le noyau vérifie les périphériques PCI pour les conflits d'adresse au démarrage.

Des raccourcis existent et le logiciel PnP peut les utiliser. Un d'eux est de garder la trace de la façon dont sont assignées les ressources bus lors de la dernière configuration (lorsque l'ordinateur a été utilisé pour la dernière fois) et de réutiliser cette information. Les BIOS le font ainsi que MS Windows mais le Linux standard ne le fait pas. Mais, d'une certaine façon, il le fait car il utilise souvent ce que le BIOS a fait. Windows stocke cette information dans sa base de registres sur le disque dur et un BIOS PnP/PCI le stocke dans une mémoire non volatile de votre PC (mémoire appelée ESCD ; voir la section intitulée « La base de données ESCD du BIOS » [p 25]). Certains disent que ne pas avoir de registres (ce qui est le cas de Linux) est mieux car, avec Windows, la base de registres peut être corrompue et elle est de toute façon difficile à éditer. Mais PnP sur Linux a aussi des problèmes.

Alors que MS Windows (sauf en ce qui concerne Windows 3.x et NT4) est un système d'exploitation PnP, Linux n'était pas originellement un système d'exploitation PnP mais l'est devenu graduellement. Au début, PnP a fonctionné avec Linux parce qu'un BIOS PnP configurait les ressources bus et que les pilotes de périphériques récupéraient cette information en utilisant des programmes apportés par le noyau Linux. Aujourd'hui, la plupart des pilotes peuvent envoyer des commandes pour réaliser leur propre configuration et n'ont pas besoin de se reposer toujours sur le BIOS. Malheureusement, un pilote pourrait utiliser une ressource bus dont un autre périphérique aurait besoin plus tard. D'autres pilotes de périphériques enregistrent la dernière configuration dans un fichier de configuration et l'utilisent la prochaine fois que l'ordinateur est allumé.

Si le périphérique physique se rappelle de son ancienne configuration, alors il n'y aura pas de matériel à configurer au prochain démarrage, mais il semble oublier sa configuration lors de l'arrêt. Certains périphériques disposent d'une configuration par défaut (mais pas nécessairement la dernière utilisée). Donc, un périphérique PnP a besoin d'être reconfiguré à chaque fois que le PC est allumé. De la même manière, si un nouveau périphérique est ajouté, alors il a besoin d'être configuré. Allouer des ressources bus pour ce nouveau périphérique peut nécessiter de récupérer des ressources données auparavant à un autre et d'affecter à ce dernier de nouvelles ressources. Actuellement, Linux ne peut pas gérer ce niveau de sophistication (et MS Windows XP pourrait aussi ne pas en être capable).

Démarrer le PC

Quand le PC est lancé la première fois, le BIOS lance son programme pour démarrer le PC (la première étape étant de vérifier le matériel de la carte mère). Si le système d'exploitation est stocké sur disque dur (ce qui est habituellement le cas), alors le BIOS doit disposer de certaines informations sur le disque dur. Si ce disque est PnP, le BIOS peut utiliser des méthodes PnP pour le trouver. De même, pour permettre à l'utilisateur de configurer manuellement le CMOS du BIOS et de répondre aux messages d'erreur lors du démarrage, un écran (et donc une carte vidéo) et un clavier sont aussi requis. Donc, le BIOS doit toujours configurer via PnP les périphériques nécessaires au chargement du système d'exploitation à partir du disque dur.

Une fois que le BIOS a identifié le disque dur, la carte vidéo et le clavier, il est prêt à commencer le démarrage (charger le système d'exploitation en mémoire). Si vous avez indiqué au BIOS que vous disposez d'un système d'exploitation PnP, il devrait commencer le chargement comme indiqué ci-dessus et laisser le système d'exploitation finir la configuration PnP. Autrement, un BIOS-PnP essaiera de configurer le reste des périphériques (mais sans en informer leur pilote). peuvent toujours trouver les informations nécessaires en utilisant les fonctions disponibles dans le noyau Linux.

Les bus

Pour voir ce qui se trouve sur le bus PCI, exécutez **lspci** ou **lspci -vv**. Ou vous pouvez aussi saisir **scanpci -v** pour la même information dans le format de code numérique où le périphérique est indiqué par numéro (par exemple : « device 0x122d ») au lieu du nom, et cætera. Dans de rares cas, **scanpci** trouvera un périphérique que **lspci** n'arrive pas à trouver.

Les messages au démarrage sur votre écran affichent les périphériques qui ont été trouvés sur les différents bus (utilisez shift-PageUp pour les voir). Voir Messages au démarrage [p 33] .

ISA est l'ancien bus des PC compatibles IBM alors que PCI est le nouveau bus, plus rapide, d'Intel. Le bus PCI a été conçu pour ce qui est appelé de nos jours PnP. Ceci rend facile (comparé à ce qu'il faut faire pour le bus ISA) la découverte de l'affectation des ressources bus PnP aux périphériques matériels.

Pour le bus ISA, il existait un problème réel avec l'implémentation de PnP car personne n'avait en tête PnP lorsque ce bus a été conçu et il existe encore moins d'adresses d'entrées/sorties disponibles pour PnP pour envoyer des informations de configuration à un périphérique physique. Du coup, la façon dont PnP a été réalisé pour le bus ISA est très complexe. Des livres entiers ont été écrits à ce sujet. Voir notamment ce livre [p 30] . Entre autres choses, il requiert que soit assigné par le programme PnP à chaque périphérique PnP un point d'ancrage temporaire pour que tout le monde puisse faire la configuration PnP. Assigner ces « points d'ancrage » est appelé « isolation ». Voir la section intitulée « Isolation ISA » [p 50] pour des détails complexes.

Au fur et à mesure de la disparition du bus ISA, PnP sera un peu plus facile. Il ne sera pas seulement plus simple de savoir comment le BIOS a configuré le matériel mais il y aura aussi moins de conflits, le PCI pouvant partager des interruptions. Il y aura toujours besoin de faire correspondre un pilote de périphérique à un matériel et il y aura toujours besoin de configurer les périphériques ajoutés lors du lancement du PC. Le sérieux problème des quelques périphériques non supportés par Linux restera présent.

Comment Linux gère-t-il le PnP

Linux a eu de sérieux problèmes dans le passé pour la gestion de PnP mais la plupart de ces problèmes sont maintenant résolus (vers mi-2004). Linux est parvenu d'un système non PnP à un système qui peut être PnP si le noyau est compilé avec certaines options. Le BIOS peut affecter des IRQ mais Linux peut aussi affecter certains d'entre eux ou même les refaire ce que le BIOS a déjà fait. La partie de configuration de ACPI (*Advance Configuration and Power Interface*) est conçu pour faciliter la propre configuration des systèmes d'exploitation. Linux peut utiliser l'ACPI si le noyau a été compilé avec son support.

Dans Linux, il est traditionnel qu'un pilote de périphérique fasse sa propre configuration de bas niveau. C'était difficile jusqu'au moment où le noyau Linux a fourni le logiciel que les pilotes peuvent utiliser pour se faciliter le travail. Aujourd'hui (2005), c'est arrivé à un point où le pilote peut simplement appelé la fonction pci_enable_device() du noyau et où le périphérique se voit configuré en étant activé et en récupérant une IRQ (si nécessaire) et les adresses affectées au périphérique. Cette affectation pourrait être ce qui avait été affecté auparavant par le BIOS ou ce que le noyau avait réservé quand le périphérique PCI ou ISAPNP a été détecté par le noyau. Il existe même une option ACPI pour que le noyau affecte toutes les IRQ des périphériques lors du démarrage.

Donc aujourd'hui, d'une certaine façon, les pilotes font toujours la configuration mais ils peuvent la faire en demandant simplement au noyau de s'en charger (et Linux pourrait ne pas avoir besoin de faire grand chose car il est quelque fois capable d'utiliser ce qui a déjà été configuré par le BIOS ou par lui-même). Donc, c'est vraiment la partie du noyau, hors du pilote, qui fait la grande partie de configuration. Du coup, il pourrait être correct d'appeler Linux un système d'exploitation PnP, au moins pour les architectures communes.

Ensuite, lorsqu'un pilote découvre son périphérique, il demande à connaître les adresses et IRQ affectés (par le BIOS ou par Linux) et, habituellement, les accepte simplement. Mais, si le pilote le souhaite, il peut essayer de modifier les adresses en utilisant les fonctions fournies par le noyau. Cependant, le noyau n'acceptera pas d'adresses entrant en conflit avec d'autres périphériques ou des adresses que le matériel ne peut pas supporter. Quand le PC démarre, vous pouvez noter les messages à l'écran montrant que certains pilotes de périphériques Linux ont trouvé leurs périphériques matériels et quels sont leurs IRQ et adresses.

Donc, le noyau fournit des fonctions (programmes) que les pilotes peuvent utiliser pour trouver si leur périphérique est présent, la façon dont il est configuré et les fonctions qui permettent de modifier la configuration si nécessaire. Le noyau 2.2 pouvait faire ceci pour le bus PCI seulement mais le noyau 2.4 contient cette fonctionnalité pour les bus ISA et PCI (à condition que les options PnP et PCI appropriés ont été sélectionnées avant la compilation du noyau). Le noyau 2.6 est arrivé avec une meilleure utilisation de l'ACPI. Ceci ne garantie en rien que tous les pilotes utilisent complètement et correctement ces fonctionnalités. Les périphériques propriétaires que le BIOS ne connait pas pourraient ne pas être configurés tant qu'une configuration manuelle ne soit effectuée.

De plus, le noyau tente d'éviter les conflits d'adresses en ne permettant pas à deux périphériques d'utiliser les même ressources bus en même temps. Au début, ce n'était valable que pour les IRQ et les DMA mais, maintenant, cela s'adresse aussi aux adresses.

Si vous avez un ancien bus ISA, le programme isapnp est exécuté pour trouver et configurer les périphériques PnP sur le bus ISA.

Pour voir quelle aide le noyau peut apporter aux pilotes de périphériques, consultez le répertoire `/usr/.../.../Documentation` où un des « ... » contient le mot « kernel-doc » ou quelque chose d'approchant. Attention : cette documentation a tendance à ne plus être à jour pour avoir les dernières informations donc vous aurez besoin de lire les messages des listes de diffusion des développeurs du noyau et peut-être aussi de lire le code source et les commentaires qu'ils ont écrits. Dans le répertoire de la documentation du noyau, voir pci.txt (« How to Write Linux PCI Drivers », c'est-à-dire « Comment écrire des pilotes PCI pour Linux ») ainsi que le fichier `/usr/include/linux/pci.h`. Si vous êtes un gourou des pilotes et si vous connaissez la programmation en C, ces fichiers sont écrits d'une telle façon qu'il ne vont pas vous permettre d'écrire un pilote. Mais cela vous donnera une idée des fonctions type PnP disponibles pour les pilotes.

Pour le noyau 2.4, voir `isapnp.txt`. Pour le noyau 2.6, `isapnp.txt` est remplacé par `pnp.txt` qui est totalement différent et gère en plus le bus PCI. Voir aussi le livre d'O'Reilly : Linux Device Drivers, 3è édition, 2005. Le texte complet est disponible sur Internet.

Problèmes avec Linux PnP

Il existe un certain nombre d'autres points qu'un système d'exploitation PnP devrait mieux gérer :

- Allouer des ressources bus quand il en existe peu par une réallocation des ressources si nécessaire ;
- Gérer le choix d'un pilote lorsqu'il en existe plus d'un par périphérique ;

Comme chaque pilote s'occupe de lui mais pas des autres, un pilote pourrait récupérer les ressources bus nécessaires à d'autres périphériques (et non encore alloués à ceux-ci par le noyau). Donc un noyau Linux PnP plus perfectionné serait bien mieux, un noyau qui pourrait s'occuper de l'allocation une fois toutes les requêtes envoyées. Une alternative serait d'essayer de réallouer les ressources déjà affectées quand un pilote n'obtient pas toutes les ressources qu'il a demandées.

Le problème de la « rareté des ressources bus » devient de moins en moins réel pour deux raisons : la première est que le bus PCI est en train de remplacer le bus ISA. Le bus PCI n'a pas ce type de problème pour les IRQ car celles-ci peuvent être partagées (bien que ce partage rende le système un peu moins efficace). De plus, le PCI n'utilise pas les ressources DMA (bien qu'il dispose d'un équivalent sans avoir besoin des ressources).

La deuxième raison est que plus d'espace d'adresses est disponible pour les entrées/sorties des périphériques. Alors que l'espace d'adresses conventionnel du bus ISA était limité à 64 Ko, le bus PCI dispose de 4 Go. Comme plus de périphériques physiques utilisent les adresses en mémoire principale au lieu de l'espace d'adresses des entrées/sorties, il existe toujours un peu de place disponible, y compris sur le bus ISA. Sur les PC 32 bits, il y a un espace d'adressage de 4 Go en mémoire principale et la plupart de ces ressources bus est disponible pour les entrées/sorties des périphériques (à moins que vous ayez 4 Go de mémoire principale installée).

Il y a eu au moins une tentative pour faire de Linux un vrai système d'exploitation PnP. Voir http://www.astarte.free-online.co.uk. Bien que développé dès 1998, elle n'a jamais été intégrée au noyau (mais aurait certainement dûe l'être).

Configurer un BIOS PnP

Lorsque l'ordinateur est démarré, le BIOS est lancé avant que le système d'exploitation ne soit chargé. Les BIOS modernes sont PnP et peuvent configurer la plupart des périphériques PnP. Quelques anciens BIOS PCI vont seulement configurer le bus PCI. Voici quelques choix qui pourraient exister dans le menu CMOS de votre BIOS :

- la section intitulée « Avez-vous un système d'exploitation PnP ? » [p 20] ;
- la section intitulée « Affecter les ressources par le BIOS ? » [p 23] ;
- la section intitulée « Réinitialiser la configuration » [p 23] .

Avez-vous un système d'exploitation PnP ?

Quelle que soit votre réponse au BIOS, le BIOS PnP utilisera PnP pour paramétrer le disque dur, le lecteur de disquette, la carte vidéo et le clavier, afin de permettre au système de démarrer. Si vous dites avoir un système d'exploitation PnP, il laissera la fin de la configuration au système d'exploitation (ou aux pilotes de périphériques). Si vous dites ne pas avoir de système d'exploitation PnP, alors le BIOS devra tout configurer.

Comment répondre à cette question de votre BIOS ? Si vous avez au moins un noyau 2.4, vous pourriez répondre ce que vous voulez et Linux fonctionnera habituellement correctement. Même si vous avez Windows 2000 ou XP sur le même PC, cela devrait fonctionner de toute façon tout simplement parce que Windows et Linux sont tous les deux à priori des systèmes d'exploitation PnP et que si le système d'exploitation est PnP, il devrait être capable de gérer le cas où le BIOS a tout configuré lui-même (si vous avez répondu que le système d'exploitation n'est pas PnP). Mais, je continue à suggèrer de répondre qu'il n'est pas PnP sauf si une raison valable vous oblige à faire autrement.

Linux avant le noyau 2.4

La réponse n'est pas souvent claire dans ce cas. Si isapnp était utilisé par Linux, alors Linux fera la configuration et il était indiqué qu'il est mieux de dire qu'il s'agit d'un système d'exploitation PnP. La raison pour laquelle isapnp aurait des problèmes en présence de périphériques déjà configurés par le BIOS n'est pas claire mais de tels problèmes arrivent quelquefois et sont corrigés en stoppant la configuration du BIOS (en répondant oui, c'est un système d'exploitation PnP). Il existe quelques cas où dire non résolvait un problème. Donc, si isapnp n'est pas utilisé, non est généralement mieux. Les pilotes Linux de périphériques PCI devraient configurer correctement ces périphériques. Mais pour le cas où les périphériques PCI pilotés par des pilotes non PCI, alors vous pourriez dire que le système d'exploitation n'est pas PnP pour obtenir du BIOS qu'il les configure directement.

Windows 2000 et XP

Si vous utilisez aussi des systèmes d'exploitation Windows sur le même PC, vous pourriez dire que vous n'avez pas un système d'exploitation PnP. C'est ce que MS vous suggère de faire. Peut-être que MS souhaite que le BIOS fasse un meilleur travail pour la configuration que Windows ne le fera. Ceci est sensé parce que le BIOS devrait être conçu pour les particularités spécifiques de la carte mère, et tout spécialement de nos jours où beaucoup de périphériques sont intégrés à celle-ci. Dire non devrait aussi être bon pour les noyaux Linux 2.4 et ultérieurs. Mais pour les noyaux précédents, ce n'est pas si clair (voir la section ci-dessous). Donc, si vous avez des problèmes avec Linux, vous pourriez essayer de dire que vous avez un système d'exploitation Linux mais ceci va contre ce que raconte MS (mais fonctionnera probablement bien de toute façon).

Lorsque le BIOS configure un périphérique différemment de ce qui est stocké dans la base de registres de Windows, celui-ci vous dira qu'il a découvert un nouveau matériel. Ce qu'il est réellement en train de faire est de trouver l'ancien matériel qui a été configuré différemment. De toute façon, il enregistre la configuration que le BIOS a utilisée dans ses registres et le périphérique devrait bien fonctionner à partir de ce moment.

MS Windows 95, 98 (et Me ?)

Pour Windows9x, MS suggère de dire au BIOS que vous avez un système d'exploitation PnP (l'opposé complet du cas pour Windows 2000 et XP). Ceci devrait être bon pour Linux si vous disposez d'un noyau 2.4 ou ultérieur. Mais si vous avez un noyau Linux précédent le 2.4, alors il est mieux pour Linux de dire qu'il ne s'agit pas d'un système d'exploitation PnP. Une façon de résoudre ce dilemme est de le configurer pour le système d'exploitation que vous utilisez le plus fréquemment. Ensuite, au démarrage de l'autre système d'exploitation, modifiez manuellement la configuration du BIOS. C'est très ennuyant mais c'est faisable si vous n'utilisez pratiquement jamais l'autre système d'exploitation. Sinon, il existe de meilleurs façons de résoudre ce dilemme.

La deuxième façon de résoudre ce dilemme est de faire en sorte que Linux configure toutes les ressources. Voir la section intitulée « Linux avant le noyau 2.4 » [p 20] . Ensuite, vous dites au BIOS qu'il s'agit d'un système d'exploitation PnP.

La troisième façon de résoudre ce dilemme est de dire au BIOS qu'il ne s'agit pas d'un système d'exploitation PnP. Ceci va à l'encontre de ce que dit MS mais il est possible d'obtenir un bon fonctionnement de MS Windows9x si vous comprenez ce que vous faites (et pourquoi). Si vous dites au BIOS qu'il ne s'agit pas d'un système d'exploitation PnP, MS Windows ne devrait-il pas détecter la façon dont le BIOS a configuré les périphériques et modifier cela s'il n'aime pas ce que le BIOS a fait ? Cela devrait, mais malheureusement, cela ne semble pas fonctionner de cette façon.

Ce que Windows 9x semble faire lorsqu'il trouve un matériel déjà configuré par le BIOS est de le laisser seul et de ne pas le reconfigurer. Maintenant, Windows 9x garde une trace de la configuration des ressources bus dans sa base de registres. Si la configuration du BIOS est différent, il devrait soit corriger ce qui se trouve dans sa base de registres soit tout reconfigurer suivant les indications de cette même base. Mauvaise nouvelle : il semble ne rien faire et pense que la configuration actuelle est la même que celle de la base de registres alors qu'en fait elles sont différentes.

Mais si la base de registre contient une configuration des ressources bus identique à celle du BIOS, alors tout fonctionnera bien. Un périphérique fonctionnera bien si le BIOS l'a configuré de la même façon que ce qui est enregistré dans la base de registres. Donc, le moyen de faire fonctionner correctement MS Windows est d'obtenir que la base de registres soit synchronisée avec la configuration du BIOS. Comme mentionné précédemment, le BIOS configure les éléments suivant son ESCD (qui est quelque chose comme la base de registres mais pour le BIOS). Voir la section intitulée « La base de données ESCD du BIOS » [p 25] . Donc, nous avons besoin d'obtenir la synchronisation des registres avec l'ESCD du BIOS pour que la base de registres et ESCD contiennent la même configuration. Dans certains cas, ces deux arrivent à être synchrones et vous n'avez pas besoin de faire quoi que ce soit.

Une question à laquelle vous pourriez penser est : comment l'ESCD du BIOS et la base de registres Windows peuvent-ils se désynchroniser ? Voici un scénario. Vous installez Windows avec le BIOS configuré pour un système d'exploitation PnP. Alors, Windows configure la plupart des éléments et sauvegarde sa configuration dans sa base de registres. Puis, plus tard, vous changez la configuration du BIOS en précisant qu'il ne s'agit pas d'un système d'exploitation PnP. Ensuite, après un redémarrage, le BIOS configure tout et il ne fait pas exactement ce que Windows a fait. Donc, la configuration actuelle du matériel et ce que Windows dispose dans sa base de registres sont maintenant différents.

Une façon d'essayer d'obtenir que la base de registres et l'ESCD disposent des mêmes informations est d'installer (ou de réinstaller) Windows lorsque le BIOS est configuré pour un système d'exploitation non PnP. De cette façon, Windows disposera du matériel configuré par le BIOS. Si cette configuration est faite sans conflit, Windows n'en changera pas et la sauvegardera dans sa base de registres. Et dans ce cas, l'ESCD et la base de registres seront synchronisés.

Une autre méthode est de supprimer les périphériques causant problèmes à Windows en cliquant « Supprimer » dans le gestionnaire des périphériques. Puis redémarrez avec « OS non PnP » (enregistré dans la mémoire CMOS du BIOS lorsque vous redémarrez). Windows va alors réinstaller les périphériques, en utilisant, on l'espère, les ressources bus configurées par le BIOS. Faites attention que Windows vous demandera d'insérer le CD d'installation de Windows car il peut ne pas trouver les fichiers du pilote de périphériques, même s'il sont bien là. Un contournement est de sélectionner « skip file » ce qui évitera l'installation du fichier à partir du CD. Si le fichier est toujours sur le disque dur, avec un peu de chance, le pilote et tout ira bien, même si le programme d'installation de Windows vous a demandé de l'installer à partir du CD (ce que vous avez passé).

Comme test, j'ai « supprimé » une carte réseau qui utilisait un pilote compatible Novell. Au redémarrage, Windows l'a réinstallé avec le Réseaux Microsoft plutôt qu'avec Novell. Ceci signifie que le client Novell a dû être réinstallé, un gros travail inutile. Donc, il serait mieux de ne pas continuer avec Windows 95/98 mais laisser Linux configurer les ressources bus.

Lors de l'utilisation d'un PC Window-Linux (*dual boot*), vous pouvez noter un changement dans la façon dont le BIOS configure à cause de Windows9x (et des autres versions de Windows ??) en modifiant l'ESCD. Il fait cela seulement si vous « forcez » une configuration ou une installation d'un périphérique propriétaire. Voir la section intitulée « Utiliser Windows pour configurer l'ESCD » [p 26] . Les pilotes de périphériques réalisant la configuration pourraient modifier ce que le BIOS a fait comme le font les outils PCI et isapnp.

Affecter les ressources par le BIOS ?

Les BIOS modernes vous permettent d'allouer manuellement des ressources, principalement des IRQ. Il existe normalement une option pour configurer l'allocation à « auto » de façon à ce que le BIOS décide de l'allocation des ressources. « Auto » est souvent un bon choix sauf si vous avez d'anciennes cartes ISA propriétaires non PnP.

Si vous avez de telles cartes non PnP, alors il peut être important de réserver les ressources (telles que les IRQ) pour celles-ci dans le BIOS. Sinon, le BIOS pourrait utiliser ces ressources pour d'autres périphériques et créer ainsi des conflits. Une exception concerne quelques périphériques propriétaires communs, comme les ports parallèle et séries, les disques durs. Le BIOS pourrait les trouver (jetez un ½il à l'écran au démarrage) pour que vous n'ayez pas besoin de réserver les ressources pour eux. Si vous avez utilisé Windows sur votre PC, Windows pourrait déjà avoir renseigné le BIOS en utilisant l'outil ICU (ou un outil identique) sous Windows.

Pour le PCI, le BIOS devrait aussi vous permettre d'affecter les IRQ aux emplacements de cartes 1, 2, 3, 4, et cætera. Si vous le faites, vous devez connaître les emplacements où se trouvent les cartes. En fait, chaque emplacement dispose de quatre IRQ PCI : A, B, C et D. Si le menu du BIOS ne vous dit pas laquelle (A, B, C, D) est affectée à un numéro d'IRQ, il est probable qu'il affecte seulement le numéro d'IRQ à l'IRQ PCI A. Mais, beaucoup de cartes utilisent seulement l'IRQ A donc il s'agit surtout d'affecter une IRQ à un emplacement. Voir les interruptions PCI [p 39] .

Réinitialiser la configuration

C'est aussi un peu risqué. Ceci va écraser les données du BIOS contenues dans l'ESCD indiquant la façon dont vos périphériques PnP ont été configurés et comment les périphériques non PnP ont été configurés manuellement. Ne faites jamais ceci à moins que vous ne soyez convaincu que la base de données est mauvaise et a besoin d'être reconstruite. Il était indiqué quelque part que vous deviez faire ceci seulement lorsque vous n'arrivez plus à démarrer. Si le BIOS perd les données sur les périphériques ISA non PnP, alors vous devrez relancer ICA une nouvelle fois sous DOS/Windows pour ré-enregistrer les données.

Gérer les cartes PnP

Introduction à la gestion des périphériques PnP

De nos jours, pratiquement toutes les nouvelles cartes internes sont Plug-and-Play. Du coup, la configuration des ressources bus devraient être dans la plupart des cas entièrement automatique. Si un périphérique ne fonctionne pas, vérifiez s'il a été détecté, par exemple en redémarrant. Si le pilote de périphérique ne peut pas configurer les ressources, alors probablement une ou plus des méthodes du 2.6 le feront :

- la section intitulée « Configuration du pilote de périphérique, réservation des ressources » [p 24] ;
- la section intitulée « /sys : interface de configuration pour l'utilisateur » [p 24] le noyau 2.6+ (pas encore pour le PCI et quelques autres sévères limitations) ;

- la section intitulée « Configuration du BIOS » [p 25] (pour le bus PCI, vous avez seulement besoin d'un BIOS PCI, sinon vous avez besoin d'un BIOS PnP) ;
- la section intitulée « ISA seulement : Désactiver PnP ? » [p 27] par des cavaliers ou avec un logiciel DOS/Windows (mais la plupart des cartes ne le font pas) ;
- la section intitulée « Bus ISA : Isapnp (outil faisant partie d'isapnptools) » [p 28] est un programme que vous pouvez toujours utiliser pour configurer les périphériques PnP du bus ISA (seulement),
- la section intitulée « Les utilitaires PCI » [p 29] permet de configurer le bus PCI mais le pilote de périphérique devrait le gérer ;
- la section intitulée « Configuration de Windows » [p 29] et alors vous démarrez Linux à partir de Windows/DOS. A utiliser en dernier recours.

N'importe lequel configurera les ressources bus au niveau matériel mais seul le premier (voire le second) indiquera au pilote ce qui a été fait. La façon dont le pilote est informé dépend du pilote. Vous pouvez avoir besoin de faire quelque chose pour l'informer. Voir la section intitulée « Indiquer au pilote la configuration ?? » [p 30] .

Configuration du pilote de périphérique, réservation des ressources

Les pilotes de périphériques (avec l'aide de fonctions du noyau) peuvent être écrits pour utiliser des méthodes PnP pour configurer les ressources bus du matériel mais seulement pour le périphérique qu'ils contrôlent. Mais beaucoup de pilotes de périphériques acceptent directement ce que le BIOS ou Linux a configuré et utilise le code fourni par le noyau pour découvrir comme ce périphérique a été configuré. Comme le pilote a vérifié la configuration et la certainement reconfiguré, il connaît de façon évidente la configuration et il n'y a aucun besoin de lui donner cette information. C'est dont la façon la plus simple de le faire car vous n'avez rien à faire si le pilote fait tout.

Si vous avez un matériel datant d'avant l'ISA PnP, le logiciel PnP Linux pourrait ne pas le savoir et pourrait ne pas connaître les ressources bus qu'il réclame. Donc, il pourrait allouer de façon erronée des ressources dont cet ancien matériel a besoin à un autre périphérique. Le résultat est un conflit de ressources mais il existe un moyen de l'éviter. Vous pouvez réserver les ressources dont cette ancienne carte ISA a besoin en configurant le BIOS au démarrage (habituellement), au module isa-pnp ou au noyau (si le support de PnP est intégré dans le noyau). Par exemple, pour réserver l'IRQ 5, donnez cet argument au module isa-pnp (ou au noyau) : isapnp_reserve_irq=5. Voir le Guide pratique sur l'invite de démarrage (*BootPrompt-HOWTO*). Au lieu de ..._irq, il existe aussi _io, _dma et _mem.

Pour les périphériques PCI, la plupart des pilotes configureront PnP. Malheureusement, un pilote peut récupérer des ressources bus nécessaires à d'autres périphériques (mais non alloués à eux par le noyau). Donc, un noyau Linux PnP plus perfectionné serait meilleur là où le noyau fait l'allocation pour toutes les demandes envoyées. Voir la section intitulée « Comment Linux gère-t-il le PnP » [p 18] .

/sys : interface de configuration pour l'utilisateur

Depuis le noyau 2.6, il existe une nouvelle façon pour que l'utilisateur configure les ressources grâce au répertoire /sys. Mais, jusqu'à août 2004, il ne peut pas être utilisé pour une configuration dans la plupart des cas. Voir la section intitulée « Le répertoire /sys » [p 35] .

Configuration du BIOS

Introduction à l’utilisation de la configuration PnP faite par le BIOS

Si vous avez un BIOS PnP, il peut configurer le matériel. Si le pilote ne peut pas le faire, le BIOS le peut probablement. Ceci veut dire que votre BIOS lit les besoins en ressources de tous les périphériques et les configure (en leur allouant les ressources bus). C’est un substitut pour l’OS PnP sauf que le BIOS ne peut faire correspondre les pilotes avec leur périphériques et ne peut pas non plus indiquer aux pilotes la façon dont il a configuré les périphériques. Il devrait normalement utiliser la configuration enregistrée dans sa mémoire non volatile (ESCD). S’il trouve un nouveau périphérique ou s’il existe un conflit, le BIOS devra effectuer les changements nécessaires et pourrait ne pas utiliser la même configuration que celle de l’ESCD. Dans ce cas, il devra mettre à jour l’ESCD pour refléter la situation.

Votre BIOS doit gérer une telle configuration, mais il existe des cas où il ne le fait pas correctement ou pas complètement. Le BIOS a aussi besoin de savoir via le menu CMOS si le système d’exploitation est PnP. Alors que la plupart des pilotes de périphériques seront capables de détecter automatiquement ce que le BIOS a fait, dans certains cas, vous aurez besoin de le déterminer (ce qui n’est pas toujours facile). Voir la section intitulée « Comment puis-je trouver les périphériques et comment sont-ils configurés ? » [p 32] . Un avantage possible à laisser le BIOS faire cette configuration est qu’il fait son boulot avant de lancer Linux, donc c’est fait très tôt dans le processus de démarrage.

La plupart des BIOS créés après 1996 ?? peuvent configurer les ressources des bus PCI et ISA. Mais, il a été dit que certains anciens BIOS peuvent uniquement s’occuper du PCI. Pour essayer d’en savoir plus sur votre BIOS, cherchez sur le web. Merci de ne pas me demander car je n’ai pas toutes les données là-dessus. Les détails du BIOS que vous souhaitez connaître peuvent être difficiles à trouver. Certains BIOS pourraient avoir des capacités PnP minimales et attendre que le système d’exploitation fasse la configuration PnP. Si cela arrive, vous devrez soit trouver une autre méthode soit essayer d’enregistrer les informations dans la base de données ESCD si le BIOS en a une. Voir la prochaine section.

La base de données ESCD du BIOS

Le BIOS maintient une base de données non volatile contenant la configuration PnP qu’il essaiera d’utiliser (si vous aviez indiqué qu’il ne s’agit pas d’un système d’exploitation PnP). Elle s’appelle l’ESCD (acronyme pour Extended System Configuration Data, soit Données pour une Configuration Étendue du Système). Encore une fois, l’ESCD est optionnel mais la plupart des BIOS PnP en disposent. L’ESCD enregistre non seulement la configuration des ressources pour les périphériques PnP mais aussi celle des périphériques non PnP (et les indique en tant que tels) pour éviter les conflits. Les données de l’ESCD sont habituellement enregistrées sur un composant et restent intactes lorsque la machine est arrêtée, mais c’est parfois stocké sur un disque dur ??

L’ESCD a pour but de conserver la dernière configuration utilisée. Mais comme Linux peut modifier la configuration des périphériques (en incluant l’utilisateur avec les outils PCI ou isapnp), l’ESCD ne sera pas au courant de cette modification et ne sauvegardera pas cette configuration. Un bon système d’exploitation PnP devrait mettre à jour l’ESCD, pour que les informations qui y sont stockées puissent être utilisées par un système d’exploitation non PnP (comme un Linux standard). MS Windows 9x ne le fait que dans certains précis. Voir la section intitulée « Utiliser Windows pour configurer l’ESCD » [p 26] . À partir du noyau 2.6, Linux est capable de modifier l’ESCD mais cela n’est pas encore utilisé (août 2004).

Pour utiliser ce qui a été enregistré dans l'ESCD, assurez-vous d'avoir bien spécifié que l'OS n'est pas PnP dans le CMOS du BIOS. Par la suite, à chaque fois que le BIOS démarre (avant que l'OS Linux ne soit chargé), il devrait tout configurer de cette façon. Si le BIOS détecte une nouvelle carte PnP non indiquée dans l'ESCD, alors il allouera des ressources bus à la carte et mettra à jour l'ESCD. Il pourrait même changer les ressources bus assignées aux cartes PnP existantes et modifier l'ESCD de manière concordante.

Un programme vous permet de visualiser le contenu de l'ESCD. Il affiche les IRQ, les adresses d'entrées/sorties, et cætera mais les noms de périphériques manquent (seulement les numéros d'identifiant des périphériques EISA). Il est disponible sur l'index de /home/gunther.mayer/lsescd.

Si chaque périphérique sauvegardait sa dernière configuration au niveau du matériel, la configuration matérielle ne serait pas nécessaire à chaque démarrage du PC. Mais cela ne fonctionne pas ainsi. Donc, toutes les données de l'ESCD ont besoin d'être actualisées si vous utilisez le BIOS pour PnP. Il existe des BIOS ne disposant pas d'ESCD mais ayant une mémoire non volatile pour stocker des informations concernant l'attribution des ressources bus aux cartes non PnP. Beaucoup de BIOS disposent des deux.

Utiliser Windows pour configurer l'ESCD

Éventuellement, Linux pourrait initialiser l'ESCD. Depuis Linux 2.6, une fonction du nouveau code pourrait le faire si le noyau a été compilé avec PNPBIOS. Mais elle reste pour l'instant inutilisée.

Si le BIOS ne configure pas l'ESCD de la façon souhaitée (ou de la bonne façon), alors il serait bien de disposer d'un utilitaire Linux pour le faire. Donc, vous pourriez vouloir utiliser Windows (si vous l'avez sur le même PC) pour faire cela.

Il existe trois façons d'utiliser Windows pour tenter de modifier l'ESCD. La première est d'utiliser l'utilitaire ICU pour DOS ou Windows 3.x. Il devrait aussi fonctionner pour Windows 9x/2k ?? Une autre façon est de configurer les périphériques manuellement (« en forçant ») sous Windows 9x/2k de façon à ce que Windows enregistre les informations dans l'ESCD lorsque Windows est arrêté normalement. La troisième façon est possible uniquement pour les périphériques non PnP. Si Windows connaît quelque chose sur eux, notamment quelles ressources bus ils utilisent, alors Windows enregistrera cette information dans l'ESCD.

Si les périphériques PnP sont configurés automatiquement par Windows sans que l'utilisateur ait besoin de forcer cette reconnaissance, alors ces paramétrages ne se trouveront probablement pas dans l'ESCD. Bien sûr, Windows pourrait bien décider de lui-même de configurer ce qui est enregistré dans l'ESCD, ce qui pourrait aboutir au même par coïncidence.

Windows 9x est un système d'exploitation PnP et configure automatiquement via PnP les périphériques. Il maintient leur propre base de données PnP dans la base de registre (fichiers binaires de Windows). Beaucoup d'autres données de configuration résident dans la base de registre en plus des ressources bus PnP. Il y a à la fois une configuration des ressources PnP actuelles et une autre (peut-être la même) enregistrée sur le disque dur. Pour voir ça avec Windows 98, ou pour forcer l'enregistrement des modifications, utilisez le gestionnaire des périphériques.

Dans Windows 98, il existe deux façons d'arriver au gestionnaire des périphériques :

- 1. Poste de travail --> Panneau de configuration --> Système --> Gestionnaire de périphériques

- 2. (clic droit) Poste de travail --> Propriétés --> Gestionnaire de périphériques.

Ensuite, dans ce gestionnaire, vous sélectionnez un périphérique (parfois un processus en plusieurs étapes s'il existe plusieurs périphériques de la même classe). Ensuite, cliquez sur « Propriétés » puis « Ressources ». Pour essayer de modifier la configuration des ressources manuellement, décochez « Utilisez la configuration automatique » puis cliquez sur « Changer la configuration ». Maintenant, essayez de modifier les paramétrages. Il peut ne pas vous laisser les modifier. S'il vous le permet, vous avez « forcé » un changement. Du coup, un message devrait vous avertir que vous avez forcé cette modification. Si vous souhaitez garder le paramétrage existant affiché par Windows, mais que vous voulez forcer, alors vous devrez forcer un autre changement et de nouveau forcer sa modification en sa valeur précédente.

Pour voir ce qui a été forcé sous Windows 98, regardez la liste des matériels « forcés » : Démarrer --> Programme --> Accessoires --> Outils système --> Information système --> Ressources matérielles --> Matériel forcé. Lorsque vous « forcez » un changement des ressources bus dans Windows, il devrait enregistrer votre modification dans l'ESCD (à condition que vous ayez quitté Windows normalement). À partir de la fenêtre « Informations système », vous pouvez aussi voir comment les IRQ et les ports d'entrées/sorties ont été alloués par Windows.

Même si Windows ne montre aucun conflit des ressources bus, il peut exister un conflit sous Linux. Ceci est dû au fait que Windows peut affecter des ressources bus différentes de celles de l'ESCD. Dans le cas rare où les périphériques sous Windows sont soit non PnP soit « forcés », alors la configuration Windows et celle de l'ESCD devraient être les mêmes.

Ajouter un nouveau périphérique (sous Linux ou Windows)

Si vous ajoutez un nouveau périphérique PnP et avez configuré le BIOS à « pas un système d'exploitation PnP », alors le BIOS devrait automatiquement le configurer et enregistrer la configuration dans l'ESCD. S'il ne s'agit pas d'un périphérique non PnP (ou un utilisant les cavaliers), alors il existe quelques options pour le gérer.

Vous pouvez indiquer directement au BIOS (via le menu de configuration CMOS) que certaines ressources bus qu'il utilise sont réservées et ne peuvent pas être allouées avec PnP. Ceci ne met pas cette information dans l'ESCD. Il existe un menu de sélection du BIOS permettant d'indiquer si les choix CMOS supplantent ceux de l'ESCD en cas de conflit. Une autre méthode revient à lancer ICU sous DOS/Windows. Encore une autre permet de l'installer manuellement sous Windows 9x/2k puis de s'assurer que cette configuration est « forcée » (voir la section précédente). Si elle l'est, Windows devrait mettre à jour l'ESCD à l'arrêt du PC.

ISA seulement : Désactiver PnP ?

Les périphériques PCI sont PnP à la base donc cela ne peut pas être désactivé. Mais quelques périphériques ISA ont des options pour désactiver PnP par l'intermédiaire de cavaliers ou en lançant un programme Windows fourni avec le périphérique (configuration logicielle). Si le pilote du périphérique ne peut pas le configurer, ceci évitera la tâche probablement compliquée de la configuration PnP. N'oubliez pas de dire au BIOS que ces ressources bus sont réservées. Mais comme le support de Linux pour le PnP a été amélioré, vous ne voulez généralement pas désactiver PnP. Voici quelques arguments pour lesquels vous ne voudrez pas désactiver PnP :

- Si vous avez Windows sur la même machine, alors vous pouvez permettre à PnP de configurer les périphériques différemment entre Windows et Linux.
- L'ensemble des choix pour les numéros d'IRQ (ou ports d'adresse) peut être assez limité sauf si vous utilisez PnP.
- Vous pourriez avoir un pilote de périphérique Linux utilisant des méthodes PnP pour rechercher le périphérique qu'il contrôle.
- Si vous avez besoin de modifier la configuration plus tard, il serait plus facile de faire ceci avec PnP (sans utiliser de cavaliers ou d'avoir à lancer un programme Dos/Windows).

Une fois vos périphériques configurés sans PnP, ils ne peuvent plus être configurés par un logiciel PnP ou par un BIOS PnP (jusqu'à ce que vous changiez les cavaliers ou utilisiez le logiciel de configuration Dos/Windows).

Bus ISA : Isapnp (outil faisant partie d'isapnptools)

Le programme **isapnp** est utilisé uniquement pour les périphériques PnP du bus ISA (donc non PCI). Il était vraiment nécessaire avant les noyaux 2.4. Avec le noyau 2.4, qui a apporté des fonctionnalités permettant aux pilotes de gérer le PnP sur le bus ISA, le programme **isapnp** devient moins important. De plus, le BIOS pourrait configurer ISA PnP de manière satisfaisante. Mais, le module isa-pnp (ou l'équivalent intégré au noyau) est déjà très satisfaisant car de nombreux pilotes de périphériques ISA l'appellent pour configurer les ressources du bus. Avant le noyau 2.6, cela résultait en un « fichier » `/proc/isapnp` pouvant être utilisé pour configurer manuellement (voir isapnp.txt dans la documentation du noyau).

Dans certains cas, les distributions Linux ont été configurées pour lancer **isapnp** automatiquement au démarrage. Il est toujours utilisé en 2004 mais il n'est pas réellement nécessaire si les pilotes de périphériques fonctionnent bien. Si vous avez besoin de le configurer vous-même, la grande partie de la documentation d'isapnp est difficile à comprendre sauf si vous possédez des notions de base de PnP. Ce guide pratique devrait vous aider à le comprendre, ainsi que la FAQ qui accompagne **isapnp**. Lancer le programme **isapnp** au démarrage vous permettra de configurer ces périphériques suivant les valeurs spécifiées dans `/etc/isapnp.conf`. Il est possible de créer ce fichier de configuration automatiquement mais vous devrez alors l'éditer manuellement pour choisir entre les différentes options. Puis pour que le pilote connaisse ces ressources, vous avez souvent besoin de les spécifier en tant que paramètres pour les modules appropriés (pilotes). Ceci se fait avec des fichiers de configuration, généralement dans le répertoire `/etc`. Cherchez-y des fichiers nommés `mod*`, et cætera. Si le pilote est intégré au noyau, alors ils pourraient parfois être donnés comme paramètre du noyau. Voir le guide pratique des options de démarrage.

Avec **isapnp**, il existait un risque qu'un pilote de périphérique, intégré au noyau, soit lancé trop tôt, avant qu'**isapnp** n'ait pu configurer les adresses, et cætera au niveau matériel. En conséquence, le pilote de périphérique ne serait plus capable de trouver le périphérique. Le pilote essaie la bonne adresse mais cette adresse n'est pas configurée au niveau matériel. Cela est-il toujours un problème ??

Si votre distribution Linux a automatiquement installé **isapnptools**, **isapnp** est probablement lancé au démarrage. Dans ce cas, il ne vous reste qu'à éditer `/etc/isapnp.conf` suivant **`man isapnp.conf`**. Notez que cela revient à configurer manuellement PnP car vous prendrez les décisions sur la façon de configurer lors de l'édition du fichier de configuration.

Si le fichier de configuration est mauvais ou n'existe pas, vous pouvez utiliser le programme **pnpdump** pour vous aider à créer le fichier de configuration. Il crée pour vous un fichier de configuration mais vous devrez l'éditer avec intelligence avant de l'utiliser. Il contient quelques commentaires pour vous aider. Alors que le BIOS pourrait aussi avoir configuré les périphériques ISA (si vous lui avez dit que vous ne disposez pas de système d'exploitation PnP), **isapnp** le refera.

La terminologie utilisée dans le fichier `/etc/isapnp.conf` peut sembler étrange au début. Par exemple, pour une adresse d'entrée/sortie 0x3e8, vous pourriez voir « (IO 0 (BASE 0x3e8)) » à la place. « IO 0 » veut dire qu'il s'agit de la première plage d'adresses que ce périphérique utilise. Une autre façon d'exprimer ceci serait : « IO[0] = 0x3e8 » mais **isapnp** ne le fait pas de cette façon. « IO 1 » voudrait dire qu'il s'agit de la deuxième plage d'adresse utilisée par ce périphérique, et cætera. « INT 0 » a une signification similaire mais pour les IRQ (interruptions). Une carte simple peut contenir plusieurs périphériques physiques mais l'explication ci-dessus était seulement pour un des périphériques.

Les utilitaires PCI

Le paquetage des utilitaires PCI (**pciutils**, quelque fois appelé « pcitools ») vous permet de configurer manuellement via PnP le bus PCI (avec difficulté). **lspci** ou **scanpci** liste les ressources bus alors que **setpci** enregistre les allocations des ressources (sauf les IRQ) dans les périphériques physiques. Il semble que **setpci** soit principalement utilisé dans des scripts et en fait, vous aurez besoin de comprendre le détail des registres de configuration du PCI pour pouvoir l'utiliser. Ce thème n'est pas expliqué ici, et pas plus dans la page de manuel de **setpci**.

Les gens l'ont utilisé pour configurer les périphériques PCI dont le pilote a échoué dans cette action. Un exemple est disponible dans le guide pratique sur les modems et le guide pratique sur les ports séries dans la sous-section « PCI : Activer un port désactivé ». Néanmoins, activer un périphérique n'est d'aucune utilité si vous n'avez pas de pilote fonctionnel pour ce périphérique.

Configuration de Windows

Cette méthode utilise MS Windows pour configurer et devrait être utilisée seulement si tout le reste échoue. Si vous avez Windows 9x (ou 2k) sur le même PC, alors lancez simplement Windows et laissez-le configurer PnP. Puis lancez Linux à partir de Windows (ou DOS) en utilisant, par exemple, **loadlin.exe**. Il peut y avoir un problème avec les IRQ pour les périphériques PCI. Quand Windows s'arrête (sans messages) pour laisser la place à Linux, il pourrait écraser l'IRQ (en y mettant 0) qui est stocké dans un des registres de configuration du périphérique PCI. Linux se plaindra de trouver une IRQ 0.

Ce qu'on vient d'aborder arrive lorsque vous lancez Linux en utilisant un raccourci (fichier PIF). Mais un moyen de contourner ce problème est connu si vous utilisez toujours le raccourci PIF. Un raccourci est en quelque sorte l'équivalent du lien symbolique sous Linux, mais il est en fait plus que ça car il est paramétrable. Pour lancer Linux, à partir de DOS, vous créez un fichier batch (script) qui lance Linux. (Le programme qui lance Linux est dans le paquet appelé loadlin.) Ensuite, créez un raccourci PIF vers ce fichier batch et allez dans la fenêtre des propriétés du raccourci. Sélectionnez « Avancé », puis vérifiez que le « mode MS-DOS » est bien coché.

Maintenant, voici une astuce empêchant de mettre à zéro les IRQ PCI. Cochez « Spécifier une nouvelle configuration MS-DOS ». Ensuite, soit vous acceptez la configuration par défaut qui vous est proposée soit vous cliquez sur « Configuration » pour la modifier. Maintenant, lorsque vous lancerez Linux en cliquant sur le raccourci, des nouveaux fichiers de configurations (`config.sys` et `autoexec.bat`) seront créés pour votre nouvelle configuration.

Les anciens fichiers sont enregistrés comme `Config.wos` et `Autoexec.wos`. Une fois que vous avez terminé d'utiliser Linux et que vous avez arrêté votre PC, vous aurez encore besoin de ces fichiers pour pouvoir lancer DIS la prochaine fois que vous démarrerez votre PC. Vous devez vous assurer que les noms redeviennent `*.sys` et `*.bat`. Lorsque vous quittez Windows/DOS pour aller sous Linux, Windows s'attend que, une fois que vous avez fini avec Linux, vous retourniez à Windows pour que celui-ci puisse restaurer ces fichiers avec leur noms originaux. Mais ceci n'arrivera pas car lorsque vous quitterez Linux, vous éteindrez votre PC et ne retournerez pas sous Windows. Donc, comment renommer ces fichiers ? C'est facile, placez ces commandes dans votre fichier batch de lancement de Linux pour qu'il renomme les fichiers. Mettez ces commandes de renommage dans votre fichier batch juste avant la ligne qui charge Linux.

De la même façon, il a été rapporté que vous devez cliquer sur l'onglet « Général » (de la fenêtre « Propriétés » de votre raccourci) et cochez « Lecture seule ». Sinon, Windows pourrait remettre à zéro les « Paramétrages avancées » en « Utilisez la configuration MS-DOS courante » et les IRQ PCI se retrouveraient à zéro. Comme Windows efface les IRQ lorsque vous utilisez la configuration MS-DOS courante mais il n'efface pas une nouvelle configuration (qui peut configurer tout de manière identique à l'ancienne configuration). Windows ne semble pas très cohérent.

Documents/Logiciels PnP

- Page d'accueil d'**Isapnptools** ;
- Proposition pour un gestionnaire de configuration pour Linux 1999 (n'a jamais fait partie du noyau) ;
- Spécifications PnP de Microsoft ;
- Livre : PCI System Architecture, quatrième édition par Tom Shanley +, MindShare 1999. Couvre les fonctionnalités PnP du bus PCI ;
- Livre : Plug and Play System Architecture, par Tom Shanley, Mind Share 1999. Détails sur PnP pour le bus ISA. Une vue de PnP avec le bus PCI ;
- Livre : Programming Plug and Play, par James Kelsey, Sams 1995. Détails sur la programmation pour communiquer avec un BIOS PnP. Couvre les bus ISA, PCI et PCMCIA.

Indiquer au pilote la configuration ??

Introduction

Un pilote moderne trouvera pour un périphérique la configuration des ressources bus sans que vous ayez besoin de lui dire quoi que ce soit. Il pourrait même enregistrer les ressources bus au niveau matériel en utilisant des méthodes PnP. Certains périphériques ont plus d'une façon pour trouver comment leur périphérique physique est configuré. Dans le pire des cas, vous devez coder en dur les ressources bus dans le noyau (ou un module) et recompiler.

En un juste milieu, il existe des cas tels que le lancement d'un programme pour donner les informations des ressources bus au pilote ou pour mettre les informations dans un fichier de configuration. Dans certains cas, le pilote peut chercher le périphérique aux adresses où il suppose qu’il réside (mais il ne trouvera jamais un périphérique PnP s’il n’a pas été activé par des méthodes PnP). Il peut même essayer de tester différentes IRQ pour voir laquelle fonctionne. Il peut, ou non, le faire automatiquement.

Dans d’autres cas, le pilote peut utiliser des méthodes PnP pour trouver le périphérique et la façon dont les ressources bus ont été configurées par le BIOS, et cætera mais ne les modifiera pas. Il peut aussi regarder dans certains des « fichiers » du répertoire `/proc`.

Il peut aussi dire « manuellement » au pilote les ressources bus qu’il doit utiliser. Vous donnez ces ressources bus en tant que paramètre au noyau ou à un module. Si le pilote est intégré au noyau, vous passez les paramètres au noyau via l’invite du démarrage. Voir le Guide pratique sur l’invite de démarrage (*BootPrompt-HOWTO*) pour la description de quelques ressources bus et autres paramètres. Une fois que vous savez quels paramètres donner au noyau, vous pouvez les enregistrer dans un fichier de configuration du chargeur. Par exemple, mettez **`append="..."`** dans le fichier `lilo.conf` puis lancez lilo pour qu’il mette à jour les informations de lancement.

Si le pilote est chargé comme module, dans plusieurs cas, le module trouvera les ressources bus nécessaires et les enregistrera dans le périphérique. Dans les autres cas (généralement pour les anciens PC), vous pouvez avoir besoin de donner les ressources bus comme paramètres du module. Les paramètres d’un module peuvent être spécifiés dans `/etc/modules.conf`. Ce sont généralement des outils utilisé pour modifier ce fichier et qui sont dépendant de la distribution. Les commentaires inclus dans ce fichier devraient vous aider sur la façon de le modifier. De même, tout module que vous placez dans `/etc/modules` se verra charger avec ses paramètres.

Bien qu’il ait une grande hétérogénéité sur la façon dont les pilotes trouvent leur ressources bus, le but final est le même. Si vous avez des problèmes avec un pilote, vous pouvez avoir besoin de regarder la documentation du pilote (vérifier la documentation du noyau). Quelques exemples brefs de pilotes sont présentés dans les sections suivantes :

Exemple de pilote de port série

Pour les ports séries PCI (et pour les ports série ISA PnP après le noyau 2.4), le pilote série détecte le type de port série et le configure via PnP. Malheureusement, quelques ports série PCI ne sont pas encore gérés.

Pour le pilote du port série ISA standard avec les anciennes versions du noyau et pour le pilote série (ne faisant pas partie des cartes multiports), le pilote travaille sur deux adresses standards pour les ports série. Il ne cherche pas d’IRQ mais il affecte l’IRQ « standard » aux deux premiers ports séries. Ceci peut être mauvais.

Pour tout autre chose dans le fichier de configuration, le programme **setserial** doit être modifié manuellement. Voir le Guide pratique sur la programmation des ports série pour plus de détails. Vous utilisez **setserial** pour informer le pilote de l’adresse d’entrée/sortie et **setserial** est souvent exécuté à partir d’un fichier de démarrage. Dans les versions récentes, il existe un fichier `/etc/serial.conf` (ou `/var/lib/setserial/autoconfig`) que vous « éditez » en lançant simplement la commande **`setserial`** de façon ordinaire et ce que vous configurez avec **setserial** est sauvegardé dans le fichier de configuration `serial.conf`. Le fichier `serial.conf` devrait être consulté lorsque la commande **setserial** est lancée à partir d’un fichier de démarrage. Votre distribution peut, ou non, avoir configuré ceci pour vous.

Il existe deux façons d'utiliser **setserial** suivant les options que vous lui donnez. Une possibilité est de dire manuellement au pilote la configuration. L'autre méthode est de tester une adresse donnée et d'indiquer si un port série existe à cet endroit. Il peut aussi tester cette adresse et essayer de détecter l'IRQ utilisée par ce port.

Même avec des noyaux modernes, **setserial** est quelque fois nécessaire si le pilote échoue lors de la détection du port série ou si vous avez un très ancien matériel.

Comment puis-je trouver les périphériques et comment sont-ils configurés ?

La recherche des périphériques et la découverte de la configuration sont liés

Une fois que vous avez trouvé votre matériel, le même programme qui l'a trouvé vous indique normalement comment il est configuré. Donc, trouver comment un matériel est configuré revient habituellement à trouver le matériel.

Les périphériques pourraient avoir deux « configurations »

Ici, « configuration » correspond à l'assignation des ressources bus PnP (adresses, IRQ et DMA). Pour chaque périphérique, il existe deux parties à cette question de configuration :

- Que croit le pilote en ce qui concerne la configuration matérielle ?
- Quelle configuration (s'il y en a une) est réellement enregistrée au niveau du matériel ?

Chaque partie devrait avoir la même réponse (la même configuration). La configuration du périphérique physique et de son pilote devrait être évidemment la même (elle l'est habituellement). Mais les choses ne fonctionnent pas toujours bien et c'est pourquoi il existe une différence. Ceci veut dire que le pilote dispose d'une mauvaise information sur la configuration actuelle du matériel. Les problèmes arrivent. Si le logiciel que vous utilisez ne vous dit pas exactement ce qui ne va pas (ou ne configure pas automatiquement correctement), alors vous avez besoin de chercher comment vos périphériques physiques et vos pilotes sont configurés. Alors que les pilotes de périphériques Linux devraient tout vous dire, dans certains cas, il n'est pas facile de déterminer ce qui a été enregistré au niveau matériel.

Un autre problème est que lors de la visualisation des messages de configuration à l'écran, vous devez savoir si la configuration affichée est celle du pilote, du périphérique physique ou des deux. Si le pilote de périphérique a soit enregistré la configuration soit vérifié le matériel, alors le pilote devrait avoir les bonnes informations.

Mais, quelquefois, le pilote a obtenu des ressources incorrectes par un script ou un fichier de configuration, par des paramètres de ressources incorrectes données à un module, ou peut-être même que la configuration ne lui a pas été fournie complètement et qu'il essaie d'utiliser par défaut des ressources incorrectes. Par exemple, quelqu'un peut utiliser **setserial** pour indiquer au pilote de périphérique une configuration incorrecte des ressources et le pilote les acceptera sans broncher. Mais le port série ne fonctionne pas bien (s'il fonctionne tout court).

Trouver le matériel

Un problème habituel est que le logiciel ne détecte pas votre périphérique ou ne détermine pas le bon pilote pour celui-ci. Pour les périphériques PnP, les détecter est facile avec un logiciel PnP sauf pour le cas inhabituel où le matériel a été désactivé. Le BIOS peut parfois être initialisé pour désactiver les périphériques PnP ou un cavalier/interrupteur sur le périphérique physique lui-même pourrait le désactiver. Dans ce cas, le matériel ne peut pas être détecté du tout jusqu'à ce que vous re-configuriez le BIOS ou que vous changiez le cavalier/interrupteur.

Comme le bus PCI est intrinsèquement PnP, il n'y a pas de périphériques cachés. Même si les périphériques PnP sont faciles à trouver avec les méthodes PnP, si le pilote n'utilise pas les méthodes PnP mais utilise l'ancienne méthode de recherche aux adresses supposées, ils pourraient ne pas être trouvés. Ceci est dû au fait que, avant que les ressources bus aient été initialisées (par le BIOS ou Linux), le périphérique pourrait ne pas avoir d'adresses du tout, donc parcourir les adresses habituelles n'apportera rien. Pour l'ancien bus ISA, quelques-uns des périphériques pourraient être non PnP et, de ce fait, les anciennes méthodes pourraient fonctionner. Donc, de nombreux pilotes continuent à parcourir les adresses habituelles en plus d'utiliser les méthodes PnP (parcours PnP, quelquefois appelé simplement parcours).

Façons de détecter des périphériques matériels (et leur configurations) : (suivre le lien pour plus de détails)

- Vérifier le BIOS pour vous assurer qu'ils ne sont pas désactivés ;
- Regarder les messages de démarrage à l'écran (voir les messages au démarrage [p 33]) ;
- Regarder dans la section intitulée « Le répertoire /proc » [p 34] ;
- Outils pour détecter et configurer tout le matériel [p 38] ... lsdev, hwinfo, discover, kudzu ;
- Outils pour détecter ou configurer un type de matériel [p 39] ;
- PCI : la section intitulée « Inspection du bus PCI » [p 36]
- Bus ISA : la section intitulée « Introduction au bus ISA » [p 36]
- Bus ISA : la section intitulée « Cartes ISA PnP » [p 37]
- Bus ISA : la section intitulée « Cartes non PnP » [p 37]
- Bus ISA : la section intitulée « Cartes non PnP avec cavaliers » [p 38]
- Bus ISA : la section intitulée « Cartes non PnP et sans cavaliers » [p 38]
- Et la section intitulée « Utilisez MS Windows » [p 39]

Messages de démarrage

Des informations significatives sur la configuration peuvent être obtenues en lisant les messages affichés par le BIOS et par Linux lors du démarrage de l'ordinateur. Ces messages disparaissent souvent trop rapidement pour être lus mais, une fois le défilement terminé, tapez **Majuscule-Page Suivante** plusieurs fois pour revenir en arrière. Pour aller en avant, faites **Majuscule-Page**

Précédente. Utilisez **dmesg** à la console à n'importe quel moment pour afficher seulement les messages du noyau. Vous ne verrez pas certains messages les plus importants provenant généralement du BIOS. Les messages affichés par Linux peuvent parfois n'afficher que ce que le pilote de périphérique croit être configuré, peut-être à cause d'un mauvais fichier de configuration. Vérifier les fichiers de traces dans `/var/log` peut être utile.

Pour le bus PCI, la notation 00:1a:0 signifie le bus PCI 00 (le bus PCI principal), la carte PCI (ou le composant) 1a et la fonction 0 (le premier périphérique) sur la carte ou le composant. Le deuxième périphérique sur la carte (ou sur le composant) devrait être 00:08:1.

Les messages du BIOS s'affichent en premier et montrent la configuration du matériel à ce moment, mais **isapnp**, ou les utilitaires PCI, voire les pilotes de périphériques peuvent le changer plus tard. Dans certains cas, il n'affiche pas les périphériques que le BIOS n'a pas configuré.

Si les messages du BIOS ne s'affichent pas en revenant au début des messages du BIOS avec **Majuscule**-**Page Suivante**, essayez de mettre en pause lorsqu'ils apparaissent, en utilisant la touche **Pause** dès que les premiers mots apparaissent. Appuyez sur n'importe quelle touche pour continuer. Il est souvent difficile d'appuyer sur **Pause** exactement au bon moment. Assurez-vous d'appuyer continuellement sur la touche **Shift** avant d'appuyer sur **Pause** car **Pause** est une touche nécessitant l'utilisation de **Shift**. Si vous n'avez pas réussi, appuyez sur **Ctrl**-**Alt**-**Del** au lancement de Linux pour le relancer et essayer de nouveau. Une fois que les messages de Linux commencent à être visibles, il est trop tard pour utiliser **Pause** car cela ne gèlera pas les messages de Linux.

Pour initialiser des éléments du BIOS comme les IRQ réservés au matériel propriétaire, aux adresses des ports série, et cætera vous aurez besoin d'entrer dans les menus de configuration du BIOS (CMOS) au lancement. Chaque marque de BIOS a différentes touches pour ce faire. Il existe des listes sur Internet. Parfois en gelant l'affichage des messages du BIOS ou en regardant l'écran, la touche que vous devez appuyer sera indiquée dans un message tel que « Press DEL for setup » (« Appuyez sur DEL pour la configuration »). Mais il pourrait disparaître si rapidement que vous ne le verrez pas. Bien sûr, vous ne modifiez rien dans le BIOS que vous ne comprenez pas. Le cas contraire, votre PC pourrait être désactivé.

Les messages du BIOS au démarrage vous indiquent ce que fut la configuration du matériel. La configuration actuelle pourrait toujours être la même que ce qu'a fait le BIOS et que Linux devrait accepté si c'est bon. Les messages de Linux pourraient provenir des pilotes utilisant les fonctions PnP du noyau pour inspecter ou configurer les ressources bus. Cela devrait être correct mais attention aux messages qui affichent seulement ce que le pilote a lu de son fichier de configuration. Cela pourrait être faux. Bien sûr, si le périphérique fonctionne bien, alors il est configuré probablement de la même façon par le pilote.

Le répertoire /proc

Depuis le noyau 2.6, il existe aussi un répertoire `/sys` en plus du répertoire `/proc`. Ces répertoires sont utiles pour récupérer les configurations des ressources et périphériques. Les fichiers qu'ils contiennent représentent des données provenant de la mémoire du noyau et n'existent pas du tout sur votre disque dur. Les programmes comme `lspci` récupèrent leurs informations du répertoire `/proc` et ils doivent les afficher d'une façon plus lisible qu'en lisant directement le contenu des fichiers de ce répertoire. Voici quatre fichiers provenant de `/proc` qui montrent les ressources enregistrées dans le noyau par les pilotes de périphériques.

Comme le Plug-And-Play de Linux fonctionne en laissant les pilotes de périphériques alloués les ressources pour leur périphérique, il pourrait ne pas y avoir de ressources utilisées par certains de vos matériels si le pilote n'a pas encore réclamé que ces ressources lui soient réservées. Pour les cas des modules du noyau (pilotes de périphérique chargeables), si le module n'est pas encore chargé, le noyau ne connaît pas les ressources dont le module a besoin. Quelque fois, le module se charge seulement quand vous lancez une application qui en a besoin. Donc, si un certain matériel est manquant parmi les fichiers de `/proc`, cela pourrait signifier que le matériel n'a pas encore été utilisé. Par exemple, même si votre lecteur de disquette dispose d'une disquette et est prêt à être utilisé, son interruption n'apparaîtra tant que le lecteur n'est pas utilisé.

`/pts` affiche les adresses d'entrée/sortie. S'il y a une erreur (mauvaise adresse), cela pose problème car le périphérique n'obtiendra pas les octets qui lui sont envoyés. `/proc/iomem` affiche les adresses mémoires d'entrée/sortie qui sont réservées. `/proc/interrupts` affiche les interruptions en cours d'utilisation. `/proc/dma` affiche les allocations de canaux DMA pour le bus ISA.

Dans le passé, l'auteur a observé une liste d'interruptions qui n'existaient pas. Dans certains cas, cela montrait que quelques interruptions étaient vraiment envoyées. Ceci peut être dû à des matériels défectueux envoyant des interruptions erronées.

`/proc/bus/` contient les sous-répertoires `/input/`, `/pci/` et `/isapnp/`. Le format de la plupart des fichiers dans ce répertoire est vraiment cryptique, souvent une simple copie des octets de l'espace de configuration. Donc, utilisez-les seulement en dernier ressort. Le sous-répertoire `input/` a des informations sur les périphériques en entrée comme le clavier ou la souris. Elles ne sont pas aussi complexes que les autres répertoires sous `/proc/bus` et pourraient fournir des informations utiles sur les périphériques d'entrées qui sont sur les ports PS2 ou sur le bus LPC (voir la section intitulée « Bus LPC » [p 37]). Malheureusement, ce que j'ai vu ne dit pas que c'est sur le bus LPC où il est probablement. Dans `/pci/00/` se trouve un fichier binaire pour chaque périphérique PCI où les noms des fichiers sont les numéros des emplacements PCI. Le 00 signifie le bus PCI 0.

Le répertoire /sys

À partir du noyau 2.6, il existe un nouveau répertoire /sys pour la configuration de PnP. Il s'agit d'un système de fichiers de type sysfs et c'est un équivalent du système de fichiers /proc car les « fichiers » représentent une information de la mémoire du noyau et non pas un vrai fichier de votre disque dur. Cependant, il n'est pas aussi utile que le système de fichiers /proc. Au début (pour les noyaux 2.5), il s'appelait le « système de fichiers des pilotes » et avait comme type « driverfs ».

Dans ce système de fichiers, chaque périphérique existant sur votre système a son propre répertoire contenant des fichiers spécifiant les ressources qui lui sont affectées. Ces répertoires ont des noms comme `0000:00:12.0@` ou `00:06@`. Quels sont ces périphériques ? Le premier est une carte PCI dans l'emplacement 12 de votre PC. L'emplacement pourrait être appelé PCI2 dans votre PC (2 au lieu de 12) tout simplement parce que les emplacements ayant des numéros fiables sont utilisés par les emplacements intégrés à la carte mère et n'utilisent pas les emplacements physiques. Dans cet exemple, les emplacements 1 à 10 seraient intégrés alors que les emplacements 11 à 14 seraient appelés de 1 à 4. En exécutant **lspci**, vous connaîtrez la correspondance entre les numéros (comme 0000:00:12.0) et les noms (identique à l'interface IDE). Exécutez la commande **lspci -vv**, ou **lspci -vv** si vous voulez en voir plus.

Alors, qu'est-ce que `00:06` ? C'est une carte ISA (ou un périphérique intégré) mais ce n'est pas l'emplacement 6 du bus ISA (contrairement au numérotage PCI). Quand une recherche est faite pour les périphériques PnP ISA, il a été le sixième découvert. Plus précisément, il était le septième trouvé car il existe un périphérique numéroté 00:00. Donc, comment les identifier ? Vous pouvez lancer « cat */* » et afficher tous les fichiers pour tous les périphériques mais même à ce moment-là vous ne verrez pas les noms des périphériques (mais vous verrez l'information qui vous permettra de l'identifier). Ce problème sera corrigé dans le futur.

Non seulement ces fichiers apportent des informations sur la configuration des ressources du bus (d'une manière un peu cryptée) et sur les pilotes (dans les répertoires « drivers ») mais, dans le futur, vous devriez être capable de les utiliser pour modifier la configuration des ressources. Actuellement (août 2004), vous ne pouvez pas configurer le bus PCI avec cela. Une sérieuse limitation est qu'avec le « modèle de pilote » actuel, vous ne pouvez pas changer la ressource d'un périphérique qui a été affecté à un pilote, ce qui signifie généralement que vous aurez besoin de décharger le module du pilote pour pouvoir l'utiliser. Si le pilote est intégré, il n'y a aucun espoir. Ces sérieuses limitations seront éliminées dans le futur avec un peu de chance. Dans la documentation du noyau se trouve un fichier `pnp.txt` indiquant comment réaliser la configuration. En août 2004, il était obsolète mais l'auteur travaille sur une mise à jour. Utiliser le répertoire `/sys` pour configurer les ressources est connue comme l'« interface utilisateur pour le Plug and Play de Linux ».

L'autre partie de « Linux Plug and Play » est l'interface noyau utilisée par les pilotes de périphériques. Elle a beaucoup changé depuis le début du noyau 2.6 mais la plupart des pilotes utilisent toujours l'ancienne interface (août 2004). Il est aussi possible pour les pilotes (ou vous) d'utiliser l'interface utilisateur qui a besoin d'améliorations.

Inspection du bus PCI

Il est facile de trouver quelles ressources bus ont été assignées aux périphériques du bus PCI avec les commandes **lspci** et **scanpci**. Les options -v et -vv vous donneront plus de détails. Dans certains cas, **scanpci** trouvera un périphérique que **lspci** ne peut pas trouver. Ceci est dû au fait que **scanpci** recherche les périphériques directement sur le bus PCI (via l'espace de configuration) et n'utilise pas les données obtenues par le noyau (qui pourraient être fausses à cause d'un bogue du noyau - je viens de trouver un tel cas).

Cette information d'un format crypté est disponible dans les « fichiers » situés dans les répertoires `/sys` et `/proc`. Dans `/sys/bus/pci/devices`, le fichier `vendor` contiendra le numéro d'identifiant du vendeur, comme 0x4B8C, et cætera. Dans un format encore moins compréhensible, il se trouve dans `/proc/bus/pci`. De telles informations dans les anciens noyaux (avant le 2.6) se trouvaient dans `/proc/pci` (compréhensible malgré que les IRQ soient en hexadécimal) ou dans `/proc/buspci/devices` (affichage non compréhensible).

Dans la plupart des cas pour le PCI, vous verrez seulement comment le matériel est maintenant configuré et pas quelles ressources sont nécessaires. Dans certains cas, vous verrez seulement l'adresse de base (le début des plages d'adresses) mais pas celle de fin. Si vous disposez de l'espace complet, alors vous pourrez déterminer combien d'octets de ressources sont nécessaires.

Introduction au bus ISA

Pour les cartes du bus ISA, ce n'est pas aussi simple que pour le bus PCI qui est conçu pour le PnP. Les cartes ISA récentes étaient PnP contrairement aux anciennes. De même, certaines cartes PnP ont leur partie PnP désactivée par des logiciels spéciaux ne fonctionnant que sous MS Windows. Les cartes non PnP sont configurées avec des cavaliers sur la carte ou par des logiciels sous MS Windows.

Cartes ISA PnP

Si c'est une carte PnP, vous pouvez essayer **`pnpdump --dumpregs`** mais ce n'est pas une certitude. Le résultat peut sembler crypté mais il peut être déchiffré. Ne confondez pas les adresses de port de lecture que **pnpdump** utilise pour communiquer avec les cartes PnP avec l'adresse d'entrées/sorties du périphérique trouvé. Elles ne sont pas identiques.

Bus LPC

LPC (acronyme de *Low Pin Count*, soit petit nombre de connecteurs) est une interface type bus souvent utilisée sur les portables et de plus en plus utilisée sur les machines de bureau. Pour savoir si vous disposez d'un bus LPC, saisissez la commande **lspci** et cherchez quelque chose comme « LPC ». Il y a d'autres mots prêt de « LPC » comme « ISA Bridge ... LPC Interface Controller » ou « LPC Bridge », et cætera. LPC n'est pas réellement de l'ISA mais il se substitue à un bus ISA.

L'ancien bus ISA était lent et les périphériques qui avaient besoin de plus de rapidité étaient placés sur le nouveau bus PCI. Mais les périphériques qui n'avaient pas besoin d'une grande vitesse étaient souvent implémentés par des composants sur la carte mère et restaient sur le bus ISA même s'il n'y avait aucun emplacement pour des cartes ISA. Puis le bus LPC est arrivé pour remplacer les cartes ISA restantes. LPC est bien plus petit que l'ISA et aussi rapide car son horloge est quatre fois plus rapide que celle du bus ISA. Son bus multiplexé pour les données/adresses et le contrôle est composé de quatre fils. Envoyer un octet requiert la séparation de l'octet en deux demi-octets et leur réassemblage après. Cela explique la signification de l'acronyme LPC : Low Pin Count (petit nombre de broches). Il y a aussi quelques lignes sur le bus.

Cette petite interface LPC est utilisé pour les périphériques propriétaires lents comme les ports séries, les ports parallèles et les lecteurs de disquette. Donc, un ordinateur utilisant le bus LPC aura tous ces périphériques rapides sur le bus PCI, et cætera et les périphériques lents sur le bus LPC. Tous les périphériques LPC seront sur la carte : il n'existe pas d'emplacement pour carte LPC.

Un composant majeur du bus LPC est le composant superio, contenant des périphériques d'entrées/sorties propriétaires : ports série et parallèle, lecteur de disquette, contrôleur de clavier, et cætera. Le BIOS pourrait même se trouver sur le bus LPC. Le clavier et la souris (périphériques en entrée) devraient être listés dans `/proc/bus/input/devices` mais, au lieu de voir « lpc », il semble afficher « isa0060/serio0 », et cætera même s'ils se trouvent sur le bus LPC, et non pas sur le bus ISA.

X-bus

Avant que le bus LPC devienne populaire, il existait un bus X (pas couvert dans ce guide pratique) qui a servi dans le même but que le bus LPC mais qui n'était pas aussi compact que le bus LPC. Certains PC disposent des deux bus.

Cartes non PnP

En contraste avec les cartes PnP, les cartes non PnP ont toujours leurs ressources configurées au niveau matériel. C'est-à-dire qu'elles ont toujours une adresse et une IRQ sauf s'il existe une configuration par cavalier, et cætera pour désactiver le périphérique. Quelquefois, le pilote du périphérique, ou un autre logiciel, peut trouver les ressources utilisées simplement en cherchant sur chaque adresse. Par exemple, **scanport** (Debian uniquement ?) cherche sur la plupart des adresses d’entrées/sorties et peut trouver des périphériques ISA. Mais, attention, cela peut bloquer votre PC. Quelque fois, il échouera dans sa recherche du matériel disponible (car le matériel a 0xff dans ses registres). Même s’il trouve le matériel, il n’affichera pas l’IRQ ou n’identifiera pas positivement le matériel.

Donc, une façon de trouver ce matériel est de lancer un pilote, qui pourrait chercher un tel matériel. En regardant dans les messages du démarrage, vous pourriez voir un pilote se lancer et découvrir le matériel. Sinon, vous pourriez avoir besoin de trouver un pilote et de le lancer (par exemple, en le chargeant comme un module).

Trouver le bon pilote peut être difficile. Quelquefois, il n’existe tout simplement pas de pilote car certains périphériques ne sont pas (encore) gérés par Linux. Pour déterminer le pilote dont vous avez besoin, jetez un ½il sur toute documentation pouvant vous permettre d’identifier la carte. Si ceci échoue, jetez un ½il à la carte elle-même, avec les noms/numéros importants inscrits sur les composants. Mais l’identification du module de pilote dont vous avez besoin pourrait n’être pas disponible sur la carte. Vous pouvez trouver l’identifiant FCC sur la carte, puis chercher sur Internet avec ce numéro pour essayer de trouver plus d’informations sur la carte (ou sur les composants en faisant partie).

Cartes non PnP avec cavaliers

Si la carte dispose de cavaliers pour configurer les ressources, alors vous pouvez regarder la façon dont ils sont installés. Il existe des cartes qui ont à la fois le support de PnP et des cavaliers. Elles fonctionnent comme des cartes à cavalier si PnP a été désactivé d’une façon ou d’une autre. Vous pourriez avoir besoin de la documentation (soit imprimée soit sur disquette) venant avec la carte. Peut-être pourrez-vous la trouver sur Internet.

Cartes non PnP et sans cavaliers

Un des cas les plus difficiles est quand du logiciel fonctionnant sous MS Windows a été utilisé pour configurer une carte non PnP ou une carte PnP où la partie PnP a été désactivée. Donc, vous ne pouvez la configurer par PnP ou par des cavaliers. Dans ce cas, votre seul espoir est de chercher les adresses comme décrit dans la section intitulée « Cartes non PnP » [p 37] . Ou essayez de trouver le logiciel MS qui l’a configuré.

Outils pour détecter ou configurer le matériel

Dans un effort dupliqué, plusieurs distributions majeures de Linux ont développé leur propre outil de détection et de configuration du matériel. Ils configurent généralement bien plus que les ressources Plug-and-Play. C’est une configuration générale qui est bien au-delà du domaine couvert par ce guide pratique.

Puis, d’autres distributions, comme Debian, pouvaient obtenir des copies des outils et les offrir à leurs utilisateurs comme option ou comme outil en cas de problème. Ces outils utilisent généralement les outils Linux standard pour détecter le matériel, comme par exemple **lspci**. Dans la liste d’outils qui suit, le nom de la distribution qui l’a conçu est entre parenthèses mais l’outil est certainement disponible aussi pour les autres distributions.

- **hardinfo** ;
- **hwinfo** (SuSE) détecte plus de choses que **discover** ;
- **discover** (Progeny, utilisé par Debian) ;
- **Kudzu** (RedHat) détecte et configure ;
- **lsdev** (commande Linux standard) ;
- **hwsetup-knoppix** (Knoppix, basé sur Kudzu).

Outils pour détecter et configurer un type de matériel

Il existe différents outils disponibles pour trouver et, quelque fois, configurer différents types de périphériques. Cette configuration est généraliste et n'est pas couverte dans ce guide pratique.

- **read-edid** (**get-edid**) : récupère les paramètres des moniteurs VESA (à part les très anciens) ;
- **sndconfig** : pour les cartes son ;
- **printtool** : imprimantes (X-window doit être en cours d'exécution) ;
- **pconf-detect** : ports parallèles ;
- **mdetect** : détecte et configure les souris (mais n'a pas pu trouver la mienne). Connaît-il les souris sur `/dev/input/` ?
- **nictools-pci** (et **nictools-nopci**) pour les cartes ethernet ;
- **hdparm** : configure les disques durs ;
- **hotplug** : utilisé par le noyau ;
- **xvidtune** : configure la vidéo avec Xwindows (voir XFree86-Video-Timings-HOWTO).

Utilisez MS Windows

Quelques personnes ont essayé d'utiliser Windows pour voir comment les ressources bus étaient configurées. Malheureusement, comme le matériel PnP oublie sa configuration de ressources bus à l'arrêt, la configuration peut ne pas être identique lors du redémarrage sous Linux pour le matériel non PnP (ou lorsque quelqu'un a désactivé PnP dans le périphérique soit par des cavaliers soit en utilisant des logiciels Windows). Même pour PnP, cela peut être le cas parce que dans beaucoup de cas, Windows et Linux acceptent simplement ce que le BIOS a fait. Mais là où Windows et Linux font une configuration, ils peuvent le faire différemment. Donc ne comptez pas à ce que les périphériques soient configurés de la même manière.

Interruptions PCI

Chaque périphérique PCI qui a besoin d'une interruption vient avec une interruption PCI fixe qui ne peut pas être modifié. Mais, cette interruption PCI peut être redirigé vers une interruption ISA par un composant sur la carte-mère. Du coup, quand une carte PCI génère une interruption PCI, elle devient une interruption ISA, même s’il n’y a plus de bus ISA.

Il existe deux nouveaux développements dans les interruptions PCI. Ils sont particulièrement importants pour les PC multiprocesseurs. Un cas possible est l’APIC, *Advanced Programmable Interrupt Controller* qui peut fournir 24 interruptions, voire plus. Voir le fichier `IO-APIC` dans le répertoire `i386` de la documentation du noyau et dans le guide pratique sur l’ACPI. Ne confondez pas APIC et ACPI (*Advanced Configuration and Power Interface*) qui pourrait être utilisé par le noyau pour configurer l’APIC.

Un autre nouveau développement est les interruptions signalées par message (MSI, acronyme de *Message Signalled Interrupts* (MSI) où l’interruption est juste un message envoyé à une adresse spéciale sur le bus principal de l’ordinateur (pas de ligne d’interruption nécessaire). Mais le périphérique qui envoie un tel message doit tout d’abord obtenir le contrôle du bus principal de façon à ce qu’il puisse envoyer le message d’interruption. Un tel message contient plus d’informations que « J’envoie une interruption ».

La redirection des interruptions vers des interruptions ISA est réalisée soit par un routeur programmable d’interruptions soit par un APIC. Les deux sont programmables par le BIOS et par Linux.

Les interruptions PCI peuvent être partagées, signifiant que deux périphériques voire plus utilisent la même IRQ. C’est faisable, il est généralement préférable de ne pas les partager. Le partage ne fonctionne pas bien pour les très anciens matériels PCI (avant 1995 ?) et pour les matériels PCI défectueux dès l’usine (ils ont été créés ainsi). Par exemple, si un périphérique PCI sur l’IRQ 9 réclamait par erreur que toute IRQ 9 était pour lui, alors les autres périphériques utilisant l’IRQ 9 verraient toutes leurs demandes d’interruption ignorées. Sans partage, ce problème est évité.

Pour un exemple de partage d’une même IRQ entre deux périphériques PCI, voir Partage d’interruption PCI [p 49] . Cette capacité de partage est intégrée au matériel et tous les pilotes de périphériques sont supposés la supporter. Notez que vous ne pouvez pas partager la même interruption entre le bus PCI et le bus ISA. Néanmoins, de tels partages illégaux fonctionneront si seulement un des deux périphériques en conflit est utilisé à un moment donné. Utilisé signifie ici qu’un programme a ouvert le périphérique (et ne l’a pas fermé) dans son code C, même si un tel programme peut être temporairement endormi.

Lier les interruptions PCI

Voici quelques détails sur le système d’interruptions PCI. Chaque carte PCI (et les périphériques montés sur la carte-mère) a quatre interruptions possibles : INTA#, INTB#, INTC#, INTD#. À partir de maintenant, nous les appelerons simplement A, B, C et D. Chacune a sa propre broche sur le connecteur d’une carte PCI. Donc, pour un système comprenant sept emplacements (pour sept cartes), il pourrait y avoir 28 (7 x 4) différentes lignes d’interruption pour ces cartes. Les périphériques intégrés à la carte-mère ont aussi des interruptions supplémentaires. Mais les spécifications permettent un nombre plus réduit de lignes d’interruption, donc certains bus PCI semblent n’avoir que quatre ou huit lignes d’interruption. Ceci n’est pas trop restrictif car les interruptions pourraient être partagées. Pour une ligne à quatre interruptions (LNKA, LNKB, LNKC, LNKD), il y a un composant appelé « routeur programmable d’interruptions » qui redirige LNKA, LNKB, LNKC, LNKD vers les IRQ sélectionnées. Ce routage peut être modifié par le BIOS ou par Linux. Par exemple, LNKA peut être routé vers l’IRQ 5. Supposons que nous désignons l’interruption B de l’emplacement 3 comme l’interruption 3B. Les interruptions 3B et 2A pourraient être connectées de façon permanente à LNKA qui est routé vers l’IRQ 5. Ces deux interruptions, 3B et 2A, sont partagées en permanence par le cablage sur la carte-mère.

Saisir **dmesg** sur la ligne de commande permet de voir comment les lignes d'interruption style LNKA sont redirigées (ou routées) vers les IRQ (*5 signifie que c'est lié à l'IRQ 5). Recherchez *PCI Interrupt Link*. Notez que « link » est utilisé ici avec deux significations : 1. le lien (routage) des lignes d'interruptions PCI vers les les IRQ, 2. le label d'une ligne d'interruption comme LNKB (lien B). Les labels de la ligne d'interruption semblent être fournis par le BIOS (??) et pourraient avoir des noms différents comme : LNKC, LNK2, APCF, LUBA, LIDE, et cætera. Question : quand un grand nombre de lignes d'interruption sont affichées comme étant désactivées, existent-elles toutes physiquement sur la carte-mère ? ou existent-elles seulement dans la partie ACPI du BIOS pour que ce dernier puisse fonctionner avec les cartes-mères qui ont un grand nombre de lignes d'interruption ?

Connecter toutes les interruptions A (INTA#) à la ligne LNKA, toutes les B à la ligne LNKB, et cætera. est une méthode simple pour connecter en dur ces lignes des périphériques PCI (comme le 3B) aux interruptions LNKA, et cætera. Cette méthode a été utilisée une fois plusieurs années auparavarant mais ce n'est pas la bonne solution. Voici pourquoi. Si une carte a seulement besoin d'une interruption, elle doit utiliser A. Si elle a besoin de deux interruptions, elle doit utiliser A et B. Du coup, INTA# est utilisé bien plus fréquemment que INTD#. Donc, on va se trouver avec un nombre excessif d'interruptions partageant la première ligne (LNKA connecté à toutes les INTA#). Pour dépasser ce problème, il vaut mieux les connecter de façon aléatoire pour que chacune des quatre lignes d'interruptions (LNKA, LNKB, LNKC, LNKD) partagent à peu près le même nombre d'interruptions PCI.

Une façon de le faire est de lier en dur LNKA avec les interruptions 1A, 2B, 3C, 4D, 5A, 6B, 7C. Ceci se fait en connectant physiquement le fil W aux fils 1A, 2B, et cætera. De la même façon, le fil LNKB pourrait être connecté aux fils 1B, 2C, 3D, 4A, 5B, 6C, 7D, et cætera. Puis, au démarrage, le BIOS dirige les LNKB, LNKA, LNKC, LNKD aux IRQ. Après cela, il écrit l'IRQ que chaque périphérique utilise dans un registre de configuration du matériel dans chaque périphérique. À partir de maintenant, tout programme interrogeant ce registre peut savoir l'IRQ utilisée par le périphérique. Notez qu'écrire simplement l'IRQ dans un registre sur une carte PCI ne configure en aucun cas l'IRQ pour ce périphérique.

Une utilisation pratique de cette information est qu'en dernier ressort, une personne pourrait modifier les IRQ d'une carte PCI en l'insérant dans un emplacement différent. Dans l'exemple ci-dessus, INTA# d'une carte PCI sera connecté au fil LNKA si la carte est insérée dans l'emplacement 1 (1A correspond à LNKA) mais INTA# sera connecté au fil LNKB si elle est insérée dans l'emplacement 4 (4A correspond à LNKB).

Une carte dans un emplacement pourrait avoir jusqu'à huit périphériques mais il n'y a que quatre interruptions PCI pour elle (A, B, C, D). Cela suffit car les interruptions pourraient être partagées pour que chacun des huit périphériques (s'ils existent) puisse avoir une interruption. La lettre de l'interruption PCI d'un périphérique est souvent fixée et codée en dur dans le périphérique. L'affectation des interruptions est réalisée par soit le BIOS ou par Linux, établissant une correspondance entre les interruptions PCI et les interruptions ISA comme mentionné ci-dessus.

S'il n'existe que quatre lignes (LNKA, LNKB, LNKC, and LNKD) comme dans l'exemple ci-dessus, les choix de correspondance pour le BIOS sont limités. Certaines cartes-mère peuvent utiliser plus de lignes et ont donc plus de choix. Par exemple, de LNKA à LNKH (8 lignes). Les messages au démarrage (**dmesg**) peuvent les afficher et indiquer leur correspondance. Le BIOS sait comment elles sont câblées.

Sur le bus PCI, le BIOS (ou Linux) affecte des IRQ (interruptions) de façon à éviter les conflits avec les IRQ qu'il sait affectés au bus ISA. Quelque fois, le menu du CMOS du BIOS peut vous autoriser à affecter des IRQ aux cartes PCI ou indiquer au BIOS les IRQ réservées aux périphériques ISA. Les affectations sont connues sous le nom d'une table de routage. Sous MS WIndows, c'est appelé *IRQ steering* mais cela couvre aussi le cas d'un routage dynamique des IRQ après le démarrage. Le BIOS peut supporter son propre *IRQ steering*.

Vous pourriez penser que, comme le PCI utilise des IRQ (conçues pour le bus ISA), cela pourrait être lent car le bus ISA est lent. Pas vraiment. Le composant de contrôle des interruptions ISA a un fil d'interruption direct le reliant au CPU pour qu'il obtienne une attention immédiate. Bien que les signaux sur les bus d'adresses et de données de l'ancien ISA sont lents pour arriver au CPU, les signaux d'interruptions y arrivent rapidement. Même l'ACPI a un bus allant directement au CPU.

PnP pour les périphériques externes et ajoutés

Bus USB

L'USB (*Universal Serial Bus*, c'est-à-dire Bus Universel Série) est un bus à grande vitesse sur un câble externe qui se connecte au PC. Le bus externe a ses propres protocoles de communication et n'utilise pas les IRQ, adresses d'entrées/sorties (ou tout autre ressource bus) sur les câbles bus externes. La communication se fait par paquets comme sur Internet, seulement sur des allocations de tranches de temps, ce qui empêche un périphérique de manger le bus si d'autres périphériques en ont besoin. Il existe des tranches de temps libre qui permettent à tout périphérique d'envoyer un message court au contrôleur de bus sans avoir besoin des IRQ sur le bus.

Néanmoins, le contrôleur de bus USB intégré au PC a une IRQ et une adresse sur le bus PCI (ou ISA), utilisées pour la communication entre le CPU et tous les périphériques USB. Donc, il n'y a pas d'allocations de ressources nécessaires pour les périphériques individuels sur le bus USB. Vous pouvez aussi imaginer que tous les périphériques sur le bus USB partagent la même interruption et la même adresse. Si un périphérique est sur l'USB, il a besoin d'un pilote qui comprenne l'USB.

Mais, chaque périphérique USB a un identifiant, comme les cartes du bus PCI. Linux maintient donc une table des identifiants de façon à ce que les pilotes de périphérique puissent les vérifier et trouver ainsi leur périphérique. L'USB supporte aussi le « hot plug ». Pour trouver ce qui est placé sur le bus USB, vous pouvez utiliser un outil généraliste de détection de matériel comme **discover** ou **hwinfo**.

Hot Plug

« Hot plug » correspond à la connexion d'un périphérique sur un PC (habituellement avec un câble) et à sa détection immédiate. Si nécessaire, il est configuré avec les ressources bus. Son pilote est aussi lancé, peut-être en chargeant le module correspondant. Pour que ceci fonctionne, le matériel utilisé doit être conçu de façon approprié. Vous pouvez utiliser cette fonctionnalité avec certaines cartes PCI (Cardbus), périphériques USB et IEEE 1394 (Firewire).

Lorsqu'un nouveau périphérique est détecté, ses registres sont lus de façon à récupérer un numéro d'identifiant du périphérique. Pour trouver un pilote, Linux doit maintenir une table réalisant la correspondance entre numéro de périphérique et pilote. Cette table existe dans le noyau depuis la version 2.4. Elle est nommée MODULE_DEVICE_TABLE.

Hot Swap

« Hot Swap » vous permet de remplacer un ancien périphérique en l’enlevant et en branchant le nouveau. Vous avez donc interverti (*swapped*) les périphériques. Maintenant, en plus d’être capable de détecter le branchement d’un nouveau périphérique, la suppression d’un ancien périphérique doit aussi être détectée.

PnP détecte les périphériques connectés aux ports séries

Les périphériques externes connectés par le port série via un câble (comme les modems externes) sont aussi appelé Plug-and-Play. Comme seul le port série a besoin de ressources bus (une IRQ et une adresse d’entrées/sorties), il n’y a pas de ressources bus à allouer pour ces périphériques. Dans ce cas, PnP est utilisé uniquement pour l’identification du modem (lire le numéro/code du modèle). Ceci est important dans le cas où ce modem est un modem logiciel (linmodem) et requiert un pilote spécifique. Il existe une spécification PnP pour de tels périphériques séries externes (quelque chose connecté au port série).

Linux ne supporte pas encore ceci ? Pour un modem matériel, le pilote série ordinaire suffira, donc il n’y a pas besoin de chercher un pilote avec **serialpnp**. Vous devez toujours indiquer au programme de communication sur quel port se trouve le modem. Avec PnP, vous n’auriez même pas besoin de faire ça. Avec l’arrivée des modems logiciels disposant de pilotes Linux (linmodem), il serait bien que les pilotes appropriés s’installent automatiquement via PnP.

Messages d’erreurs

Unexpected Interrupt (Interruption inattendue)

Ceci signifie qu’une interruption est survenue alors qu’aucun pilote ne l’attendait. Il est improbable que le matériel ait généré une interruption par erreur. Il est plus probable que le logiciel comporte un petit bogue et n’a pas réalisé qu’un logiciel a fait quelque chose qui a généré cette interruption. Dans la plupart des cas, vous pouvez ignorer ce message en toute sécurité, et tout particulièrement si cela n’est arrivé qu’une ou deux fois lors du démarrage. Pour les messages du démarrage, regardez les messages qui lui sont proches pour trouver une indication sur ce qui s’est passé. Par exemple, si une recherche était en cours, il est possible que cela ait activé un périphérique physique qui en retour a généré une interruption, interruption que le pilote n’attendait pas. Le pilote n’écoutait peut-être pas le bon numéro d’IRQ.

Erreur de configuration Plug and Play (BIOS Dell)

Le BIOS a été incapable de configurer les ressources bus. Il peut exister un conflit d’interruptions qui ne peut être évité. Dell vous suggère que vous enleviez certaines des cartes non essentielles pour voir si le problème disparaît. Dans un cas, le problème était dû à une carte mère défectueuse.

isapnp: Write Data Register 0xa79 already used (à partir des journaux)

Si vous utilisez isa-pnp, l’adresse d’entrées/sorties 0xa79 ne doit jamais être utilisée par un périphérique, quel qu’il soit. Donc, si un autre matériel utilise 0xa79 lorsque vous essayez de charger le module isa-pnp, vous obtiendrez ce message dans vos journaux de trace et isa-pnp quittera. Une façon de corriger cela est de charger le module isa-pnp bien plus tôt avant que tout autre matériel ne soit initialisé. Pour le PCMCIA, ceci impose de charger isa-pnp avant de lancer les modules cb et le service associé.

Impossible d'allouer la région (PCI)

Ici, « région » signifie un ensemble d'adresses. Un périphérique PCI qui a besoin de deux régions aura la région 0 comme première adresse et la région 1 pour deuxième adresse. Utilisez la commande **lspci --v** pour voir les différentes régions de ressources (souvent appelé régions) et si l'adresse est de type entrée/sortie ou mémoire. Dans le jargon PCI, la région 2 est l'« adresse de base 2 » (ou « registre d'adresse de base 2 »), et cætera.

Partage et conflit d'interruption

Quand deux périphériques ou plus utilisent la même ligne d'interruption, (et le même numéro d'IRQ), il s'agit soit d'un « partage d'interruption » soit d'un « conflit d'interruption ». Le bus PCI autorise tous les périphériques PCI à partager des interruptions avec les autres, ce qui est appelé le partage. Mais si un périphérique ISA (ou un périphérique LPC ??) essaie d'utiliser la même interruption (IRQ) qu'un autre périphérique (PCI, ISA ou LPC ??), il y habituellement un conflit d'interruption. Ceci signifie que, lors de l'arrivée d'une interruption, le mauvais périphérique est souvent appelé et des problèmes surviennent lors des dépassements de tampons (perte de données).

Les conflits d'nterruptions étaient communs quand les IRQ étaient configurées grâce à des cavaliers sur les cartes (bus ISA). Le Plug-and-Play ISA (aucun cavalier) a beaucoup aidé car le logiciel pouvait modifier les IRQ. L'abandon d'ISA en faveur du PCI a pratiquement éliminé les conflits IRQ. Malgré tout, votre PC peut toujours avoir des périphériques sur un bus ISA, LPC ou X. Mais le noyau devrait savoir comment les configurer et donc éviter de les utiliser pour les périphériques PCI, évitant ainsi les conflits d'interruption.

Mais, bien qu'ayant éliminé le problème des conflits, le partage d'IRQ sur le bus PCI a introduit un nouveau problème, le problème d'équilibre des IRQ. Si des périphériques utilisant beaucoup les interruptions partagent la même IRQ, cela pourrait amener des délais dans la récupération des IRQ et pourrait même amener à des dépassements de tampon et d'autres erreurs. Ceci n'est pas dû à la façon dont le logiciel détermine le périphérique qui a lancé cette interruption.

Il y a aussi le cas du conflit d'interruption survenant quand un périphérique ISA est activé mais ne peut pas se voir affecter une interruption (IRQ) car il n'en reste plus. C'est une sorte de conflit d'interruption (IRQ) car, si une interruption avait été affectée, cela aurait créé un conflit. Ceci peut aussi arriver même si une interruption est disponible mais ne peut pas être utilisé car le matériel du périphérique qui a besoin d'une interruption ne supporte le numéro de l'interruption disponible (ou la carte-mère ne la supporte à cause de problèmes de routage, voir Interruptions PCI [p 39]). Si les périphériques ISA utilisent toutes les interruptions, alors une ou plusieurs cartes PCI pourraient être en conflit car elles ne peuvent obtenir d'IRQ.

Normalement, le BIOS affectera des interruptions et ne créera pas de conflits. Mais il pourrait être forcé de créer des conflits s'il tombe à court d'IRQ. Ceci peut survenir si quelqu'un a configuré le BIOS pour réserver certaines IRQ pour les périphériques ISA qui ne sont pas PnP. Ces paramétrages pourraient être mauvais et devraient être vérifiés, tout spécialement si vous avez des problèmes. Par exemple, quelqu'un pourrait avoir réservé une IRQ pour une carte ISA qui a été enlevé du PC depuis longtemps. Si vous récupérez cette IRQ, alors elle est disponible et un conflit disparaît.

Quelque fois, le BIOS résoudra le problème du manque d'IRQ en utilisant l'IRQ 0. Elle n'existe pas mais signifie que le pilote devrait utiliser la demande au lieu des IRQ. Ceci signifie que le pilote devra vérifier fréquemment le périphérique (lui demander) pour voir si le périphérique a besoin d'un service de la routine d'interruptions. Bien sûr, cela gâche du temps processeur et il y a plus de risques d'un dépassement de tampon du périphérique car il pourrait ne pas être servi assez rapidement par le pilote.

Quelques périphériques ISA ont été conçus pour autoriser les conflits d'interruption (entre deux périphériques ISA ??) mais les deux périphériques ISA doivent être conçus de cette façon et piloté par du logiciel connaissant les interruptions partagées. La carte mère doit aussi le supporter.

Annexe

Universal Plug and Play (UPnP)

C'est en quelque sorte un réseau Plug-and-Play développé par Microsoft mais utilisable sous Linux. Vous connectez quelque chose sur un réseau et ce quelque chose n'a pas besoin d'être configuré mais ne va communiquer qu'avec des périphériques UPnP du réseau. Ici, « configurer » est utilisé dans le sens large et ne signifie pas simplement configurer les ressources bus. Un des objectifs est de permettre au gens connaissant peu de choses sur les réseaux ou sur la configuration et l'installation d'un routeur, d'une passerelle, d'une imprimante réseau, et cætera de le faire. Une utilisation majeure de UPnP serait dans les réseaux sans-fil.

UPnP utilise :

- un protocole de découverte des services (*Simple Service Discovery Protocol*) pour trouver les périphériques,
- une architecture de notification générale d'événements (*General Event Notification Architecture*),
- un protocole d'accès aux objets (*Simple Object Access Protocol*) pour assurer le contrôle des périphériques.

Ce guide pratique ne couvre pas UPnP. UPnP pour Linux est supporté par Intel qui a développé un logiciel spécifique. Il existe d'autres programmes qui font à peu près la même chose que UPnP. Une comparaison de ceux-ci est disponible sur http://www.cs.umbc.edu/~dchakr1/papers/mcommerce.html. Un projet UPnP pour Linux se trouve sur Sourceforge : Kit UPnP pour Linux

Détails des adresses

Il existe trois types d'adresses : adresses en mémoire principale, adresses d'entrées/sorties (ports) et adresses de configuration. Sur le bus PCI, les adresses de configuration constituent une plage d'adresses séparée un peu comme les adresses d'entrées/sorties. Sauf dans le cas compliqué des adresses de configuration ISA, qu'une adresse sur le bus soit ou non une adresse en mémoire principale, une adresse d'entrées/sorties ou une adresse de configuration dépend seulement du voltage sur certains fils du bus. Pour plus de détails sur les adresses de configuration du bus ISA, voir la section intitulée « Adresses de configuration du bus ISA (Port de lecture et cætera) » [p 47] .

Plages d’adresses

Le terme « adresse » est quelque fois utilisé dans ce document pour signifier un ensemble contigu d’adresses. Les adresses sont en unité d’octets. Donc, par exemple, un port série sur l’espace d’adressage 3F8-3FF sera souvent juste référencé par son adresse de base, 3F8. 3F8 est l’emplacement du premier octet de la plage (espace d’adressage). Pour visualiser les espaces d’adressage, jetez un ½il à `/proc/iomem` et `/proc/ioports`.

Plage d’adresses

Pour accéder à la fois aux espaces d’adresses mémoire principale et d’entrées/sorties, le même bus d’adresses est utilisé (les fils utilisés pour l’adresse sont partagés). Comment le périphérique sait-il si l’adresse apparaissant sur le bus est une adresse mémoire ou d’entrées/sorties ? En fait, pour l’ISA (pour le PCI, lisez aussi ceci), il existe quatre fils dédiés sur le bus qui amènent ce type d’informations. Si un de ces quatre fils est « activé », cela indique que le CPU veut lire une adresse d’entrées/sorties et la mémoire principale ignore l’adresse sur le bus. En tout, les fils de lecture et écriture existent à la fois pour les adresses de mémoire principale et pour les adresses d’entrées/sorties (quatre fils en tout).

Pour le bus PCI, il s’agit de la même idée de base (utilisant aussi quatre fils) mais réalisée un peu différemment. Au lieu d’avoir un seul des quatre fils activé, un nombre binaire est placé sur les fils (d’où 16 possibilités différentes). Donc, il est possible de véhiculer plus d’informations sur ces quatres fils. Quatre de ces 16 nombres sont utilisés pour les espaces en mémoire principale et d’entrées/sorties comme indiqué dans le paragraphe ci-dessus. En plus, il existe aussi un espace d’adressage de configuration qui utilise plus de deux chiffres supplémentaires. Cela laisse dix autres nombres disponibles pour d’autres utilisations.

Espace d’adresses pour la configuration PCI

Ceci est différent des espaces d’adresses mémoire et d’entrées/sorties parce que l’espace d’adresses de configuration est « géographique ». Chaque emplacement d’une carte a un numéro d’emplacement faisant parti de l’adresse. De cette façon, Linux (ou le BIOS) peut adresser un certain emplacement et trouver le type de carte fiché dans cet emplacement. Chaque périphérique a des registres standards de 64 bits et quelques uns d’entre eux contiennent des numéros qui peuvent identifier de façon non ambiguë le périphérique. Comme le nombre d’emplacements est limité comme le sont le nombre de périphériques PCI construit dans la carte mère, Linux (ou le BIOS) a besoin de vérifier un nombre limité d’adresses pour trouver tous les périphériques PCI. S’il ne lit que des uns (0xFF en hexadécimal) à partir du premier registre d’un périphérique, alors cela signifie qu’aucun périphérique n’est présent. Comme il n’y a aucune carte ou périphérique fournissant tous les numéros un (0xFF), le « host bridge » PCI sur la carte mère fournit ce numéro pour tous les périphériques inexistants.

Le numéro d’emplacement PCI est appelé (dans le jargon PCI le numéro de périphérique et comme une carte peut avoir au plus huit périphériques sur elle, un numéro de fonction (allant de 0 à 7) identifie le périphérique qui se trouve sur une carte PCI. Ces numéros font partie de l’adresse géographique. Les développeurs Linux l’appellent *pci-slot-name*. Du coup, ce que Linux appelle un périphérique est en fait une fonction dans le jargon PCI. Le numéro du bus PCI (souvent 00) devient aussi une partie de l’adresse géographique. Par exemple, 0000:00:0d.2 correspond au bus PCI 0, emplacement 0, fonction 2. Pour l’adresse géographique complète, vous devez inclure le numéro sur deux mots des registres de configuration du périphérique auquel on veut l’accès. Les 0000 en tête (en 1999) étaient réservés pour une utilisation future.

Comment le processeur désigne-t-il qu'une lecture ou une écriture doit se faire dans l'espace de configuration PCI ? Il ne le fait pas, en tout cas pas directement. À la place lorsque l'accès à l'espace de configuration est désiré, il fait une écriture sur 32 bits (un mot double) pour écrire 0cf8-0cfb en espace d'entrées/sorties et écrit l'adresse géographique complète ici. Le *host bridge* PCI écoute à cette adresse et nous assure que la prochaine écriture de données sera 0cfc-0cff. C'est enregistré dans des registres de configuration du périphérique spécifié. Le pont fait les deux en envoyant un signal spécial à la carte PCI spécifiée (ou ce qui y ressemble) sur un fil dédié qui va seulement à l'emplacement où la carte est connectée. Il place aussi des bits sur le bus de contrôle indiquant que ce qui est sur le bus d'adresse maintenant est une adresse géographique de l'espace de configuration.

Pourquoi ne pas faire simple et demander simplement au processeur de placer les bits sur le bus de contrôle pour indiquer que l'adresse sur le bus principal est une adresse géographique pour la configuration du PCI ? Et bien, la plupart des processeurs ne sont pas capables de le faire donc le « host bridge » PCI le fait à la place.

Vérification de la plage (test ISA pour les conflits d'adresses d'entrées/sorties)

Sur le bus ISA, il existe une méthode intégrée dans chaque carte PnP pour vérifier qu'aucune autre carte n'utilise la même adresse d'entrées/sorties. Si deux cartes ou plus utilisent la même adresse d'entrées/sorties, les cartes ont peu de chance de fonctionner correctement (voire de fonctionner tout court). Un bon logiciel PnP devrait allouer les ressources bus de manière à éviter ce conflit, mais même dans ce cas, une carte non PnP pourrait avoir la même adresse.

Le test fonctionne par une carte plaçant un nombre de test connu dans ses propres registres d'entrées/sorties. Puis, le logiciel PnP le lit et vérifie que ce qu'il lit correspond bien au numéro de test connu. Il répète le même test avec un autre numéro. Comme il vérifie l'ensemble des adresses d'entrées/sorties allouées à la carte, il est appelé un vérificateur de plage. Il pourrait être appelé plus logiquement un testeur de conflit d'adresses. Si un conflit est détecté, vous obtenez un message d'erreur.

Communiquer directement via la mémoire

Traditionnellement, la plupart des périphériques d'entrées/sorties utilisent seulement la mémoire d'entrées/sorties pour communiquer avec le processeur (CPU). Le pilote de périphérique, exécuté sur le processeur lira et écrira des données de/vers l'espace d'adressage des entrées/sorties et la mémoire principale. Malheureusement, cela nécessite deux étapes. Par exemple, 1. lire les données à partir d'un périphérique (en espace d'adressage) et les stocker temporairement dans le CPU ; 2. écrire ces données en mémoire principale. Une façon plus rapide serait que le périphérique place lui-même les données directement en mémoire principale. Une façon de faire ceci est d'utiliser la section intitulée « DMA (accès direct à la mémoire) ou maîtrise du bus » [p 13] ISA ou la maîtrise du bus PCI. Le périphérique physique peut aussi détenir un peu de mémoire principale (aux adresses supérieures pour éviter les conflits avec les adresses des composants de la mémoire principale). De cette façon, le périphérique lit et écrit directement dans son espace mémoire interne sans avoir à s'embêter avec le DMA ou la maîtrise du bus. De tels périphériques pourraient aussi utiliser des adresses d'entrées/sorties.

Adresses de configuration du bus ISA (Port de lecture et cætera)

Ces adresses sont aussi connues comme les « ports d'auto-configuration ». Pour le bus ISA, il n'existe pas techniquement de plage d'adresses de configuration, mais le CPU utilise une façon spéciale d'accéder aux registres de configuration PnP sur les cartes PnP. Dans ce but, trois adresses d'entrées/sorties sont allouées et chacune adresse un seul octet (il n'y a pas à proprement parler d'espace ou de plage). Il ne s'agit pas de trois adresses pour chaque carte mais de trois adresses partagées par toutes les cartes ISA-PnP.

Ces trois adresses sont nommées port de lecture (*read-port*), port d'écriture (*write-port*) et port d'adresse (*address-port*). Chaque port a une taille d'un octet. Chaque carte PnP dispose d'un grand nombre de registres de configuration, donc même les trois adresses ne sont pas suffisantes pour les registres de configuration d'une seule carte. Pour résoudre ce problème, chaque carte se voit affecter un numéro de carte en utilisant une technique appelée « isolation ». Voir la section intitulée « Isolation ISA » [p 50] pour des détails plus complexes.

Ensuite, pour configurer une certaine carte, son numéro de carte est envoyé via l'adresse du port d'écriture pour indiquer à cette carte qu'elle doit écouter sur son port d'adresse. Toutes les autres cartes notent que ce n'est pas leur numéro de carte et donc n'écoutent pas. Ensuite, l'adresse d'un registre de configuration est envoyé sur le port d'adresse (à toutes les cartes, mais une seule écoute). Enfin, le transfert de données prend place avec ce registre de configuration sur cette carte soit en faisant une lecture sur le port de lecture soit en faisant une écriture sur le port d'écriture.

Le port d'écriture est toujours A79 et le port d'adresse est toujours 279 (en hexadécimal). Le port de lecture n'est pas fixe mais dépend du logiciel de configuration (tout en restant dans la plage 203-3FF) qui avec un peu de chance n'entrera pas en conflit avec les autres cartes ISA. Si un conflit se déclare, il changera l'adresse. Toutes les cartes PnP sont « programmées » avec cette adresse. Donc, si vous utilisez **isapnp** pour enregistrer ou connaître la configuration, celui-ci doit d'abord déterminer l'adresse du port de lecture.

Détails sur les interruptions

Interruptions sérialisées

Il a été dit précédemment qu'il existe un fil pour chaque interruption. Mais l'interruption sérialisée (ou interruption série) est une exception. Un seul fil est utilisé pour toutes les interruptions qui sont multiplexées sur ce fil. Chaque interruption a un créneau horaire sur la ligne d'interruption. Il est utilisé sur le bus LPC mais aussi sur le bus PCI bien que cela soit plus rare pour ce dernier ?

DMA

Avant de se plonger dans le détail des interruptions, il existe une autre façon pour que les périphériques initient la communication en dehors de l'envoi d'une interruption. Cette méthode est une requête DMA (*Direct Memory Access*) pour prendre le contrôle de l'ordinateur à partir du CPU pour un temps limité. Sur le bus PCI, il n'utilise aucune ressource. Tous les périphériques ne sont pas capables de faire du DMA. Voir la section intitulée « DMA (accès direct à la mémoire) ou maîtrise du bus » [p 13] .

Interruptions logicielles

Il existe aussi un autre type d'interruption nommée « interruption logicielle », non couverte par ce guide pratique et n'utilisant pas de ressources. Alors qu'une interruption matérielle est générée par le matériel, une interruption logicielle est initiée par le logiciel. Il existe plusieurs façons pour ce faire. Une façon est que le logiciel dise au processeur d'exécuter une interruption (une instruction d'interruption). Une autre façon consiste, pour le logiciel, à envoyer des messages aux autres processus pour les interrompre même s'il n'est pas clair qu'on puisse appeler ça une interruption. Le processus ksoftirq process, que vous pouvez trouver dans la liste des processus sur un PC Linux, est un programme qui lance ce type d'interruption pour gérer les pilotes de périphériques. Le pilote de périphérique commence à s'exécuter à cause d'une interruption matérielle mais, plus tard, des interruptions logicielles sont utilisées pour la deuxième moitié de la routine d'interruption du pilote. Donc, le processus ksoftirq est aussi connu comme la « seconde moitié ». Pour plus de détails, voir la documentation du noyau.

Interruptions matérielles

Les interruptions amènent beaucoup d'informations mais seulement indirectement. Le signal de demande d'interruption (un voltage sur un fil) indique seulement au composant, appelé le contrôleur d'interruption, qu'un certain périphérique demande l'attention. Le contrôleur d'interruption envoie le signal au CPU. Le CPU s'interrompt dans ce qu'il faisait, trouve le pilote de ce périphérique et exécute une partie de celui-ci nommée « routine d'interruption » (ou « gestionnaire d'interruption »). Cette « routine » essaie de trouver ce qui est arrivé et gère ensuite le problème. Par exemple, le périphérique peut avoir besoin d'envoyer/recevoir des octets. Ce programme (cette routine) peut facilement comprendre ce qui s'est passé car le périphérique dispose de registres disponibles sur des adresses connues par le pilote (à condition que le numéro d'IRQ et que les adresses d'entrées/sorties soient correctement configurés). Ces registres contiennent l'état du périphérique. Le logiciel lit le contenu de ces registres et en inspectant le contenu, trouve ce qui est arrivé et réalise l'action appropriée.

Donc, chaque pilote de périphérique a besoin de savoir le numéro d'interruption (IRQ) où écouter. Sur le bus PCI (et dans certains cas spéciaux, sur le bus ISA), il est possible que deux (voire plus) périphériques partagent le même numéro d'IRQ. Notez que vous ne pouvez pas partager une interruption PCI avec une interruption ISA (y a-t'il des exceptions ?). Quand une interruption partagée est lancée, le processeur exécute toutes les routines du service d'interruption séquentiellement pour tous les périphériques utilisant cette interruption. La première action qu'entreprend la première routine lancée est de vérifier les registres du périphérique pour voir si une interruption a été générée par son périphérique. S'il se trouve que ce n'est pas le cas (fausse alarme), il s'arrêtera immédiatement et la prochaine routine commence pour le deuxième périphérique qui utilise cette même interruption, et cætera. Il vérifie le périphérique comme décrit ci-dessus. Cette séquence est répétée jusqu'à la découverte du périphérique qui a lancé cette interruption. Toutes les routines d'interruption pour une interruption constituent une chaîne. Donc, la chaîne est traversée jusqu'à ce qu'une routine de la chaîne réclame l'interruption en disant : cette interruption est pour moi. Après avoir géré l'interruption, les routines suivantes du service d'interruption ne sont pas exécutées.

Mettre un certain voltage sur une ligne IRQ revient seulement à demander que le CPU s'interrompe de façon à exécuter la routine du pilote du périphérique. Dans pratiquement tous les cas, le CPU est interrompu par la requête. Mais les interruptions du CPU peuvent être temporairement désactivées ou « faire la queue », et donc, dans de rares cas, une interruption peut ne pas être gérée (ou peut subir un certain délai). Donc, ce qui a été auparavant appelé une interruption est plus précisément une « demande d'interruption », ce qui explique l'acronyme d'IRQ (« Interrupt ReQuest », c'est-à-dire ReQuête d'Interruption).

Comment le pilote de périphérique récupère son interruption

L'indication précédente, à savoir que les pilotes de périphérique écoutent leur interruption, était une explication très simplifiée. En fait, il s'agit d'un composant (ou d'une partie d'un composant) embarqué sur la carte-mère, appelé le contrôleur d'interruptions. Il va écouter toutes les interruptions. Quand le contrôleur récupère une interruption, il envoie un signal au CPU pour lancer la routine du service d'interruption du pilote de périphérique approprié pour gérer cette interruption.

Il existe différents types de contrôleurs d’interruptions. L’un d’entre eux est l’APIC (acronyme de Advanced Programmable Interrupt Controller) qui a des broches en entrée pour toutes les interruptions, y compris les interruptions PCI. Les anciens contrôleurs ont seulement des broches pour les interruptions ISA mais ils peuvent toujours gérer les interruptions PCI car il s’agit d’un routeur programmable d’interruptions qui convertit les interruptions PCI en interruptions ISA et les envoie vers certaines broches (c’est-à-dire vers certaines IRQ). Un autre schéma est que le routeur informe directement le CPU des interruptions PCI et qu’il laisse le contrôleur d’interruption ISA gérer les interruptions ISA en contactant le CPU via le routeur.

S’il y a plus d’un pilote de périphérique, au lieu de lancer un service h(), comment écoutent-ils ? Un pilote de périphérique fonctionnel est seulement du code dans la mémoire de l’ordinateur. Comment une interruption qui est une simple impulsion électrique sur un fil IRQ de la carte-mère, atteint ce code du pilote et lance l’exécution de la routine du service d’interruption ?

Il existe une pièce matérielle connue sous le nom de contrôleur d’interruption. Un exemple moderne est l’APIC (*Advanced Programmable Interrupt Controller*).

Isolation ISA

C’est uniquement pour l’ancien bus ISA. L’isolation est une méthode complexe d’affectation d’un point temporaire (numéro d’identifiant ou CSN, *Card Select Number*) à chaque périphérique PnP du bus ISA. Comme il existe des moyens plus efficaces (mais plus complexes) de le faire, certains pourraient dire qu’il s’agit d’une méthode simple. Seule une adresse d’écriture est utilisée pour les écritures PnP vers tous les périphériques pour que toutes les écritures vers cette adresse aillent sur tous les périphériques PnP. Cette adresse d’écriture est utilisé pour envoyer (affecter) un numéro de carte unique à chaque périphérique PnP. Pour être assigné, ce numéro de carte nécessite qu’un seul périphérique soit en écoute lorsque le numéro de carte est envoyé (écrit) à cette adresse commune. Tous les périphériques PnP ont un numéro de série unique qu’ils utilisent lors du processus d’isolation. Faire l’isolation est comme un jeu. Cela se fait en utilisant l’équivalent d’un bus commun de fils connectant tous les périphériques PnP au programme d’isolation.

Pour le premier tour du « jeu », tous les périphériques PnP écoutent sur ce fil et envoient simultanément une séquence de bits sur le fil. Les bits autorisés sont soit un 1 (voltage positif) soit un « 0 ouvert » sans voltage (circuit ouvert ou trois-états). Pour cela, chaque périphérique PnP commence à envoyer séquentiellement son numéro de série sur ce fil, voltage (circuit ouvert ou trois-états). Pour faire cela, chaque périphérique PnP lance simplement son numéro de série sur les fils, bit à bit, en commençant par le plus haut. Si un périphérique envoie un 1, un 1 sera entendu par tous les autres périphériques. Si tous les périphériques envoient un « 0 ouvert », rien ne sera entendu sur le fil. Le but est d’éliminer (à la fin du premier tour) tous les périphériques sauf celui possédant le numéro de série le plus important. « Éliminer » signifie enlever de ce tour du jeu et donc cesser temporairement d’écouter tout ce qui passe sur le fil. (Notez que tous les numéros de série ont la même taille.) Quand il ne reste qu’un seul périphérique en écoute, un numéro de carte lui est donné.

Tout d’abord, considérez seulement le bit le plus haut du numéro de série qui est placé sur le fil par tous les périphériques qui n’ont pas encore de numéro de carte. Si un périphérique PnP envoie un 0 (0 ouvert) mais entend un 1, cela signifie qu’un ou plusieurs autres périphériques PnP a un numéro de série plus important, donc il se supprime temporairement pour ce tour. Maintenant, les périphériques restant en jeu (pour ce tour) ont tous le même bit de haut niveau (un 1), donc nous pouvons supprimer ce bit et continuer avec le « reste du numéro de série » pour la suite du tour. Ensuite, recommencez depuis le début de ce paragraphe et répétez jusqu’à ce que le numéro de série soit examiné en entier pour chaque périphérique (voir plus bas pour les cas « tous à 0 »).

Donc, il est clair que seules les cartes avec un petit numéro de série sont éliminées lors d'un tour. Mais qu'arrive-t-il si tous les périphériques du jeu envoient un 0 comme leur bit de haut niveau ? Dans ce cas, un « 0 ouvert » est envoyé sur la ligne et tous les participants restent en lice. S'ils ont tous un 0 au début, alors les 0 sont supprimés comme les 1 du paragraphe ci-dessus. Le jeu continue alors avec le bit suivant du numéro de série).

A la fin du tour (après que le dernier bit ait été envoyé), seul un périphérique PnP, celui possédant le plus haut numéro de série, reste en jeu. Il se voit attribuer un numéro de carte et quitte le jeu définitivement. Ensuite, tous les autres périphériques du tour précédent (qui n'ont donc pas de numéro de carte) reviennent dans le jeu et un nouveau tour commence, avec un participant en moins. Éventuellement, tous les périphériques PnP se voient assigner un numéro de carte. Il est facile de prouver que cet algorithme fonctionne. L'algorithme actuel est un peu plus complexe que celui présenté ci-dessus car chaque étape est répétée deux fois pour s'assurer, et ces répétitions sont faites d'une façon un peu différente (mais en utilisant la même idée de base).

Une fois tous les numéros de carte assignés, ils sont utilisés pour s'adresser à chaque périphérique PnP pour envoyer/lire des données de configuration. Notez que ces numéros de carte sont seulement utilisés pour la configuration PnP et ne sont pas utilisés pour les communications normales avec le périphérique PnP. Lorsque l'ordinateur démarre, un BIOS PnP fera l'isolation puis s'occupera de la configuration PnP. Après ça, tous les numéros de carte sont « perdus » d'une telle façon que si quelqu'un veut changer (ou inspecter) la configuration une nouvelle fois, l'isolation devra être refaite intégralement.

Maîtrise du bus et ressources DMA

Si un bus dispose d'une fonctionnalité de maîtrise du bus, il est peu probable que des ressources seront nécessaires pour le DMA sur ce bus. Par exemple, le bus PCI n'a pas besoin des ressources DMA car il dispose de cette fonctionnalité. Néanmoins, la « maîtrise du bus » est souvent appelée DMA. Mais, comme il ne s'agit pas strictement de DMA, il ne nécessite aucune ressource DMA. Les bus locaux ISA et VESA n'ont pas de maîtrise du bus. Les anciens bus MCU et EISA l'avaient.

Historique et obsolescence

Pilote son OSS-Lite

Vous devez donner l'adresse d'entrée/sortie, l'IRQ et le canal DMA comme paramètres au module ou les compiler dans le noyau. Mais certaines cartes PCI seront automatiquement détectées. RedHat fournit un programme **sndconfig** qui détecte les cartes ISA PnP et configure automatiquement les modules en chargeant les ressources bus détectées.

ALSA (Architecture son avancée pour Linux) en l'an 2000

Ceci détectera la carte par des méthodes PnP, puis sélectionnera le pilote et le chargera. Il configurera aussi les ressources bus sur les cartes ISA-PnP et sur les cartes PCI. Il remplace OSS (Open Sound System), auparavant populaire.

Notes sur MS Windows Notes

Windows NT4 ne supportait pas ISAPNP mais dispose d’ un programme PNPISA que vous pouvez utiliser « à vos risques et périls ». Pour NT4, les utilisateurs se sont vus conseiller de ne pas configurer le BIOS avec l’indication que le système d’exploitation est PnP de façon à ce que le BIOS s’occupe de la configuration des ressources. Du coup, MS Windows et Linux étaient auparavant dépendants de la configuration du BIOS (et le sont toujours).

A. Adaptation française

Traduction

La traduction française de ce document a été réalisée par Guillaume Lelarge <`gleu CHEZ wanadoo POINT fr`>.

Préparation de la publication

La publication de ce document a été préparée par Jean-Philippe Guérard <`fevrier CHEZ tigreraye POINT org`>.