Le HOWTO *Plug-and-Play* pour Linux

David S.Lawyer *dave@lafn.org* v0.08, 28 Novembre 1999

Aide à l'utilisation et à la compréhension des périphériques Plug-and-Play. Comment rendre votre système Linux apte à gérer le Plug-and-Play.

Contents

Adaptation française : Albert-Paul Bouillot *apb@club-internet.fr*

Définition : ***Plug-and-Play*** (Installez-et-Utilisez),**PnP**. C'est du jargon. C'est un matériel ou un logiciel, qui, après avoir été installé ("plugged in"), peut être utilisé immédiatement ("played with"), par opposition au matériel ou au logiciel qui nécessite d'être configuré.

1 Introduction

1.1 Copyright, Marques déposées, Limitation de garantie & Crédits

1.1.1 Copyright

1.1.2 Marques déposées

Si certains mots sont des marques déposées, le contexte doit en mettre l'appartenance en évidence. Par exemple "MS Windows" (ou simplement "Windows") implique que "Windows" appartient à Microsoft.

1.1.3 Limitation de garantie

Une grande partie des informations contenues dans cet HOWTO ont été obtenues d'Internet, de livres qui peuvent être erronés ou obsolètes etc. Bien que je n'aie pas essayé de vous induire en erreur sciemment, il peut y avoir un certain nombre d'inexactitudes dans ce document. Ayez l'obligeance de me le faire savoir. Dans la mesure où cette documentation est gratuite, il est bien évident que ni moi, ni les auteurs précédents (ni le traducteur ;-)) ne pourront être tenus pour légalement responsables des erreurs éventuelles.

1.2 Pour le futur : Vous pouvez contribuer

Merci de me faire part de toutes erreurs dans les faits, les opinions, la logique, l'orthographe et la grammaire, la clarté, les liens etc. que vous pourrez rencontrer. Mais d'abord, vérifiez que le document ne soit pas trop ancien et que vous possédez la dernière version. Merci de m'envoyer toutes les infos que vous pensez devoir concerner ce document.

Je n'ai pas étudié en détail ni les outils "isapnptools" ni les correctifs du noyau de David Howells (mais j'envisage de le faire). De même, je ne comprends pas totalement comment le BIOS configure le PnP (cela dépend de la version du BIOS) ni comment Windows9x met à jour l'ESCD. Donc, cet HOWTO est encore incomplet et peut se révéler inexact (faites-moi savoir où sont les erreurs). Dans cet HOWTO j'ai parfois utilisé;nbsp;?? pour indiquer que je ne connais pas vraiment la réponse.

1.3 Nouvelles versions de cet HOWTO

Les nouvelles versions du Plug-and-Play-HOWTO devraient être disponibles à peu près tous les mois pour consultation et/ou télé-chargement sur les sites miroirs du LDP. Pour obtenir une liste de ces sites, consultez :

`http://metalab.unc.edu/LDP/mirrors.html` . Différents formats sont disponibles. Si vous voulez rapidement vérifier la date de la dernière version, regardez à :

`http://metalab.unc.edu/LDP/HOWTO/Plug-and-Play-HOWTO.html` .

2 Que devrait faire le PnP : Allouer les "Ressources du bus"

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

En simplifiant beaucoup, le *Plug-and-Play* est une méthode pour indiquer automatiquement au logiciel (pilotes de périphériques) où trouver différents éléments matériels (périphériques) tels que modems, cartes réseau, cartes son etc. La tâche du *Plug-and-Play* consiste à mettre en correspondance les périphériques physiques et les logiciels (pilotes de périphériques) qui les font fonctionner et à établir un canal de communication entre chaque périphérique et son pilote. Pour ce faire, le PnP alloue les "ressources bus" suivantes à la fois aux pilotes et au matériel : adresses d'E/S, IRQ, canaux DMA (uniquement pour le bus ISA), régions mémoire. Si vous ne comprenez pas ce que signifient ces quatre notions, lisez les paragraphes suivants. Après que ces ressources du bus ont été assignées (et si les pilotes adéquats sont installés), les noms des unités périphériques dans le répertoire /dev sont prêts à être utilisés.

Cette méthode d'affectation de certaines ressources du bus est quelquefois désignée par le terme "configuration", mais c'est uniquement d'une configuration de bas niveau qu'il s'agit. Même lorsque le *PnP* est utilisé au maximum, une bonne partie de la configuration des périphériques est réalisée par autre chose que le *PnP*. Par exemple, pour la configuration du modem, une "chaîne de caractères d'initialisation" est envoyée au modem à travers le "canal" adresse d'E/S. Cette "chaîne de caractères d'initialisation" n'a rien à voir avec le PnP bien que le "canal" utilisé pour l'envoyer ait été alloué par le PnP. Le réglage de la vitesse (et de beaucoup d'autres paramètres) d'un port série est réalisé en envoyant, de programmes exécutés par l'utilisateur (souvent automatiquement, au démarrage), des messages au pilote de périphérique. Et cette configuration n'a rien à voir avec le *PnP*. Donc, quand on parle de la configuration effectuée par le*PnP*, cela ne concerne qu'un certain type de configuration. Alors que d'autres documentations (telle celle de MS Windows) parlent simplement de "ressources" quand il s'agit des ressources du bus, j'ai volontairement utilisé le terme "ressources du bus" pour les distinguer de la multitude d'autres ressources disponibles.

2.2 Comment un ordinateur trouve-t-il les périphériques (et inversement)

Un ordinateur est constitué d'un CPU/processeur qui effectue les traitements, et de mémoire pour stocker les programmes et les données. De plus, il y a un certain nombre de périphériques tels que différents types d'unité de disques, une carte vidéo, un clavier, des cartes réseau, des cartes modem, des cartes son, des ports série et parallèle etc. Il y a également une alimentation pour fournir l'énergie nécessaire, différents bus sur une carte mère pour connecter les périphériques au CPU, ainsi qu'un boîtier pour contenir le tout.

Dans les temps anciens, la plupart des périphériques avaient leur propre carte d'interface (circuits imprimés). Aujourd'hui, en plus des cartes d'interface, beaucoup de "périphériques" sont de petits circuits montés de façon permanente sur une carte unique appelée "carte mère". Les cartes qui se connectent sur la carte mère peuvent contenir plus d'un périphérique. Les circuits mémoire sont quelquefois également considérés comme des périphériques mais ne sont pas *plug-and-play* au sens de cet HOWTO.

Pour qu'un système informatique fonctionne correctement, chaque unité périphérique doit être sous le contrôle de son "pilote de périphérique" Ce logiciel fait partie du système d'exploitation (ou d'un module) et tourne sur le CPU. Les pilotes de périphériques sont associés à des "fichiers spéciaux" dans le répertoire /dev, bien que ce ne soient pas vraiment des fichiers. Ils ont des noms tels que hda1 (première partition sur le disque dur a), ttyS0 (premier port série), eth1 (deuxième carte Ethernet) etc. Pour rendre les choses plus compliquées, le pilote de périphérique sélectionné, disons pour eth1, dépendra du type de carte Ethernet que vous aurez. Donc, eth1 ne peut pas être affecté à n'importe quel pilote Ethernet, mais à un certain pilote qui fonctionnera avec le type de carte que vous avez installé. Pour gérer un périphérique, le CPU (sous le contrôle du pilote de périphérique), envoie des commandes (et des données) vers les différents périphériques et en lit les infos. Pour cela, chaque pilote de périphérique doit connaître l'adresse utilisée pour le périphérique qu'il gère. Connaître une telle adresse correspond à mettre en oeuvre un canal de communication, même si le "canal" physique est en réalité un bus de données dans le PC qui est partagé avec pratiquement tout le reste.

Le canal de communication est en réalité un peu plus compliqué que la description qui en est faite ci-dessus. Une "adresse" est en réalité un ensemble d'adresses et il y a un canal inverse (les interruptions) qui permet aux périphériques d'envoyer, en urgence, une requête de "demande d'aide" au pilote de périphérique.

2.3 Adresses d'E/S etc.

Les PC utilisent trois espaces d'adresses : mémoire principale, E/S, et configuration (uniquement dans le cas du bus PCI). Les trois types d'adresses partagent le même bus à l'intérieur du PC. Mais la tension sur certains fils spécialisés indique à tous les périphériques à quel "espace" une adresse appartient : E/S, mémoire centrale, ou configuration. Consultez 8.1 (Adresses) pour plus de détails. Les périphériques sont normalement situés dans l'espace d'adressage E/S bien que, maintenant ils utilisent l'espace d'adressage mémoire centrale. Une adresse E/S (Entrée/Sortie) (ou I/0, pour Input/Output) est quelquefois appelée simplement "I/O", "IO", "i/o" ou "io". Le terme "port d'E/S" (ou "I/O port") est également utilisé. Il y a deux étapes principales pour allouer les adresses d'E/S (ou d'autres ressources telles que les interruptions) :

1. Affecter l'adresse d'E/S etc. sur la carte (dans l'un de ses registres)
2. Communiquer cette adresse d'E/S etc. au pilote.

Le processus en deux étapes ci-dessus ressemble un peu au problème de la recherche du numéro de la maison de quelqu'un dans une rue. Vous devez obtenir (et noter) le numéro de la maison et quelqu'un doit avoir apposé le numéro sur la maison afin qu'on puisse le trouver. Dans les ordinateurs, le pilote de périphérique doit obtenir l'adresse, et le matériel du périphérique doit installer la même adresse dans l'un de ses registres. Ces deux opérations doivent être réalisées, mais certains font l'erreur de ne réaliser que l'une de ces deux étapes et se demandent ensuite pourquoi l'ordinateur ne peut pas trouver le périphérique. Par exemple, ils utiliseront "setserial" pour attribuer une adresse à un port série, sans penser que cela indique seulement au pilote quelle est cette adresse. Cela ne sélectionne pas l'adresse physique du port série lui-même. Si le port série a, en réalité, une adresse différente (ou pas d'adresse du tout)et que vous indiquiez une adresse erronée a setserial, vous avez des problèmes.

Une autre obligation, évidente, consiste a avoir paramétré l'adresse d'E/S sur la carte avant que le pilote de périphérique n'essaie d'utiliser cette adresse. Comme les pilotes de périphériques démarrent souvent dès la mise en route de l'ordinateur, ils essaient quelquefois d'accéder à une carte (pour tester sa présence etc.) avant que l'adresse ait été paramétrée dans la carte par le programme de configuration *PnP*. Ainsi, vous voyez des messages d'erreur indiquant qu'une carte est introuvable bien qu'elle soit présente (mais son adresse n'a pas encore été attribuée).

Ce qui vient d'être dit dans les deux derniers paragraphes concernant les adresses d'E/S s'applique avec la même force aux autres ressources : 2.4 (Les IRQ –Vue d'ensemble), 2.5 (Canaux DMA),et 2.6 (Espace mémoire). C'est ce que nous allons expliquer dans les trois paragraphes suivants.

2.4 Les IRQ, vue d'ensemble

Après avoir lu ce paragraphe, il faut lire 8.2 (Interruptions –Détails) pour avoir quelques détails complémentaires. Ce qui suit est volontairement très simplifié : en plus de l'adresse, il faut également prendre en compte un numéro d'interruption (tel que IRQ5), que l'on appelle également numéro d'IRQ (IRQ = Interrupt ReQuest, demande d'interruption). Nous avons déjà mentionné plus haut que le pilote de périphérique doit connaître l'adresse de la carte de façon à pouvoir communiquer avec elle. Mais comment se fait la communication dans l'autre sens ? Supposons que le périphérique ait besoin de dire quelque chose à son pilote immédiatement. Par exemple, le périphérique peut venir de recevoir une rafale d'octets destinés à aller en mémoire centrale et que le périphérique ait besoin de demander à son pilote de venir les récupérer immédiatement de sa mémoire tampon pratiquement pleine pour les transférer en mémoire centrale.

Comment le périphérique peut-il appeler son pilote à l'aide ? Il ne peut pas utiliser le bus de données principal puisque celui-ci est déjà en cours d'utilisation. Au lieu de cela, il lance un appel en mettant une certaine tension sur une ligne d'interruption (qui fait partie du bus). Il y a l'équivalent de 16 lignes, et chacune d'elle correspond (indirectement) à un certain pilote de périphérique. Chaque ligne possède un numéro d'interruption (IRQ) unique (IRQ = Interrupt ReQuest). Le périphérique doit envoyer son interruption sur la bonne ligne et le pilote de périphérique doit attendre l'interruption sur cette ligne. Le numéro d'IRQ stocké dans le périphérique détermine quelle est la ligne concernée. Ce même numéro d'IRQ doit être connu du pilote de périphérique de manière que celui-ci sache quelle ligne d'IRQ le concerne.

Lorsqu'un pilote de périphérique reçoit une interruption (un appel à l'aide) il doit se renseigner sur la cause de l'émission de cette interruption et effectuer les actions appropriées pour y répondre. Sur un bus ISA chaque périphérique a besoin d'un numéro d'interruption unique (sauf que deux ports série ou plus peuvent partager la même interruption depuis le noyau 2.2). Sur un bus PCI, le partage des interruptions est permis.

2.5 Les canaux DMA

Les canaux DMA ne concernent que le bus ISA. DMA signifie "Accès Direct Mémoire" (Direct Memory Access). Dans ce cas, le périphérique est autorisé à prendre le pas sur le CPU pour le contrôle du bus et à transférer les octets directement en mémoire centrale. Normalement le CPU effectuerait un tel transfert en deux étapes :

1. Lecture de l'espace mémoire E/S du périphérique et stockage de ces octets dans le CPU lui-même
2. écriture de ces octets du CPU dans la mémoire centrale.

Avec le DMA le processus de transfert des octets du périphérique vers la mémoire centrale se déroule normalement en une seule étape. Les périphériques doivent avoir cette capacité intégrée dans le matériel, et donc tous les périphériques ne fonctionnent pas en DMA. Pendant que le transfert DMA est en cours d'exécution, le CPU ne peut pas faire grand-chose puisque le bus principal est utilisé par ce transfert DMA.

Le bus PCI n'a pas vraiment de DMA, mais plutôt quelque chose de mieux : le "contrôle du bus" (bus mastering). Cela fonctionne un peu comme le DMA et on l'appelle quelquefois DMA (par exemple, les unités de disques durs se nomment "UltraDMA"). Cette technique permet aux périphériques de prendre temporairement le contrôle du bus et de transférer des octets exactement comme le fait le CPU lorsqu'il a le contrôle du bus. On n'utilise pas de numéros de canaux car le bus PCI est constitué de telle manière que le matériel sait quel est le contrôleur du bus actuel et quel est le périphérique qui demande à le devenir. Il n'y a donc pas d'allocation de canaux DMA dans le cas d'un bus PCI.

Quand un périphérique, sur un bus ISA, désire effectuer un transfert DMA, il génère une requête de DMA en utilisant une ligne spécialisée comme pour une demande d'interruption. Le DMA aurait pu être réalisé en utilisant les interruptions mais cela aurait introduit des retards, donc il est plus rapide d'utiliser un type d'interruption spécial connu sous le nom de requête DMA. Comme les interruptions, les requêtes DMA sont numérotées de façon à identifier le périphérique qui a envoyé cette requête. Ce numéro est appelé canal DMA. Puisque tous les transferts DMA utilisent le bus principal (et qu'un seul utilisateur ne peut utiliser ce bus à un instant donné) ils utilisent tous le même canal, mais le numéro de "canal DMA" sert à identifier qui est en train d'utiliser le "canal". Des registres existent sur la carte mère pour stocker l'état courant de chaque "canal". Donc, pour émettre une requête DMA, le périphérique doit connaître son numéro de canal DMA qui doit être stocké dans un registre sur le périphérique physique.

2.6 Espace mémoire

Certains périphériques se voient assigner des adresses dans l'espace de la mémoire principale. On parle souvent de "mémoire partagée" ou "d'E/S en zone mémoire". Quelquefois, il s'agit de mémoire morte (ROM) sur le périphérique. A propos de ressources *PnP*, on parle simplement de "mémoire". Un tel périphérique peut également utiliser l'espace d'adressage E/S.

Lorsque vous branchez une telle carte, en réalité vous connectez un module mémoire pour la mémoire principale. Cette mémoire peut être soit de la mémoire morte (ROM = Read Only Memory) soit de la mémoire partagée. La mémoire partagée est partagée entre le périphérique et l'Unité Centrale (CPU qui exécute le pilote de périphérique). Cette mémoire peut servir de moyen de "transfert" direct des données entre le périphérique et la mémoire principale. En réalité, ce n'est pas un transfert à proprement parler puisque le périphérique met ses données dans sa propre mémoire qui se trouve être également la mémoire centrale. À la fois la carte et le pilote de périphérique doivent savoir où cela se passe. Il faut utiliser des adresses mémoires hautes de façon à éviter tout conflit avec les circuits mémoires des adresses basses (de la mémoire centrale) de votre ordinateur.

Pour la ROM, c'est différent. C'est comme un programme (peut-être le pilote de périphérique) qui serait utilisé avec le périphérique. Heureusement, il peut tourner avec Linux et pas seulement avec Windows ?? Il peut être nécessaire de le dupliquer en mémoire centrale pour le faire tourner plus vite. Une fois dupliqué, il n'est plus "en lecture seule".

2.7 Les "ressources" pour le périphérique et son pilote

Donc, les pilotes de périphériques doivent être "attachés" d'une façon quelconque au matériel qu'ils contrôlent. Cela est réalisé en fournissant des "ressources" (E/S, Mémoire, IRQ, DMA) à la fois au périphérique physique et au logiciel de pilotage. Par exemple, un port série n'utilise seulement que 2 ressources (parmi les 4 possibles) : une IRQ et une adresse d'E/S. Ces deux valeurs doivent être fournies au pilote de périphérique et au périphérique physique. On donne alors un nom (tel que ttyS1) au pilote (et à l'unité périphérique correspondante) dans le répertoire /dev. L'adresse et le numéro d'IRQ sont stockés dans un registre sur la carte du périphérique physique (ou dans un circuit sur la carte mère). Si l'on utilise de cavaliers, cette information est toujours stockée dans la partie "matériel" du périphérique (sur la carte etc.). Mais dans le cas du *PnP*, la donnée enregistrée est généralement perdue lors de l'arrêt du PC (mise hors tension) de sorte que cette donnée doit être fournie à chaque fois que le PC est remis en fonction.

2.8 Le problème

L'architecture du PC fournit seulement un nombre limité d'IRQ, de canaux DMA, d'adresses d'E/S et de régions mémoire. S'il n'y avait que quelques périphériques et qu'ils aient des ressources standardisées (comme des adresses d'E/S et des numéros d'IRQ uniques) il n'y aurait aucun problème pour attacher des pilotes de périphériques aux unités périphériques. Chaque unité aurait des ressources fixes qui n'entreraient en conflit avec aucune autre unité périphérique de l'ordinateur. Deux périphériques n'auraient pas la même adresse d'E/S, la même IRQ etc. Chaque pilote pourrait être programmé avec l'adresse d'E/S, l'IRQ etc. codées en dur dans le programme. La vie serait simple.

Mais ce n'est pas le cas. Non seulement il existe tellement de périphériques différents aujourd'hui que les conflits sont fréquents, mais, de plus, on a souvent besoin d'avoir plus d'un périphérique d'un type donné. Par exemple, on peut vouloir disposer de plusieurs unités de disques différentes, de plusieurs ports série etc. Pour ces raisons, les périphériques doivent être paramètrables afin que l'on puisse leur attribuer les adresses, IRQ etc. que l'on désire pour éviter les conflits. Mais quelques IRQ et adresses sont tout à fait standard comme celles de l'horloge et du clavier. Elles ne nécessitent pas une telle adaptabilité.

En dehors du problème de conflit d'allocation des ressources, il y a le problème de l'erreur en communiquant la nature des ressources au pilote de périphérique. Par exemple, supposons que vous ayez entré IRQ 4 dans un fichier de configuration alors que le périphérique utilise, en réalité, l'IRQ 5. C'est un autre type d'erreur dans l'allocation des ressources du bus.

L'allocation des ressources du bus, si elle est correctement réalisée, établit des canaux de communication entre les périphériques physiques et leurs pilotes. Par exemple, si une certaine gamme d'adresses d'E/S (ressource) est allouée à la fois à un pilote de périphérique et à un matériel, alors on a établi un canal de communication à sens unique entre eux. Le pilote peut envoyer des commandes et des informations au périphérique. C'est, en réalité, un peu plus qu'un canal à sens unique puisque le pilote peut obtenir des informations du périphérique en consultant ses registres. Mais le périphérique ne peut pas prendre l'initiative d'une communication de cette façon. L'allocation d'une IRQ est nécessaire pour créer un canal de communication bi-directionnel, où à la fois le pilote et le périphérique peuvent prendre l'initiative d'une communication.

2.9 Le *PnP* détecte les périphériques connectés aux ports série

Les périphériques externes qui se connectent au port série par un câble (comme les modems externes) peuvent également être appelés *Plug-and-Play* Puisque seul le port série lui-même nécessite des ressources (une IRQ et une adresse d'E/S), il n'y a pas de ressources à allouer à un périphérique qui se connecte à un tel port. Donc, le *PnP* n'a pas vraiment de raison d'être pour ces périphériques. Cela, même s'il y a des spécifications *PnP* pour de tels périphériques série.

Un système d'exploitation *PnP* les détectera et saura lire leur numéro de modèle etc. Il sera donc alors capable de trouver un pilote de périphérique adapté et vous n'aurez pas à dire à un programme d'application que vous utilisez disons, /dev/ttyS1. Mais puisque vous devriez être capable d'indiquer manuellement à votre ordinateur (par l'intermédiaire d'un fichier de configuration etc.) sur quel port série votre périphérique est connecté (et éventuellement le modèle dont il s'agit) vous n'avez pas réellement besoin de cette fonctionnalité "port série" du *PnP*.

3 La solution *Plug-and-Play* (PnP)

3.1 Introduction au *PnP*

Le terme *Plug-and-Play* (PnP) (Installez et Utilisez) a plusieurs sens. Au sens large c'est uniquement une auto-configuration où l'on installe simplement un périphérique et que celui-ci se configure par lui-même. Au sens où je l'entends dans cet HOWTO, la configuration ne concerne que la configuration de la ressource PnP et la communication de l'information au pilote de périphérique. Dans un sens plus étroit c'est simplement le paramètrage des ressources des périphériques. Cela peut aussi concerner les spécifications qui, (entre autres choses) indiquent comment les données PnP de la ressource doivent être lues ou écrites dans les périphériques (souvent des cartes) sur le bus ISA. Les spécifications du standard PCI (et non PnP) sont de même nature pour le bus PCI.

Le *Plug-and_Play (PnP)* met en relation les périphériques et leurs pilotes et spécifie leurs "canaux" de communication. Sur le bus ISA, avant le *Plug-and-Play*, les ressources étaient configurées sur le matériel par des cavaliers et les pilotes de périphériques étaient assignés aux ressources par des fichiers de configuration (ou quelque chose du même style) ou par des tests des périphériques à des adresses où ils étaient supposés se trouver. Le bus PCI a été compatible PnP dès le début et il a été trivial d'implémenter le PnP pour ce bus. Mais comme les spécifications du bus PCI n'utilisent pas le terme PnP on ne sait pas si l'on doit dire que le bus PCI est PnP (mais il supporte le matériel que l'on appelle aujourd'hui PnP).

3.2 Comment cela fonctionne-t-il ? (explication simplifiée)

Voici une explication très simple de la façon dont fonctionne le *PnP*. Le programme de configuration *PnP* (éventuellement un programme du BIOS) recherche tous les périphériques *PnP* et demande à chacun les ressources du bus dont il a besoin. Ensuite, il regarde les ressources du bus (IRQs, etc.) qu'il doit attribuer. Bien sûr, s'il y a des ressources du bus réservées par des périphériques (anciens) non *PnP* (dont il a connaissance), il ne les attribue pas. Puis il utilise un certain critère (non précisé dans les spécifications *PnP*) pour attribuer les ressources de façon à ce qu'elles n'entrent pas en conflit et que tous les périphériques obtiennent ce dont ils ont besoin. Il indique alors à chaque périphérique physique quelles ressources du bus lui ont été affectées et chacun se configure alors pour n'utiliser que les seules ressources du bus qui lui ont été affectées.

Par exemple, supposons qu'une carte carte ait besoin d'une interruption (d'un numéro d'interruption) et de 1 MB de mémoire partagée. Le programme *PnP* lit cette requête dans la carte. Il attribue alors le numéro d'interruption 5 (IRQ 5) et un méga-octet d'espace mémoire à partir de l'adresse 0xe9000000. Ce n'est pas toujours aussi simple car la carte peut utiliser uniquement certains numéros d'interruption (seulement pour ISA) ou n'accepter qu'un certain domaine d'adressage. Les détails sont différents pour les bus ISA et PCI, les choses étant plus complexes pour le bus ISA

Le logiciels *PnP* peut utiliser quelques raccourcis. L'un d'entre eux consiste à garder une trace de la manière dont il a assigné les ressources du bus lors de la dernière configuration (lors de la dernière utilisation de l'ordinateur) et la réutiliser. Windows9x et les BIOS PNP le font mais ce n'est pas le cas du Linux standard. Windows9x enregistre ces informations dans ses "Registres" sur le disque dur et un BIOS PnP le fait dans la mémoire non volatile de votre PC (connu sou le nom d' ESCD; voir 5.3.2 (La Base de données ESCD du BIOS)).

Sous Linux, c'est le règne du "chaque périphérique pour lui-même" et il n'existe aucun registre centralisé des affectations de ressources. Certains pilotes de périphériques enregistrent la dernière configuration qu'ils ont utilisé et la réutilisent à la prochaine remise en route de l'ordinateur. Implicitement, ils considèrent que le reste du matériel n'aura pas besoin d'utiliser ses ressources du bus.

Si les périphériques se souvenaient de leur configuration précédente, il n'y aurait aucun matériel pour les reconfigurer au prochain démarrage, mais ils semblent oublier leur configuration lorsque l'on stoppe l'alimentation. Certains périphériques ont une configuration par défaut (mais pas nécessairement la dernière utilisée). Le programme de configuration *PnP* doit être exécuté à chaque mise sous tension du PC. De même, si l'on ajoute un nouveau périphérique, celui-ci a besoin d'être configuré. L'allocation de ressources du bus à ce nouveau périphérique peut nécessiter d'en enlever à un périphérique existant et de donner à celui-ci des ressources du bus différentes qu'il pourra utiliser à la place.

3.3 Démarrage du PC

Lors de la mise sous tension initiale du PC le circuit BIOS exécute son programme pour démarrer l'ordinateur (la première étape consistant à tester le matériel). Si le système d'exploitation est stocké sur le disque dur (ce qui constitue le cas normal), le BIOS doit trouver le disque dur. Si le disque dur est *PnP*, alors le BIOS doit utiliser une méthode PnP pour le trouver. Donc, pour permettre à l'utilisateur de configurer manuellement la mémoire CMOS du BIOS et de réagir aux messages d'erreur apparaissant au démarrage de l'ordinateur, un écran (une carte vidéo) et un clavier sont également nécessaires. Donc, le BIOS doit configurer, selon la procédure PnP, ces périphériques lui-même.

Une fois que le BIOS a identifié le disque dur, la carte vidéo et le clavier il est prêt à entamer le processus de démarrage (chargement du système d'exploitation du disque dur vers la mémoire centrale). Si vous avez indiqué à votre BIOS que vous aviez un système d'exploitation *PnP* (SE PnP ou PnP OS), il devrait commencer à démarrer le PC de la manière indiquée ci-dessus et laisser le soin de terminer la configuration *PnP* au système d'exploitation. Autrement, un BIOS-PnP essaiera de réaliser le reste de la configuration *PnP* des périphériques avant le chargement du système d'exploitation (mais ne chargera pas leurs pilotes).

3.4 Les bus

Le bus ISA est le vieux bus du vieil IBM PC alors que le bus PCI est un bus plus récent et plus rapide de Intel. Le bus PCI a été conçu pour ce que l'on appelle aujourd'hui le *PnP*. Il est facile (si l'on compare au bus ISA) de voir comment les ressources du bus ont été assignées aux périphériques. Pour voir comment cela s'est passé consulter le "fichier" /proc/pci (/proc/bus/pnp/devices pour les noyaux 2.2+), les messages de démarrage sur votre écran (utilisez shift-PageUp pour revenir en arrière), ou utilisez les utilitaires PCI (pour les noyaux 2.2+).

Pour ce qui concerne le bus ISA, il y a un vrai problème pour implémenter le *PnP* car personne n'y pensait lors de la conception de ce bus et il n'y a pratiquement pas d'adresse d'E/S disponible pour communiquer les informations de configuration aux périphériques physiques. Il en résulte que la manière dont le *PnP* a été implanté sur le bus ISA est très compliquée. Un livre entier a été rédigé à ce sujet. Voir 5.9 (Le livre du PnP). Entre autres choses, le programme *PnP* doit attribuer à chaque périphérique *PnP* un "identificateur" provisoire de façon à pouvoir l'adresser pour la configuration *PnP*. L'assignation de ces "identificateurs" provisoires est appelée "isolation". Voir 8.4 (Isolation) pour de plus amples détails.

Finalement, le bus ISA finira par disparaître. Ce jour là, le *PnP* deviendra plus simple puisqu'il sera facile de voir comment le BIOS a configuré le matériel. Il restera cependant nécessaire de mettre en correspondance les périphériques et leurs pilotes ainsi que de configurer les périphériques ajoutés alors que le PC est en fonctionnement. Ces besoins seraient satisfaits si Linux était un système d'exploitation *PnP*.

3.5 Linux doit mieux prendre en compte le *PnP*

Le *PnP* (pour le bus ISA) a été inventé par Compaq, Intel et Phoenix. Microsoft a été un des leaders de sa promotion. Linux aurait été meilleur si le *PnP* n'avait jamais été "inventé". Finalement, le bus ISA aura disparu et le bus PCI à la mode *PnP* prévaudra de sorte que nous aurons alors un *PnP* facile à implanter. Mais, que l'on veuille ou pas, la plupart des nouveaux matériels ISA aujourd'hui sont *PnP* et Linux ne peut pas faire autrement que de gérer efficacement le *PnP*. Cependant le Linux standard (celui du début 1999) rend la gestion du *PnP* compliquée (spécialement pour le bus ISA) alors que le but du *PnP* était de rendre les choses faciles.

D'une certaine façon, Linux est un peu *PnP* pour le bus PCI. Quand le PC démarre vous pouvez noter, en lisant les messages sur la console, que certains pilotes de périphériques Linux trouvent leurs périphériques (et les ressources du bus que le BIOS leur a attribué). Mais il y a des situations où un système d'exploitation *PnP* pourrait prendre les choses en compte de manière plus satisfaisante :

1. En cas de pénurie de ressources du bus;
2. S'il y a plus d'un pilote pour un périphérique physique;
3. Si un pilote de périphérique activé ne peut pas trouver le périphérique physique;
4. Installation à chaud d'un périphérique.

Les utilisateurs de Linux ne devraient pas avoir à plonger dans les détails du *PnP* pour configurer leurs périphériques *PnP* ISA comme ils doivent le faire. Une solution pourrait consister à avoir une version standardisée du noyau qui supporterait le *Plug-and-Play* sur les bus ISA, PCI et autres. Un correctif pour le noyau a été écrit, bien que la plupart des pilotes ne le gèrent pas. Il ne fait pas partie du Linux standard. Voir 5.6 (Correctif noyau).

4 Configuration d'un BIOS PnP

À la mise sous tension de l'ordinateur, le BIOS s'exécute avant que le système d'exploitation ne soit chargé. Les nouveaux BIOS sont PnP et configurent quelques-uns ou la totalité des périphériques PnP. Il n'est pas possible de désactiver la plupart des BIOS PnP, donc il faut faire avec. Voici quelques-uns des choix possibles dans le menu BIOS (quelquefois appelé CMOS) :

- 4.1 (Avez-vous un système d'exploitation PnP ?)
- 4.2 (Comment doivent être contrôlées les ressources du bus ?)
- 4.3 (Réinitialiser la configuration ?)

4.1 Avez-vous un système d'exploitation PnP ?

Si vous répondez oui, alors le BIOS PnP commencera à configurer le disque dur, la carte vidéo et le clavier pour permettre le démarrage du système. Mais il laissera le soin de terminer la configuration au système d'exploitation. Il peut faire une 8.4 (isolation) sur le bus ISA en laissant les périphériques désactivés, mais prêts à être configurés par le système d'exploitation. Pour Linux, vous devrez probablement indiquer que votre système d'exploitation n'est pas *PnP*. Si vous ne le faites pas, le BIOS risque de laisser vos périphériques ISA, qu'il n'a pas configurés,en état désactivé ?? Et les périphériques PCI peuvent également ne pas être configurés ?

Si vous répondez non, alors le BIOS fera la configuration lui-même. Il utilisera la configuration précédente, sauvegardée dans la mémoire non volatile (ESCD), à moins que vous n'ayez ajouté de nouveaux périphériques PnP. Voir 5.3.2 (La base de données ESCD du BIOS) Si votre dernière session d'utilisation de l'ordinateur a été faite sous Linux, il n'y aura pas de modification de configuration. Voir 5.3 (Le BIOS configure le PnP). Mais si cette dernière session a été faite sous Windows 9x (qui est PnP), alors Windows peut avoir modifié l'ESCD. Normalement, il ne le fera que si vous "forcez" une configuration ou si vous installez un périphérique ancien. Voir 5.3.3 (Utilisation de Windows pour paramétrer l'ESCD). Si vous utilisez le(s) programme(s) utilitaire(s) isapnp ou PCI pour réaliser la configuration, ils s'exécuteront après le BIOS et modifieront les choses de la façon que vous leur aurez indiqué.

4.1.1 Cohabitation avec Windows9x

Si vous faites tourner Windows sur le même PC, comment répondre à la question du BIOS : Avez-vous un Système d'Exploitation (SE) *PnP* ? Normalement (et avec confiance) vous devriez dire non pour Linux standard et oui pour Windows9x. Mais il est particulièrement pénible d'avoir à paramétrer le menu CMOS manuellement à chaque fois que vous changez de SE. Une solution consiste à paramétrer la CMOS pour un SE non *PnP*, même si vous démarrez sous Windows. On peut espérer que Windows sera capable de prendre en compte cette situation où tout le matériel a été entièrement configuré par le BIOS. De plus, on peut penser que même si Windows ne s'aperçoit pas que le matériel a déjà été configuré, il fera la configuration lui-même et que tout ira bien. Mais il semble que cela ne se passe pas ainsi. Windows ne semble seulement capable d'indiquer aux pilotes de périphériques que ce qui est enregistré dans les registres Windows. Mais, la configuration matérielle réelle (effectuée par le BIOS) est celle qui est enregistrée dans l'ESCD et elle peut être différente de celle des registres => donc ennuis.

Une solution pour essayer d'avoir les mêmes infos dans les registres et dans l'ESCD consiste à installer (ou réinstaller) Windows après que le BIOS ait été configuré pour un "SE non *PnP*". Cela devrait permettre de présenter à Windows un système configuré par le BIOS. Si cette configuration se fait sans conflits, avec un peu de chance, Windows s'en contentera et la sauvegardera dans ses registres. Si cela fonctionne pour vous (et c'est la dernière version de cet HOWTO), faites-le moi savoir car je n'ai eu qu'une seule indication me disant que cela marchait.

Une autre méthode consiste à "enlever" les périphériques qui causent des problèmes à Windows en cliquant sur "enlever" dans le gestionnaire de périphériques. Ensuite, redémarrez le PC avec "SE non *PnP*" (paramétré dans la CMOS au moment du redémarrage°. Windows réinstallera alors les périphériques, en utilisant, on l'espère, le paramètrage des ressources du bus tel que configuré par le BIOS. Pensez que Windows peut vous demander d'insérer le CD d'installation de Windows dans le lecteur car, quelquefois, il ne trouve pas les fichiers de pilotes (et autres) même s'ils sont déjà présents. Pour le vérifier, j'ai "enlevé" la carte NIC (carte interface réseau) qui avait un pilote compatible Novell. Au redémarrage, Windows l'a réinstallée pour un réseau Microsoft au lieu de Novell. Ce qui signifie que le client Novell a dû être réinstallé. Indiquez moi vos problèmes lors de l'utilisation de cette méthode (uniquement pour la dernière version de cet HOWTO).

4.2 Comment doivent être contrôlées les ressources du bus ?

cela peut simplement signifier de décider comment allouer les ressources du bus pour les IRQ et le DMA. Sur "auto", le bios se chargera de l'allocation. Sur manuel, vous réserverez manuellement certaines IRQ à utiliser pour des cartes anciennes (cartes non-pnp). Maintenant le BIOS peut, ou non, reconnaître vos vieilles cartes. Le BIOS ne reconnaîtra vos cartes anciennes que si vous faites tourner ICU (ou quelque chose comme cela) sous Windows pour les faire reconnaître par le BIOS. S'il les reconnaît, alors essayez l'option "auto". S'il ne les reconnaît pas, alors réservez manuellement les IRQ nécessaires pour les cartes ISA anciennes et laissez BIOS PnP allouer le reste.

4.3 Réinitialiser la configuration ?

Cette opération effacera de la base de données ESCD du BIOS la manière dont les périphériques PnP doivent être configurés ainsi que la liste de configuration des périphériques anciens (non-PnP). Ne jamais faire cela à moins d'être convaincu que cette base de données est erronée et doit être refaite. Il a été dit quelque part que vous ne devriez faire cela que dans le cas où votre ordinateur refuserait de démarrer. Si le BIOS perd les données sur les périphériques anciens, alors vous devrez faire tourner à nouveau ICA sous Windows pour retrouver ces données.

5 Comment agir avec les cartes PnP

5.1 Introduction

Aujourd'hui, la plupart des nouvelles cartes internes sont *Plug-and-Play (PnP)*. Bien qu'il existe quelques logiciels sous Linux pour gérer le *PnP* ils ne sont pas toujours d'utilisation facile. Il y a 6 méthodes différentes, dont la liste suit, pour s'occuper du *PnP* (mais certaines peuvent ne pas être utilisables dans notre situation). Celle(s) à utiliser dépend(ent) de nos objectifs. Ce qui semble être le plus efficace dans l'immédiat peut se révéler ne pas être le plus facile et le meilleur dans le long terme. Une méthode qui paraît simple consiste à ne rien faire et à laisser le BIOS-PnP effectuer la configuration, mais vous devrez faire quelques investigations pour découvrir ce qu'a fait le BIOS. Il faudrait que quelqu'un qui a tester toutes les méthodes en fasse une comparaison et l'écrive. Vous pouvez avoir à utiliser plus d'une méthode pour arriver à vos fins.

- 5.2 (Désactiver le PnP) par cavaliers (mais ce n'est pas possible pour de nombreuses cartes) ou par un logiciel spécial
- 5.3 (Le BIOS configure le PnP) (Pour le bus PCI, vous avez seulement besoin d'un BIOS PCI, autrement, il vous faut BIOS PnP)
- 5.4 (Isapnp) c'est un programme que vous pouvez toujours utiliser pour configurer les périphériques PnP sur le bus ISA uniquement
- 5.5 (Utilitaires PCI) pour effectuer la configuration du bus PCI
- 5.7 (Configuration par Windows) puis vous lancez Linux à partir de Windows/DOS. À utiliser en dernier recours
- 5.6 (Patch Kernel) pour transformer Linux en un système d'exploitation PnP.
- 5.8 (Configuration par le pilote de périphérique), mais peu le font.

Chacune des solutions ci-dessus configurera les ressources du bus dans le matériel. Seule les deux dernières solutions devraient informer le pilote du périphérique de ce qui a été fait. Mais seule la dernière le fait sûrement (puisque c'est le pilote lui-même qui le fait). La façon dont le pilote se trouve informé dépend de lui et vous pouvez avoir à faire quelque chose pour le renseigner. Voir 6 (Indiquer la configuration au pilote)

5.2 Désactiver le *PnP* ?

De nombreux périphériques sont uniquement PnP, sans option pour le désactiver. Mais, pour certains, vous pouvez désactiver le *PnP* avec des cavaliers ou en utilisant un programme sous Windows qui est fourni avec le périphérique (configuration sans cavaliers). Cela évite la tâche souvent compliquée de configuration *PnP*. N'oubliez d'indiquer au BIOS que ces ressources du bus sont réservées. Il y a aussi plusieurs raisons pour lesquelles vous pouvez ne pas vouloir désactiver le *PnP* :

1. Si vous avez MS Windows sur la même machine, vous voulez permettre au PnP sous MS Windows de configurer les périphériques de façon différente que sous Linux.
2. Le champ des possibilités de choix des numéros d'IRQ (ou des adresses de ports) etc. peut être particulièrement restreint si l'on n'utilise pas le PnP.
3. Vous avez un pilote de périphérique sous Linux qui utilise les méthodes *PnP* pour chercher le périphérique qu'il contrôle.

4. Si vous avez à changer la configuration dans le futur, cela peut être plus facile en utilisant le *PnP* (pas de cavaliers à déplacer ou de programme Dos/Windows à faire tourner).
5. Vous pouvez avoir (ou aurez) d'autres périphériques *PnP* qui auront besoin d'être configurés et vous voulez donc le prévoir (ou l'étudier) de toute façon.

Une fois qu'ils sont configurés comme des périphériques non-PnP, ils ne peuvent plus être configurés par un logiciel PnP ou par le BIOS (à moins que vous ne déplaciez les cavaliers et/ou n'utilisiez à nouveau le logiciel de configuration Dos/Windows).

5.3 Le BIOS configure le PnP

5.3.1 Introduction à l'utilisation du BIOS pour configurer le *PnP*

Si vous avez un BIOS PnP, il peut configurer le matériel. Cela signifie que votre BIOS lit les besoins en ressources de tous les périphériques et les configure (leur alloue les ressources du bus). C'est un substitut de SE *PnP* à l'exception près que le BIOS ne fait pas le lien entre les périphériques et leurs pilotes et qu'il n'indique pas aux pilotes comment il a fait la configuration. Il devrait normalement utiliser la configuration qu'il a stocké en mémoire non volatile (ESCD). S'il trouve un nouveau périphérique ou en présence d'un conflit, le BIOS devrait effectuer les modifications nécessaires dans la configuration et ne pas utiliser exactement ce qu'il y avait dans l'ESCD.

Votre BIOS doit supporter une telle configuration mais il y a eu des cas où il ne l'a pas fait correctement ou l'a fait de manière incomplète. Un avantage de l'utilisation du BIOS réside dans sa simplicité puisque, dans la plupart des cas, il n'y a rien à paramétrer (sauf à indiquer dans le menu CMOS du BIOS que le SE n'est pas PnP). Alors que quelques pilotes de périphériques peuvent être capables de détecter automatiquement ce que le BIOS a fait, dans certains cas, vous devrez le faire (ce qui n'est pas toujours facile). Voir 7 (Quelle est ma configuration actuelle ?) Un autre avantage possible vient de ce que le BIOS fait son boulot avant que Linux ne démarre et donc que toutes les ressources du bus sont prêtes à être utilisées (et trouvées) par les pilotes de périphériques qui démarrent plus tard.

Selon MS le fait qu'un BIOS PNP soit capable de configurer au sens PnP les périphériques (sans l'aide de MS Windows) est optionnel (pas obligatoire). Mais il semble que la plupart de ceux créés après 1996 ?? ou aux environs en sont capables. Nous devrions leur envoyer des lettres de remerciements s'ils le font correctement. Ils configurent à la fois les bus ISA et PCI, mais on a dit que quelques vieux BIOS ne le faisaient que pour le PCI. Pour essayer d'avoir plus d'informations concernant votre BIOS, regardez sur le Web. Merci de ne pas me poser de questions sur ce sujet car je n'ai aucunes données. Les détails que vous aimeriez connaître concernant votre BIOS peuvent être difficiles à trouver (ou pas disponibles). Quelques BIOS ont des fonctionnalités *PnP* minimales et essaient de se décharger de la partie difficile de la tâche de configuration sur des utilitaires Windows. Si cela arrive, soit vous devrez trouver une autre méthode (comme les outils isapnp) soit vous devrez essayer de paramétrer la base de données ESCD si le BIOS en possède une. Consulter le paragraphe suivant.

5.3.2 La base de données ESCD du BIOS

Le BIOS entretient une base de données non volatile contenant la configuration *PnP* qu'il essaiera d'utiliser. On l'appelle ESCD (Extended System Configuration Data). Encore une fois, l'existence de l'ESCD est optionnelle mais la plupart des BIOS-PnP la possède. L'ESCD non seulement stocke la configuration des ressources des périphériques *PnP* mais aussi stocke les informations de configuration des périphériques non-PnP (et les repère comme étant non-PnP) pour éviter les conflits. Les données ESCD sont habituellement sauvegardées dans un circuit qui conserve ces données quand on arrête l'alimentation, mais quelquefois elles sont stockées sur un disque dur ??

L'objectif de l'ESCD est de garder la dernière configuration utilisée, mais si vous utilisez un programme tel que isapnp sous Linux ou des utilitaires PCI (qui ne mettent pas à jour l'ESCD) alors, l'ESCD n'en n'aura aucune connaissance et ne sauvegardera pas cette configuration dans l'ESCD. Un bon SE *PnP* doit mettre à jour l'ESCD de façon à ce que vous puissiez l'utiliser plus tard avec un SE non-PnP (comme Linux standard). Windows peut le faire dans certaines circonstances. Voir 5.3.3 (Utilisation des Windows pour paramètrer l'ESCD).

Pour utiliser ce qui a été paramétré dans l'ESCD assurez-vous de choisir "SE non-PnP" équivalent dans la CMOS du BIOS. Ensuite, chaque fois que le BIOS démarre (avant le chargement du SE Linux) il doit configurer les choses de cette façon. Si le BIOS détecte une nouvelle carte *PnP* qui n'est pas dans l'ESCD, il doit alors allouer des ressources du bus à cette carte et mettre l'ESCD à jour. Il peut même avoir à modifier les ressources du bus déjà assignées à des cartes *PnP* existantes et modifier l'ESCD en conséquence.

Si chaque périphérique a enregistré sa dernière configuration, il ne devrait pas être nécessaire de les reconfigurer à chaque redémarrage de votre PC. Mais cela ne fonctionne pas comme cela. Donc toutes les données ESCD doivent être correctes si vous utilisez le BIOS pour le *PnP*. Il y a quelques BIOS qui n'ont pas d'ESCD mais possèdent une sorte de mémoire non volatile pour stocker les informations concernant les ressources du bus qui ont été réservées par les cartes non-PnP. De nombreux BIOS possèdent les deux.

5.3.3 Utilisation de Windows pour paramétrer l'ESCD

Si le BIOS ne paramètre pas l'ESCD de la façon que vous le voulez (ou de la façon dont il faudrait), il serait intéressant d'avoir un utilitaire Linux qui le fasse. Au début 1999, il n'y en avait pas. Il faut donc essayer d'utiliser Windows pour le faire (si vous l'avez sur le même PC).

Il y a trois manières d'utiliser Windows pour essayer de paramétrer/modifier l'ESCD. L'une consiste à utiliser l'utilitaire ICU conçu pour DOS ou Windows 3.x. Cela devrait aussi fonctionner correctement pour Windows 9x/2k ?? Une autre méthode consiste à initialiser les périphériques manuellement ("forcée") sous Windows 9x/2k de sorte que Windows mette cette information dans l'ESCD lorsqu'on stoppe Windows normalement. La troisième méthode est destinée aux périphériques anciens qui ne sont pas *plug-and-play*. Si Windows les reconnaît et sait quelles ressources du bus ils utilisent, alors Windows devrait mettre cette info dans l'ESCD.

Si les périphériques *PnP* sont configurés automatiquement par Windows sans que l'utilisateur ne le "force" à changer les paramètres, alors ceux-ci ne seront probablement inscrit dans l'ESCD. Naturellement Windows peut décider de lui-même de les configurer en utilisant des paramètres identiques à ceux qui sont dans l'ESCD mais ce n'est qu'une coïncidence.

Les systèmes d'exploitation Windows 9x sont *PnP* et configurent, au sens *PnP* les périphériques. Ils entretiennent leur propre base de données *PnP* au plus profond des Registres (stockés dans des fichiers binaires Windows). Il y a également tout un tas de renseignements concernant la configuration dans les registres, en plus des ressources du bus pour le *PnP*. Il y a à la fois la configuration courante des ressources *PnP* en mémoire et une autre (peut-être à peu près la même) stockée sur le disque dur. Pour la consulter (celle qui est en mémoire ?)indirectement dans Windows98 ou pour y forcer des modifications vous utilisez le gestionnaire de périphériques.

Dans Windows98, il y a 2 manières d'invoquer le gestionnaire de périphériques :

1. Poste de travail –> Panneau de configuration –> Système –> Gestionnaire de périphériques.
2. (cliquer sur le bouton droit de la souris) Poste de travail –> Propriétés –> Gestionnaire de périphériques.

Puis dans le gestionnaire de périphériques vous sélectionnez un périphérique (quelquefois en plusieurs étapes s'il y plusieurs périphériques de la même classe). Puis cliquez sur Propriétés puis sur Ressources. Pour essayer de modifier la configuration de la ressource manuellement, désélectionnez "utiliser les paramètres automatiques" puis cliquez sur "Modification des paramètres". Maintenant essayer de modifier les paramètres, mais il se peut que cela ne soit pas possible. Si vous le pouvez, vous avez "forcé" une modification. Un message doit vous informer de ce qui a été forcé. Si vous voulez conserver le paramètrage existant de Windows mais que vous vouliez le "forcer", alors, vous devrez forcer une modification quelconque puis forcer à nouveau les paramètres originaux.

Pour voir ce qui a été "forcé" sous Windows98 regardez dans la liste du "matériel forcé" : Démarrer –> Programmes –> Accessoires –> Outils système – Information système –> Ressources matérielles –> Matériel forcé. Quand vous "forcez une modification des ressources du bus sous Windows, il doit mettre votre modification dans l'ESCD (à condition que vous quittiez Windows normalement).

> De la fenêtre "Information système" vous pouvez également contrôler comment les IRQ et les ports d'E/S ont été alloués sous Windows.

Même si Windows ne signale pas de conflit de ressources du bus, il peut en exister sous Linux. Cela vient du fait que Windows peut assigner les ressources du bus de manière différente de l'ESCD. Dans les rares cas où tous les périphériques sous Windows sont soit des périphériques anciens soit des périphériques "forcés", il faut que les configurations Windows et ESCD soient identiques.

5.3.4 Ajout d'un nouveau périphérique (sous Linux ou sous Windows)

Si vous ajoutez un nouveau périphérique *PnP* et que vous ayez paramétré votre BIOS pour un "SE non PnP", alors le BIOS devrait le configurer automatiquement et en enregistrer la configuration dans l'ESCD. S'il s'agit d'un périphérique *non PnP* ancien (ou d'un périphérique configurable par cavaliers, etc...), vous n'avez alors que peu d'options pour le prendre en compte.

Vous devez être capable d'indiquer directement au BIOS (par l'intermédiaire des menus de configuration de la CMOS) que certaines ressources du bus qu'il utilise sont réservées et ne doivent pas être allouées par le *PnP*. Cette information n'est pas sauvegardée dans l'ESCD. Mais, en cas de conflit, il peut y avoir une option du menu BIOS permettant de rendre prépondérant ou non, sur le contenu de l'ESCD, les choix faits au niveau de la CMOS. Une autre méthode consiste à faire tourner ICU sous DOS/Windows. Encore une autre consiste à installer manuellement le périphérique sous Windows 9x/2k et de s'assurer que cette configuration a été "forcée" (voir le paragraphe précédent). Si elle a été "forcée" Windows devrait avoir mis à jour l'ESCD lors de l'arrêt du PC.

5.4 Isapnp

Une bonne partie de la documentation concernant isapnp est difficile à comprendre si l'on ne connaît pas les bases du *PnP*. Cet HOWTO devrait aider à la comprendre ainsi que les FAQ (Questions fréquemment posées) qui l'accompagne. `isapnp` ne concerne que les périphériques PnP sur le bus ISA (non PCI). En lançant le programme Linux "isapnp" au démarrage, on configure ces périphériques avec les valeurs indiquées dans /etc/isapnp.conf. Il est possible de créer ce fichier de configuration automatiquement mais vous devrez l'éditer manuellement pour faire des choix entre différentes options. Avec isapnp, un pilote de périphérique, faisant partie du noyau, peut être lancé prématurément, avant que isapnp ait paramétré l'adresse, etc. dans le matériel. Ce qui entraîne que le pilote de périphérique ne sera pas capable de trouver le périphérique correspondant. Le pilote utilise la bonne adresse alors que celle-ci n'a pas encore été paramétrée dans le matériel.

Si votre distribution Linux installe automatiquement isapnptools, isapnp peut déjà être lancé au démarrage. Dans ce cas, tout ce que vous avez besoin de faire est d'éditer /etc/isapnp.conf selon "`man isapnp.conf`". Notez que cela revient à configurer manuellement PnP puisque vous avez à prendre les décisions sur la manière de configurer à mesure que vous éditez le fichier de configuration. Vous pouvez utiliser le programme "pnpdump" pour vous aider à créer le fichier de configuration. Si vous utilisez "isapnp" tel quel et que vous ayez un BIOS *PnP*, vous devrez probablement indiquer au BIOS (quand vous le paramétrerez) que vous n'avez pas de SE *PnP* puisque vous voudrez toujours que que le BIOS configure les périphériques PCI. Alors que le BIOS peut également configurer les périphériques ISA, isapnp le refera.

5.5 Utilitaires PCI

Le nouveau paquetage Utilitaires PCI (= pciutils, improprement appelé "pcitools"), devrait vous permettre de configurer manuellement, *à la PnP*, le bus PCI. "lspci" donne la liste des ressources bus, alors que "setpci" paramètre les allocations de ressources dans le matériel.

5.6 Corriger le noyau pour rendre Linux compatible PnP

David Howells a créé une telle mise à jour pour le faire. C'est le "Gestionnaire de configuration/ressource du noyau Linux" ("Linux Kernel Configuration/Resource Manager") quelquefois appelé "Gestionnaire de configuration matérielle". Cette adaptation peut ne pas correspondre au noyau le plus récent. Le noyau résultant est réputé stable mais cependant des bogues ont été signalés. Une documentation est incluse comme serial.txt, pour montrer comment faire pour un port série. Elle fournit des "fichiers" dans l'arborescence /proc pour que vous puissiez voir ce qui se passe et peut enregistrer des commandes dans l'un de ces fichiers pour une configuration personnalisée. Un problème vient de ce que de nombreux pilotes de périphériques ne le prennent pas en compte et qu'il vous faut encore utiliser les fichiers de configuration traditionnels etc. pour effectuer la configuration. Consultez `http://www.astarte.free-online.co.uk`

5.7 Configuration par Windows

Si vous avez Windows9x (ou 2k) sur le même PC, démarrer alors simplement Windows et laissez-le configurer le *PnP*. Puis lancez alors Linux à partir de Windows (ou du DOS). On a signalé que Windows effaçait les IRQ des registres des périphériques PCI. Dans ce cas, Linux signale qu'il a trouvé une IRQ à zéro. Dans ce cas, vous ne pouvez pas utiliser cette méthode.

5.8 Configuration par les pilotes de périphériques

Quelques pilotes de périphériques utilisent les méthodes *PnP* pour paramétrer les ressources du bus au point de vue matériel et seulement pour les périphériques qu'ils contrôlent. Dans ce cas, puisque le pilote a effectué la configuration, il connaît évidemment la configuration et il n'y a aucun besoin de lui fournir ces informations.

Le problème ici porte sur deux points. Il est difficile de mettre dans le pilote tout ce qui est nécessaire et le pilote peut accaparer des ressources du bus qui seraient nécessaires pour d'autres périphériques. Cela serait plus facile pour l'utilisateur, mais un noyau Linux *PnP* serait une meilleure solution. Voir 3.5 (Linux doit mieux prendre en compte le PnP)

5.9 Logiciels et documents *PnP*

- *Page d'acceuil Isapnptools* `http://www.roestock.demon.co.uk/isapnptools/`
- *Mise à jour pour rendre le noyau Linux PnP* `http://www.astarte.free-online.co.uk`
- *Projet de pilote PnP* `http://www.io.com/~cdb/mirrors/lpsg/pnp-linux.html`

- *Spécifications PnP de Microsoft* `http://www.microsoft.com/hwdev/respec/pnpspecs.htm`
- Le livre : PCI System Architecture, 3rd ed. by Tom Shanley +, MindShare 1995. Traite des fonctionnalités style PnP pour le bus PCI bus.
- Le livre : Architecture système *Plug and Play*, de Tom Shanley, Mind Share 1995. Détaille le *PnP* sur le bus ISA. Uniquement un survol concis du *PnP* sur le bus PCI.

6 Communication de la configuration au pilote

6.1 Introduction

La manière exacte dont cela s'effectue dépend du pilote. Certains pilotes disposent de plus d'une méthode pour détecter la configuration du périphérique physique. À l'extrême, on trouve le cas où il faut paramétrer "en dur" les ressources du bus dans le noyau et recompiler celui-ci. À l'autre extrême, le pilote fait tout automatiquement et vous n'avez rien à faire. Il peut même paramétrer les ressources du bus dans le matériel en utilisant les méthodes *PnP*.

Entre les deux, on trouve les cas où il faut faire tourner un programme pour faire passer les infos au pilote ou mettre ces infos dans un fichier. Dans certains cas, le pilote peut tester le périphériques à des adresses ou il pense trouver le périphérique. Il peut alors essayer de tester différentes IRQ pour voir laquelle fonctionne (IRQ = Interrupt ReQuest / ReQuête d'Interruption). Ce qu'il peut faire automatiquement ou non. Dans d'autres cas, le pilote utilise des méthodes *PnP* pour trouver le périphérique et la façon dont les ressources du bus ont été affectées, mais ne pas les paramétrer lui-même. Il peut aussi regarder certains fichiers du répertoire /proc.

On peut avoir besoin d'indiquer les ressources du bus comme paramètre au noyau pour un module chargeable. Voir /usr/lib/modules_help/descr.gz pour avoir une liste des paramètres possibles. Le modules à charger est indiqué dans /etc/modules avec ses paramètres. Dans certains autres cas les ressources du bus peuvent être indiquées au noyau par des paramètres. Ceux-ci sont mis dans le fichier lilo.conf sous la forme append="...". Dans ce cas le programme lilo doit être exécuté pour enregistrer ces données dans le code de démarrage du noyau.

Alors qu'il y a une grande diversité dans la manière dont les pilotes peuvent trouver les ressources du bus, le but final est le même. Il y a tellement de périphériques différents et de pilotes de périphériques pour les gérer que vous pouvez avoir besoin de consulter la documentation de votre pilote particulier pour trouver comment celui-ci gère les ressources du bus et ce que vous avez besoin de faire pour vous assurer qu'il trouve les infos dont il a besoin. Quelques brèves indications sur un petit nombre de pilotes de périphériques sont données dans les paragraphes suivants.

6.2 Pilote du port série : setserial

Pour le pilote de port série standard (et non pas pour les cartes multi-ports) vous utilisez setserial pour en configurer le pilote. Ce programme est souvent lancé à partir d'un fichier de démarrage. Dans les versions les plus récentes, on trouve un fichier /etc/serial.conf que vous pouvez "éditer" en utilisant simplement la commande setserial de la manière habituelle et ce que vous paramétrez en utilisant `setserial` est sauvegardé dans le fichier `serial.conf`. Le fichier `serial.conf` devrait être consulté lors de l'exécution de la commande `setserial` lancée par un fichier de démarrage. Votre distribution peut le faire ou ne pas le faire pour vous.

Il y a deux manières différentes d'utiliser `setserial` selon les options que vous choisissez. Une méthode consiste à indiquer la configuration manuellement au pilote. L'autre méthode consiste à tester une adresse donnée et à voir s'il y a un port série à cette adresse. On peut aussi tester cette adresses et essayer de détecter si une interruption est utilisée pour ce port. Le pilote exécute quelque chose comme `setserial` au démarrage, mais ne teste pas les interruptions, il affecte simplement l'interruption "standard" ce qui peut ne pas correspondre à la réalité. Il teste uniquement l'existence d'un port. Consulter le Serial-HOWTO pour avoir des détails complémentaires.

6.3 Pilotes de cartes son

6.3.1 OSS-Lite

Vous devez passer IO, IRQ, et DMA en tant que paramètres à un module ou les compiler dans le noyau. Mais quelques cartes PCI seront détectées automatiquement (par exemple en prenant les infos dans /proc/pci ou quelque chose d'équivalent). RedHat fournit un programme "sndconfig" qui détecte les cartes ISA PnP et paramètre automatiquement les modules pour les charger avec les ressources de bus trouvées.

6.3.2 OSS (Open Sound System) et ALSA

Ils détecteront les cartes par les méthodes *PnP* puis choisiront le pilote approprié et le chargeront. Ils paramètreront également les ressources du bus sur une carte ISA-PnP. Vous pouvez avoir à intervenir manuellement pour résoudre les problèmes de conflits. Pour le pilote ALSA, la prise en charge de l'ISA-PnP est optionnelle et vous pouvez utiliser isapnp si vous le voulez.

7 Quelle est ma configuration actuelle?

Ici "configuration" concerne l'attribution des ressources *PnP* du bus (adresses, IRQ, et DMA). C'est une question à deux volets pour chaque périphérique, et chacun d'eux doit avoir le même réponse.

1. Quelle est la configuration du logiciel du pilote de périphérique ? Autrement dit : comment le pilote pense-t-il que le matériel est configuré ?
2. Quelle est la configuration (s'il y a lieu) du matériel périphérique ?

Naturellement la configuration matérielle du périphérique et celle de son pilote doivent être les mêmes (et normalement, c'est le cas). Mais, si les choses ne se passent pas bien, c'est qu'il y a peut-être une différence. Cela signifie que le pilote a des données incorrectes concernant la configuration réelle du matériel, et dysfonctionnement. Si le logiciel que vous utilisez ne vous indique pas correctement ce qui ne va pas (ou n'effectue pas la configuration correcte automatiquement), il vous faudra chercher la configuration matérielle des périphériques et celle des pilotes correspondants. Alors que les pilotes de périphériques de Linux devraient "tout nous dire", il y a des cas où il n'est pas facile de déterminer comment le matériel a été configuré.

Un autre problème vient de ce que les messages à l'écran indiquant la configuration ne sont pas toujours clairs quant à la nature de la configuration qu'ils concernent : le pilote de périphérique, le périphérique lui-même ou les deux. Si le pilote de périphérique s'est vu attribué une configuration et qu'ils teste le matériel pour vérifier qu'il est configuré de la même manière, alors la configuration indiquée par le pilote devrait être à la fois celle du pilote et du matériel.

Mais certains pilotes ne se comportent pas comme cela et peuvent accepter une configuration qui ne puisse pas se vérifier. Par exemple, "setserial" acceptera une configuration qui ne se vérifie pas (même si vous demandez un test des ressources du bus). Dans ce cas "setserial" peut uniquement vous indiquer la configuration du pilote et pas celle du matériel.

Quelques infos concernant la configuration peuvent être obtenues en lisant les messages du BIOS et de Linux qui apparaissent à l'écran au démarrage initial de l'ordinateur. Souvent, ces messages défilent trop rapidement pour pouvoir être lus, mais lorsque leur défilement s'arrête, tapez plusieurs fois sur la touche "Majuscule + défilement arrière" (Shift-PageUp) pour les visualiser. Pour visualiser la suite, taper sur la touche "Majuscule + défilement avant" (Shift-PageDow). La commande "dmesg" n'importe quand n'affichera que les messages du noyau Linux et les messages les plus importants peuvent manquer (y compris ceux du BIOS). Les messages de Linux peuvent quelquefois n'indiquer que la vue de la configuration par les pilotes de périphérique, et ces informations peuvent parfois provenir d'un fichier de configuration incorrect.

Les messages du BIOS affichent la configuration réelle du matériel à cet instant donné, mais un SE *PnP*, isapnp ou les utilitaires PCI peuvent la modifier ultérieurement. Les messages du BIOS sont affichés avant ceux de Linux. Comme alternative au rappel des messages par les touches "majuscule + défilement arrière", essayez de geler l'affichage en tapant sur la touche "Pause". Taper sur n'importe quelle touche pour continuer. Mais, une fois que les messages de Linux commencent à apparaître, c'est trop tard pour utiliser la touche "Pause" puisqu'alors, les messages de Linux ne sont pas gelés.

7.1 Comment mes pilotes de périphériques sont-ils configurés ?

Il y a des programmes que l'on peut lancer de la ligne de commande (tel que "setserial" pour les ports série) pour le déterminer. L'arborescence du répertoire /proc est utile. /proc/ioports indique les adresses d'E/S utilisées par les pilotes (ou essaye si c'est faux). Ils peuvent ne pas être configurés de cette façon dans le matériel.

/proc/interrupts n'indique que les interruptions utilisées actuellement et beaucoup de celles qui ont été allouées aux pilotes ne sont pas affichées car elles ne sont pas actuellement utilisées. Par exemple, même s'il y a une disquette dans l'unité et qu'elle est prête à être utilisée, l'interruption la concernant ne sera pas affichée, à moins qu'elle ne soit effectivement en cours d'utilisation. De plus, une interruption indiquée ici ne signifie pas qu'elle existe dans le matériel. Un indice du fait qu'elle n'existe pas dans le matériel consiste à vérifier que le nombre d'interruptions générées est 0. Et encore, même si l'on voit que quelques interruptions ont été générées, cela peut signifier que cette interruption n'existe pas pour ce périphérique mais par un autre périphérique qui n'est pas en cours d'utilisation mais qui a pu générer une interruption ou deux. Dans le noyau 2.2 l'arborescence /proc a été modifiée.

7.2 Comment mes périphériques sont-ils configurés ?

C'est facile de trouver quelles ressources du bus ont été attribuées à des périphériques sur un bus PCI : on utilise soit la commande "lspci" ou, pour les noyaux <: 2.2: voir `/proc/pci`; pour les noyaux 2.2+: `/proc/bus/pci/devices`. Pour le bus ISA, on peut essayer de faire tourner `pnpdump -dumpregs`, mais ce n'est pas une méthode sûre. Les résultats peuvent être difficile à décoder. Ne pas confondre les adresses de port de lecture que `pnpdump` "essaye" (et où il trouve quelque chose) avec les adresses d'E/S du périphérique trouvé. Ce ne sont pas les mêmes.

Les messages en provenance du BIOS, lors du démarrage, indique la configuration matérielle au moment du démarrage. Si vous vous basez sur le BIOS pour faire votre configuration, ils faut qu'elle soit encore valable. Les messages de Linux peuvent provenir des pilotes qui ont fait des essais pour vérifier la présence du matériel (et éventuellement des essais sur les IRQ et le DMA). Naturellement, si le périphérique fonctionne correctement, alors il doit être configuré comme son pilote.

8 Appendice

8.1 Adresses

Il y a trois types d'adresses : les adresses mémoire centrale, les adresses d'E/S et les adresses de configuration. Pour le bus PCI, les adresses de configuration constituent un espace d'adressage séparé, exactement comme le sont les adresses d'E/S. Sauf dans le cas des adresses de configuration ISA, qu'une adresse sur le bus d'adresses (ou sur le bus partagé adresses-données dans le cas du PCI) soit ou non une adresse mémoire, une adresse d'E/S ou une adresse de configuration dépend uniquement des tensions sur d'autres fils (pistes) du bus.

8.1.1 Adresses de configuration sur le bus ISA (Port de lecture etc.)

En ce qui concerne le bus ISA, il n'y a pas, techniquement, d'espace d'adressage de configuration, mais il existe, pour le CPU, une méthode spéciale d'accès aux registres de configuration sur les cartes PnP. Trois adresses d'E/S sont réservées à cet usage. Ces 3 adresses ne sont pas différentes pour chaque carte, mais sont partagées par toutes les cartes.

Les trois adresses sont appelées port de lecture, port d'écriture et port d'adresse. Ces ports ont la taille d'un octet. Chaque carte PnP possède un certain nombre de registres car ces trois adresses seules ne sont pas suffisantes pour une simple carte. Pour communiquer avec une carte, le numéro de la carte (handle) est envoyé à toutes les cartes sur le port d'écriture. Une seule carte reconnaît son numéro et se met à l'écoute. Alors l'adresse du registre concerné est envoyée sur le port adresse (de toutes les cartes, mais une seule est à l'écoute). Ensuite, on fait une écriture sur le port d'écriture ou une lecture sur le port de lecture.

Le port d'écriture est toujours à l'adresse A79 et le port d'adresse est toujours 279. Le port de lecture n'est pas déterminé et est initialisé par le logiciel de configuration à une adresse qui est supposée ne pas être en conflit avec une autre carte ISA. S'il y a conflit, il modifiera l'adresse. Pour toutes les cartes PnP, cette adresses est "programmée". Donc, si vous utilisez disons, isapnp, pour initialiser ou tester des données de configuration, vous aurez besoin de connaître l'adresse du port de lecture.

8.1.2 Champs d'adresses

Le terme "adresse" est quelquefois utilisé dans ce document pour parler d'une série d'adresses contiguës. Comme les adresses sont données en octets, une simple adresses contient uniquement un octet mais les adresses E/S (et mémoire centrale) ont besoin de plus que cela. Donc un espace de, disons, 8 octets est souvent utilisé pour les adresses d'E/S alors que le nombre d'adresses en mémoire centrale allouées à un périphérique est beaucoup plus grand. Pour un port série (périphérique d'E/S), il suffit de donner l'adresse de départ du périphérique (par exemple 3F8) puisqu'il est bien connu que le champ d'adresses de ce type de périphérique est seulement de 8 octets. Cette adresse de départ est connue sous le nom "d'adresse de base".

8.1.3 Espace d'adresses

En ce qui concerne l'ISA, pour accéder à la fois aux "espaces" d'adresses E/S et mémoire (centrale) on utilise le même bus d'adresses (les fils utilisés pour les adresses sont partagés). Comment le périphérique reconnaît-il si une adresse qui apparaît sur le bus d'adresses est une adresse mémoire ou une adresse E/S ? Eh bien, il y a quatre lignes spécialisées du bus qui véhiculent cette information et un peu plus. Si l'un de ces quatre fils est alimenté, cela indique que le CPU veut lire une adresse E/S, et la mémoire centrale ne tient pas compte de l'adresse présente sur les fils d'adresses du bus. Les trois autres fils sont utilisés à des fins identiques. En résumé : il existe des lignes de lecture et d'écriture pour les adresses mémoire centrale et pour celles d'E/S (4 fils en tout).

Pour le bus PCI, l'idée de base est identique et on utilise également 4 fils, mais c'est réalisé de façon légèrement différente. Au lieu d'avoir uniquement l'un des quatre fils alimenté, on code un nombre binaire sur ces fils (16 possibilités différentes). On peut donc véhiculer plus d'informations. Quatre de ces 16 valeurs sont utilisées pour les espaces mémoire et E/S comme dans le paragraphe ci-dessus. En plus, on trouve l'espace des adresses de configuration qui utilise deux valeurs supplémentaires. Les dix possibilités restantes sont réservées pour d'autres utilisations.

8.1.4 Vérification des champs d'adresses (Test des conflits d'adresses E/S ISA)

Pour le bus ISA, il existe un système implanté sur chaque carte PnP pour vérifier que d'autres cartes n'utilisent pas la même adresse. Si deux cartes ou plus utilisent les mêmes adresses d'E/S, aucune des cartes ne fonctionnera correctement. Un bon logiciel *PnP* devrait affecter les ressources du bus pour éviter de tels conflits, mais, même dans ce cas, une carte ancienne pourrait se cacher quelque part avec la même adresse.

Le test consiste, pour la carte, à donner, sur le bus, comme numéros de test ses propres registres d'E/S. Le logiciel *PnP* les lit et vérifie qu'il lit les mêmes numéros. Si tel n'est pas le cas, quelque chose ne va pas (comme, par exemple, une autre carte à la même adresse). Il répète alors le test avec un autre numéro de test. Puisqu'en réalité il teste le champ des adresses d'E/S assigné à cette carte, on l'appelle " test des champs d'adresses". On devrait plutôt l'appeler "test des conflits d'adresses". S'il y a un conflit d'adresses, on obtient un message d'erreur qu'il faut résoudre par ses propres moyens.

8.1.5 Communication directe via la mémoire.

Traditionnellement, la plupart des périphériques d'E/S utilisaient les E/S mémoire pour communiquer avec le CPU. Par exemple, le port série fonctionne de cette façon. Le pilote de périphérique, qui tourne sur le CPU lit et écrit les données de/vers l'espace des adresses d'E/S et la mémoire centrale. Une méthode plus rapide consisterait à faire mettre les données directement en mémoire centrale par le périphérique. Une manière de le réaliser est d'utiliser l'accès direct mémoire (2.5 (Canaux DMA)). Une autre méthode consiste à affecter un espace de la mémoire centrale au périphérique. Comme cela, le périphérique lit et écrit directement en mémoire centrale sans avoir à se préoccuper de l'accès direct mémoire (DMA). De tels périphériques possèdent normalement des adresses d'E/S et des adresses mémoire.

8.2 Les interruptions - Description détaillée.

Le système d'interruption véhicule un paquet d'informations, mais seulement de façon indirecte. Le signal d'interruption (une tension sur un fil) indique simplement à un circuit appelé contrôleur d'interruptions qu'un certain périphérique a besoin que l'on s'occupe de lui. Le contrôleur d'interruption le signale alors au CPU. Le CPU va chercher le pilote de ce périphérique et en exécute une partie que l'on désigne sous le nom de "routine de service de l'interruption" (ou "routine de gestion de l'interruption"). Cette "routine" essaie de voir ce qui s'est passé et se charge de traiter la question comme, par exemple, de transférer des octets du (ou vers le) périphérique. Ce programme (cette routine) peut facilement analyser ce qui s'est passé puisqu'il connaît les adresses des registres (à condition que les numéros d'interruption (IRQ) et que les adresses d'E/S aient été correctement renseignées). Ces registres contiennent les informations concernant l'état du périphérique. Le logiciel lit le contenu de ces registres et, en examinant ce contenu, voit ce qui s'est passé et entreprend l'action appropriée.

Donc, chaque pilote de périphérique a besoin de savoir quel est le numéro d'interruption (IRQ) qui le concerne. Sur le bus PCI (et pour les ports série sur un bus ISA depuis la version du noyau 2.2) il est possible que deux périphériques ou plus se partagent le même IRQ (numéro d'interruption). Cela est rendu possible en faisant exécuter par le CPU toutes les routines de service d'interruption de tous les périphériques qui utilisent cette interruption. La première chose que réalise une telle routine de service consiste à tester si l'interruption concerne son périphérique. Si tel n'est pas le cas (fausse alarme), elle se termine et le traitement continue avec la routine de service suivante etc.

8.3 Interruptions PCI

Les interruptions PCI sont différentes, mais comme elles correspondent normalement aux IRQ, elles se comportent à peu près de la même façon. Le fait que les interruptions PCI puissent être partagées constitue une différence majeure. Par exemple l'IRQ5 peut être partagée entre deux périphériques PCI. Cette possibilité de partage est automatique : il n'est pas nécessaire de disposer d'un matériel ou d'un logiciel spécial. On a entendu parler de situations où un tel partage ne fonctionne pas, mais la cause réside certainement dans le logiciel du pilote de périphérique. Tous les pilotes de périphérique PCI sont supposés fonctionner avec partage des interruptions. Il faut cependant noter que le partage d'une même interruption entre un bus ISA et PCI n'est pas possible. Mais un partage illégal d'interruption fonctionnera à condition que les périphériques en conflit ne soient pas utilisés en même temps. "En même temps" signifie ici qu'un programme en cours d'exécution "ouvre" le périphérique dans le programme C.

Vous pouvez avoir besoin de connaître quelques-uns des détails du système d'interruption PCI de façon à pouvoir paramétrer la mémoire CMOS du BIOS ou pour positionner les cavaliers sur de vieilles cartes PCI. Le bus PCI possède quatre lignes d'interruptions de INTA# à INTD# (A, B, C et D). Pour un système à 7 connecteurs, d'après les spécifications, on aurait la possibilité de disposer de 7 x 4 = 28 lignes d'interruptions différentes. Mais ces spécifications ne permettent qu'un nombre inférieur de lignes d'interruption. Ce n'est pas trop contraignant puisque l'on peut partager les interruptions. Beaucoup de bus PCI ne paraissent disposer que de 4 lignes d'interruption. Appelons ces lignes (fils ou pistes) W, X, Y et Z. Supposons que l'on désigne l'interruption B de l'emplacement numéro 3 comme étant l'interruption 3B. Alors, on peut utiliser le fil W pour partager les interruptions 1A, 2B, 3C, 4D, 5A, 6B, 7C. On le réalise physiquement en connectant le fil W aux fils 1A, 2B etc... De même, le fil X pourrait être connecté aux fils 1B, 2C, 3D, 4A, 5B, 6C, 7D. Au démarrage, le BIOS fait correspondre X, W, Y, Z aux interruptions. Ensuite il écrit dans un registre physique de chaque périphérique quelle interruption lui a été affectée de sorte que le périphérique (et tout ce qui interroge le périphérique) sache quelle IRQ il utilise.

Les fils X, W, Y et Z mentionnés ci-dessus sont désignés dans les spécifications PCI par les noms INTA#, INTB#, INTC# et INTD#. Cette dénomination officielle PCI porte à confusion puisque, maintenant, INTA# a deux significations selon que l'on parle du connecteur ou du bus PCI. Par exemple, si 3C correspond à X alors on dira que l'INTC# du connecteur 3 est reliée à l'INTA# (X) du bus PCI. C'est une notation confuse.

Il y a également une autre obligation. Un connecteur PCI doit utiliser les premières lettres d'interruption en premier. Donc, s'il n'y a qu'un connecteur à utiliser une interruption, ce doit être L'INTA#. S'il utilise 2 interruptions ce doivent être INTA# et INTB# etc. Jusqu'à 8 périphériques peuvent être connectés à une carte à un emplacement donné, mais ils ne peuvent disposer que de 4 interruptions PCI. Cela ne pose pas de problème puisque les interruptions peuvent être partagées et donc, chacun des 8 périphériques (s'ils sont présents) pourra disposer d'une interruption. La lettre d'interruption PCI d'un périphérique est souvent fixe et câblée dans le périphérique.

Le BIOS affecte les IRQ(demandes d'interruptions) de façon à éviter les conflits avec les IRQ qu'il connaît sur le bus ISA. Quelquefois, dans le menu CMOS, on peut affecter des IRQ aux cartes PCI (mais ce n'est pas aussi facile à faire que ce qui a été expliqué ci-dessus). Il existe une situation dans laquelle Windows met à zéro tous les numéros d'interruption dans les cartes PCI après que l'affectation des numéros d'interruption a été effectué. Alors, quelqu'un qui utilise Windows et qui lance Linux à partir de Windows verra Linux ne trouver que des IRQ incorrectement paramétrées à zéro.

Vous pourriez penser que l'utilisation par le PCI des IRQ (bus ISA) peut être lent etc. Pas vraiment. Le(s) circuit(s) contrôleur(s) d'interruptions ISA possède(nt) un fil d'interruption directement relié au CPU afin que celui-ci puisse réagir immédiatement. Alors que les signaux sur les bus d'adresse et de données ISA doivent cheminer à travers le bus PCI pour atteindre le CPU, les signaux d'interruption IRQ lui parviennent pratiquement directement.

8.4 Isolation

C'est uniquement valable pour le bus ISA. L'isolation est une méthode complexe d'assignation d'un identificateur temporaire (id number = numéro d'identification ou Card Select Number (CSN) = numéro de sélection de carte) à chaque périphérique *PnP* sur le bus. Puisqu'il existe des moyens plus efficaces (mais plus complexes) pour le faire, certains pourront affirmer que c'est une méthode simpliste. On n'utilise qu'une seule adresse d'écriture pour toutes les écritures sur tous les périphériques *PnP* connectés. Cette adresse est utilisée pour envoyer (assigner) un identificateur unique à chaque périphérique *PnP*. L'attribution de cet identificateur impose qu'un seul périphérique soit à l'écoute lorsque cet identificateur est envoyé (écrit) à cette adresse commune. Tous les périphériques *PnP* ont un numéro de série unique qu'ils utilisent dans le processus d'isolation. La réalisation de l'isolation ressemble à un jeu. Elle est réalisée en utilisant l'équivalent d'un bus à un seul fil reliant tous les périphériques *PnP* et du programme d'isolation.

Lors de la première manche de ce "jeu", tous les périphériques *PnP* sont à l'écoute sur ce fil et envoient simultanément une séquence de bits sur le fil. Les valeurs permises sont soit des 1 (tension positive) soit des "0 ouverts" sans tension (circuit ouvert ou troisième état). Chaque périphérique *PnP* commence à envoyer séquentiellement son numéro de série, bit par bit, en commençant par le bit de poids fort, sur le fil. Si l'un des périphériques envoie un 1 sur le fil, un 1 sera reçu par tous les autres. Si tous les périphériques envoient un "0 ouvert", on n'entendra rien sur le fil. L'objectif est d'éliminer (à la fin de cette première manche) tout le monde sauf le périphérique ayant le numéro de série le plus élevé. Par "éliminer", on entend cesser d'écouter plus avant l'adresse d'écriture que tous les périphériques encore dans la course continuent d'écouter. On appelle également cela "se retirer". (Il faut noter que tous les numéros de série sont de même longueur).

En premier lieu, ne prenons en considération que le bit de poids le plus élevé du numéro de série mis sur le fil par tous les périphériques n'ayant pas encore d'identificateur. Si l'un des périphériques *PnP* envoie un 0 (0 ouvert) mais reçoit un 1, cela signifie qu'un autre périphérique *PnP* possède un numéro de série supérieur au sien. Donc, il se retire provisoirement du jeu et n'écoute plus ce qui se passe sur la ligne jusqu'à la fin de cette manche (quand un identificateur est attribué au gagnant : celui qui a le numéro de série le plus élevé). Alors, les périphériques encore de la partie possèdent tous le bit de poids fort (un 1), donc, nous pouvons supprimer ce bit et prendre en compte uniquement le "numéro de série tronqué" résultant pour continuer à participer à cette manche. Retournez au début de ce paragraphe et répétez le processus jusqu'à ce que le numéro de série complet ait été traité pour chacun des périphériques (voir ci-dessous comment est traité le cas où il n'y a que des 0).

Il est donc clair que le numéro de série le plus élevé ne sera pas éliminé de la partie. Mais qu'en est-il si les chiffres de tête (du numéro de série éventuellement tronqué) sont tous des 0 ?. Dans ce cas un "0 ouvert" est envoyé sur la ligne et tous les participants restent en lice. S'ils ont tous des zéros en tête, alors les 0 sont éliminés exactement comme les 1 au paragraphe ci-dessus. La partie continue et les chiffres suivants (du numéro de série) sont envoyés.

À la fin de la manche (lorsque le bit de poids faible du numéro de série du concurrent restant a été émis), seul le périphérique *PnP* ayant le numéro de série le plus élevé est présent. On lui donne alors un identificateur et il quitte la partie définitivement. Ensuite, tous les éliminés de la dernière manche (ceux qui n'ont pas encore obtenu d'identificateur) reviennent dans le jeu et une nouvelle manche commence avec un concurrent de moins. Finalement, tous les périphériques *PnP* se verront attribuer un identificateur. Il est facile de prouver que cet algorithme fonctionne.

Une fois que tous les identificateurs ont été attribués,ils sont utilisés pour s'adresser à chacun des périphériques *PnP* pour les configurer et lire leur configuration. On notera que ces identificateurs ne sont utilisés que pour la configuration *PnP* et ne sont pas utilisés pour les communications normales avec le périphérique *PnP*. Au démarrage de l'ordinateur, tous les identificateurs sont perdus et, donc, un BIOS PnP refait normalement ce processus d'isolation à chaque fois que vous remettez votre PC en service.

FIN DU Plug-and-Play-HOWTO