El puerto USB: mas que un puerto serie con energía.

Lisandro Damián Nicanor Pérez Meyer

CIC-LMNE-GISEE - Universidad Nacional del Sur
http://www.gisee.uns.edu.ar/
Bahía Blanca - Argentina
–
The Debian Project
http://www.debian.org/

9 de julio de 2012

Temario

- Motivación.
- ¿Qué es el puerto USB?
- Conceptos importantes.
- Objetivos de diseño.
- Conectores / parte eléctrica.
- Versiones y velocidades.
- Limitaciones.
- Comunicación en el bus.
- Descriptores.
- Clases estándares de dispositivos.
- Desarrollando software para el host.
  - Driver del kernel o driver en espacio de usuario.

Comisión de Investigaciones Científicas

Motivación

Aprender sobre el puerto USB en general, con énfasis en los dispositivos.

- Conocer es perder el miedo.
- No depender siempre del puerto serie.
- Dar a conocer las ventajas de desarrollar “dispositivos multiplataforma”.

Comisión de Investigaciones Científicas

¿Que es el puerto USB?

Un estandar manejado por el “USB Implementers Forum, Inc.” en `http://usb.org/`.
El estandar incluye:

- Cableado.
- Conectores.
- Energizado.
- Señalización.
- Protocolo de comunicaciones.

Conceptos importantes.

El host:

- Chipsets de PC.
- System-onChip (SoC).
- Microcontroladores “grandes”.

Conceptos importantes.

El host:

- Chipsets de PC.
- System-onChip (SoC).
- Microcontroladores “grandes”.

El dispositivo:

- Todo lo que uno pueda conectar.
- Alimentado por el bus o de forma propia.

Conceptos importantes.

El host:

- Chipsets de PC.
- System-onChip (SoC).
- Microcontroladores “grandes”.

El dispositivo:

- Todo lo que uno pueda conectar.
- Alimentado por el bus o de forma propia.

Existe hardware que puede ser host y device al mismo tiempo. Por ejemplo, osciloscopios digitales que permiten que se conecte a su puerto host un pendrive y que permiten que una PC tome sus datos a través de su puerto device.

Conceptos importantes.

El host:

- Chipsets de PC.
- System-onChip (SoC).
- Microcontroladores “grandes”.

El dispositivo:

- Todo lo que uno pueda conectar.
- Alimentado por el bus o de forma propia.

Existe hardware que puede ser host y device al mismo tiempo. Por ejemplo, osciloscopios digitales que permiten que se conecte a su puerto host un pendrive y que permiten que una PC tome sus datos a través de su puerto device.
Los dispositivos se describen a sí mismos.

- Descriptores.
  Configuraciones, interfaces y endpoints.

Conceptos importantes.

El host:

- Chipsets de PC.
- System-onChip (SoC).
- Microcontroladores “grandes”.

El dispositivo:

- Todo lo que uno pueda conectar.
- Alimentado por el bus o de forma propia.

Existe hardware que puede ser host y device al mismo tiempo. Por ejemplo, osciloscopios digitales que permiten que se conecte a su puerto host un pendrive y que permiten que una PC tome sus datos a través de su puerto device.
Los dispositivos se describen a sí mismos.

- Descriptores.
  Configuraciones, interfaces y endpoints.

El hub: un divisor (splitter).

Objetivos de diseño.

- Conectar periféricos a una PC.
- Un alaternativa a los puertos antiguos (que los haga desaparecer).
- Plug-and-Play.
  - Mechanical connector keying (o como evitar enchufar las cosas mal).
- Los dispositivos deben ser simlpes y baratos de fabricar.
  - Fuertemente basados en el host.
  - La comunicación es manejada enteramente por el host $\rightarrow$ polling de los dispositivos.

Se calcula que existen 8 billones de puertos host, +2 billones por año.

Conectores / parte eléctrica.

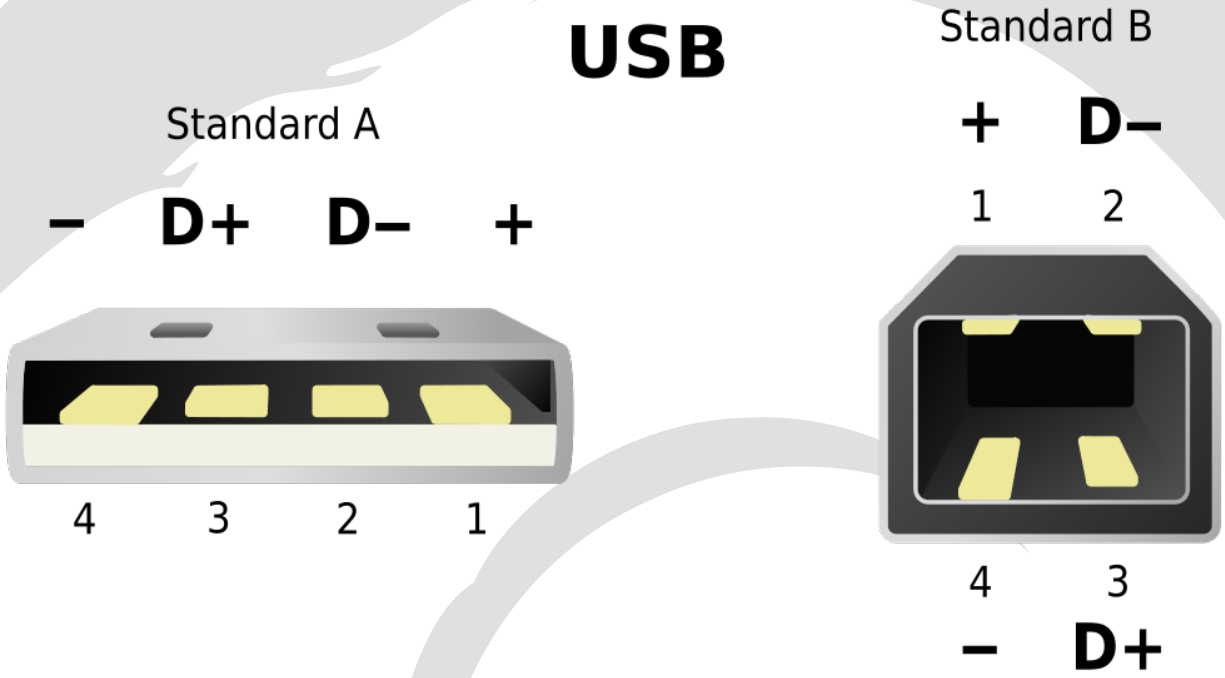

Figura: Conectores USB estandar. Fuente: Simon Eugster/Wikipedia. Bajo licencia CC-BY-SA 3.0 unported (entre otras).

Cuatro cables: +5V, D+, D- y GND. Señalización diferencial.
Y algunos detalles mas de señalización de bajo nivel.

Versiones y velocidades.

1996.1998: 1.0-1.1

- Low speed, 1.5 Mbps.
- Full speed, 12 mbps.
- Host controller interface: UHCI y OHCI.

Versiones y velocidades.

1996.1998: 1.0-1.1

- Low speed, 1.5 Mbps.
- Full speed, 12 mbps.
- Host controller interface: UHCI y OHCI.

2000: 2.0

- High speed, 480 Mbps.
- Host controller interface: EHCI.
- 2.0 on-the-go: conexiones sin host.

Versiones y velocidades.

1996.1998: 1.0-1.1

- Low speed, 1.5 Mbps.
- Full speed, 12 mbps.
- Host controller interface: UHCI y OHCI.

2000: 2.0

- High speed, 480 Mbps.
- Host controller interface: EHCI.
- 2.0 on-the-go: conexiones sin host.

2008: 3.0

- SuperSpeed, 5Gbps.
- Host controller interface: XHCI.

Limitaciones.

127 dispositivos por bus, incluyendo hubs.

Limitaciones.

127 dispositivos por bus, incluyendo hubs.
Siete niveles de profundidad como máximo.

- Root hub = 1.
- Dispositivo = 1.
- $\rightarrow$ Máximo de 5 hubs entre el host y el device.

Limitaciones.

127 dispositivos por bus, incluyendo hubs.
Siete niveles de profundidad como máximo.

- Root hub = 1.
- Dispositivo = 1.
- $\rightarrow$ Máximo de 5 hubs entre el host y el device.

Cables de hasta 5 metros

Limitaciones.

127 dispositivos por bus, incluyendo hubs.
Siete niveles de profundidad como máximo.

- Root hub = 1.
- Dispositivo = 1.
- $\rightarrow$ Máximo de 5 hubs entre el host y el device.

Cables de hasta 5 metros $\rightarrow$ Distancia entre host y device $<$ 30 metros.

Limitaciones.

127 dispositivos por bus, incluyendo hubs.
Siete niveles de profundidad como máximo.

- Root hub = 1.
- Dispositivo = 1.
- $\rightarrow$ Máximo de 5 hubs entre el host y el device.

Cables de hasta 5 metros $\rightarrow$ Distancia entre host y device $<$ 30 metros.
Corriente máxima que puede consumir un dispositivo:

- 500 mA conectado directamente.
- 100 mA si está conectado a un hub.
- 500 μA cuando se encuentra suspendido.

Comisión de Investigaciones Científicas

Comunicación en el bus.

Es un bus serial de un canal $\rightarrow$ una comunicación en un sentido por vez.

Comunicación en el bus.

Es un bus serial de un canal $\rightarrow$ una comunicación en un sentido por vez.
Frames:

- Un frame dura 1 ms.
- Un microframe dura 125 μs (solo en high-speed).

Comunicación en el bus.

Es un bus serial de un canal $\rightarrow$ una comunicación en un sentido por vez.
Frames:

- Un frame dura 1 ms.
- Un microframe dura 125 μs (solo en high-speed).

Transferencias:

- Compuesta de varios tipos de paquetes.
- Sync/setup, datos, handshake (ACK/NAK/STALL).

Tipos de transferencias.

Transferencias de control

- “Best Effort”, max 10 %/20 % (full-low/high speed).
- Datos y estado, paquetes de 64 bytes máximo.

Transferencias de interrupción

- Periódicas, cada 1 ms máximo, 80 %/90 % por frame (high/full speed).
- Datos unidireccionales solamente, máximo 64 o 1024 bytes (full/high speed).

Transferencias en masa (bulk)

- Entrega garantizada pero sin ancho de banda o latencia garantizado.
- Paquetes de 64/512 bytes (full/high speed) máximo.
- **No se permite** en dispositivos de baja velocidad.

Transferencias isocrónicas

- Latencia garantizada a velocidad constante, no se reintentan.
- Paquetes de 1023/1024 (full/high speed) bytes máximo.

Comisión de Investigaciones Científicas

Endpoints

Fuente o destino de todas las transferencias.

- Tiene un número (dirección) y sentido.
- 0–15, IN **ó (xor)** OUT.

Una comunicación bidireccional necesita de dos endpoints.
El endpoint 0 es el endpoint "por defecto": tiene que ser un endpoint de control.
La funcionalidad de los otros endpoints son determinados por el desarrollador del dispositivo.
Un conjunto de endpoints se agrupan y describen en una **Interface**.

Descriptores

Descriptor de dispositivo

Descriptores

Descriptor de dispositivo

Descriptor de configuración

Descriptores

- Descriptor de dispositivo
  - Descriptor de configuración
    - Descriptor de interfaz

Descriptores

- Descriptor de dispositivo
  - Descriptor de configuración
    - Descriptor de interfaz
      - Descriptor de endpoint

Descriptores

Descriptor de dispositivo
  Descriptor de configuración
    Descriptor de interfaz
      Descriptor de endpoint

Múltiples interfaces en cada configuración.

- Concurrentes e independientes.

Las configuraciones son mutuamente exclusivas: una sola al mismo tiempo.

- Pero se pueden usar varias interfaces en una configuración.
- Se indica la máxima corriente que se puede consumir.

Descriptor de dispositivo

- Versión y clase del dispositivo.
- ID numérico del vendor y del producto.
- Cadenas de identificación: (nombre del vendor/producto, número de serie, ...)

Descriptores

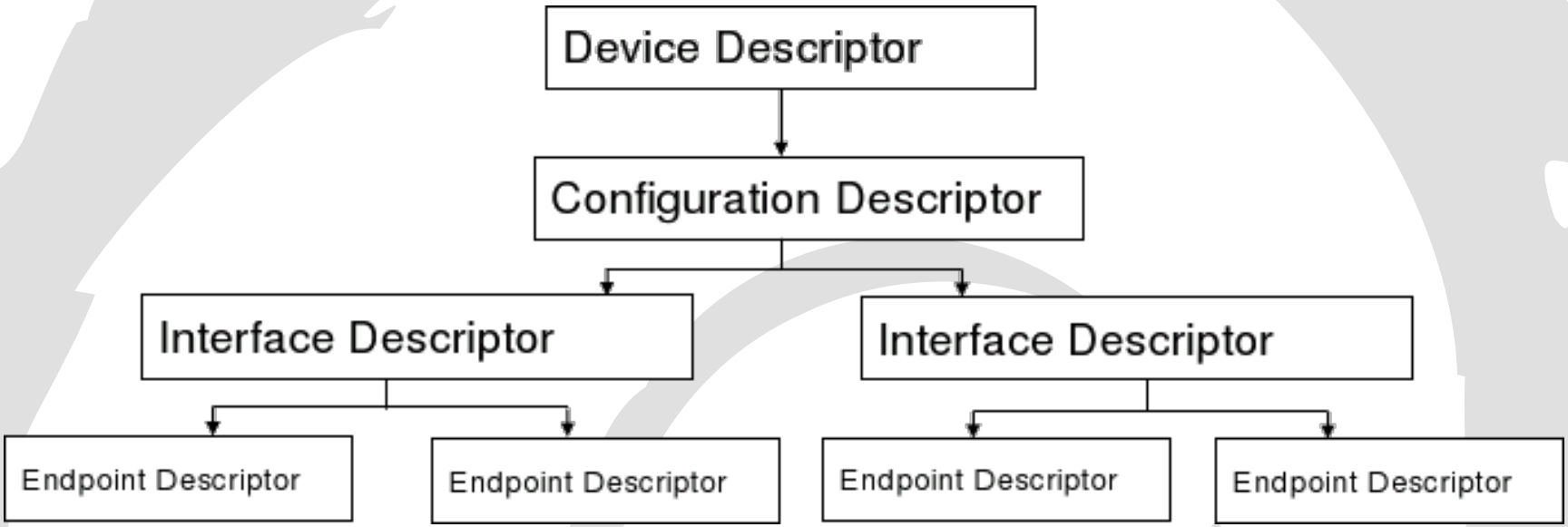

Figura: Arquitectura lógica de un dispositivo USB.

Clases

Algunas interfaces comunes han sido estandarizadas:

- HID: Human Interaction Device.
  Teclado, mouse, joystick,...
- MSC: Mass Storage Class.
  Discos rígidos, memorias flash,...
- CDC: Communications Device Class.
  Puertos serie, ethernet,...
- Audio.
  Altavoces, micrófono.
- DFU: Device Firmware Upgrade.
- TMC: Test and Measurement Class.
- Y muchos mas...

Un solo driver puede manejar muchos dispositivos similares =)

Comisión de Investigaciones Científicas

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).
- Combinación óptima de features.

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).
- Combinación óptima de features.
- Hay que pensar bien al escribir el driver...

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).
- Combinación óptima de features.
- Hay que pensar bien al escribir el driver...
  - "Desconectar" un driver existente puede ser complicado o molesto.
  - Puede requerir permisos de admin/superusurio.
  - Puede requerir reiniciar el sistema.

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).
- Combinación óptima de features.
- Hay que pensar bien al escribir el driver...
  - "Desconectar" un driver existente puede ser complicado o molesto.
  - Puede requerir permisos de admin/superusurio.
  - Puede requerir reiniciar el sistema.
  - El driver puede ser muy específico de la plataforma...

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).
- Combinación óptima de features.
- Hay que pensar bien al escribir el driver...
  - "Desconectar" un driver existente puede ser complicado o molesto.
  - Puede requerir permisos de admin/superusurio.
  - Puede requerir reiniciar el sistema.
  - El driver puede ser muy específico de la plataforma... o no.

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).
- Combinación óptima de features.
- Hay que pensar bien al escribir el driver...
  - "Desconectar" un driver existente puede ser complicado o molesto.
  - Puede requerir permisos de admin/superusurio.
  - Puede requerir reiniciar el sistema.
  - El driver puede ser muy específico de la plataforma... o no.

Generalmente se puede acceder al dispositivo sin utilizar un driver...

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).
- Combinación óptima de features.
- Hay que pensar bien al escribir el driver...
  - "Desconectar" un driver existente puede ser complicado o molesto.
  - Puede requerir permisos de admin/superusurio.
  - Puede requerir reiniciar el sistema.
  - El driver puede ser muy específico de la plataforma... o no.

Generalmente se puede acceder al dispositivo sin utilizar un driver...
excepto en Windows. Pero se puede usar WinUSB.sys o...

Interfaces específicas de los vendors

"Vendor specific" es normalmente una cosa mala...
Para el caso de USB, puede ser muy buena.

- No hay restricción de endpoints (aparte del EP 0).
- Combinación óptima de features.
- Hay que pensar bien al escribir el driver...
  - "Desconectar" un driver existente puede ser complicado o molesto.
  - Puede requerir permisos de admin/superusurio.
  - Puede requerir reiniciar el sistema.
  - El driver puede ser muy específico de la plataforma... o no.

Generalmente se puede acceder al dispositivo sin utilizar un driver...
excepto en Windows. Pero se puede usar WinUSB.sys o...

libusb

¿Dentro o fuera del kernel?

Cuando desarrollo un dispositivo USB puedo llegar a necesitar desarrollar un software para manejarlo: driver o biblioteca.
La pregunta es ¿que opción elijo?
Debo considerar:

- ¿Ya hay soporte previo? (“El osciloscopio imager”).
- ¿Afecta al rendimiento del sistema? (Placa de red).

La mayoría de las veces la información termina en el usuario.

libusb

O como hacer código una vez y que ande en todos lados.

- Soporta Windows, Mac, Linux, Solaris y FreeBSD.
- API sincrónica y asincrónica.
- Thread-safe a partir de la versión 1.0.

Question mark in Esbjerg.
CC-BY-SA por alexanderdrachmann.
http://www.flickr.com/photos/drachmann/

Comisión de Investigaciones Científicas

Thank you.

http://www.flickr.com/photos/psd/

Comisión de Investigaciones Científicas

Basado en la charla “USB and libusb 27C3. So much more than a serial port with power” por Peter Stuge <peter@stuge.se>, bajo licencia CC-BY-SA 3.0.

El look Debian lo tomé de `http://rkd.zgib.net/wiki/DebianBeamerThemes`.

Los logos de Debian se encuentran bajo la licencia Expat/MIT.

El logo de la CIC es ©Comisión de Investigaciones Científicas de la provincia de Buenos Aires.