eCos
an Open Source Embedded Operating System

Dr. Ralf Schlatterbeck
Open Source Consulting

Email:	office@runtux.com
Web:	http://www.runtux.com
Tel.	+43/650/621 40 17

Contents

eCos Overview

- single process, multiple thread operating environment
- single linked ELF binary image
- History: Cygnus later RedHat, then eCosCentric
- eCosCentric sells a closed „industrial strength eCosPro“
- The free version is on ecos.sourceware.org
- All symbols in eCos Libraries and include files still begin with `cyg_` or `CYG_`
- Hardware Abstraction Layer (HAL) for many CPUs
- Closely tied to RedBoot Bootloader, uses the HAL

eCos Overview

- good docs, both reference and introductory
- lists what is and what is not implemented (!)
- needed when porting code (e.g. from Linux)
- needed when writing packages for distribution
- eCos User Guide
- RedBoot User’s Guide
- eCos Reference Manual
- eCos Component Writer’s Guide

RedBoot

- Uses eCos HAL
- Command-Line Interface
- Can provide jump table to application $\rightarrow$ code-size
- manage flash images (by name)
- inspect, copy, checksum, compare, fill memory
- simple bootup script
- software update via serial or ethernet interface
- command-line history (!)
- gdb support: start image in debugger

Debugging Support

- GDB stub built into RedBoot
- funnels all stdio/debug output via GDB protocol
- support for breakpoints
- break with Ctrl-C possible during output
- support for image-upload with debugger
- no eCos thread support
  (can't find out why a thread is waiting)
- eCos has separate „synthetic" target
- . . . which runs natively on the devel machine
- $\rightarrow$ nice for debugging algorithmic problems

Native API

Schedulers:

- Different schedulers configurable
- configurable number of priority levels
- Bitmap scheduler: only one thread per prio
- Multi-level queue (MLQ) scheduler: more flexible
- MLQ: configurable time-slicing
- optional priority queuing for threads waiting on a resource
- some functions only available with MLQ
  (e.g. protection against priority inversion)
- Stopping of threads is compile-time option

Native API: Other primitives

- Counters: count operating system events,
  e.g. clock ticks
- Clocks: special form of counter
- Alarms: take action after certain number of events
  $\rightarrow$ used together with counters/clocks
- Mail boxes: Wait for event and pass data
- Flags: atomically wait for one of several events
- Spinlocks (multiprocessor synchronisation)
  low-level primitive, used by eCos kernel
- Counting Semaphores
- Mutex and Condition variables on next slides

Excursus: Priority Inversion

- Three threads with different priorities
- thread A has highest prio, B intermediate, C lowest
- C holds semaphore (mutex)
- A waits on semaphore
- B preempts C
- Priority Inversion: $\Rightarrow B$ de facto preempts A
- Solutions: Priority Inheritance, Priority Ceiling
- High profile case: Mars Pathfinder Rover

Native API: Mutex

Support for Priority Ceiling

```
static cyg_mutex_t lock;
static cyg_priority_t prio = MAX_TASK_PRIO;
cyg_mutex_init (&lock);
cyg_mutex_set_ceiling (&lock, prio);
...
cyg_mutex_lock (&lock);
/* critical section */
cyg_mutex_unlock (&lock);
```

Native API: Condition variables

```
static cyg_mutex_t lock;
static cyg_cond_t wait;
cyg_mutex_init (&lock);
cyg_cond_init (&wait, &lock);
cyg_mutex_lock (&lock);
while (count == 0)
    cyg_cond_wait (&wait);
/* take resource */
cyg_mutex_unlock (&lock);
/* somewhere else: */
cyg_cond_signal (&wait);
```

POSIX API: Networking

- Two networking stacks: FreeBSD and OpenBSD
- standard TCP/IP and UDP/IP
- `MSG_DONTWAIT` on recv/send is accepted during compile but ignored at runtime (!)
- bridging, routing
- . . . unfortunately only one routing table
- IPv6 support (untested by me)
- DNS
- Libraries for IPSEC, PPP, . . .
- Client Libraries for TFTP, NTP, SNMP, FTP
- two HTTP Server versions

POSIX API: Filesystem

Low-level I/O and stdio are supported

```
const char *fspath = "/dev/flash/fis/jffs2";
int ret = mount (fspath, "/dir", "jffs2");
...
int flags = O_CREAT | O_TRUNC | O_RDWR;
int fd = open ("/dir/file", flags);
```

- `/dev/flash/fis`: RedBoot flash image system
- Needs jffs2 option configured
- Only works if application is linked against RedBoot jump table (!)

Compatibility Libraries: PPP

- Full PPP implementation
- . . . including a chat program
- Support for HW-flow control depends on driver
- prio of PPP-Thread configurable
- . . . must not be better than calling thread!
- ppp works fine with GPRS modems
- . . . even without a driver supporting flow control
- but without modem signals ppp sometimes blocks forever
- . . . need to call `cyg_ppp_down` in other thread

Packages: Example OpenSSL

- CDL: configuration description language based on TCL, see eCos Component Writer’s Guide
- packages come with a config snipped
- packages and options are enabled, e.g.
```
ecosconfig add CYGPKG_POSIX
ecosconfig add ppp
ecosconfig import .../mysettings.cdl
```
- Example for setting an option
```
cdl_option CYGOPT_FS_JFFS2_WRITE {
    user_value 1
};
```

Packages: Example OpenSSL

- CDL in a packages specifies config options and how to compile, e.g.
```
cdl_package CYGPKG_OPENSSL {
    display        "OpenSSL"
    parent         CYGPKG_NET
    requires       CYGPKG_IO
    compile                               \
            crypto/asn1/a_object.c  \
```
- I’ve upgraded an existing port of OpenSSL 0.9.8 to the latest OpenSSL version 0.9.8r
- Available on request