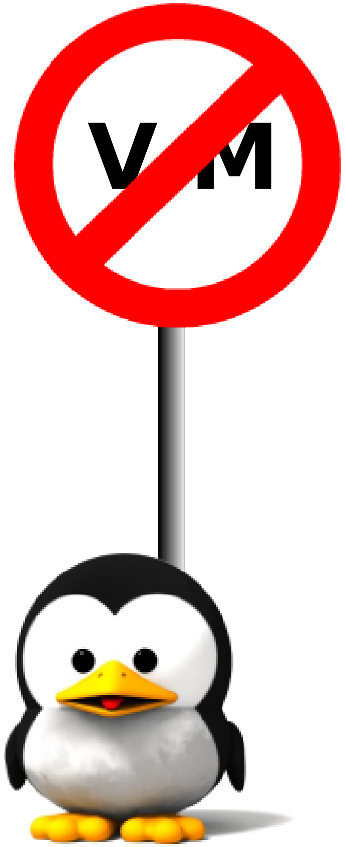
Introduction to uClinux



Introduction to uClinux

Michael Opdenacker

Free Electrons

http://free-electrons.com

Created with OpenOffice.org 2.x

Thanks to Nicolas Rougier (Copyright 2003, http://webloria.loria.fr/~rougier/) for the Tux image

Rights to copy

COMMONS DEED

Attribution – ShareAlike 2.5

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions

Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/2.5/legalcode

© Copyright 2004-2007
Free Electrons
feedback@free-electrons.com

Document sources, updates and translations:
http://free-electrons.com/articles/uclinux

Corrections, suggestions, contributions and translations are welcome!

Best viewed with...

This document is best viewed with a recent PDF reader or with OpenOffice.org itself!

- Take advantage of internal or external hyperlinks. So, don't hesitate to click on them!
- Find pages quickly thanks to automatic search.
- Use thumbnails to navigate in the document in a quick way.

If you're reading a paper or HTML copy, you should get your copy in PDF or OpenOffice.org format on http://free-electrons.com/articles/uclinux!

Contents

- uClinux project overview
- uClinux implementation details
- Using uClinux

Acronyms

- MMU: Memory Management Unit
- MPU: Memory Protection Unit
- GOT: Global Offset Table - Used in executable formats.
- ELF: Executable and Linkable Format
  A file format describing executables, object code, shared libraries and core dumps. The OS uses it to know how to load executables and shared libraries.
- PIC: Position Independent Code
  Object code without absolute addresses, which can execute at different locations in memory.
  See http://en.wikipedia.org/wiki/Position_independent_code

The MMU job

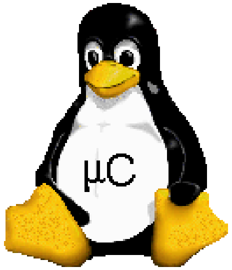
MMUs included in many general purpose processors available today

- Virtual to physical address translation.
  Allows processes to run in their own virtual contiguous address space. No need for relocating process addresses. Possible to expand the address space of a running process.
  The MMU raises an exception when no physical address is available, making it possible to implement swapping to disk.
- Address protection
  Actually done by the MPU available in most MMUs.
  Prevents processes from accessing unauthorized memory addresses.
  Note that some systems just have an MPU, but no MMU.

uClinux project overview

The uClinux project

http://www.uclinux.org/ - Linux for micro-controllers



Deliveries:

- Linux kernel supporting MMU-less processors (at least 32 bit) Supported in mainstream sources or through patches.
- Software distribution (source only): http://www.uclinux.org/pub/uClinux/dist/
- uClibc: a lightweight though highly compatible C library. Now an independent project. Used by Linux too!
- Cross-compiling toolchains.

uClinux devices

Just a few examples!

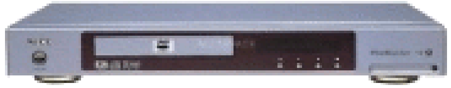
Send us more!

uClinux is often so deeply embedded

that it's difficult to identify 😉

Multimedia

Apple iPod (not shipped with uClinux)



Sigma Designs EM8500 based DVD players

Industrial

IntelliCom remote control system

Tiny Single Board Computers

C Data Solutions CF computer

Simtec SBCs

picotux RJ45 size computer

Network devices

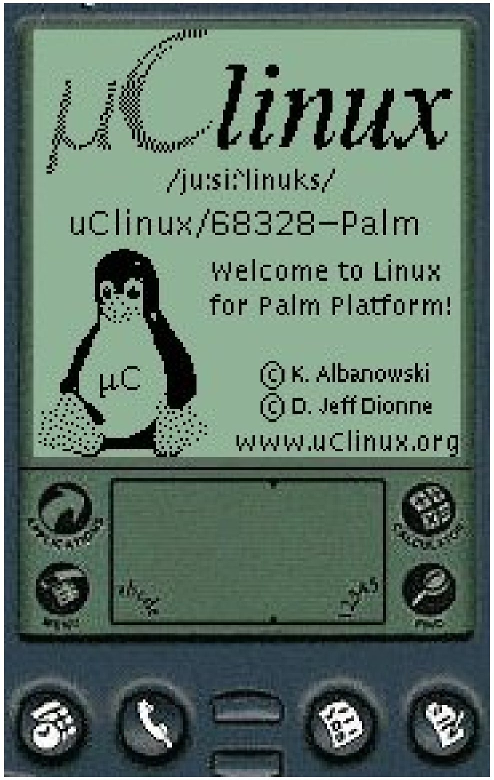
SnapGear LITE2 VPN/Router

StarDot NetCam

Aplio/PRO IP Phone

uClinux history

- First release in 1998 (Linux 2.0), for the Motorola 68000 processor. Demonstrated on Palm Pilot III.
- 1999: Motorola ColdFire support
- 2001: Linux 2.4 support. ARM7 support
- 2004: Linux 2.6 support for ARM
- 2007: You're reading this document 😉



µClinux
/ju:si:'linuks/
uClinux/68328-Palm
Welcome to Linux
for Palm Platform!
µC
© K. Albanowski
© D. Jeff Dionne
www.uClinux.org

Reasons for using uClinux (1)

- Linux
  Built-in IP connectivity, reliability, portability, filesystems, free software...
- Lightweight
  Full Linux 2.6 kernel under 300K, binaries much smaller with uClibc.
- XIP (Execute In Place)
  Don't have to load executables in RAM. May run slower though.
- Cheaper
  MMU-less arm cores are smaller.
- Sufficient
  A large number of embedded systems applications can do without an MMU.
- Faster
  Faster context switches: no cache flushes.

Reasons for using uClinux (2)

- User access to the hardware

  User applications can access the whole system, including device registers.

- Full Linux API

  Can use most Linux system calls with minor exceptions. Ported applications distributed with uClinux.

- Full Linux 2.6 kernel features

  stability, preemptible kernel, drivers...

- Full multi-tasking

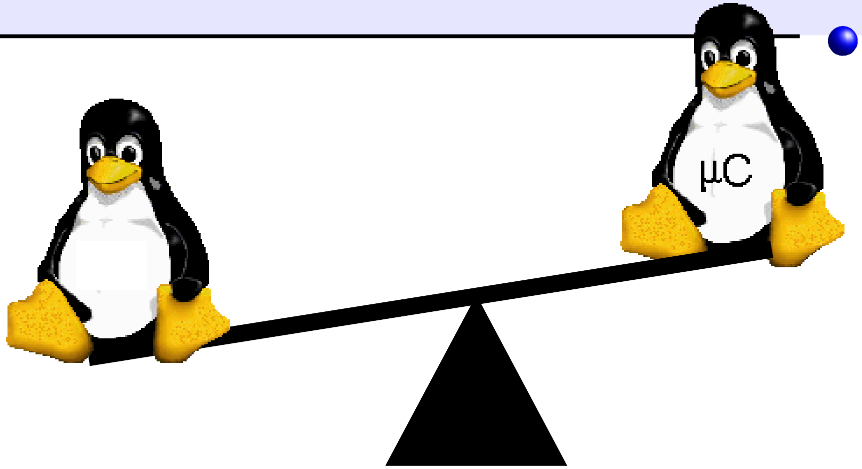
  Just minor limitations

- Supported on many processors, which wouldn't be supported by Linux otherwise.

  See http://www.uclinux.org/ports/

  Even running on DSP processors (ADI Blackfin, TI DM64x)!

uClinux weaknesses



- Less momentum than Linux.
  Much smaller community.
- Much less on-line documentation and resources available
- Lack of updates on pages and deliverables on http://uclinux.org. Lots of links and resources older than 2002.
- Lots of projects still using Linux 2.4.

However uClinux development is still active: kernel and distribution.
uClinux releases available for each Linux 2.6.x version,
released just a few weeks after.

uClibc

http://www.uclibc.org/ for CodePoet Consulting

- Lightweight C library for small embedded systems, with most features though.
- Originally developed for uClinux. Now an independent project.
- The whole Debian Woody (thousands of programs) was recently ported to it... You can assume it satisfied most needs!
- Example size (ARM): approx. 400K (vs. 1700 K for glibc)
- uClibc vs. glibc size comparison (busybox example, static build): 311 K vs. 843 K!

uClinux limitations

In a nutshell

- Virtual memory = physical memory
- Fixed memory for processes, can't fragment the memory: more memory consumption
- No memory protection

Very useful details (by David McCullough):
http://www.linuxjournal.com/article/7221

uClinux implementation details

No memory management

- No virtual memory
  Programs addresses need to be preprocessed ("relocated") before running to obtain unique address spaces.
- No on-demand paging
  Need to load whole program code in RAM
  (instead of just loading pages when they are effectively accessed).
- No memory protection
  Any program can crash another program or the kernel. Corruption can go unnoticed and surface later... difficult to track down!
  Design your code carefully. Be careful of data from the outside!
- No swapping
  Not really an issue on the tiny embedded devices

Better performance

uClinux can be significantly faster than Linux on the same processor!

- MMU operation can represent a significant time overheard. Even when an MMU is available, it is often turned off in systems with real-time constraints.

- Context switching can be much faster on uClinux. On ARM9, for example, the VM based cache has to be flushed at each context switch. No such need when all the processes share the same address space. See an interesting benchmark from H.S. Choi and H.C. Yun: http://opensrc.sec.samsung.com/document/uc-linux-04_sait.pdf

Different executable format

- Standard formats (such as ELF) rely on VM to create the address space of a process.
- flat format: condensed executable format storing only executable code and data, plus the relocations needed to load the executable into any location in memory.
- uClinux specific toolchains are needed to create executables in this format.

Different mmap implementation

- Unless the file is stored sequentially and contiguously, `mmap` needs to allocate memory! Only `romfs` can guarantee this.
- Another condition is that the storage can directly be accessed in the CPU physical address space. Can work with flash or ROM, but not with disk storage.
- Only read-only mappings can be shared (no copy-on-write) without allocating memory.
- In a nutshell, the `mmap` system call is available, but application developers should be aware that it has performance issues in the cases mentioned above.

Other kernel differences

- No tmpfs

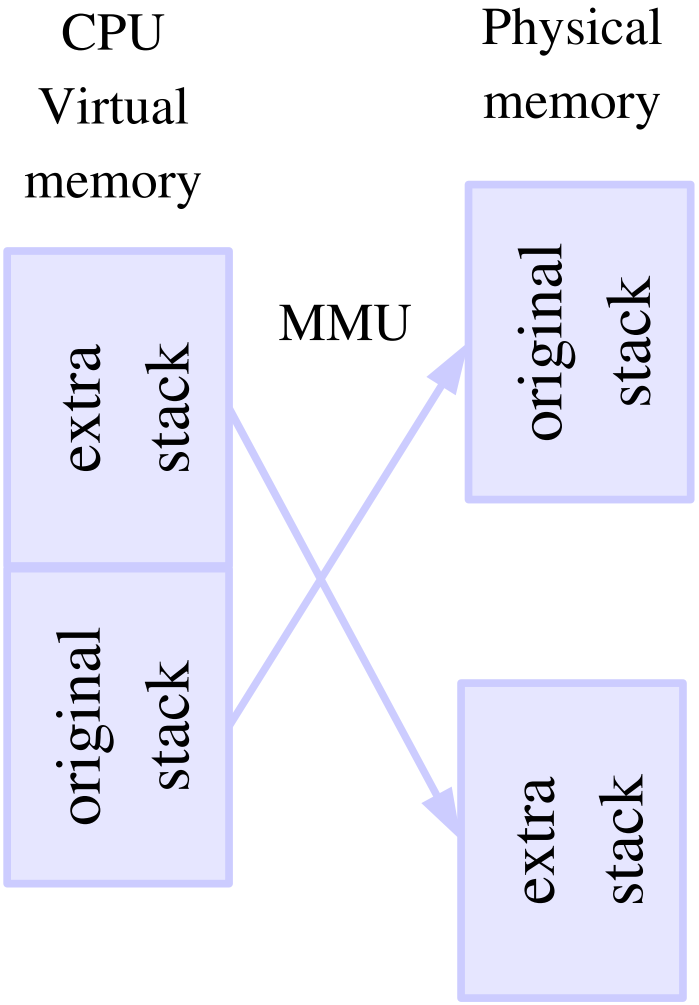
  Cannot use the tmpfs filesystem relying on VM.

  Need to use fixed size ramdisks.

No dynamic stack (1)

Linux

- With VM, can grow the stack of a running process whenever needed.
- Whenever an application tries to write beyond the top of its stack, the MMU raises an exception. This causes some new memory to be allocated and mapped in at the top of the stack.



No dynamic stack (2)

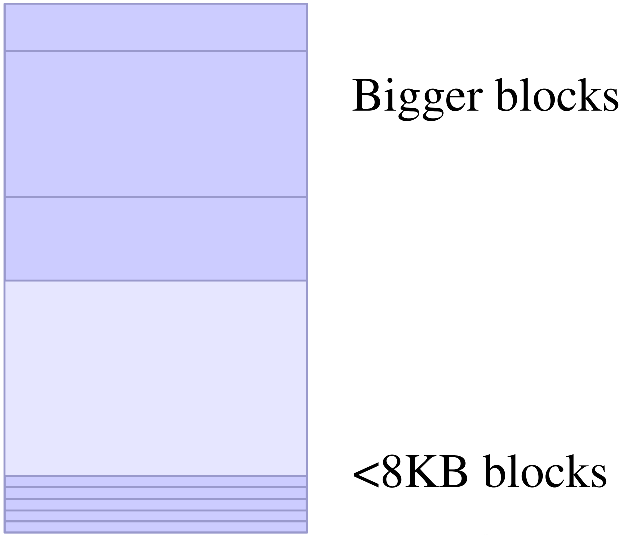
uClinux

- Stack size must be allocated at compile time: 4 KB by default.
- No exception raised when a process writes beyond the top of its stack! The consequences of this could surface much later.
- If strange crashes happen, try to increase the stack size of programs:
  - Either recompile:
    run `export FLTFLAGS=-s <stacksize>` before recompiling.
  - Or run `flthdr -s <stacksize> <executable>`

Memory allocation

- Standard Linux allocator: allocates blocks of 2^n size.
  If 65 KB are requested, 128KB will be reserved, and the remaining 63KB won't be reusable in Linux.

- uClinux 2.4 memory allocator: `kmalloc2` (aka `page_alloc2`)
  - Allocates blocks of 2^n size until 4KB
  - Uses 4KB pages for greater requests
  - Stores amounts not greater than 8KB on the start of the memory, and larger ones at the end. Reduces fragmentation.
  - Not available yet for Linux 2.6!



No dynamic process size (1)

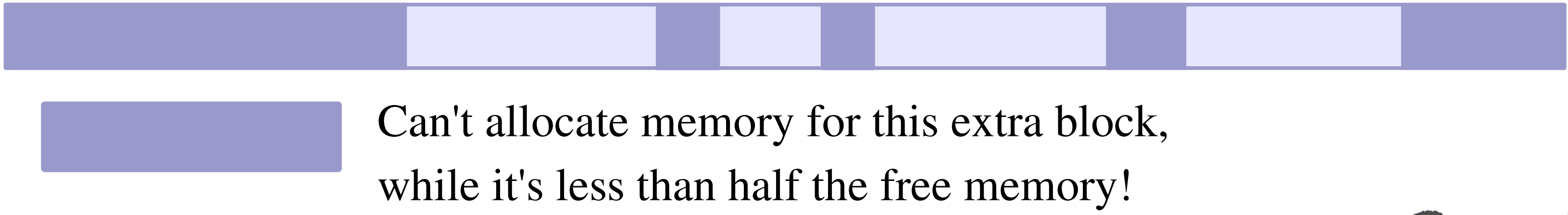
Linux

- With VM, can increase or decrease the process size with the `brk()` and `sbrk()` system calls.
- `malloc()` implementation based on `brk()` and `sbrk()`.

No dynamic process size (1)

uCLinux

- Memory has to be allocated from a global, shared memory pool
- Different `malloc()` implementation (`malloc-simple()`), accessing memory in this pool, managed by the kernel allocator.
- Fragmentation: can be unable to allocate enough contiguous memory
  Such situations can be detected through `/proc/mem_map` (kmalloc2)
- See http://www.cyberguard.info/snapgear/tb20020530.html for details about allocating memory in uClinux.



Can't allocate memory for this extra block, while it's less than half the free memory!

Tips for reducing memory fragmentation

- Have your programs allocate smaller chunks of memory, rather that allocating big ones at once.
- If possible, stop and restart applications when memory is too fragmented. Design your applications so that they can be shut down and restarted.

Tips for avoiding memory issues

Memory issues can be very difficult to track in uClinux.

- Fortunately, they should only happen with your own applications, not with widely tested tools from your uClinux distribution.
- Design with care and with a low memory budget in mind.
- Idea: first develop and profile your application on Linux (typically on your `i386` desktop). There are several utilities to detect memory issues: Valgrind, memcheck, ElectricFence. See http://free-electrons.com/articles/swdev for details.

No fork()

- Linux `fork()`

  The child process is a clone of the parent, using the same virtual memory space. New memory allocation happens for the child only when it modifies a page (“copy on write”).

- uClinux only implements `vfork()`
  - The parent execution is stopped and new memory is created before the child process is executed. Consumes more memory!
  - Need to replace all `fork()` calls by `vfork()`
  - No significant impact on multitasking though.

Execute In Place (XIP)

- Allows to start an application without loading it in RAM.
- Applies also to multiple instances of the same program. Saves a lot of RAM!
- Only supported by romfs (need continuous, non compressed storage).
- Only supported by the Position-Independent Code (PIC) flavor of the flat format (must be supported by the compiler).
- Caution: XIP may be much slower if storage access time is high.

Shared libraries

- Pretty different under uClinux.
- Different compiling options...
  Making your own won't be familiar.
- Must be compiled for XIP. Without XIP, shared libraries result in a full copy of the library for each application using it, which is worse than statically linking your applications.

See http://tree.celinuxforum.org/CelfPubWiki/uClinuxSharedLibrary for implementation details.

uClinux and Linux 2.6

- Most of uClinux code now merged with mainstream Linux.
- uClinux m68k tree now released in mainstream:
  `arch/m68knommu` (Motorola's m68k embedded CPUs)
- Other supported architectures:
  Hitachi's H8/300 series, NEC v850 processor, ADI Blackfin
- `arch/armnommu` not merged yet.
  Still available as a separate patch.
- Linux 2.6 can be built with no virtual memory system
  in a few platforms.

Using uClinux

uClinux on ARM MMU-less platforms

http://opensrc.sec.samsung.com/

- Patches against the standard Linux 2.6 kernel released by Hyok S. Choi ("-hsc" patches)
- Get the kernel patches and recent toolchains from http://opensrc.sec.samsung.com/download.html
- Supported processors
  - ARM7TDMI: Atmel AT91xxx, Samsung S3C3410X, S3C4510B, S3C44B0
  - ARM920T: S5C7375
- Feb. 2007: no new release since Linux 2.14 (Nov. 2005)! **Caution**: recent kernel versions not supported.

uClinux on m68knommu platforms

http://uclinux.org/ports/coldfire/

- Kernel sources with the standard Linux kernel (`arch/m68knommu/`)
- Supported processors: pretty long list! (see `arch/m68knommu/Kconfig`)
- Binary filesystem images also available on http://uclinux.org/ports/coldfire/binary.html

uClinux on ADI Blackfin

http://blackfin.uclinux.org/

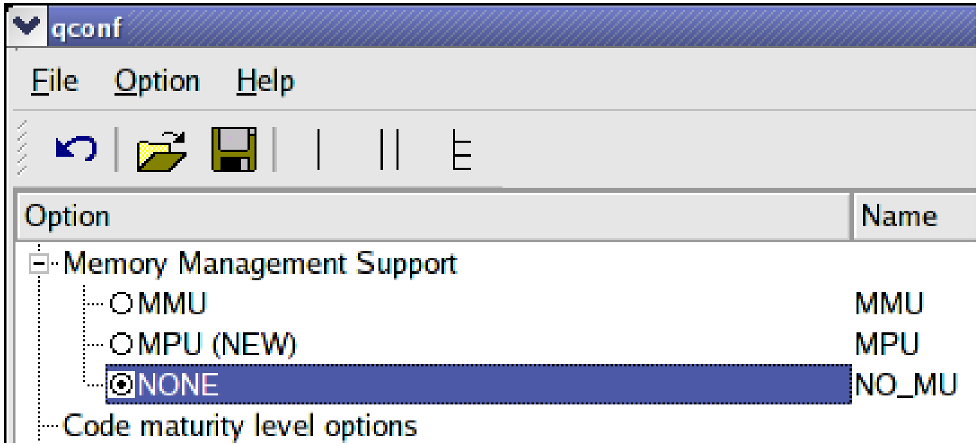
- Supported in mainstream Linux since 2.6.22
- A nice site and an active developer community
- Ships binary bootloader, kernel and distribution images.
- People from other uClinux ports can learn from their resources and experience.
- Actually, Blackfin has no VM, but some memory protection. See a nice article on Blackfin specifics:

http://docs.blackfin.uclinux.org/doku.php?id=operating_systems#introduction_to_uclinux

uClinux on other architectures

- uClinux on the MicroBlaze FPGA processor
  http://www.itee.uq.edu.au/~jwilliams/mblaze-uclinux/
  Nice and active community. Commercial support also available.
- More resources available on http://uclinux.org/ports/
  (Caution: lots of broken hyperlinks!)

Disabling the MMU on CPUs with MMU



Option	Name
⊟ Memory Management Support	
○ MMU	MMU
○ MPU (NEW)	MPU
● NONE	NO_MU
Code maturity level options	

arm

- Get the latest kernel patch from http://opensrc.sec.samsung.com/
- You can then disable the MMU.

sh

- `CONFIG_MMU` can be disabled. Not tested.

Other platforms: `i386`, `m68k`, `mips`, `ppc`

- Can't configure out MMU usage with the Vanilla Kernel sources

uClinux distributions

uClinux.org distribution

- http://uclinux.org/pub/uClinux/dist/
- Just distributes toolchains and sources

SnapGear (major contributor to uClinux) distribution

- http://www.snapgear.org/snapgear/downloads.html
- Just distributes toolchains and sources
- A bit less complete, though very similar.

Best places to pick sources for use with uClinux,
even if you don't use the distributions.

uClinux toolchains

Need to get uClinux specific toolchains

- Available from 2 main locations.
  uClinux.org: http://uclinux.org/pub/uClinux/
  SnapGear: http://snapgear.org/snapgear/downloads.html
- Regular toolchain (compiled to support PIC and XIP), plus flat format utilities: `elf2flt` and `flthdr`.

Compiling the uClinux distribution (1)

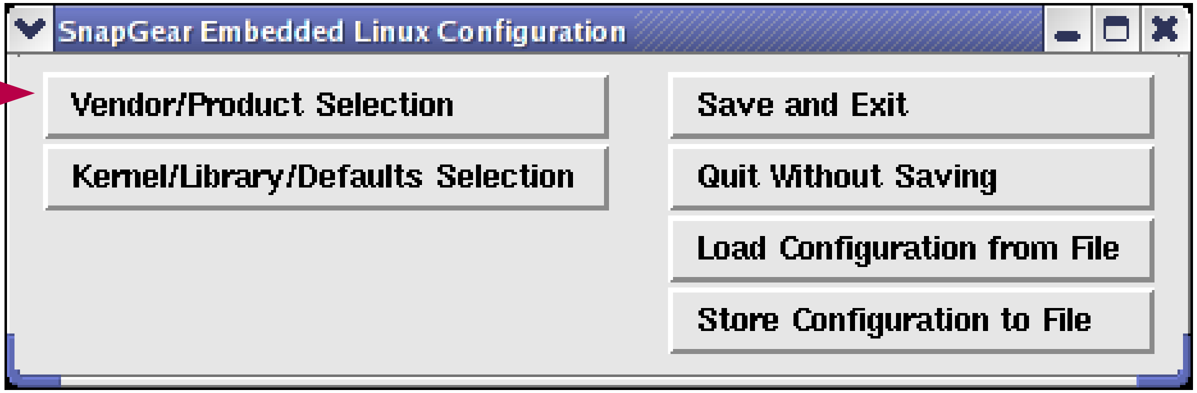
- Add your uClinux cross-compiling toolchain to your PATH:

```
export PATH=/usr/local/arm-elf/bin:$PATH
```

- In the toplevel `uClinux-dist` directory:

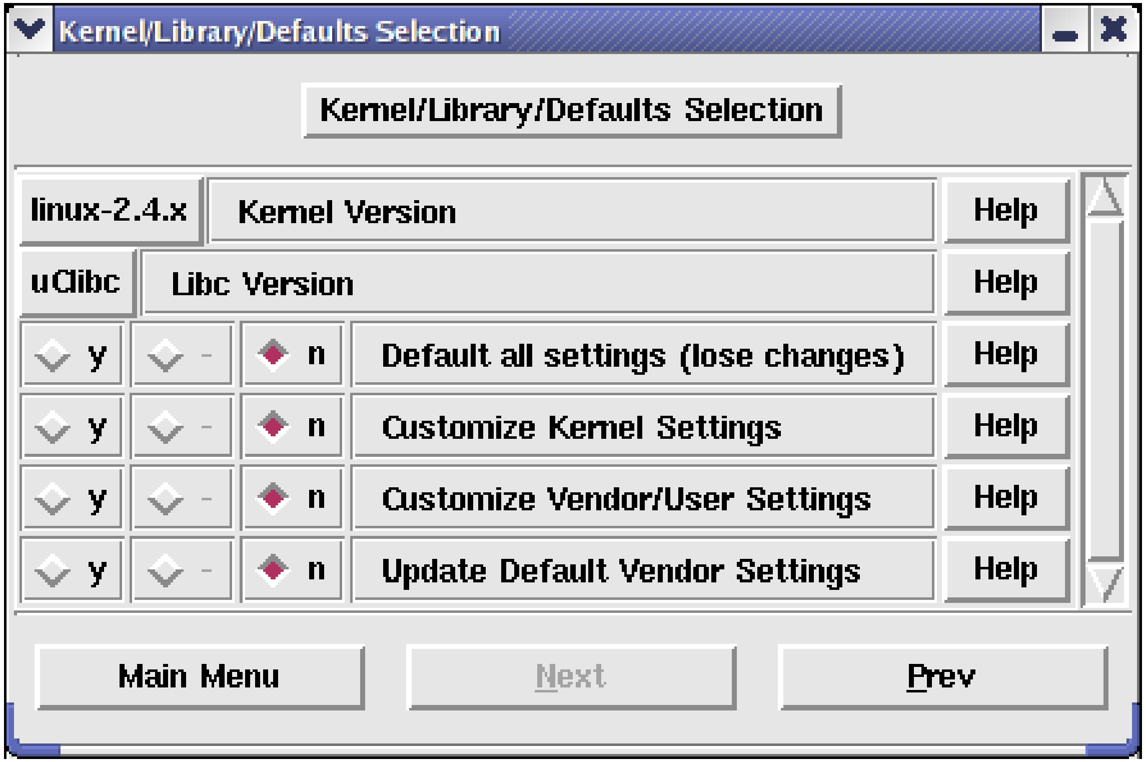
```
make xconfig
```

- Choose your platform (vendor and product)


SnapGear Embedded Linux Configuration
Vendor/Product Selection
Kernel/Library/Defaults Selection
Save and Exit
Quit Without Saving
Load Configuration from File
Store Configuration to File

Compiling the uClinux distribution (2)

- Choose your kernel and C library version
- Customize or not kernel and tool settings
- Compile:
  `make dep`
  `make`
  This generates binary images for your target in the `images/` directory.



Kernel/Library/Defaults Selection

Kernel/Library/Defaults Selection

linux-2.4.x Kernel Version Help

uClibc Libc Version Help

○ y ○ - ● n Default all settings (lose changes) Help

○ y ○ - ● n Customize Kernel Settings Help

○ y ○ - ● n Customize Vendor/User Settings Help

○ y ○ - ● n Update Default Vendor Settings Help

Main Menu | Next | Prev

Toolchain installation example

- Download the `arm-elf-tools-20030314.sh` file from the SnapGear site.
- Make the file executable (self extracting archive):

```
chmod a+rx arm-elf-tools-20030314.sh
```

- Switch to the `root` user:

```
su -
```

- Install the toolchain in `/usr/local/arm-elf`:

```
./arm-elf-tools-20030314.sh
```

  Caution: toolchains are not relocatable!
  They cannot be moved to another location
  (unless you create links in the default location)

Summary

- Worthy to use uClinux (rather than a proprietary RTOS) on processors without an MMU. Surprisingly, with uClinux, there are only minor differences with what you can do with a full Linux kernel.
- In some cases, worthy to use uClinux rather than Linux on processors with an MMU, for performance reasons.
- Though a lot of work has already been done and most applications have already been ported, some uClinux experience or learning is definitely needed when you start a new project.

Resources

- uClinux development mailing list
  https://mailman.uclinux.org/mailman/listinfo/uclinux-dev
  Archives: http://mailman.uclinux.org/pipermail/uclinux-dev/
- http://ucdot.org/
  A wealth of resources, FAQs, forums for uClinux developers!
  Don't miss the very complete FAQ:
  http://www.ucdot.org/faq.pl

eCos: an alternative to uClinux

- eCos: http://ecos.sourceware.org/
- Very lightweight real-time embedded system, originally contributed by Red Hat / Cygnus solutions.
- Supported by GNU development tools.
- POSIX compliant API (can run standard Unix/Linux tools).
- Highly configurable to remove unneeded features. Highly portable thanks to a Hardware Abstraction Layer.
- Also supports 16 bit processors (32 and 64 bit too) (16 bit CPUs not supported by uClinux and Linux).

Related documents

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

How to help

If you support this work, you can help ...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order training sessions performed by the author of these documents (see http://free-electrons.com/training)
- By speaking about it to your friends, colleagues and local Free Software community.
- By adding links to our on-line materials on your website, to increase their visibility in search engine results.

Thanks

- To the uClinux community for their work and documentation, in particular to David McCullough for his very useful articles.
- To the OpenOffice.org project, for their presentation and word processor tools which satisfied all my needs.
- To the members of the whole Free Software and Open Source community, for sharing the best of themselves: their work, their knowledge, their friendship.

Free Electrons services

Embedded Linux Training

Unix and GNU/Linux basics
Linux kernel and drivers development
Real-time Linux
uClinux
Development and profiling tools
Lightweight tools for embedded systems
Root filesystem creation
Audio and multimedia
System optimization

Consulting

Help in decision making
System architecture
Identification of suitable technologies
Managing licensing requirements
System design and performance review

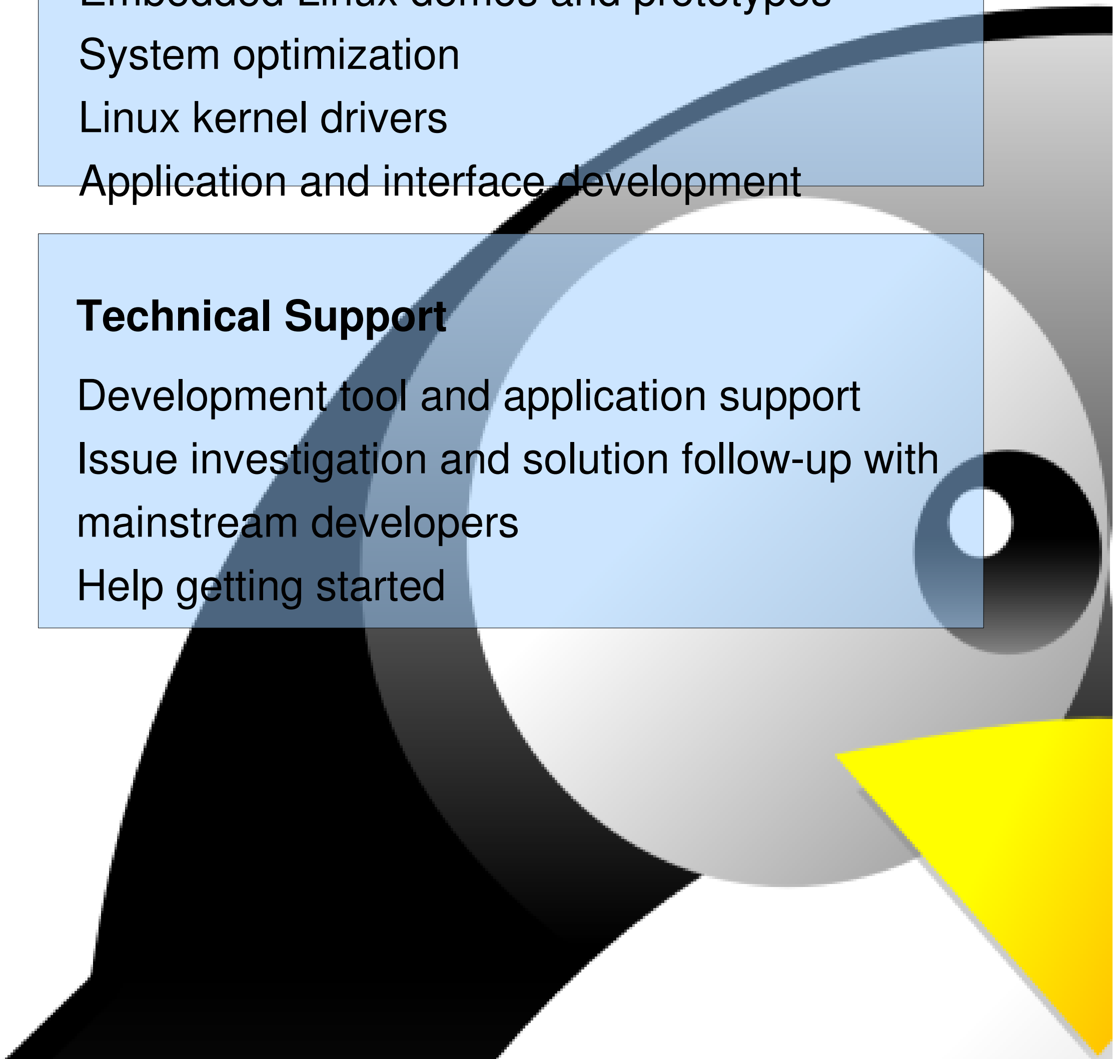
Custom Development

System integration
Embedded Linux demos and prototypes
System optimization
Linux kernel drivers
Application and interface development

Technical Support

Development tool and application support
Issue investigation and solution follow-up with mainstream developers
Help getting started

http://free-electrons.com



Free Electrons
Free Software for Embedded Systems