Ottawa Linux Symposium 2007

Tiny qemu arm system with a DirectFB interface

Michael Opdenacker
Free Electrons
http://free-electrons.com

Created with OpenOffice.org 2.2

Free embedded Linux training materials

All the technical presentations and training materials created and used by Free Electrons, available under a free documentation license (more than 1500 pages!).

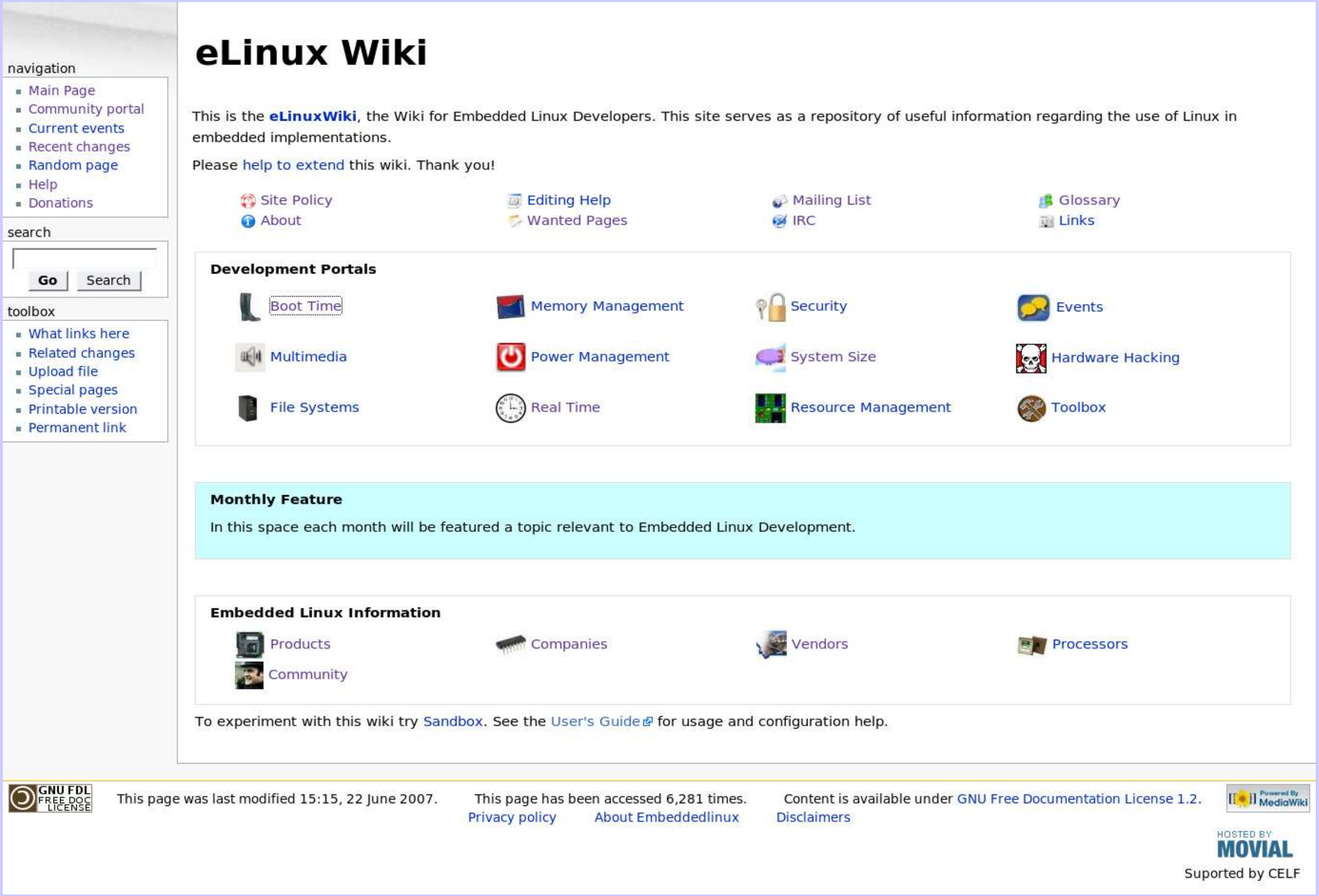
http://free-electrons.com/training

- Introduction to Unix and GNU/Linux
- Embedded Linux kernel and driver development
- Free Software tools for embedded Linux systems
- Audio in embedded Linux systems
- Multimedia in embedded Linux systems

http://free-electrons.com/articles

- Advantages of Free Software in embedded systems
- Embedded Linux optimizations
- Embedded Linux from Scratch... in 40 min!
- Linux USB drivers
- Real-time in embedded Linux systems
- Introduction to uClinux
- Linux on TI OMAP processors
- Free Software development tools
- Java in embedded Linux systems
- Introduction to GNU/Linux and Free Software
- Linux and ecology
- What's new in Linux 2.6?
- How to port Linux on a new PDA

Embedded Linux Wiki BOF



navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help
- Donations

search

toolbox

- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link

eLinux Wiki

This is the **eLinuxWiki**, the Wiki for Embedded Linux Developers. This site serves as a repository of useful information regarding the use of Linux in embedded implementations.

Please help to extend this wiki. Thank you!

Site Policy	Editing Help	Mailing List	Glossary
About	Wanted Pages	IRC	Links

Development Portals

Boot Time	Memory Management	Security	Events
Multimedia	Power Management	System Size	Hardware Hacking
File Systems	Real Time	Resource Management	Toolbox

Monthly Feature

In this space each month will be featured a topic relevant to Embedded Linux Development.

Embedded Linux Information

Products	Companies	Vendors	Processors
Community			

To experiment with this wiki try Sandbox. See the User's Guide for usage and configuration help.

This page was last modified 15:15, 22 June 2007. This page has been accessed 6,281 times. Content is available under GNU Free Documentation License 1.2.

Privacy policy About Embeddedlinux Disclaimers

Suported by CELF

Demo features

- VersatilePB board emulated by qemu,
  16 MB of RAM (needs at least 10 MB)
- Uses the uClibc C library
  (4x smaller than glic)
- DirectFB library and example programs
- Busybox: shell scripts and command line utilities
- Original size: 74 MB
- Size after optimization: 2.2 MB (kernel + rootfs)
  (31% of which is images!)

Built with Scratchbox

http://scratchbox.org/ - A cross-compiling toolkit

- Makes it easier to cross-compile a complete embedded Linux system. A cornerstone of Nokia's Maemo development environment.
- Works by allowing tools to be cross-compiled in a transparent way, making building tools believe they are doing a native compile job.
- Supported platforms: `arm`, `x86`
  Uses the `qemu` emulator to transparently run built target binaries.
  Experimental support for `ppc`, `mips` and `cris`.

Embedded Linux Optimizations

Reducing system size

Removing unused files

- `/usr/include` (48 M), `/usr/local/include` (2.3M): C headers.
- `/usr/local/share/man` (236 K): manual pages.
- `*.a` library object files (13.2 M) and `*la` links to them: only needed for compiling.
- `/usr/lib/libfakeroot` (60 K), `/usr/local/lib/pkgconfig` (28K), `/usr/bin/gdbserver`, `/usr/bin/strace` (304 K): programs no longer needed in production.
- `/usr/local/share/aclocal` (14K): just needed for development.

Detecting unused libraries

- Run the `readelf` command from your cross-compiling toolchain on all your executables (`ldd` often not available) . It tells you which shared libraries are used.
- Remove the unused shared libraries
- Back to our example:
  `/usr/lib/libstdc++.so.6.0.3` (2.4 M)

Automatic removal of unused files

Implementation ideas:

- Mount the root filesystem with the `atime` option.
- Run your complete test suite
- Then, use the `find` command to identify files with an access time older than that of your first user-space program.
  Or implement that through a BusyBox applet.
- Drawback: has to be done on a real, read-write filesystem, if you are using a read-only filesystem like SquashFS.
- Of course, also takes care of detecting unused shared libraries.

Merging duplicate files

Software compiling and installing often create duplicate files...
Check that your root filesystem doesn't contain any!

- dupmerge2: http://sourceforge.net/projects/dupmerge
  Replaces duplicate files by hard links.
- clink: http://free-electrons.com/community/tools/utils/clink
  Replaces duplicate files by symbolic links.
  Example: saves 4% of total space in Fedora Core 5.
- finddup: http://www.shelldorado.com/scripts/cmds/finddup
  Finds duplicate files.

sstrip: “super strip”

http://muppetlabs.com/~breadbox/software/elfkickers.html

- Goes beyond strip and can strip out a few more bits that are not used by Linux to start an executable.
- Can be used on libraries too. Minor limitation: processed libraries can no longer be used to compile new executables.
- Can also be found in toolchains made by Buildroot (optional)

	Hello World	*Busybox*	*Inkscape*
Regular	4691 B	287783 B	11397 KB
stripped	2904 B (-38 %)	230408 B (-19.9 %)	9467 KB (-16.9 %)
sstripped	1392 B (-70 %)	229701 B (-20.2 %)	9436 KB (-17.2 %)

Best for tiny executables!

Reducing the kernel size

Using Linux-Tiny options (`CONFIG_EMBEDDED`)

- Removing `printk`, `BUG`, `panic`...
- Removing unused features like core dumps, etc.
  Who needs all features in a special purpose system?
- Using compiler optimizations for size.
- Compressed kernel size before: 632 KB
- Compressed kernel size after: 420 KB!

Using initramfs

Booting on an root filesystem in an initramfs:

- Saves size: compressed cpio archive within the kernel.
  Just 1 file to handle in the bootloader.
- Simpler: no block driver, no filesystem driver (smaller kernel)
- Saves RAM compared to an init ramdisk
- Can also be used to carry non GPL files:
  firmware, pictures, proprietary drivers...

Embedded Linux Optimizations

Reducing kernel boot time

Disable console output

- The output of kernel bootup messages to the console takes time! Even worse: scrolling up in framebuffer consoles! Console output not needed in production systems.

- Console output can be disabled with the `quiet` argument in the Linux kernel command line (bootloader settings)
- Example:

```
root=/dev/ram0 rw init=/startup.sh quiet
```

- Benchmarks: can reduce boot time by 30 or even 50%!

See http://elinux.org/Disable_Console

Preset loops_per_jiffy

- At each boot, the Linux kernel calibrates a delay loop (for the `udelay` function). This measures a `loops_per_jiffy` (`lpj`) value. This takes about 25 jiffies (1 jiffy = time between 2 timer interrupts). In embedded systems, it can be about 250 ms!
- You just need to measure this once! Find the `lpj` value in kernel boot messages (if you don't get it in the console, boot Linux with the `loglevel=8` parameter). Example:

```
Calibrating delay loop... 187.59 BogoMIPS (lpj=937984)
```

- At the next boots, start Linux with the below option:
  `lpj=<value>`

Techniques not used yet

- Library optimizer: http://libraryopt.sourceforge.net/
  Removes unused symbols in shared libraries.
- LZMA SquashFS
  http://www.squashfs-lzma.org/
- Puppies!

References

- http://elinux.org/
- More specifically:
  http://elinux.org/System_Size
  http://elinux.org/Boot_Time
  http://free-electrons.com/articles/optimizations/
- The demo, its sources and technical details can be found on:
  http://free-electrons.com/community/demos/qemu-arm-directfb/

Thank you! Questions or suggestions?