LYNXLINE

Professional Software Development Services

Home Blogs Projects About Services Career Contact Us

Lab 19, keyboard through serial, fixes to get Ls

By ADMIN | Published: DECEMBER 1, 2013

In previous Lab we got a screen working, but still no way to send commands into Raspberry computer. My guess that for time being it just simple enough to make the keyboard input coming from serial cable, so then I can play with Sh on raspberry and do some simple experiments onboard until I will have the Usb subsystem working and it recognize keyboard connected to RPi.

To work with input from Plo11 we should have it in non-blocking way. To do so we just make some function like *plo11init()* and check the serial io port for waiting bytes with timer (using *addclockolink()*). Take some consulting with port/devuart.c and see that for 115200 it is enough to check it every 22 ms. You may see that the programming is really simple:

```
void
pl011init(void)
{
    /*
    * at 115200 baud, the 1024 char buffer takes 56 ms to process,
    * processing it every 22 ms should be fine
    */
    addclock0link(pl011_clock, 22);
}
```

And in plo11_clock we just check for waiting bytes:

```
static void
02
    pl011_clock(void)
04
        char c;
05
        int i;
06
        if (pl011_tstc()) {
            c = pl011 getc();
            if (c == 13) {
98
                 pl011_putc('\r');
09
                 pl011_putc('\n');
10
                 kbdputc(kbdq,'\r');
11
12
                 kbdputc(kbdq,'\n');
13
                 return;
14
            pl011_putc(c);
15
16
            kbdputc(kbdq,c);
17
18
```

But now if we try to execute something from **osinit.b** like **Sh**:

```
init(nil: ref Draw->Context, nil: list of string)
02
     {
          shell := load Sh "/dis/sh.dis";
03
04
          sys = load Sys Sys->PATH;
          sys->print("init: starting shell\n");
05
06
          sys->bind("#i", "/dev", sys->MREPL);
sys->bind("#c", "/dev", sys->MAFTER);
sys->bind("#u", "/dev", sys->MAFTER);
07
                                                              # draw device
08
                                                              # console device
09
                                                              # usb subsystem
10
11
          spawn shell->init(nil, nil);
12
```

Categories

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

We get a fault with lock:

```
1  $ panic: ilock: no way out: pc 0x30f1c: lock 0x0 held by pc 0xe59ff018
2  ktrace/kernel/path 00009578 00ad07f4 00000000
```

Tracing of the problem leads us to the fact that we haven't initialized *kbdq* (Keyboard Queue). Simple fact but hard to reveal and sure you can not google about ⁽¹⁾ . So add to *plo11init()*:

```
1 if(kbdq == nil)
2 kbdq = qopen(4*1024, 0, 0, 0);
```

Now Sh looks as started and when we try to type symbols on our serial console they also appear on screen connected to Raspberry Pi. Success! No, wait, attempt to run Ls give us the another fault:

```
01  $ ls
02  panic: Undefined instruction
03  ktrace/kernel/path 0000968c 00add608 00000000
04  [Ls] Broken: "sys: trap: fault pc=000095a4"
05  3 "Ls":sys: trap: fault pc=000095a4
06  ,$ pwd
07  /
08  $ echo 123
09  123
10  $
```

But *echo* and *pwd* are working. So, what it can be? After long study and experiement first I found that when **trap.c** was written I put *panic()* for *PsMund*, but Inferno+Arm uses "undefined instructions" to work with FPU. Oops, we do not have any FPU codes currently. So started to migrate some code again from 9pi. Added **fpiarm.c**, had to adjust it to fit Inferno, but little. From port added **fpi** and **fpimem**. Some polishing, compiling and still faulting, but at least I know that it if FPU codes, so for now I just disabled *fpemu()* in *fpiarm()*. And I got Ls finally working!

```
Enabled irq 3
ARM 700 MHz id 410fb767
Inferno OS Fourth Edition (20130906) Vita Nuova
Ported to Raspberry Pi (BCM2835) by LynxLine
Starting init()
Initial Dis: "/osinit.dis
init: starting shell
                                 0 Nov 30 22:39 chan
   xr-xr-x / 0 inferno inferno
                                 0 Nov 30 22:39 dev
             O inferno inferno
                                 0 Nov 30 22:39 dis
       xr-x / 0 inferno inferno
             0 inferno inferno
                                 0 Nov 30 22:39 env
                                 0 Nov 30 22:39 fd
       xr-x / 0 inferno inferno
                                 0 Nov 30 22:39 net
      -xr-x / 0 inferno inferno
             O inferno inferno 341 Nov 30 22:39 osinit.dis
                                0 Nov 30 22:39 prog
d-r-xr-xr-x / 0 interno interno
  1s -1 /dev
 d-r-xr-xr-x i 0 inferno inferno 0 Nov 30 22:39 /dev/draw
 -r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/drivers
-rw-r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/hostowner
 rw-rw-rw- c 0 inferno inferno 0 Nov 30 22:39 /dev/jit
  rw-rw-rw- c 0 inferno inferno 0 Nov 30 22:39 /dev/keyboard
  r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/klog
    -r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/kprint
  r--r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/memory
  r--r--r-- c 0 inferno inferno 12 Nov 30 22:39 /dev/msec
  r--r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/notquiterando
 -rw-rw-rw- c 0 inferno inferno 0 Nov 30 22:39 /dev/null
 -r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/random
 -r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/scancode
 -rw-r--r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/sysct1
 -rw-rw-r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/sysname
 -rw-rw-r-- c 0 inferno inferno 0 Nov 30 22:39 /dev/time
 -rw-r--r- c 0 inferno inferno 0 Nov 30 22:39 /dev/user
S pwd
$ cat /dev/msec
       86090$ cat /dev/hostowner
infernos ls -1 /dev/draw
--rw-rw-rw- i 0 inferno inferno 0 Nov 30 22:39 /dev/draw/new
```

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

« Lab 18, we have a screen!



Copyright LynxLine. All rights reserved. Powered by lynxline.com, WordPress