Trabajo Práctico N°7

Práctica con Linux

Buses

Los componentes individuales de la placa del sistema (motherboard) están interconectados por un sistema de conexión múltiple conocido como *colector* (más conocido por su término en inglés *bus*). El bus del sistema se divide en tres funciones lógicas;

- Bus de **direcciones**
  - Especifica las ubicaciones de memoria (direcciones) para las transferencias de datos y el bus de datos retiene los datos transferidos
- Bus de **datos**
  - Bidireccional, permite que los datos se lean en la CPU y se escriban desde la CPU
- Bus de **control**
  - Contiene varias líneas utilizadas para transferir señales de temporización y control en todo el sistema

Existen muchos tipos de bus, por ejemplo, los buses SATA y PCI son formas populares de conectar periféricos al sistema, como vemos en la Figura 7.1.

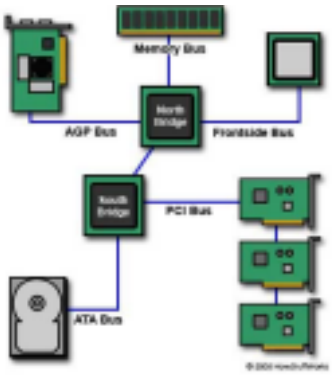

Figura 7.1.: Colectoras en la placa madre

Los **periféricos** son dispositivos reales, como tarjetas gráficas o discos controlados por chips controladores o controladores.
Los **discos SATA** están controlados por el chip del controlador SATA.
Los **discos SCSI** por los chips del controlador de disco SCSI, etc. Estos controladores están conectados a la CPU y entre sí mediante una variedad de buses. La mayoría de los sistemas construidos ahora usan buses PCI y AGP para conectar los componentes principales del sistema.

Los **controladores** son procesadores como la propia CPU, pueden verse como ayudantes inteligentes con la CPU en el control general del sistema. Todos los controladores son diferentes, pero tienen registros que los controlan. El software que se ejecuta en la CPU debe poder leer y escribir esos registros de control.

Un **registro** puede contener el estado que describe un error. Otro podría usarse para fines de control; cambiando el modo del controlador. Cada controlador en un bus puede ser direccionado individualmente por la CPU, debe ser así para que el controlador del dispositivo de software pueda escribir en sus registros y así controlarlo.

Dispositivos en /dev

Como decíamos al principio de la materia, UNIX modela a los dispositivos de entrada y salida como si fueran archivos, de esta forma las mismas operaciones (llamadas a sistema) utilizadas para acceder a un archivo (leer, escribir, buscar y posicionarse) se utilizan sobre los dispositivos.

Entonces «imprimir» implica «escribir en la impresora», mostrar un carácter (o un texto) en la pantalla implica escribir en ella, para escuchar mi tema preferido de música tengo que escribir sobre mi dispositivo de audio. Análogamente hago operaciones de lectura del teclado y del mouse.

Estos «archivos dispositivos» son una interfaz al manejador de dispositivos (device driver), que aparecen en el árbol del sistema de archivos como si fueran archivos verdaderos pero no lo son, si bien el usuario y las aplicaciones los perciben como tales.
Por razones organizativas están en el directorio /dev. De hecho, si listamos el contenido de este directorio los veremos como en la Figura 7.2:

```
ubuntu@ubuntu:~$ ls  /dev
autofs          fd          lightnvm          null      snd     tty20  tty38  ttyS5     ttyS13  ttyS30  vcsa
block           full        log               port      sr0     tty21  tty39  ttyS6     ttyS14  ttyS31  vcsa1
bsg             fuse        loop0             ppp       stderr  tty22  tty4   ttyS7     ttyS15  ttyS4   vcsa2
btrfs-control   fw0         loop1             psaux     stdin   tty23  tty40  ttyS8     ttyS16  ttyS5   vcsa3
bus             gpiochip0   loop2             ptmx      stdout  tty24  tty41  ttyS9     ttyS17  ttyS6   vcsa4
cdrom           gpiochip1   loop3             pts       tty     tty25  tty42  tty6      ttyS18  ttyS7   vcsa5
cdrw            hpet        loop4             random    tty0    tty26  tty43  tty60     ttyS19  ttyS8   vcsa6
char            hugepages   loop5             rfkill    tty1    tty27  tty44  tty61     ttyS2   ttyS9   vfio
console         hwrng       loop6             rtc       tty10   tty28  tty45  tty62     ttyS20  uhid    vga_arbiter
core            i2c-0       loop7             rtc0      tty11   tty29  tty46  tty63     ttyS21  uinput  vhci
cpu             i2c-1       loop-control      sda       tty12   tty3   tty47  tty7      ttyS22  urandom  vhost-net
cpu_dma_latency i2c-2       mapper            sda1      tty13   tty30  tty48  tty8      ttyS23  userio  vhost-vsock
cuse            i2c-3       mcelog            sda2      tty14   tty31  tty49  tty9      ttyS24  vcs     zero
disk            i2c-4       mem               sda3      tty15   tty32  tty5   ttyprintk ttyS25  vcs1
dri             i2c-5       memory_bandwidth  sda4      tty16   tty33  tty50  ttyS0     ttyS26  vcs2
dvd             initctl     mqueue            sg0       tty17   tty34  tty51  ttyS1     ttyS27  vcs3
dvdrw           input       net               sg1       tty18   tty35  tty52  ttyS10    ttyS28  vcs4
ecryptfs        kmsg        network_latency   shm       tty19   tty36  tty53  ttyS11    ttyS29  vcs5
fb0             kvm         network_throughput snapshot tty2    tty37  tty54  ttyS12    ttyS3   vcs6
ubuntu@ubuntu:~$
```

Figura 7.2.: Listado de «/dev»

Pero si utilizamos el comando «ls -l /dev» para ver el listado en formato largo, es decir cada archivo con sus detalles, veremos que al comienzo de cada línea que describe al archivo vemos una «c» o una «b», lo que indica que se trata de un archivo dispositivo orientado al carácter o al bloque respectivamente. Si busca la línea que corresponde a su disco rígido (por ejemplo, «/dev/sda») notará que se trata de un dispositivo orientado al bloque.
En un sistema Unix tradicional (y en el núcleo Linux hasta la versión 2.5, sin udev ni devfs), en el directorio «/dev» estos archivos dispositivos eran creados mediante el comando «mknod» para cada dispositivo conocido, estuviera o no en el sistema. Se dice que es un conjunto de archivos estático, ya que los nodos no cambian.

Dispositivos en /sys

Ya habíamos observado que el sistema de archivos sysfs estaba montado en el directorio «/sys». De manera similar a procfs, este sistema de archivos volátil -sin almacenamiento permanente en el disco rígido- fue incorporado al núcleo Linux a partir de la versión 2.6 para **exportar información sobre los dispositivos y sus manejadores** (device drivers) desde el modelo de dispositivos del núcleo hacia el espacio del usuario, **también** permite **configurar parámetros**.
Para cada objeto añadido en el árbol del modelo de manejadores (manejadores y dispositivos incluyendo clases) se crea un directorio en «/sys».

- «/sys/devices/» la relación padre/hijo de la capa física se refleja con subdirectorios
- «/sys/bus/» se puebla con enlaces simbólicos, reflejando el modo en el que los dispositivos pertenecen a diferentes colectores o buses.
- «/sys/class/» muestra dispositivos agrupados de acuerdo a su clase, como por ejemplo red
- «/sys/block/» contiene los dispositivos de bloques.

Tanto para los manejadores de dispositivos como para los dispositivos se pueden crear atributos. Los atributos son simples archivos. Se estipula que sólo deben contener un valor o permitir que un solo valor se fije (a diferencia de algunos archivos en /proc, que necesitan un análisis intenso). Estos archivos están incluidos en el subdirectorio del manejador correspondiente al dispositivo. Es posible crear subdirectorios con atributos para agruparlos.

udev

A partir de la versión de desarrollo 2.5 del núcleo de Linux, todos los dispositivos físicos y virtuales en un sistema son visibles en espacio de usuario de una forma jerárquica gracias a sysfs. El programa udev, desarrollado por Kroah-Hartman (2003), proporciona entradas en «/dev» sólo para los dispositivos que están presentes en el sistema en un momento dado, y provee prestaciones con las que no se podía contar previamente, tales como:

- **Nombres fijos o persistentes** de dispositivos cuando se mueven en el árbol de dispositivos. Por ejemplo, se puede llamar cámara a una cámara fotográfica que se conecta a algunos de los puertos USB, indistintamente de cuál sea.
- **Notificación** de sistemas externos **de cambios de dispositivos**; mediante mensajes D-Bus para que cualquier programa pueda enterarse cuando un dispositivo se conecta o desconecta.
- Un **esquema** flexible **de nombres** de dispositivos, por ejemplo cámara, como ya se dijo.
- Le permite al núcleo el uso de números mayores y menores dinámicos. El programa udev no usa el número mayor o menor para reconocer al dispositivo, puede funcionar con números al azar, por lo tanto ya dejó de ser una preocupación el agotamiento de los números mayores y menores.
- Mueve toda la política de nombres fuera del núcleo.

Udev opera disparado por eventos en el núcleo (uevent). Un *evento* es un mensaje de software que indica que algo ha pasado, por ejemplo al presionar una tecla o al hacer clic con el mouse; es una

acción que se inicia fuera del ámbito de un programa pero que es manejada por una pieza de código dentro del mismo.
Si conectamos o desconectamos dispositivos a la computadora también generamos eventos.

¿Cómo puedo saber a qué subsistema pertenece un dispositivo? Por ejemplo, tengo el dispositivo «/dev/sdb». ¿Cómo puedo averiguar su subsistema «udev»? Una forma sencilla de encontrarlo es, justamente, mediante el comando «find», por ejemplo si quisiéramos primero encontrar el de «sda»:

```
[ubuntu@ubuntu:~]$ find /sys/devices/ -name sda
/sys/devices/pci0000:00/0000:00:1f.2/ata3/host2/target2:0:0/2:0:0:0/blo
ck/sda
```

El comando «udevadm» permite consultar la base de datos «udev» para obtener información de los dispositivos, en este caso hacemos una prueba cargando el módulo «scsi_debug» para generar nuestro disco rígido y generamos la consulta mediante el comando «udevadm info -q all -a /dev/sdb»

```
root@ollie1:/home/coronel#
root@ollie1:/home/coronel# cd
root@ollie1:~#
root@ollie1:~# modprobe scsi_debug
root@ollie1:~# udevadm info -q all -a /dev/sda

Udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

  looking at device '/devices/pci0000:00/0000:00:11.0/ata1/host0/target0:0:0/0:0:0:0/block/sda':
    KERNEL=="sda"
    SUBSYSTEM=="block"
    DRIVER==""
    ATTR{alignment_offset}=="0"
    ATTR{capability}=="40"
    ATTR{discard_alignment}=="0"
    ATTR{diskseq}=="9"
    ATTR{events}==""
    ATTR{events_async}==""
    ATTR{events_poll_msecs}=="-1"
    ATTR{ext_range}=="256"
    ATTR{hidden}=="0"
```

La salida mostrada por pantalla es más larga que la que vemos en la figura.

Y si quisiéramos saber algo más acerca de nuestra placa de video, podríamos hacerlo mediante el comando:

```
root@ollie1:~# udevadm info -a -p /sys/class/drm/card0-VGA-1

Udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

  looking at device '/devices/pci0000:00/0000:00:01.0/0000:01:05.0/drm/card0/card0-VGA-1':
    KERNEL=="card0-VGA-1"
    SUBSYSTEM=="drm"
    DRIVER==""
    ATTR{dpms}=="On"
    ATTR{edid}==""
    ATTR{enabled}=="disabled"
    ATTR{modes}==""
    ATTR{power/async}=="disabled"
    ATTR{power/control}=="auto"
    ATTR{power/runtime_active_kids}=="0"
    ATTR{power/runtime_active_time}=="0"
    ATTR{power/runtime_enabled}=="disabled"
    ATTR{power/runtime_status}=="unsupported"
    ATTR{power/runtime_suspended_time}=="0"
    ATTR{power/runtime_usage}=="0"
    ATTR{status}=="disconnected"

  looking at parent device '/devices/pci0000:00/0000:00:01.0/0000:01:05.0/drm/card0':
    KERNELS=="card0"
    SUBSYSTEMS=="drm"
    DRIVERS==""
```

También en este caso la salida es más larga que la mostrada.

El colector o bus PCI

El estándar PCI se ha convertido en el estándar de facto para los colectores o buses del sistema. Linux proporciona un amplio soporte para PCI y contiene numerosos drivers para red, almacenamiento y adaptadores de terceros. Veremos cómo el núcleo Linux representa los dispositivos PCI y mostraremos cómo decodificar dispositivos con un identificador PCI.
El núcleo Linux representa los dispositivos PCI como pseudo dispositivos en el sistema de archivos «sysfs», como vemos en la figura Figura 7.5

```
[ubuntu@ubuntu:~]$ ls -la /sys/bus/pci/devices
total 0
drwxr-xr-x 2 root root 0 oct 25 15:20 .
drwxr-xr-x 5 root root 0 oct 25 15:20 ..
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:00.0 -> ../../../devices/pci0000:00/0000:00:00.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:02.0 -> ../../../devices/pci0000:00/0000:00:02.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:02.1 -> ../../../devices/pci0000:00/0000:00:02.1
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1b.0 -> ../../../devices/pci0000:00/0000:00:1b.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1c.0 -> ../../../devices/pci0000:00/0000:00:1c.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1c.3 -> ../../../devices/pci0000:00/0000:00:1c.3
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1d.0 -> ../../../devices/pci0000:00/0000:00:1d.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1d.1 -> ../../../devices/pci0000:00/0000:00:1d.1
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1d.2 -> ../../../devices/pci0000:00/0000:00:1d.2
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1d.3 -> ../../../devices/pci0000:00/0000:00:1d.3
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1d.7 -> ../../../devices/pci0000:00/0000:00:1d.7
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1e.0 -> ../../../devices/pci0000:00/0000:00:1e.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1f.0 -> ../../../devices/pci0000:00/0000:00:1f.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1f.2 -> ../../../devices/pci0000:00/0000:00:1f.2
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1f.3 -> ../../../devices/pci0000:00/0000:00:1f.3
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:03:00.0 -> ../../../devices/pci0000:00/0000:00:1e.0/0000:03:00.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:03:01.0 -> ../../../devices/pci0000:00/0000:00:1e.0/0000:03:01.0
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:03:01.1 -> ../../../devices/pci0000:00/0000:00:1e.0/0000:03:01.1
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:03:01.2 -> ../../../devices/pci0000:00/0000:00:1e.0/0000:03:01.2
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:03:01.3 -> ../../../devices/pci0000:00/0000:00:1e.0/0000:03:01.3
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:0b:00.0 -> ../../../devices/pci0000:00/0000:00:1c.0/0000:0b:00.0
[ubuntu@ubuntu:~]$
```

Figura 7.5.: Listado de /sys/bus/pci/devices

Dada una entrada como:

```
lrwxrwxrwx 1 root root 0 oct 25 15:20 0000:00:1f.2 ->
../../../devices/pci0000:00/0000:00:1f.2
```

Podemos dividir la cadena del dispositivo «0000:00:1f.2» de la siguiente manera:

```
0000 : dominio PCI (cada dominio puede contener hasta 256 buses PCI) 00 : el número de
bus al que está conectado el dispositivo
1f : el número del dispositivo
.2 : función del dispositivo PCI
```

Si nos cambiamos dentro de ese directorio para ver qué archivos contiene y cuál es su contenido:

```
[ubuntu@ubuntu:~]$ cd "/sys/bus/pci/devices/0000:00:1f.2"
[ubuntu@ubuntu:/sys/bus/pci/devices/0000:00:1f.2]$ ls -l
(el listado es largo)
[ubuntu@ubuntu:/sys/bus/pci/devices/0000:00:1f.2]$ cat vendor
0x8086
[ubuntu@ubuntu:/sys/bus/pci/devices/0000:00:1f.2]$ cat device
0x27c4
[ubuntu@ubuntu:/sys/bus/pci/devices/0000:00:1f.2]$ cat class
0x010180
```

Cada entrada en «sysfs» contiene un dato único, como la identificación del proveedor de PCI (vendor), la clase de dispositivo (class), el identificador de dispositivo (device) e información sobre IRQ y asignaciones de recursos.

También podemos listar el contenido de los dispositivos conectados al bus PCI mediante «lspci», como vemos en la Figura 7.6:

```
ubuntu@ubuntu:~$ lspci
00:00.0 Host bridge: Intel Corporation Mobile 945GM/PM/GMS, 943/940GML and 945GT Express Memory Controller Hub (rev 03)
00:02.0 VGA compatible controller: Intel Corporation Mobile 945GM/GMS, 943/940GML Express Integrated Graphics Controller (rev 03)
00:02.1 Display controller: Intel Corporation Mobile 945GM/GMS/GME, 943/940GML Express Integrated Graphics Controller (rev 03)
00:1b.0 Audio device: Intel Corporation NM10/ICH7 Family High Definition Audio Controller (rev 01)
00:1c.0 PCI bridge: Intel Corporation NM10/ICH7 Family PCI Express Port 1 (rev 01)
00:1c.3 PCI bridge: Intel Corporation NM10/ICH7 Family PCI Express Port 4 (rev 01)
00:1d.0 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #1 (rev 01)
00:1d.1 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #2 (rev 01)
00:1d.2 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #3 (rev 01)
00:1d.3 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #4 (rev 01)
00:1d.7 USB controller: Intel Corporation NM10/ICH7 Family USB2 EHCI Controller (rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 Mobile PCI Bridge (rev e1)
00:1f.0 ISA bridge: Intel Corporation 82801GBM (ICH7-M) LPC Interface Bridge (rev 01)
00:1f.2 IDE interface: Intel Corporation 82801GBM/GHM (ICH7-M Family) SATA Controller [IDE mode] (rev 01)
00:1f.3 SMBus: Intel Corporation NM10/ICH7 Family SMBus Controller (rev 01)
03:00.0 Ethernet controller: Broadcom Limited BCM4401-B0 100Base-TX (rev 02)
03:01.0 FireWire (IEEE 1394): Ricoh Co Ltd R5C832 IEEE 1394 Controller
03:01.1 SD Host controller: Ricoh Co Ltd R5C822 SD/SDIO/MMC/MS/MSPro Host Adapter (rev 19)
03:01.2 System peripheral: Ricoh Co Ltd R5C592 Memory Stick Bus Host Adapter (rev 0a)
03:01.3 System peripheral: Ricoh Co Ltd xD-Picture Card Controller (rev 05)
0b:00.0 Network controller: Broadcom Limited BCM4311 802.11b/g WLAN (rev 01)
ubuntu@ubuntu:~$
```

Figura 7.6.: Listado de los dispositivos PCI

Cada línea en la salida anterior muestra un dispositivo PCI. A cada dispositivo se le asigna

- número de bus
- número de dispositivo
- número de función

Estos tres números se conocen comúnmente como **BDF** o B/D/F del dispositivo (bus/dispositivo/función). En Linux, a los dispositivos PCI también se les asignan *números de dominio*, pero «lspci» generalmente los omite, ya que muy a menudo todos los dispositivos tienen el mismo número de dominio (generalmente cero)[4]. Linux asigna estos cuatro números a cada dispositivo, ya sea en el arranque o cuando un dispositivo está conectado en caliente (hotplug).
En la línea 14 de la Figura 7.6 muestra un dispositivo con número de bus 00, número de dispositivo 1f y número de función 2. La clase de este dispositivo es «Interfaz IDE», su proveedor es «Intel Corporation» y su nombre es «82801GBM/GHM (ICH7-M Family) SATA Controller [IDE mode]». En otras palabras, es un controlador SATA que opera en modo IDE (Integrated Drive Electronics). Observe que en el mismo dispositivo 1f está el puente ISA (ISA bridge) y un controlador SMBus (System Management Bus): todos tienen el mismo dispositivo y números de bus pero distintos números de función.
El sistema operativo obtiene la clase, el nombre y el proveedor de un dispositivo PCI por que cada dispositivo PCI tiene un conjunto de registros denominado «espacio de configuración» del dispositivo que, entre otras cosas, muestra el ID del dispositivo (DID), el ID del proveedor (VID) y la clase del dispositivo al sistema operativo. Estos son sólo códigos numéricos que el sistema operativo asigna a nombres legibles por humanos a través de una tabla predefinida.
Los dispositivos PCI se pueden mostrar en una estructura en forma de árbol que refleja la estructura física real de los buses o colectores PCI. Para verlo, ejecute «lspci -tv» y verá una salida similar a la de la Figura 7.7:

```
ubuntu@ubuntu:~$ lspci -tv
-[0000:00]-+-00.0  Intel Corporation Mobile 945GM/PM/GMS, 943/940GML and 945GT Express Memory Controller Hub
           +-02.0  Intel Corporation Mobile 945GM/GMS, 943/940GML Express Integrated Graphics Controller
           +-02.1  Intel Corporation Mobile 945GM/GMS/GME, 943/940GML Express Integrated Graphics Controller
           +-1b.0  Intel Corporation NM10/ICH7 Family High Definition Audio Controller
           +-1c.0-[0b]----00.0  Broadcom Limited BCM4311 802.11b/g WLAN
           +-1c.3-[0c-0d]--
           +-1d.0  Intel Corporation NM10/ICH7 Family USB UHCI Controller #1
           +-1d.1  Intel Corporation NM10/ICH7 Family USB UHCI Controller #2
           +-1d.2  Intel Corporation NM10/ICH7 Family USB UHCI Controller #3
           +-1d.3  Intel Corporation NM10/ICH7 Family USB UHCI Controller #4
           +-1d.7  Intel Corporation NM10/ICH7 Family USB2 EHCI Controller
           +-1e.0-[03]--+-00.0  Broadcom Limited BCM4401-B0 100Base-TX
           |            +-01.0  Ricoh Co Ltd R5C832 IEEE 1394 Controller
           |            +-01.1  Ricoh Co Ltd R5C822 SD/SDIO/MMC/MS/MSPro Host Adapter
           |            +-01.2  Ricoh Co Ltd R5C592 Memory Stick Bus Host Adapter
           |            \-01.3  Ricoh Co Ltd xD-Picture Card Controller
           +-1f.0  Intel Corporation 82801GBM (ICH7-M) LPC Interface Bridge
           +-1f.2  Intel Corporation 82801GBM/GHM (ICH7-M Family) SATA Controller [IDE mode]
           \-1f.3  Intel Corporation NM10/ICH7 Family SMBus Controller
ubuntu@ubuntu:~$
```

Figura 7.7.: Estructura en forma de árbol del colector PCI

La opción «-v», como siempre, significa «verbose» para ver más detalles. Como la salida puede ser demasiado larga, podemos indicarle a «lspci» que muestre los detalles de un solo dispositivo utilizando el dominio del dispositivo y los números BDF. Por ejemplo, para mostrar sólo los detalles del controlador SATA mencionado, deberíamos ejecutar «lspci -v -s 00:1f.2» y veríamos algo similar a lo que se muestra en la Figura 7.8. A veces es conveniente ejecutar este comando escalando privilegios a «root» para evitar, como en este caso, el texto «<access denied>»:

```
ubuntu@ubuntu:~$ lspci -v -s 00:1f.2
00:1f.2 IDE interface: Intel Corporation 82801GBM/GHM (ICH7-M Family) SATA Controller [IDE mode] (rev 01) (prog-if 80 [Master])
        Subsystem: Dell 82801GBM/GHM (ICH7-M Family) SATA Controller [IDE mode]
        Flags: bus master, 66MHz, medium devsel, latency 0, IRQ 17
        I/O ports at 01f0 [size=8]
        I/O ports at 03f4
        I/O ports at 0170 [size=8]
        I/O ports at 0374
        I/O ports at bfa0 [size=16]
        Capabilities: <access denied>
        Kernel driver in use: ata_piix
        Kernel modules: pata_acpi
```

Figura 7.8.: Los detalles del controlador SATA

Las controladoras del disco

Por tratarse en este caso de un disco de tecnología ATA (Advanced Technologies Attachment) en general, PATA (Parallel ATA) o SATA (Serial ATA) en particular, está conectado a una controladora de esta tecnología. En efecto, algunas líneas antes de la detección del disco, están las de detección de la controladora, como vemos en el fragmento de la salida del comando «dmesg», en la Figura 7.9:

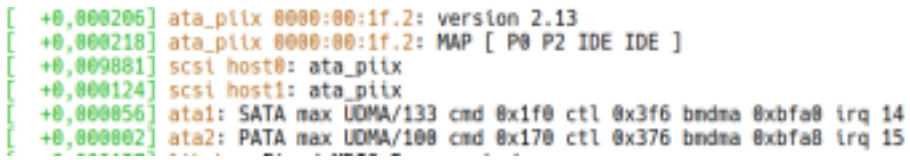

Figura 7.9.: Detección de la controladora

Estas controladoras casi siempre están incluídas en la placa base (motherboard) en el bus o colector PCI (Peripheral Component Interconnect, Interconexión de Componentes Periféricos).

El disco rígido

Al iniciar el sistema, el núcleo detecta los dispositivos presentes en la computadora. Uno de los dispositivos detectados es el disco rígido. Esto lo podemos corroborar gracias al comando «dmesg», que ya hemos utilizado. Si paginamos su extensa salida con el comando «less», de esta forma:

```
ubuntu@ubuntu:~$ dmesg | less
```

También podemos usar la opción «-H», es decir «**dmesg -H**» que sirve para paginar de manera amistosa sin necesidad de usar el pipe a «less». Para movernos podemos usar los cursores o la barra espaciadora y en algún momento nos encontraremos con la detección de nuestro disco rígido, como vemos en la Figura 7.10:

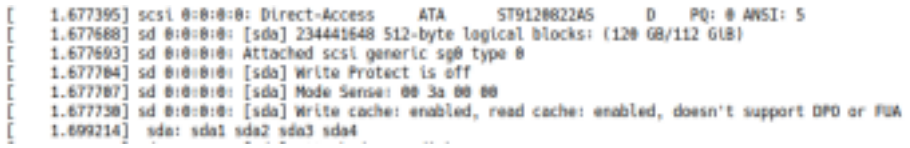

Figura 7.10.: Disco «sda» detectado por el núcleo al arranque

Un disco ATA ST9120822AS de 120 GB al que llama «sda» con cuatro particiones: «sda1», «sda2», «sda3» y «sda4». Por lo tanto a este dispositivo con sus particiones lo podemos ver dentro de «/dev»:

```
ubuntu@ubuntu:~$ ls -l /dev/sda*
brw-rw---- 1 root disk 8, 0 oct 24 20:35 /dev/sda
brw-rw---- 1 root disk 8, 1 oct 24 20:35 /dev/sda1
brw-rw---- 1 root disk 8, 2 oct 24 20:35 /dev/sda2
brw-rw---- 1 root disk 8, 3 oct 24 20:35 /dev/sda3
brw-rw---- 1 root disk 8, 4 oct 24 20:35 /dev/sda4
```

La «b» al comienzo de cada línea nos indica que es un dispositivo por **b**loques, luego vienen los permisos que tiene el archivo, el superusuario «root» es el propietario de los mismos y tiene permiso de lectura y escritura (rw-), luego vienen los permisos del grupo «disk» (rw-, lectura y escritura) y ningún permiso (---) para los otros grupos/usuarios del sistema. Luego el número contador de enlaces duros (hardlink). Posteriormente el nombre del usuario propietario (root) y el grupo (disk). Luego los números mayor y menor (major and minor numbers). El mayor es el

genérico y el menor el específico, es decir «8» es el número del manejador del dispositivo (device driver) «disco» que utiliza el núcleo, y el número menor suele identificar a un dispositivo específico, en este caso a la partición del disco.
Este esquema de modos de acceso indica que sólo «root» o quienes pertenezcan al grupo «disk» podrán ver o modificar la tabla de particiones de este disco. Los restantes ni siquiera podrán verla.

Comando lsblk Podemos listar los dispositivos por bloques mediante el comando «lsblk» como vemos en la Figura 7.11

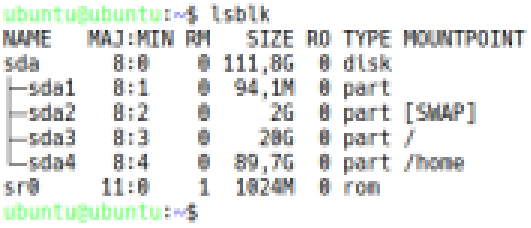

Figura 7.11.: Listado de los dispositivos por bloques

Las columnas son

- «NAME» (nombre)
- «MAJ:MIN» (números mayor y menor)
- «RM» (Removable device): «0» si el dispositivo no es removible o «1» si lo es
- «SIZE» (tamaño)
- «RO» (read-only device): «1» si el dispositivo es de sólo lectura o «0» si no lo es
- «TYPE» (tipo)
- «MOUNTPOINT» (punto de montaje)

Las líneas muestran al disco rígido «sda» y a sus particiones, también muestran al dispositivo lector de DVD (sr0). Efectivamente también es un dispositivo orientado al bloque, y así lo podemos ver en el directorio «/dev»:

```
[ubuntu@ubuntu:~]$ ls -l /dev/sr0
brw-rw----+ 1 root cdrom 11, 0 oct 22 14:04 /dev/sr0
```

Pruebe ahora insertando un pendrive con conector USB y vea qué pasa con los comandos «dmesg», «lsblk» y listando «/dev». También puede probar qué es lo que pasa insertando el módulo «scsi_debug» que vimos.

Comando lsscsi Como el sistema operativo está utilizando el manejador de dispositivos SCSI (Small Computer System Interface) para el disco SATA podemos usar el comando «lsscsi» para listar los dispositivos SCSI presentes en la computadora como vemos en la Figura 7.12:

Figura 7.12.: lsscsi

Este comando lista un dispositivo por línea, en la que vemos primero

- [H:C:T:L], es decir:
  - *scsi_host_adapter* (ID del adaptador del anfitrión SCSI)
  - *channel* (canal SCSI en el adaptador)
  - *target_number* (número identificador (ID))
  - *logical unit number* (LUN) (número de unidad lógica)
- Tipo de periférico SCSI; en lugar de usar el nombre formal (por ejemplo, «dispositivo de acceso directo») se usa un nombre más corto.
- Nombre del proveedor
- Nombre del modelo
- Cadena de revisión.
- La última entrada es el nombre del nodo del dispositivo primario.

El nombre del nodo del dispositivo «primario» está asociado con el controlador SCSI de nivel superior que «posee» el dispositivo. Ejemplos de controladores SCSI de nivel superior son «sd» para discos, «sr» para unidades ópticas y «st» para cintas.

Comando iostat El comando «iostat» es parte del paquete «sysstat» por lo que, si no existe el programa en su sistema, habrá que instalar este paquete («sudo apt install sysstat»). Este programa, si se ejecuta sin argumentos, muestra información sobre el uso de la CPU y las estadísticas de entrada y salida para cada partición del sistema.

Comando iotop Hasta ahora hemos visto cómo funciona el entramado de colectores de entrada y salida desde y hacia los dispositivos. Esto nos sirve para comprender cómo están armadas las estructuras. Pero también es importante observar cómo se desempeñan, sobre todo si notamos una cierta sobrecarga en el sistema. En este sentido, de manera similar a la utilidad «top» que vimos, tenemos a una llamada «iotop» que, si no está ya instalada, se lo hace sencillamente con:

```
[ubuntu@ubuntu:~]$ sudo apt install iotop
```

Este utilitario necesita ejecutarse con privilegios administrativos. Si se lo ejecuta sin opciones nos mostrará, al estilo de «top», una lista dinámica de todos los procesos que se están ejecutando y -ya sea que estén o no- realizando operaciones de entrada y salida. Por este motivo en varias ocasiones será deseable ejecutarlo con la opción «--only», para que sólo muestre a aquellos procesos que están efectuando operaciones de entrada o salida:

```
[ubuntu@ubuntu:~]$ sudo iotop --only
```

USB - Universal Serial Bus

El Bus Universal en Serie (BUS) (en inglés: Universal Serial Bus), más conocido por la sigla USB, es un bus de comunicaciones que sigue un estándar que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras,

periféricos y dispositivos electrónicos.
El USB es utilizado como estándar de conexión de periféricos como: teclados, ratones, memorias USB, joysticks, escáneres, cámaras digitales, teléfonos móviles, reproductores multimedia, impresoras, dispositivos multifuncionales, sistemas de adquisición de datos, módems, tarjetas de red, tarjetas de sonido, tarjetas sintonizadoras de televisión y grabadoras de DVD externas, discos duros externos y disqueteras externas.

Comando lsusb El comando «lsusb», como es de suponer, nos lista tanto el sistema como los dispositivos conectados al bus USB, como vemos en la Figura 7.13.

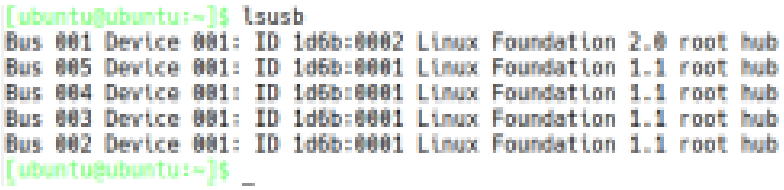

Figura 7.13.: lsusb

Las terminales y pseudo terminales

Si hemos ingresado al sistema desde la primera consola de texto como usuario «ubuntu» y colocamos el comando

```
ubuntu@ubuntu:~$ ls -l /dev/tty1
```

vemos

```
crw--w---- 1 ubuntu tty 4,1 Oct 25 11:30 /dev/tty1
```

Pero si hemos ingresado desde el ambiente gráfico y abrimos una ventana de la aplicación «Terminal», ésta opera sobre una pseudo-terminal, entonces el archivo dispositivo es, por ejemplo, «/dev/pts/0» o «/dev/pts/1»[10]. Esto es fácil de saber gracias a los comandos «w», «who» o «tty». De cualquier manera, estos archivos dispositivos de terminal han quedado de propiedad del usuario que los está utilizando.
La «c» nos indica que es un dispositivo por **c**aracteres, luego vienen los permisos que tiene el archivo, nosotros (ubuntu) somos los propietarios del mismo y tenemos permiso de lectura y escritura (rw-), luego vienen los permisos del grupo (-w-, escritura pero no lectura) y ningún permiso (---) para los otros grupos/usuarios del sistema. Luego el número contador de enlaces duros (hardlink). Posteriormente el nombre del usuario propietario (ubuntu) y el grupo (tty). Luego los números mayor y menor (major and minor numbers). El mayor es el genérico y el menor el específico, es decir «4» significa tty, terminal o consola. Si ejecutamos:

```
ubuntu@ubuntu:~$ ls -l /dev/tty2
ubuntu@ubuntu:~$ ls -l /dev/tty3
```

vemos que el número mayor es el mismo, pero cambia el menor.

Experimento con echo El comando «echo» muestra una línea de texto por pantalla, por ejemplo:

```
ubuntu@ubuntu:~$ echo "Hola, mundo"
```

Podemos redirigir la salida con «>» intentando escribir sobre otra terminal o consola (distinta a la nuestra):

```
ubuntu@ubuntu:~$ echo "Hola, mundo" > /dev/tty3
```

y el sistema nos contesta «Permiso denegado» ¿Por qué? ¿Cómo podemos solucionarlo?.
Ingresando con el mismo usuario a esa terminal o consola. Si estamos trabajando en el ambiente gráfico y hemos abierto una ventana «Terminal» podemos o bien abrir una pestaña nueva o abrir otra «Terminal». En ambas ejecutemos el comando «tty». Veríamos algo así:

```
ubuntu@ubuntu:~$ tty
/dev/pts/1
ubuntu@ubuntu:~$
```

Y en la otra mostraría, por ejemplo «/dev/pts/2». Si desde la «/dev/pts/1» ejecutamos:

```
ubuntu@ubuntu:~$ echo "Hola, mundo" > /dev/pts/2
```

Veremos el mensaje en la otra pseudo-terminal.

Los dispositivos de entrada, salida y error

Y hablando de stdin, stdout y stderr si colocamos, por ejemplo:

```
ubuntu@ubuntu:~$ ls -l /dev/stdin
```

vemos

```
lrwxrwxrwx 1 root root 17 Jun 14 17:21 /dev/stdin->../proc/self/fd/0
```

y si le seguimos la pista al enlace

```
ubuntu@ubuntu:~$ ls -l /proc/self/fd/0
```

vemos

```
lrwx------ 1 ubuntu ubuntu 64 Oct 25 18:26 /proc/self/fd/0->/dev/tty1
```

O, si estamos en una aplicación «Terminal»:

```
lrwx------ 1 ubuntu ubuntu 64 Oct 25 18:26 /proc/self/fd/0->/dev/pts/1
```

Pruebe con /dev/stdout y /dev/stderr y siga los enlaces ¿Qué conclusiones saca?.
stdin, stdout y stderr son flujos (streams) adjuntos a los descriptores de archivos 0, 1 y 2 respectivamente de un proceso.

En Linux, /dev/stdin, /dev/stdout, /dev/stderr son enlaces simbólicos a /proc/self/fd/0, /proc/self/fd/1, /proc/self/fd/2 respectivamente, y ellos mismos son enlaces simbólicos especiales al archivo real que está abierto en esos descriptores de archivo.

Dispositivo de bytes nulos

Análogamente tenemos un dispositivo «/dev/zero» que es un repositorio de ceros o mejor dicho de bytes nulos, que pueden ser útiles para generar rápidamente un archivo del tamaño que deseemos y cuyo contenido no nos importe demasiado, por ejemplo:

```
ubuntu@ubuntu:~$ dd bs=1024 count=1 if=/dev/zero of=unka
```

El comando dd (del inglés disk dump) convierte y copia un archivo,

- *bs* le indica cuántos bytes debe leer/escribir
- *count* le indica cuántos bloques debe copiar
- *if* (del inglés input file) le indica el archivo de entrada (en lugar de stdin)
- *of* (output file) el archivo de salida (en lugar de stdout).

El comando previo crea el archivo «*unka*» de 1K (1024 bytes) de tamaño.

Podemos utilizar este programa para obtener el sector de arranque (boot) de un disco rígido. Por ejemplo:

```
ubuntu@ubuntu:~$ dd if=/dev/sda of=boot-sd bs=512 count=1
```

Éste es el MBR (Master Boot Record o registro de arranque principal) y se utiliza para arrancar la computadora. El final del MBR contiene la tabla de particiones, la cual proporciona las direcciones de inicio y fin de cada partición. Eventualmente, en los primeros 466 bytes, puede haber código ejecutable por la BIOS, al que se le llama gestor de arranque (o boot loader, en inglés); por ejemplo Grub, Lilo y OS Loader. Si no hay gestor de arranque se marca una de las particiones en la tabla como activa para que la BIOS lea su primer bloque (bloque de arranque) y lo ejecute. El programa en el bloque de arranque carga el sistema operativo contenido en esa partición.
Podemos analizar el archivo generado en esta actividad con el depurador (gdb de Linux).

¿Cómo creamos archivos especiales?

Si bien con el desarrollo de «udev» ya no es necesario crear números mayores ni menores para los archivos especiales (archivos dispositivos), el comando «mknod» sigue existiendo porque es necesario seguir manteniendo compatibilidad con todo el sistema preexistente.
Los números mayores y menores están reservados, de manera que si queremos crear un manejador de dispositivo debemos referirnos a la documentación del núcleo de Linux (en este caso). En la documentación se establece que el número mayor, *42, es para uso de muestra (demo)*, para usar en código de muestra, como mero dispositivo de «ejemplo» y no debería usarse para manejador de dispositivo (device driver) que se está distribuyendo. Entonces el comando:

```
ubuntu@ubuntu:~$ sudo mknod /dev/prueba c 42 1
```

Crea un dispositivo por caracteres llamado «prueba» cuyo número mayor es 42 y cuyo número menor es 1. Vea cómo quedó con:

```
ubuntu@ubuntu:~$ ls -l /dev/prueba
crw-r--r-- 1 root root 42, 1 oct 25 23:12 /dev/prueba
```

Para borrarlo, simplemente con el comando:

```
ubuntu@ubuntu:~$ sudo rm /dev/prueba
```

y análogamente podemos crear uno por bloques

```
ubuntu@ubuntu:~$ sudo mknod /dev/prueba b 42 1
```

Tuberías con nombre

En este punto, se recomienda repasar el capítulo de comunicación y sincronización de procesos. Recordará que podíamos comunicar procesos mediante tuberías (pipes) sin nombre o con nombre, entonces con el comando:

```
ubuntu@ubuntu:~$ sudo mknod /dev/prueba p
ubuntu@ubuntu:~$ ls -l /dev/prueba
prw-r--r-- 1 root root 0 oct 25 23:17 /dev/prueba
```

creará la tubería (FIFO) «prueba». Note la «p» indicando que el archivo es un «**p**ipe». Entonces si desde una consola colocamos el comando:

```
ubuntu@ubuntu:~$ sudo -i
ubuntu@ubuntu:~# echo "Hola, mundo" > /dev/prueba
```

y desde otra colocamos el comando :

```
ubuntu@ubuntu:~$ sudo -i
ubuntu@ubuntu:~# cat /dev/prueba
```

¿Qué pasó?

En rigor de verdad no necesita ser superusuario para crear una FIFO pero necesita permiso de escritura en el directorio en el que la cree, en este caso «/dev». Pero puede como usuario «ubuntu» crearla en su propio directorio

```
ubuntu@ubuntu:~$ mknod otraprueba p
ubuntu@ubuntu:~$ echo "Hola, otra vez" > otraprueba
```

y desde otra consola como usuario «ubuntu» estando en el mismo directorio:

```
ubuntu@ubuntu:~$ cat otraprueba
```

A estas tuberías se les llama «con nombre», diferenciándolas de las vistas previamente que son «sin nombre».

Dispositivos de red

El dispositivo de red se ha convertido en un elemento fundamental de cualquier sistema informático. Por ello, el sistema operativo ha evolucionado para proporcionar un tratamiento más exhaustivo y sofisticado de este dispositivo.
Se trata de un dispositivo que presenta unas características específicas que generalmente implican

un tratamiento diferente al de otros dispositivos. Así, muchos sistemas operativos no proporcionan una interfaz basada en archivos para el acceso a la red, aunque sí lo hacen para el resto de dispositivos. En Linux, por ejemplo, no hay archivos en «/dev» que representen a los dispositivos de red. Es decir, que *no veremos* algo como «/dev/eth» o «/dev/wifi».
Recientemente los nombres de dispositivos de red cambiaron la nomenclatura siguiendo una convención que utiliza varias características del bus y puerto donde esa interfaz se encuentra conectada. De esta manera guarda mucha más relación con su ubicación física. Dada su naturaleza, para estos dispositivos siempre es recomendable poder identificarlo inequívocamente.
Para más información relacionada a esta nueva forma de nomenclar dispositivos puede consultar aquí: Predictable Network Interface Names