USB sem problema com Hotplug, Udev, HAL e D-Bus

Ligação direta

Dispositivos que funcionem imediatamente, sem intervenção, ao serem conectados são o sonho de qualquer usuário. As distribuições atuais, embora estejam caminhando nessa direção, ainda deixam a desejar no quesito. Neste artigo, investigaremos como os componentes individuais do sistema *hotplug* interagem entre si.

por Oliver Frommel, Marcel Hilzinger e René Rebe

Será que é tão difícil assim? Você só queria que o Linux chamasse o programa certo quando pluga uma câmera digital na porta USB, mas o sistema operacional simplesmente fica lá olhando para você com cara de bobo. É um absurdo que um sistema moderno e voltado para o usuário final ainda espere que alguém "monte" sua câmera ou seu chaveiro USB, mas infelizmente essa é uma situação pra lá de comum – embora tenha melhorado nos últimos anos. As distribuições têm a *obrigação* de reconhecer corretamente e configurar sozinhas *qualquer tipo de hardware* sem a intervenção do usuário. Isso é ainda mais urgente para os atuais dispositivos USB e Firewire, que devem poder ser conectados e desconectados – e, por conseguinte, "montados" e "desmontados" automaticamente – sem que seja preciso desligar o computador para tanto. Essa técnica é conhecida como *hotplugging* (em português, "conexão a quente") [1] e é vital para a sobrevivência de qualquer sistema operacional digno desse nome. Este artigo mostra o que um sistema Linux moderno faz com os dispositivos conectados a ele e tenta explicar porque nem tudo funciona como planejado.

Agentes secretos

Quando conectamos um dispositivo que deveria ser reconhecido automaticamente, o kernel envia um sinal para o susbsistema *hotplug*, que usa o *Udev* para criar um arquivo de dispositivo no diretório `/dev` e chama o agente apropriado. Um *agente* é um script, normalmente em `/etc/hotplug/`, que administra as tarefas associadas com o evento do *hotplug*. Nesse caso, o agente adiciona e registra o novo dispositivo no sistema.

As etapas seguidas pelo agente podem variar dependendo da distribuição e do dispositivo que se quer usar. Como a maioria dos dispositivos que usaremos com hotplug são USB, talvez começar por aí seja uma boa maneira de ilustrar o mecanismo. O agente USB primeiro verifica se existe um *driver* para o novo dispositivo (por exemplo *isdn*) e chama o comando `modprobe` para carregar o módulo. Se o agente encontrar um script com o mesmo nome do módulo no diretório `/etc/hotplug/usb/`, ele executa esse script.

O simples ato de carregar módulos na memória normalmente dispara outros eventos de *hotplug* que, por sua vez, ativam outros agentes. É comum que vários agentes *hotplug* trabalhem em conjunto. Por exemplo, quando se conecta um disco rígido externo à porta USB, primeiro o agente USB é carregado, em seguida o agente SCSI é chamado para montar as partições do disco como se fossem dispositivos SCSI com a ajuda do módulo *usb-storage*. Se, em vez do disco rígido, você plugar um adaptador Bluetooth, o agente USB é seguido pelo agente Bluetooth `bluetooth.agent`.

O arquivo `/etc/hotplug/blacklist` traz uma lista com os módulos que nenhum agente tem permissão de carregar. A lista inclui os módulos carregados por outros serviços, bem como alguns módulos que impedem que o subsistema de gerenciamento de energia (APM ou ACPI) funcione corretamente.

Só o necessário

O Linux normalmente precisa de um arquivo no diretório `/dev` para poder interagir com cada dispositivo conectado. Se estamos usando *hotplug*, esse arquivo tem de ser criado instantaneamente quando o dispositivo é detectado. Como vimos na seção anterior, o hotplug usa o *Udev* para criar o arquivo de dispositivo antes de convocar o agente.

O *Udev* (também disponível em [1]) é o sucessor dos arquivos permanentes de dispositivo que povoavam o diretório `/dev` até então. A maioria das distribuições Linux modernas usa o novo sistema. O propósito do *Udev* é criar arquivos de dispositivo com base em regras especificadas nos seus arquivos de configuração, que podem ser modificadas pelo usuário. Por exemplo, se você quer associar outro nome de arquivo que não `sda1` ao reprodutor de MP3, basta editar o arquivo de regras e escrever a sua própria.

As regras do *Udev* são armazenadas no diretório `/etc/udev/rules.d`. Dentro dele, há um arquivo que descreve os dispositivos mais comuns: no Ubuntu ele se chama `udev.rules`, enquanto no Fedora o nome é `50-udev.rules`. O *Udev* lê os arquivos em ordem alfabética. Se você quer que sua regra seja carregada antes das regras globais, basta nomear o arquivo de forma a respeitar a ordem – por exemplo, `10-local.rules`. A regra a seguir é suficiente para um reprodutor de MP3 de uma marca "genérica":

```
BUS="usb", SYSFS{idProduct}=➥
"1000", SYSFS{idVendor}="10d6",➥
 NAME="toca-mp3"
```

Quadro 1: Arquivos de dispositivo com o UDev

No Linux, os programas usam arquivos de dispositivo para acessar o hardware. Esses arquivos especiais, presentes no diretório `/dev`, são definidos tendo como referência seu tipo, bem como os chamados números *maior* e *menor* (*major* e *minor*). Esses números ligam o arquivo ao driver no kernel.

No passado, esse diretório era um repositório para todos os tipos de dispositivos: discos rígidos IDE e SCSI, USB, IEEE 1394 (Firewire) e dispositivos virtuais. Como todos os arquivos já estavam criados estática e permanentemente – mesmo que o dispositivo *não* estivesse realmente instalado no computador – o `/dev` acabou atulhado com centenas de arquivos.

Essa tática possui muitas desvantagens. Em primeiro lugar, é impossível saber quais dispositivos realmente estão presentes ou foram corretamente identificados por drivers. Além disso, os nomes dos arquivos podem mudar dependendo da ordem em que foram conectados. Um exemplo clássico é o do Zip Drive conectado à porta paralela, que usa emulação SCSI: às vezes ele se registra como `/dev/sda1`, outras vezes como `/dev/sda4`.

Novos mundos

O *Udev* [1] é o sucessor oficial dos arquivos estáticos no `/dev`, sendo adotado pela maioria das distribuições atuais. O *Udev* baseia-se no mecanismo de *hotplug* e gera os arquivos somente quando necessário. Quando um dispositivo muda, o kernel chama o programa especificado em `/proc/sys/kernel/hotplug` – normalmente `/sbin/hotplug`. Dependendo do tipo de dispositivo, o programa carrega módulos, altera privilégios de acesso, configura dispositivos de rede ou, no caso do Udev, administra os nós dos dispositivos.

O Udev precisa saber alguns detalhes sobre o dispositivo para poder criar o arquivo: o tipo de dispositivo (*caractere* ou *bloco*) e os números *maior* e *menor*. No kernel 2.6 (e provavelmente nos posteriores) o *Udev* consulta o sistema de arquivos *sys* para descobrir esses detalhes. O sistema de arquivos *sys* usa o subsistema *SysFS* e está montado no diretório `/sys`.

Os dispositivos de bloco (normalmente mídias como CDs, discos rígidos e disquetes) estão presentes em `/sys/block`, enquanto os dispositivos de caractere (mouse, saídas seriais) estoo em `/sys/class`. Os números *maior* e *menor* são armazenados num arquivo chamado `dev`. Por exemplo, o comando a seguir informa os números do primeiro disco rígido IDE, mais conhecido no mundo Linux como `hda`:

```
cat /sys/block/hda/dev
3:0
```

O *Udev* interpreta qualquer informação vinda do *SysFS*, como a classe do dispositivo, seu nome, números maior (*major*) e menor (*minor*) etc. A partir dessas informações, cria os dispositivos apropriados. Se os nomes forem estáveis, o *Udev* pode até mesmo executar programas complexos para decidir se a impressora recém-conectada vai para `/dev/usb/lp0` ou `/dev/usb/lp1` baseado no número de série da impressora, por exemplo. É possível até mesmo usar nomes arbitrários para os nós, como `/dev/lp-epson` e `/dev/lp-kyocera`.

Configurando o Udev

O *Udev* possui duas opções de configuração. Os arquivos em `/etc/udev/rules.d/` especificam os nomes dos dispositivos; e em `/etc/udev/permissions.d/` há outros que definem privilégios. As regras padrão criam dispositivos que usam nomes familiares para os usuários veteranos (como `hda` e `lp0`, por exemplo).

No início de cada regra, há uma ou mais normas que devem ser seguidas para que o *Udev* consiga criar os arquivos de dispositivo. O nome vem a seguir. Veja, por exemplo, uma regra para impressoras USB:

```
BUS="usb", KERNEL="lp[0-9]*", NAME="usb/%k"
```

Se a impressora for conectada à porta USB (ou seja, ao subsistema USB do kernel) e o nome interno no dispositivo for *lp* com um número entre zero e nove, o Udev cria um arquivo com o nome interno do kernel (conforme indicado pelo parâmetro `%k`) no diretório `/dev/usb`.

Além de regras estáticas como essa, é possível também chamar programas externos. A página de manual traz um exemplo para drives de CD-ROM IDE que verificam se há um diretório `/proc` para identificar o dispositivo como um leitor de CD-ROM:

```
KERNEL="hd[a-z]", PROGRAM="/bin/cat /proc/ide/%k/media", ➥
RESULT="cdrom", NAME="%k", SYMLINK="cdrom%e"
```

No exemplo, o *Udev* chama `/bin/cat` e o faz ler os arquivos no diretório */proc* que comecem por *hd*. Se o arquivo especificar `cdrom` como mídia, o Udev se "lembrará" do nome no kernel e, automaticamente, criará um link simbólico chamado `cdrom` apontando para ele. O `%e` faz com que o Udev escolha o próximo número disponível se aquele já estiver ocupado.

Usar os números de série para associar um nome menos "computês" ao dispositivo é sempre uma boa idéia:

```
BUS="usb", SYSFS{serial}="HXOLL0012202323480", NAME="lp-epson"
```

Essa regra cria um arquivo de dispositivo chamado `/dev/lp-epson` caso encontre um dispositivo com o número mencionado acima no arquivo `serial` da árvore do *SysFS*.

O *Udev* e os privilégios de acesso

As regras que definem privilégios de acesso são simples, consistindo de uma única linha com valores separados por vírgulas para nome, proprietário, grupo e privilégios.

```
usb/lp*:root:lp:0660
```

Todos os dispositivos chamados `usb/lp*` pertencem ao usuário `root` e ao grupo `lp`. Os privilégios de acesso são especificados no formato octal de sempre, da mesma forma que no comando `chmod`.

O novo modelo do sistema de hotplug é tão eficiente que é usado até para a inicialização do sistema. Durante o *boot*, o *Udev* é chamado com os valores apropriados para todos os dispositivos conhecidos nos diretórios `/sys/class` e `/sys/block`.

Com isso, o Gnome vai parar de mostrar a o rótulo genérico `sda1`; em vez disso, aparecerá o nome `toca-mp3`, muito mais intuitivo (**Figura 1**). A ferramenta `lsusb`, que apresenta uma lista com todos os dispositivos USB conectados, o ajudará a encontrar os *USB IDs* necessários. Se o nome de arquivo já existir, o comando `udevinfo -q path -n /dev/devicefile` mostra o caminho no SysFS, o sistema de arquivos de dispositivos do sistema, mas não dá o ponto de montagem em `/sys` (veja o **quadro 1: Arquivos de dispositivo com o *Udev***). É preciso ter o caminho como parâmetro (`-p`) no comando anterior para mostrar as informações sobre o SysFS:

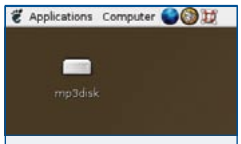

Figura 1: Depois de modificar o arquivo de configurações do Udev, o ícone mostrado no Gnome possui um nome mais intuitivo.

```
udevinfo -a -p /block/hda/hda1
...
    SYSFS{idVendor}="10d6"
...
```

Esse comando ajuda a descobrir os valores específicos para configurações especiais. Há ainda um tutorial detalhado a respeito da criação de regras para o *Udev* em [2]. O site do Fedora traz uma rápida introdução sobre o sub-sistema *Udev* [3].

Seus próprios mapas

O usuário pode criar um arquivo contendo mapeamentos com múltiplos IDs que identificam um componente de hardware. Quando um dispositivo que combina com um desses mapeamentos é conectado, o subsistema automaticamente roda o comando especificado, que pode ser um programa ou um script. Isso permite que se configure um adaptador WLAN do tipo "chaveiro" que sua distribuição não reconhece nativamente.

Em nosso laboratório, o sistema *hotplug* detectou um adaptador WLAN com o chipset *Prism2*, mas não conseguiu rodar o script que ativa as funções de rede sem fio. O comando `lsusb` nos mostra o USB ID do dispositivo: o ID do fabricante é, em nosso exemplo, `0846`, enquando o ID do dispositivo é `4110`.

Adicione os valores a seguir em um novo arquivo chamado `/etc/hotplug/usb/prism2.usermap`:

```
prism2 0x0003 0x0846 0x4110
 0x0000 0x0000 0x00 0x00 0x00
 0x00 0x00 0x00 0x0
```

Muitos mapas se parecem com esse e usam apenas os primeiros quatro valores. O primeiro especifica o programa que o *hotplug* deve executar se os valores seguintes estiverem presentes no dispositivo detectado. O primeiro valor numérico é um campo binário que indica quantos campos vêm a seguir. Para que o *hotplug* verifique os primeiros dois campos seguintes, o valor correto é `0x0003`. O primeiro bit representa 1, o segundo é 2, ambos combinados somam 3. O hotplug ignora as colunas restantes nesse arquivo, o que explica o porquê de os outros campos estarem em `0x00`.

O script que queremos executar, `prism2`, precisa estar no mesmo diretório e ser executável; para isso, basta digitar `chmod +x prism2`. Em nosso exemplo, ele roda o script de inicialização `rc.wlan`, disponível no pacote do *Prism2*. Depois, configura a interface de rede `wlan0` e solicita um endereço IP ao servidor DHCP:

```
#!/bin/sh
/etc/rc.wlan start
/sbin/ifconfig wlan0 up
/sbin/dhclient wlan0
```

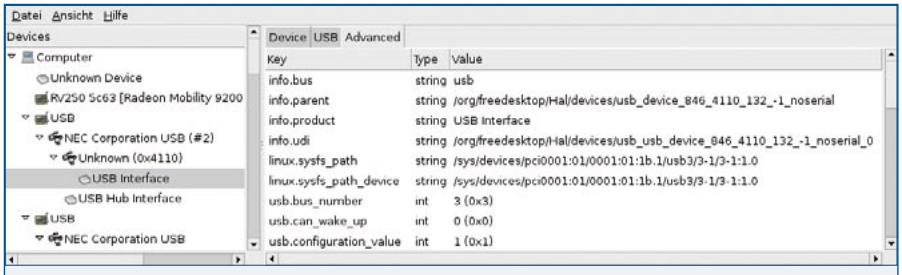

Figura 2: O *hal-device-manager* informando os detalhes do nosso já famoso adaptador WLAN com chipset *Prism2*.

Depois dessas mudanças, o adaptador WLAN do seu chaveiro USB funciona instantaneamente ao ser conectado. Infelizmente, nossas tentativas de configurar com o mesmo procedimento uma câmera de vídeo digital falharam vergonhosamente devido à lamentável implementação do subsistema Firewire no kernel do Linux. O driver IEEE1394 para o kernel atual não nos dá a informação de que precisamos via SysFS, portanto não tivemos outra alternativa senão recorrer ao utilitário *mknod* e criar permanentemente um arquivo no diretório `/dev` da maneira tradicional.

Do hardware para o aplicativo

Outra camada do subsistema *hotplug* cria uma interface entre o hardware e os programas que o acessam. Tal prodígio é obtido com o *Hardware Abstraction Layer* (HAL ou Camada de Abstração do Hardware [4]), que possui informações detalhadas sobre o hardware devidamente armazenadas em arquivos de descrição de dispositivos (*device information files*, extensão `.fdi`).

Pode-se usar o HAL para fazer mudanças em dispositivos especiais, como no caso deste usuário [5], que resolveu o problema de um iPod que se recusava a se registrar corretamente no sistema.

Os arquivos FDI usam o formato XML, que permite uma descrição detalhada dos dispositivos. O comando *lshal* mostra esses detalhes. O programa *hal-device-manager* mostra os mesmos dados em

Figura 3: No Gnome 2.8, o *gnome-volume-properties* chama o programa correto para lidar com o dispositivo detectado pelo *hotplug*.

Quadro 2: O *hotplug* em sistemas SuSE

O SuSE Linux adotou uma abordagem bastante diferente para o problema do *hotplug*, que não usa a arquitetura HAL. Em vez disso, um algoritmo próprio distingue entre dispositivos desconhecidos (ou seja, ainda não detectados) e dispositivos já configurados, usando os programas *hwup*, *hwdown*, *hwstatus* e *hwscanqueue* para lidar com os problemas da conexão "a quente". No SuSE, um arquivo de configuração em `/etc/sysconfig/hardware` guarda as informações dos dispositivos já configurados. Quando o kernel reconhece um evento hotplug, o `/sbin/hotplug` carrega o módulo apropriado.

Sem HAL

Depois disso, o `hwup` verifica se há um arquivo de configuração para o dispositivo em `/etc/sysconfig/hardware` e, caso exista, carrega os módulos especificados ali. Se o `hwup` não conseguir encontrar um arquivo de configuração, tenta localizar os módulos descritos no arquivo `*.usermap` apropriado, presente em `/etc/hotplug/`, da mesma forma que as outras distribuições. Para as próximas versões, a SuSE provavelmente irá criar um arquivo separado no diretório `/etc/sysconfig/hardware` para cada componente de hardware, sem usar os mapeamentos do usuário. Comparando com as versões mais antigas, fica claro que a SuSE já caminha nessa direção. Enquanto a 9.1 cria apenas arquivos de configuração para dispositivos de rede, a 9.2 já reconhece discos rígidos, drives de CD e DVD e um número razoável de dispositivos USB.

Depois que o `hwup` termina sua tarefa, o agente do *hotplug* é chamado ao serviço. No caso de um evento USB, obviamente o agente USB é convocado; para eventos de rede será o agente de redes e assim por diante. Se o sistema hotplug não conseguir encontrar um agente apropriado, um "genérico" é executado para criar os arquivos necessários no diretório `/dev`. No SuSE Linux 9.2, o agente também procura no diretório `/etc/sysconfig` por um arquivo de configuração para o dispositivo e, caso tenha sucesso, ativa o serviço associado.

O SuSE Linux possui um método simples para "caçar" bugs que torna o *hotplug* levemente mais tagarela. Para ativar esse modo de depuração (*debug*), atribua o valor `yes` ou mesmo `max` à variável `HOTPLUG_DEBUG`, presente no arquivo `/etc/syconfig/hotplug`. O valor `max` ordena ao sistema que registre no arquivo `/var/log/messages` cada etapa da configuração de dispositivos pelo hotplug.

Ícones da área de trabalho no SuSE Linux

O Suse Linux 9.1 e posteriores não cria ícones na área de trabalho para os dispositivos de hardware conectados ao PC: os usuários devem procurar por eles na pasta Meu Computador (*My Computer*) da mesma maneira que no Windows. Uma outra maneira é acessá-los diretamente pela URL `drives:/`. Essa solução não é lá muito boa para chaveiros de memória USB. Por padrão, o SuSE abre o Konqueror com o conteúdo da partição do chaveiro assim que conectado. Entretanto, se esse recurso for desabilitado no YaST, ou se você simplesmente fechar a janela, o único jeito de acessar o conteúdo do chaveiro novamente é reconectando – ou usando a pasta Meu Computador.

O SuSE Linux usa seus próprios ícones para a URL *drives:/*. Eles estão localizados no diretório `/usr/share/hotplug/DesktopTemplates/`. O KDE inclui o nome no arquivo `~/.kde/share/config/kio_drivesrc`. Você pode editar esse arquivo para atribuir nomes únicos e intuitivos para seus dispositivos. Por exemplo, se você possui dois chaveiros de memória, é possível editar as linhas abaixo de `[Used Names]` para criar uma distinção entre eles.

Para voltar a ter os ícones de dispositivo na área de trabalho do KDE no SuSE, instale os pacotes `kdebase3-extra` e `kdemultimedia3-extra`. Para os usuários do SuSE 9.2 um aviso: os pacotes estão apenas no DVD da distribuição. Depois de completar a instalação, clique com o botão direito na área de trabalho do KDE e escolha `Configurar área de trabalho...` (`Configure desktop...`). Na janela `Comportamento` (`Behavior`), selecione `Ícones de dispositivo` (`Device icons`) e marque a opção `Exibir ícones de dispositivos` (`Enable icons on desktop`). Com a lista habilitada, especifique quais dispositivos devem aparecer na área de trabalho. Depois de instalar os pacotes `kdebase3-extra` e o `kdemultimedia3-extra`, a URL *devices:/* passará a funcionar também no Konqueror.

uma interface gráfica bonitinha (**figura 2**). Os usuários do SuSE não podem desfrutar das benesses do HAL, já que a empresa implementou uma maneira totalmente diversa de lidar com os detalhes de hardware (ver **quadro 2: o *hotplug* em sistemas SuSE**).

No futuro, os aplicativos serão capazes de solicitar informações sobre o hardware usando a arquitetura *D-Bus* **[6]**. O *D-Bus* é um subsistema de comunicação entre programas no qual cada aplicativo "se encaixa" em um conector especial e se registra no kernel, tomando para si a responsabilidade de tratar um determinado número de eventos à sua escolha. Por exemplo, um programa de edição de vídeo precisa saber que uma nova câmera foi conectada ao PC. O utilitário *gnome-volume-properties*, presente nas versões mais recentes do Gnome, usa o *D-Bus* e o HAL para especificar qual aplicativo deve ser chamado para cada evento *hotplug* (**figura 3**).

Espera-se que brevemente o *D-Bus* tenha papel importante nas comunicações entre aplicativos do Gnome. Entretanto, até o presente momento quase nenhum deles se mostrou capaz disso.

Tudo de bom pra você – mas não hoje...

Apesar de todo o progresso já alcançado no tocante a detecção de hardware, as coisas estão longe de ser perfeitas. Os agentes precisam de informações detalhadas sobre o hardware – informações que já estarão obsoletas em sistemas com poucas semanas decorridas desde sua instalação. Um banco de dados online com os componentes de hardware ajudaria muito: os usuários que já conseguiram domar seus arquivos FDI poderiam doá-los à comunidade por meio dele.

O projeto HAL está engatinhando nessa direção, pois dá ao sistema hotplug informações das quais o kernel não tem a menor idéia. O fato de um número grande de distribuições já o estar usando é um bom sinal. Esperamos que a SuSE desista de seu sistema próprio e siga a tendência. Quanto mais consistente o gerenciamento de hardware for em todas as distribuições Linux, melhor. ■

Informações

[1] Hotplug no Linux: http://linux-hotplug.sourceforge.net

[2] Escrevendo suas próprias regras para o Udev (em inglês): http://www.reactivated.net/udevrules.php

[3] Documentação sobre o Udev do Fedora Core http://fedora.redhat.com/docs/udev

[4] HAL - Hardware Abstraction Layer: http://www.freedesktop.org/Software/hal

[5] Conectando seu iPod com a ajuda do Udev: http://www.kgarner.com/blog/archives/2005/01/11/fc3-hal-ipod/

[6] D-Bus: http://www.freedesktop.org/Software/dbus