Lab 5, Hello World

By ADMIN | *Published:* NOVEMBER 27, 2012

This lab will be short but it is going to summarize all material provided in previous labs.
Our target to develop simple “barebone” program which can be loaded and executed by U-Boot.
It should only print “Hello world!” to our serial console.

First, let’s find out how we can print something. BCM2835 manual describes that the chip has different features as Mini-UART and PL011 UART.

Instead of writing own driver for UART routines let’s better check how U-Boot performs. Checking the sources codes gives us a hint that when U-Boot pass execution control to our program we have initialized PL011 which has ports ready to transmit bytes to our serial line. It has choice for PL011 because Linux kernel (which is default target/option for U-Boot) will be throwing boot messages exactly to PL011 ports until Linux kernel initializes own driver for serial.

After checking routines we realizes that code for sending something to serial is pretty simple:

```
01 #define IOBASE          0x20000000      /* base of io registers */
02 #define PL011REGS       (IOBASE+0x201000)
03 #define UART_PL01x_FR_TXFF  0x20
04 typedef unsigned int    u32int;
05
06 void
07 pl011_putc(int c)
08 {
09     u32int *ap;
10     ap = (u32int*)PL011REGS;
11     /* Wait until there is space in the FIFO */
12     while (ap[0x18>>2] & UART_PL01x_FR_TXFF)
13         ;
14
15     /* Send the character */
16     ap[0] = c;
17
18     /* Wait until there is space in the FIFO */
19     while (ap[0x18>>2] & UART_PL01x_FR_TXFF)
20         ;
21 }
22
23 void
24 pl011_puts(char *s) {
25     while(*s != 0) {
26         if (*s == '\n')
27             pl011_putc('\r');
28         pl011_putc(*s++);
29     }
30 }
31
32 void
33 main() {
34     char * s = "Hello world!\n";
35     pl011_puts(s);
36     for (;;);
37 }
```

Again, we do not need to initialize anything, so our code should just work.

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
- Web

Compile our barenone kernel, connecting everything (serial, eth, etc), starting tftp server, juice power to rpi and:

```
U-Boot 2012.04.01-00489-gcd2dac2-dirty (Jul 07 2012 - 12:57:03)

DRAM:  128 MiB
WARNING: Caches not enabled
MMC:   bcm2835_sdh: 0
Using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Net Initialization Skipped
No ethernet found.
Hit any key to stop autoboot:  0
reading uEnv.txt

17 bytes read
Importing environment from mmc ...
reading boot.scr

274 bytes read
Running bootscript from mmc0 ...
## Executing script at 00008000
(Re)start USB...
USB:   Core Release: 2.80a
scanning bus for devices... 3 USB Device(s) found
       scanning bus for storage devices... 0 Storage Device(s) found
       scanning bus for ethernet devices... 1 Ethernet Device(s) found
Waiting for Ethernet connection... done.
BOOTP broadcast 1
*** Unhandled DHCP Option in OFFER/ACK: 95
*** Unhandled DHCP Option in OFFER/ACK: 95
DHCP client bound to address 10.0.55.120
BOOTP broadcast 2
Waiting for Ethernet connection... done.
Using sms0 device
TFTP from server 10.0.55.112; our IP address is 10.0.55.120
Filename 'irpi'.
Load address: 0x7fe0
Loading: T T T T T T T T #
done
#Bytes transferred = 634 (27a hex)
## Starting application at 0x00008000 ...
Hello world!
```

So at this point you can develop any OS you would like to have, using Plan9 Assembler and C lang, but our timeline is to have Inferno OS working native on Raspberry Pi.

FILES:
main.c

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

« *Ru: Archive: O Boost Multi-index Containers*

Lab 6, Compile something »