LYNXLINE

Professional Software Development Services

Home | Blogs | Projects | About | Services | Career | Contact Us

Lab 22, Usb keyboard

By ADMIN | *Published:* DECEMBER 26, 2013

In previous lab we approached very close to get working Usb devices. In this lab we will try to get Usb keyboard working:

First some preparations: as we need automatic drivers loading, we overwrite existing Usb codes in Inferno repository with our files affecting **appl/cmd/usb** and **appl/lib/usb**. Our Usbd which we have written as standalone limbo file, we will split into header **usb.m**, library with API, **usb.b** and functionality related to hubs and ports logic and driver loading we leave in **usbd.b**.

It is really cool with Limbo that we can dynamically load usb driver and initialize it directly with objects of usb *Dev* and then spawn the driver thread:

```
pp.dev.mod = load UsbDriver path;
if (pp.dev.mod == nil)
    sys->fprint(stderr, "usbd: failed to load %s\n", path);
else {
    rv := pp.dev.mod->init(usb, pp.dev);
    if (rv == -11) {
        sys->fprint(stderr, "usbd: %s: reenumerate\n", path);
        pp.dev.mod = nil;
        #reenumerate = 1;
    }
```

Categories
- Blog
- Boost
- C++
- Cryptography
- Embedding
- Hybrids
- Inferno OS
- MacAppStore
- Misc
- Models
- Projects
- PyQt
- PySide
- Qt
- QtSpeech
- Raspberry Pi
- Research
- Ru
- TogMeg
- Trac
- TTS
- Tutorial
- Undo
- Usb
- Web

```
    else if (rv < 0) {
        sys->fprint(stderr, "usbd: %s:init failed\n", path);
        pp.dev.mod = nil;
    }
    else if (verbose)
        sys->fprint(stderr, "%s running\n", path);
}
```

The *UsbDriver* we will define as:

```
UsbDriver: module
{
    init: fn(usb: Usb, dev: ref Usb->Dev): int;
    shutdown: fn();
};
```

Also add a member to *Dev* adt:

```
mod: UsbDriver;
```

The code for automatic driver loading is demonstrated above. It is choosing the driver dis file based on database of Usb drivers: **/lib/usbdb**. The code related to the database loading etc we will borrow from old Usb framework.

To support the keyboard I took some search through internet and amazingly found the project inferno-bls (https://code.google.com/p/inferno-bls) which is probably done almost same work of porting Plan9 Usbd to Inferno, but early version of about 2010. Because it looks compatible I attempted to use the keyboard driver from that project, but no luck, it goes into deadlock with repeater. Still with disabled repeater it is almost working but hang without reason after few seconds. So, I gave up with the driver and just go to porting Plan9 driver for Usb keybord. C to Limbo porting very straightforward with simplifications of lot of places.

The difference is that **usbkbd** writes scan codes into *"#I/kbin"*, which is handled in Plan9 by **devkbin,kbd** kernel fileserver which converts scancodes into *kbdputc()* calls. By checking 9pi it seems to use the **kbd.c** from **omap** which is designed for quote: *simulated keyboard input for systems with none (except via uart or usb)*. It compiles-in into Inferno perfectly.

When I assembled everything I got the keyboard working!

(except one strange bug that I have to put some empty *fprint()* to stderr just before writing scancode to devkbin – some sync is needed for channels? I may investigate later, or just rewrite kbin into Limbo code, will see)

FILES:
usb.m
usb.b
usbd.b
usbkbd.b

This entry was posted in *Blog*, *Inferno OS*, *Raspberry Pi*, *Research*, *Usb*. Bookmark the *permalink*. *Post a comment* or leave a trackback: *Trackback URL*.

« *Lab 21, porting usbd, fixed in allocb, see usb in actions*

0 comments

★ 0

Start the discussion…

Newest Community My Disqus Share yshurik

Be the first to comment.

Subscribe Add Disqus to your site