Plan 9's USB

Francisco J. Ballesteros

Laboratorio de Sistemas

nemo@lsub.org

Roadmap

- Introduction to USB
- Software organization
- Kernel drivers
- Enumeration and Hot-plug
- Device drivers
- File system

Acknowledgements

Before I run out of time...

- Anonymous authors of #U, usbd, ...
- The secret Plan 9 Society
- 9fans for fixes and ideas
- Geoff from breaking down my software

Anti-acknowledgements

Before I say anything else..

- To the creators of the standard
- USB2[650] + [UEO]HCI + Class specs
  - Yet most devices are not standard
  - And some controllers don't interrupt

USB

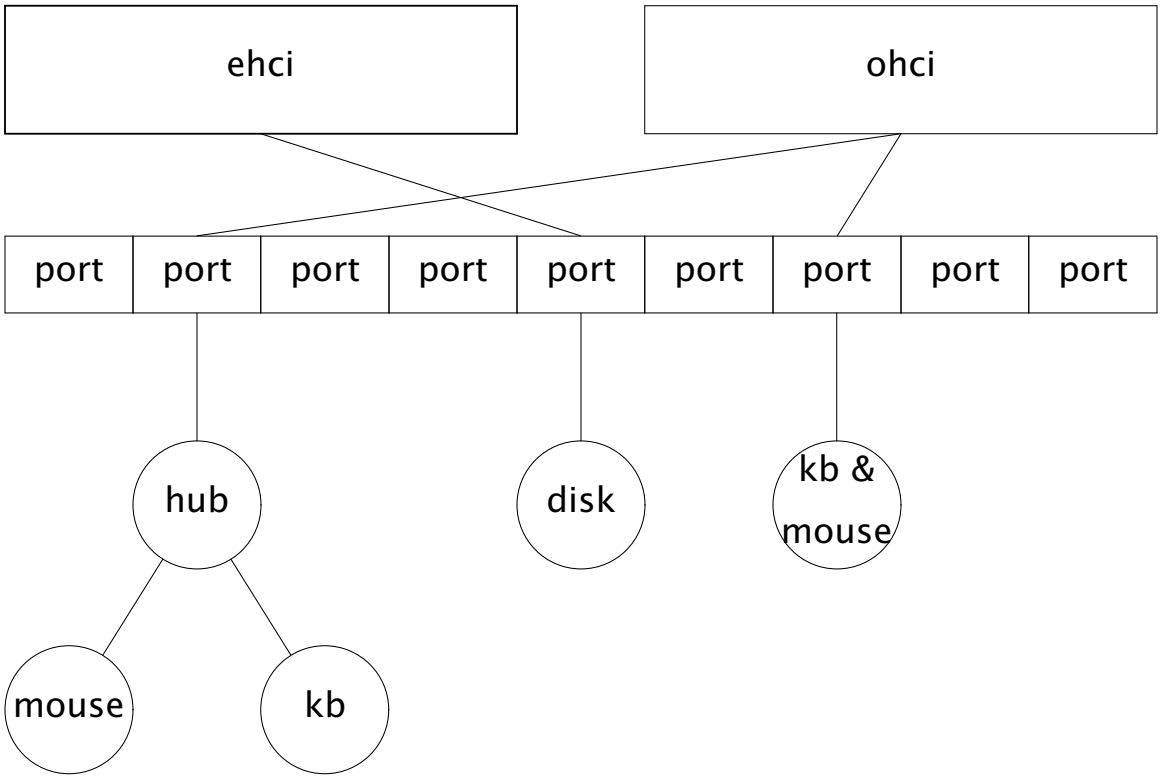
Overview

- USB 1: Low-speed (1.5M/s), Full-speed (12M/s)
- USB 2: High-speed (12M/s)
- USB 3: Super-speed (5G/s)
- Polled bus (tree) ⇒ Hosts vs devices.
- Devices ⇒ (Class, subclass, protocol)

Host controllers

- USB 1:
  - ohci
  - uhci
- USB 2:
  - ehci
- USB 3:
  - xhci (couldn't get)

Example 2.0 bus



Devices

- One or more functions.
- Endpoints
  - Addressable entities
  - In, Out, In-Out
  - With particular transfer type
- Grouped in interfaces
  - Each with a CSP

Enumeration Configuration

- Address ⇒(device, endpoint)
- On attach ⇒
  - Address 0 (config)
  - Setup endpoint (address 0)
- Software assigns unique address
- Hubs are transparent

Transfer types

- Or endpoint types:
  - Control
  - Bulk
  - Interrupt
  - Isochronous
- May be high, low, full speed.

Control transfers

- RPCs
- In or Out
- Stages:
  - Setup
  - Data
  - Status

Other transfers

- Bulk
  - Sustained In or Out xfer.
- Interrupt
  - Not an interrupt
- Iso
  - Timely

Data transfer

- Packet types (Setup, Data0, Data1, ...)
- 1-bit ack (toggle)
- Exceptions for control transfers
- Error signaling
  - In-band
  - Not standard (e.g., per-controller)
  - Ambiguous (I/O? bad request? ...)

Controller transfers

- UCHI, OHCI
- EHCI + [UHCI, OHCI]
- IO + Shared memory + Interrupts (if lucky)
- There is no root hub!
- Tranfer descriptors
  - per ctlr, per type, per speed.

Configuration

- Devices are self-describing
- Binary encoded descriptors
  - Device (class, packet size, ...)
  - Class-specific
  - Device-specific

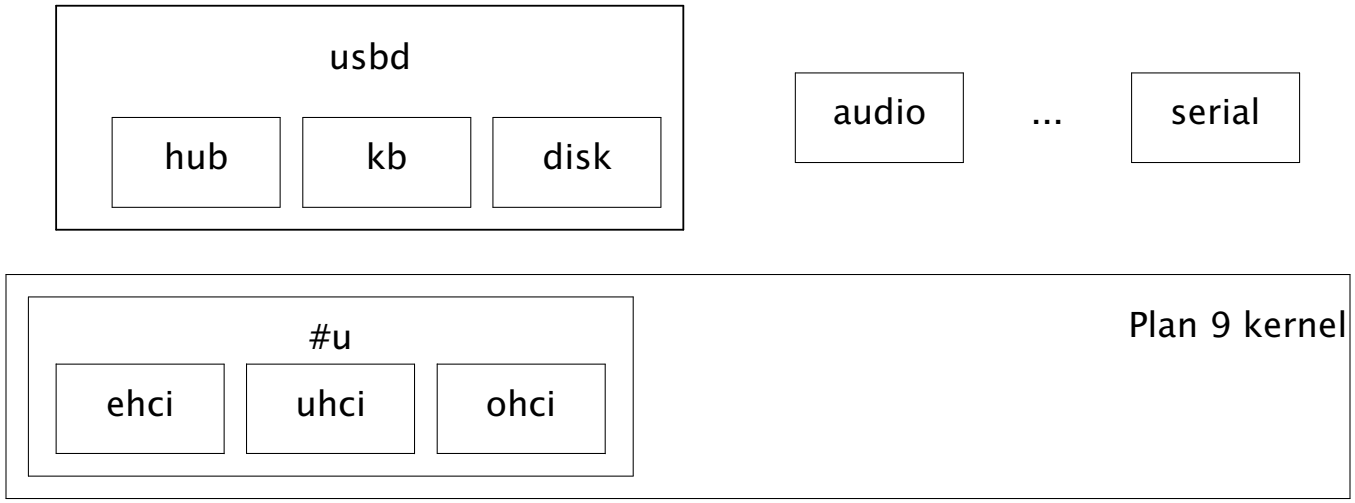
Summary

- Fake root hub; many transfer types.
- What's a device? ⇒ not clear.
- Which error? ⇒ not clear
- Devices do as they please
  - Most people do what Wind*ws does
  - If you fix a device you may break another

System software

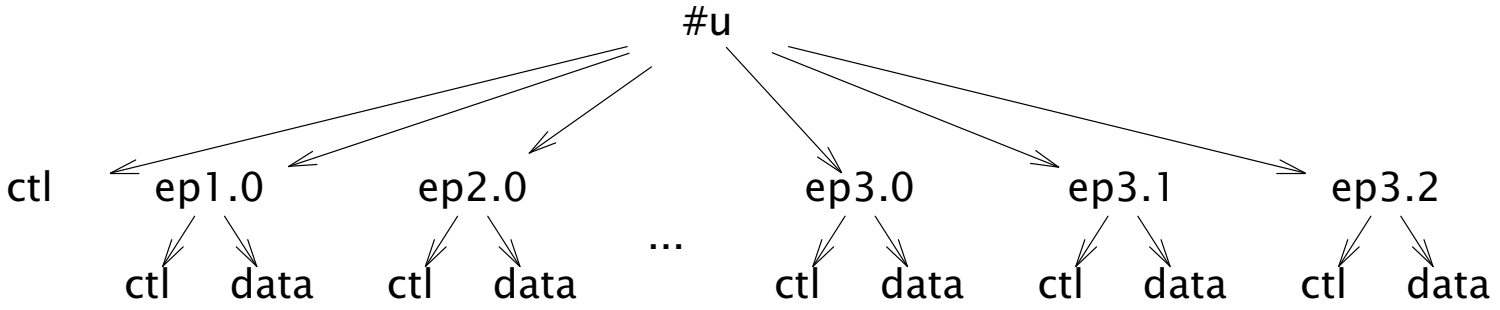
- Evolution of previous USB software
- Kernel ⇒ I/O
- Daemon ⇒ Enumeration Configuration
- User programs ⇒ Device drivers
- User library ⇒ Device drivers

System software



User interface

Not standard, but easier to understand.



Example

```
cat /dev/usb/ep3.0/ctl

enabled control rw speed full maxpkt 64
ival 0 samplesz 0 hz 0 hub 1 port 3 busy

storage csp 0x500608 vid 0x951 did 0x1613
Kingston 'DT 101 II '
```

Kernel drivers

- devusb
  - portable interface
  - ctlr independent endpoints
  - file system
- usb[ueo]hci
  - honor devusb endpoints
  - Actual I/O

Portable endpoint

```
struct Ep
{
        Ref;            /* one per fid (and per dev ep for ep0s) */

        /* const once inited. */
        int     idx;        /* index in global eps array */
        int     nb;         /* endpoint number in device */
        Hci*    hp;         /* HCI it belongs to */
        Udev*   dev;        /* device for the endpoint */
        Ep*     ep0;        /* control endpoint for its device */
        ...
```

Portable endpoint

```
/* configuration */
QLock;                  /* protect fields below */
char* name;             /* for ep file names at #u/ */
int     inuse;          /* endpoint is open */
int     mode;           /* OREAD, OWRITE, or ORDWR */
int     clrhalt;        /* true if halt was cleared on ep. */
int     debug;          /* per endpoint debug flag */
char* info;             /* for humans to read */
long  maxpkt;           /* maximum packet size */
int     ttype;          /* transfer type */
ulong         load;     /* in µs, for a transfer of maxpkt bytes */
void* aux;              /* for controller specific info */
int     rhrepl;         /* fake root hub replies */
int     toggle[2];      /* saved toggles (while ep is not in use) */
```

Portable endpoint

```
struct Ep
{
        ...
        long  pollival;         /* poll interval ([µ]frames; intr/iso) */
        long  hz;               /* poll frequency (iso) */
        long  samplesz;         /* sample size (iso) */
        int   ntds;             /* nb. of Tds per µframe */
};
```

USB device

```
struct Udev
{
        int     nb;             /* USB device number */
        int     state;          /* state for the device */
        int     ishub;          /* hubs can allocate devices */
        int     isroot;         /* is a root hub */
        int     speed;          /* Full/Low/High/No -speed */
        int     hub;            /* dev number for the parent hub */
        int     port;           /* port number in the parent hub */
        Ep*     eps[Ndeveps];   /* end points for this device (cached) */
};
```

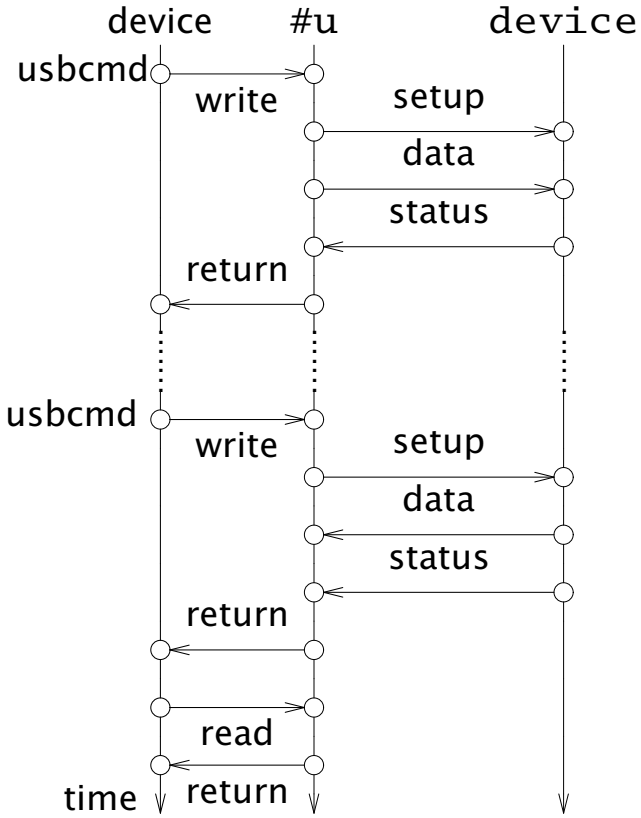
Controller interface

```
struct Hciimpl
{
        void  *aux;                             /* for controller info */
        void  (*init)(Hci*);                    /* init. controller */
        void  (*dump)(Hci*);                    /* debug */
        void  (*interrupt)(Ureg*, void*);       /* service interrupt */
        void  (*epopen)(Ep*);                   /* prepare ep. for I/O */
        void  (*epclose)(Ep*);                  /* terminate I/O on ep. */
        long  (*epread)(Ep*,void*,long);        /* transmit data for ep */
        long  (*epwrite)(Ep*,void*,long);       /* receive data for ep */
        char* (*seprintep)(char*,char*,Ep*);        /* debug */
        int   (*portenable)(Hci*, int, int);    /* enable/disable port */
        int   (*portreset)(Hci*, int, int);     /* set/clear port reset */
        int   (*portstatus)(Hci*, int);         /* get port status */
        void  (*debug)(Hci*, int);              /* set/clear debug flag */
};
```

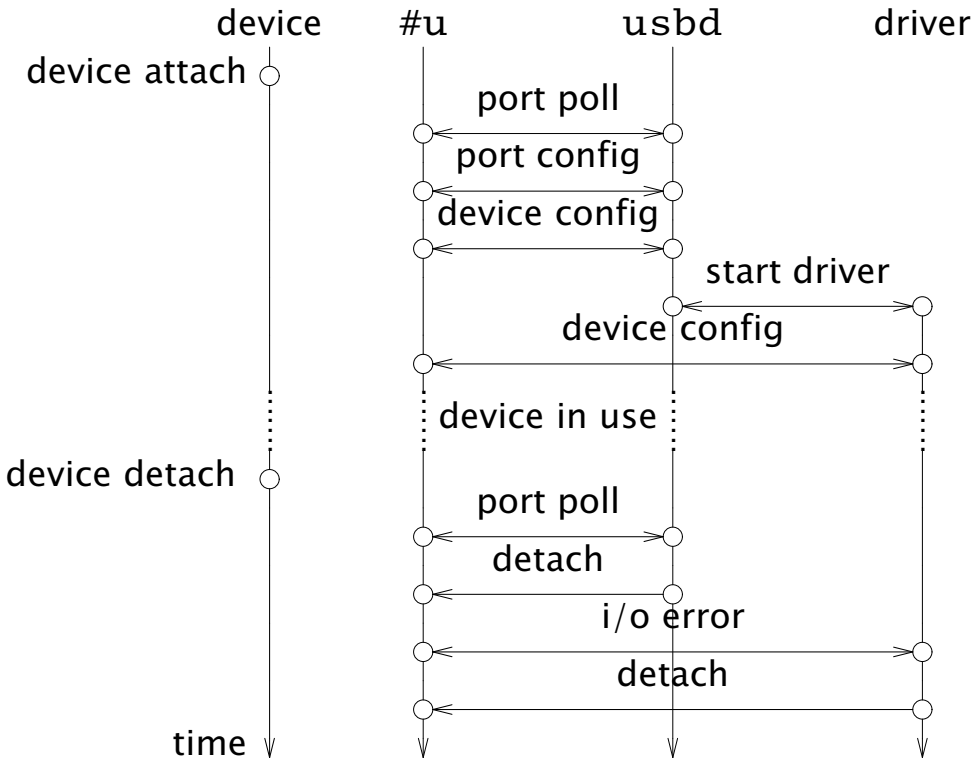
usb[ueo]hci epread/epwrite

- Endpoint I/O
- Endpoint open only while in use
  - (devusb keeps cfg. between opens)
- Fake root-hub port poll
  - But not a full hub emulation
- Control and Iso are timed out

Control transfer

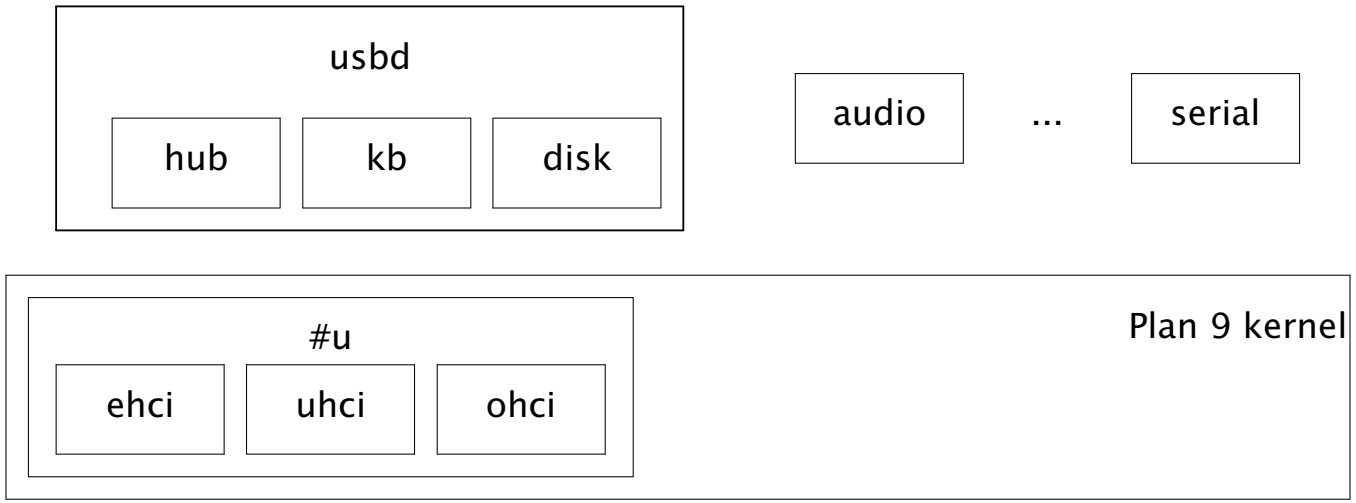


Enumeration Hot plug



Device drivers

- Not to be confussed with devusb/usb[ueo]hci



Device drivers

- usbd: ⇒
  - enumeration
  - initial configuration
- driver ⇒
  - device specific configuration
  - device I/O (unless named endpoint)
- + support library

USB device (endpoint)

```
struct Dev
{
        Ref;
        char* dir;              /* path for the endpoint dir */
        int     id;             /* usb id for device or ep. number */
        int     dfd;            /* descriptor for the data file */
        int     cfd;            /* descriptor for the control file */
        int     maxpkt;         /* cached from usb description */
        Ref     nerrs;          /* number of errors in requests */
        Usbdev*     usb;        /* USB description */
        void* aux;              /* for the device driver */
        void   (*free)(void*);  /* idem. to release aux */
};
```

USB device configuration

```
struct Usbdev
{
        ulong       csp;        /* USB class/subclass/proto */
        int     vid;            /* vendor id */
        int     did;            /* product (device) id */
        char* vendor, *product, *serial;
        int     vsid, psid, ssid;
        int     class;                      /* from descriptor */
        int     nconf;                      /* from descriptor */
        Conf*       conf[Nconf];            /* configurations */
        Ep*         ep[Nep];                /* all endpoints in device */
        Desc*       ddesc[Nddesc];          /* (raw) device specific descriptors */
};
```

USB device configuration

```
struct Ep
{
        uchar       addr;           /* endpt address, 0-15 (|0x80 if Ein) */
        uchar       dir;            /* direction, Ein/Eout */
        uchar       type;           /* Econtrol, Eiso, Ebulk, Eintr */
        uchar       isotype;        /* Eunknown, Easync, Eadapt, Esync */
        int     id;
        int     maxpkt;         /* max. packet size */
        int     ntds;           /* nb. of Tds per µframe */
        Conf*       conf;       /* the endpoint belongs to */
        Iface*iface;            /* the endpoint belongs to */
};
```

Modules?

- Code kept in library
- separate main (replaces usbd)
  - linked against library
- linked into usbd
- linked as a separate program

Usbd configuration

```
embed
        kb      csp=0x010103 csp=0x020103    args=
        disk    class=storage     args=
auto
        ether class=comms     args=
```

Example

```
int
kbmain(Dev *d, int argc, char* argv[])
{
        ARGBEGIN{
        ...
        }ARGEND;
        if(ok)
                start(d);
        else
                closedev(d);
        return 0;
}
```

Example

```
void
threadmain(int argc, char **argv)
{
        ARGBEGIN{
                as = seprint(as, ae, " -k");
                ...
        }ARGEND;
        startdevs(args, argv, argc, matchdevcsp, csps, kbmain);
        threadexits(nil);
}
```

Device file systems

- Most drivers have a file system
- FS interface is almost static
- FS problems must not abort
  - embedded devices
- Several FSs must be combined
  - Stackable (usb / usbdir / devfs)

Device file system

```
struct Usbfs
{
        char  name[Namesz];
        uvlong      qid;
        Dev*  dev;
        void* aux;
        int     (*walk)(Usbfs *fs, Fid *f, char *name);
        void  (*clone)(Usbfs *fs, Fid *of, Fid *nf);
        void  (*clunk)(Usbfs *fs, Fid *f);
        int     (*open)(Usbfs *fs, Fid *f, int mode);
        long  (*read)(Usbfs *fs, Fid *f, void *data, long count, vlong offset);
        long  (*write)(Usbfs *fs, Fid*f, void *data, long count, vlong offset);
        int     (*stat)(Usbfs *fs, Qid q, Dir *d);
        void  (*end)(Usbfs *fs);
};
```

Device file system

- Concurrent requests
  - Open, read, write
- Others automatic
  - Walk, clone, clunk, stat, end
- Per fs name.
- Decorated Qids

Implementation

1470 /sys/src/9/pc/devusb.c

705 /sys/src/9/pc/devusbcons.c

194 /sys/src/9/pc/usb.h

3340 /sys/src/9/pc/usbehci.c

147 /sys/src/9/pc/usbehci.h

2545 /sys/src/9/pc/usbohci.c

2301 /sys/src/9/pc/usbuhci.c

10702 total

Implementation

```
2448 /sys/src/cmd/usb/audio/...
2710 /sys/src/cmd/usb/disk/...
2924 /sys/src/cmd/usb/ether/...
724  /sys/src/cmd/usb/kb/...
2617 /sys/src/cmd/usb/lib/...
1272 /sys/src/cmd/usb/usbd/...
...
14240 total
```

Problems

- Iso-read on OHCI
- Too many ilocks
- 64-bit cleanup
- Profiling
- More devices
- USB 3.0