Usb serial design and experience in Plan 9

Gorka Guardiola
Francisco J. Ballesteros
Enrique Soriano Salvador

(paurea, nemo, esoriano)@lsub.org
Lsub, Universidad Rey Juan Carlos

No serial on new computers

- Most computers don't come with serial
- Specially laptops, many times servers

No problem ¿right?

- Who needs serial anyways?
- Use the network or cec or...
- But...

Need for serial

- Porting an O.S. is already hard enough
- From reboot/loop + led to interrupts and mmu working
- Couple of registers at most
- No needs for interrupts (polling mode)
- Normally comes preinitialized

Alternatives

- Usb debug plug: tried it
  - only some Ehci
  - leaves you without usb port
  - looses bytes (maybe a fixed bug?)
- Cec: need interrupts + ethernet driver +... at that point I am done porting
- Not real alternative as it is, which is why...

What vendors do

- Have secret serial ports (may need soldering)
- Include Serial to USB chip inside

Kinds of usb/serial

- Usb debug cable
- Prolific,
- Ftdi, inside the sheeva
- Cdc-acm, inside phones

Usb debug cable

- Driver in the kernel for the debugged machine
- Driver in usb/serial for console/debugger
- Only works for EHCI, cannot use the port
- Mostly working, looses bytes (probably a solved bug)
- Careful while using, (asymmetric, one side powered)

Prolific

- First one we supported (pl2303)
- Inside most cheap usb/serial cables and some old dock stations
- Driver inside usb/serial
- Seems to work, tried with two cables
- Ported the gumstix with it
- No extra modem flow control supported, just basics

Ftdi

- Inside the sheeva and most new usb devices
- Many serial devices had an ftdi added
- Driver in usb/serial
- Works well except for some unresolved issue (Geoff?)
- Multiport supported, but untried (I don't have any device, Jeff?)

Cdc-acm

- Usb standard for serial communications
- I had not seen any until recently
- Inside phones and 3G/umts modems
- It was unsupported at the time of writing now
- Now Peter Bosch and Jan Sacha seem to have one working in usb/serial (I am guessing)

Architecture

- main.c standard USB not embedded main
- serial.[ch] common infrastructure, detection, filesystem
- ftdi.[ch] for FTDI support
- prolific.[ch] Prolific support
- ucons.[ch] EHCI debug cable

Architecture

- Detection
- Configuration
- Serial ops
- Read/write Wait4data/wait4write

```
struct Serialops {
        int    (*seteps)(Serialport*);
        int    (*init)(Serialport*);
        int    (*getparam)(Serialport*);
        int    (*setparam)(Serialport*);
        int    (*clearpipes)(Serialport*);
        int    (*reset)(Serial*);
        int    (*sendlines)(Serialport*);
        int    (*modemctl)(Serialport*, int);
        int    (*setbreak)(Serialport*, int);
        int    (*readstatus)(Serialport*);
        int    (*wait4data)(Serialport*, uchar *, int);
        int    (*wait4write)(Serialport*, uchar *, int);
}
```

Procs/concurrency

- Operations in the fs proc (one per operation in USB)
- Qlock per device (not per port)
- Each device may create extra procs for reading interrupt
- In FTDI, data+metadata are mixed and periodic, and we need to read as much as we can
- RPC over channels for reading (take away headers, read lines)

Cleaning up

- There can be errors on different procs, sender/receivers
- Need to try recovering from port to device
- Need to take care of in-transit data
- Added chanclose and chanclosing to thread library
- Similar to Go (but not quite)

Close

- Chanclose closes a channel, unblocks all operations
- After chanclose it is all non-blocking
- When all the channels on an Alt are closed, error
- Chanclosing returns:
  - -1 if close was not called
  - Number of items on the channel otherwise
- Complex to get right:
  - Interrupts
  - Alt

Close

- Common pattern of use is
  - Sender closes, does not touch the channel
  - Receiver gets the error, check chanclosing if 0 frees
- More than one sender/receiver, use reference counting

Naming

- Added a U (eiaU0)
- Needed a root, because the USB fs library
- eiaU0/eiaU0.3ctl eiaU0/eiaU0.3
- Sort of redundant but need a unique name in /dev
- Fused ctl and status
- Mount in /n if run by hand, /dev if embedded
- Do not want to leave /dev broken

Performance problems

- Serial devices are faster than we thought (specially FTDI)
- Reading smaller packets does not cut it (too many context switches)
- We were loosing bytes
- Read a big chunk and then break in packets

Docs/specs

- Specs if any (if overwhelming, leave for last)
  - http://www.usb.org/developers/devclass_docs/usbcdc11.pdf
  - Vendors (various results)
- FreeBSD, cleaner, know better what they are doing
  - http://fxr.watson.org/fxr/source/dev/usb/serial/umodem.c
- Linux, hardware bugs, more complete
  - http://tomoyo.sourceforge.jp/cgi-bin/lxr/source/drivers/usb/class/cdc-acm.c
- Sniffing Linux using wireshark
- Any embedded library you can find

Conclusions

- Simple, clean usb-serial implementation
- Line disciple not mixed with driver and hand USB transactions (USB infrastructure + Plan 9 approach)
- Easy to debug and add new drivers

Future work

- More testing
  - Multiport is untested (we don't have devices)
  - Not all features have been throughly tested
- Cdc-acm soon: Jan Sacha has got it working
- JTAG