LYNXLINE

Professional Software Development Services

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

Lab 15, Eve, Hello World from Limbo!

By ADMIN | *Published:* APRIL 17, 2013

It is our 15-th Lab and it is the time came to make birth of the process Eve, run Dis virtual machine and our first Limbo program!

First create file **archrpi.c** where we all R-Pi platform specific code. We need to implement *kprocchild()* call:

```
static void
linkproc(void)
{
    spllo();
    if (waserror())
        print("error() underflow: %r\n");
    else (*up->kpfun)(up->arg);
    pexit("end proc", 1);
}

void
kprocchild(Proc *p, void (*func)(void*), void *arg)
{
    p->sched.pc = (ulong)linkproc;
    p->sched.sp = (ulong)p->kstack+KSTACK-8;
    p->kpfun = func;
    p->arg = arg;
}
```

As it is ready we just add initializations that are left to **main.c**:

```
...
trapinit();
printinit();

print("\nARM %ld MHz id %8.8lux\n", (m->cpuhz+500000)/1000000, getcpuid());
print("Inferno OS %s Vita Nuova\n\n", VERSION);

procinit();
links();
chandevreset();

eve = strdup("inferno");

userinit();
schedinit();

pl011_puts("to inifinite loop\n\n");
for (;;);
```

Where our initialization of first process *init0()* and run first Dis program:

```
void
init0(void)
{
    Osenv *o;
    char buf[2*KNAMELEN];

    up->nerrlab = 0;

    print("Starting init0()\n");
```

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

```
    spllo();

    if(waserror())
        panic("init0 %r");

    /* These are o.k. because rootinit is null.
     * Then early kproc's will have a root and dot. */

    o = up->env;
    o->pgrp->slash = namec("#/", Atodir, 0, 0);
    cnameclose(o->pgrp->slash->name);
    o->pgrp->slash->name = newcname("/");
    o->pgrp->dot = cclone(o->pgrp->slash);

    chandevinit();

    if(!waserror()){
        ksetenv("cputype", "arm", 0);
        snprint(buf, sizeof(buf), "arm %s", conffile);
        ksetenv("terminal", buf, 0);
        poperror();
    }

    poperror();

    disinit("/osinit.dis");
}

void
userinit(void)
{
    Proc *p;
    Osenv *o;

    p = newproc();
    o = p->env;

    o->fgrp = newfgrp(nil);
    o->pgrp = newpgrp();
    o->egrp = newegrp();
    kstrdup(&o->user, eve);

    strcpy(p->text, "interp");

    p->fpstate = FPINIT;

    /*  Kernel Stack
        N.B. The -12 for the stack pointer is important.
        4 bytes for gotolabel's return PC */
    p->sched.pc = (ulong)init0;
    p->sched.sp = (ulong)p->kstack+KSTACK-8;

    ready(p);
}
```

Now go to folder **../init/** and create simple Limbo program **rpiinit.b**:

```
implement Init;

include "sys.m";
    sys:    Sys;

Bootpreadlen: con 128;

Init: module
{
    init:   fn();
};

init()
{
    sys = load Sys Sys->PATH;
    sys->print("Hey, this is Hello World from Dis!\n\n");
}
```

Now edit platform definition file **rpi**, section **init** and section **root** to use **rpiinit.b**:

```
...
init
    rpiinit

root
    /chan   /
    /chan   /
    /dev    /
    /dis    /
    /env    /
    /fd     /
    /net    /
    /prog   /
    /dis/lib
    /dis/disk
    /osinit.dis
```

Compile, start Raspberry Pi device with expectations and:

```
Load address: 0x7fe0
Loading: T #T #####################################
done
Bytes transferred = 552620 (86eac hex)
## Starting application at 0x00008000 ...
Entered main() at 00009114 with SP=00002FE8
Clearing Mach:  00002000-00002060
Clearing edata: 00065B08-0006CC30
Conf: top=134217728, npage0=32659, ialloc=26750976, nproc=735

ARM 0 MHz id 410fb767
Inferno OS Fourth Edition (20121207) Vita Nuova

Starting init0()
Initial Dis: "/osinit.dis"
Hey, this is Hello World from Dis!
```

Wow, Success!

FILES:
rpi
dat.h
main.c
trap.c
mkfile
archrpi.c
../init/rpiinit.b

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

« *Lab 14, interrupts, part 3*

Lab 16, Adding clocks, timers, converging to 9pi codes »