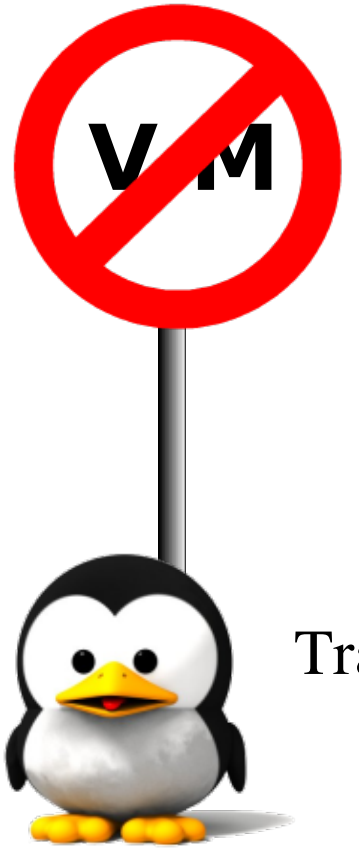



# Introduction à uClinux

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

Traduction française par Guillaume Lelarge

Créé avec OpenOffice.org 2.x

Merci à Nicolas Rougier (Copyright 2003, http://webloria.loria.fr/~rougier/) pour l'image du Tux

# Droits de copie

creative commons
COMMONS DEED

**Attribution – ShareAlike 2.0**

**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/2.0/legalcode



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

Corrections, suggestions, contributions and translations are welcome!

# Plus facile à lire avec...

Ce document est le plus facile à lire avec un lecteur PDF récent ou avec OpenOffice.org lui-même! Vous pouvez:

- Utiliser les hyperliens internes ou externes.
  Ainsi, n'hésitez pas à cliquer sur ces liens!
- Trouver facilement des pages grâce à la recherche automatique.
- Utiliser les miniatures de pages pour naviguer rapidement dans le document.

Si vous lisez une copie papier ou HTML, vous feriez mieux de récupérer une copie au format PDF ou OpenOffice.org sur http://free-electrons.com/articles/uclinux!





# Contenu

- Qu'est-ce qu'une MMU ?
- Le projet uClinux
- Quelques périphériques uClinux
- Les bonnes raisons d'utiliser uClinux
- Limitations et contraintes de uClinux
- Résumé
- Références





# Qu'est-ce qu'une MMU ?

En anglais, Memory Management Unit
(et en français, une unité de gestion de mémoire)
Inclus dans tous les processeurs à but généraliste disponibles aujourd'hui

- Traduction d'adresses
  Traduit les adresses virtuelles en adresses physiques
  Lève une exception lorsqu'aucune adresse physique n'est disponible
  Rend possible l'implémentation du swap pour les OS
- Protection des adresses
  Empêche les processus d'accéder à des adresses physiques où cela leur est interdit.

# Histoire de la MMU

- Disponible à l'origine en tant que composant séparé (MC 68851 utilisé avec le Motorola 68020, ou Z8015 utilisé avec les processeurs Zilog Z80)
- Inclus dans l'Intel 80386 (1986), le Motorola 68030 (1987) ou le Zilog Z280 (1987)
- Tout d'abord utilisé pour répondre aux limitations des tailles des mémoires sur les processeurs 16 bit.
- Utilisé après pour la protection de la mémoire.




# Le projet uClinux

http://www.uclinux.org/

Linux pour les microcontrôleurs


Fournit :

- uClibc : maintenant un projet indépendant
- Noyau Linux modifié
  Il vous est conseillé de rechercher les derniers correctifs pour votre architecture.
- Distribution du logiciel (source seulement) :
  http://www.uclinux.org/pub/uClinux/dist/
- Outils de cross-compilation





# Appareils sous uClinux

## Voici quelques exemples!

Envoyez-nous en plus !
uClinux est souvent si profondément embarqué qu'il est difficile de l'identifier.

### Périphériques réseau

VPN/Routeur SnapGear LITE2

StarDot NetCam

### Multimédia

Lecteurs DVD basés sur le Sigma Designs EM8500

Aplio/PRO IP Phone

# Histoire d'uClinux

- Première sortie en 1998 (Linux 2.0), pour le processeur Motorola 68000. Démonstration sur un Palm Pilot III.
- 1999 : Support de Motorola ColdFire.
- 2001 : Support de Linux 2.4. Support de l'ARM7.
- 2004 : Support de Linux 2.6. Support de l'ARM.
- 2004 : Vous lisez ce document.





# Raisons pour utiliser uClinux

- Linux
  Connectivité IP intégrée, fiabilité, portabilité, systèmes de fichiers, logiciels libres...
- Léger
  Noyau Linux 2.6 complet en moins de 300 Ko, binaires bien plus petits avec uClibc.
- XIP (Execute In Place)
  N'a pas besoin de charger les exécutables en mémoire, même si cela peut avoir des conséquences sur les performances.
- Moins cher
  Coeurs sans MMU environ 30% plus petits. MMU pas nécessaire dans de nombreuses applications de systèmes embarqués.
- Plus rapide
  Basculement de contexte plus rapide : pas de vidage des caches
- L'utilisateur accède au matériel
  Les applications utilisateur peuvent accéder au système complet, y compris aux registres des périphériques.
- API Linux complète
  Peut utiliser la plupart des appels système Linux avec quelques rares exceptions. Applications portées distribuées avec uClinux.
- Fonctionnalités complète du noyau Linux 2.6
  stabilité, noyau préemptif, pilotes...
- Multi-tâches complet
  Quelques limitations mineures
- Supporté par un grand nombre de processeurs
  Voir http://www.uclinux.org/ports/

# uClibc

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

- Bibliothèque C légère pour les petits systèmes embarqués, mais avec la plupart des fonctionnalités.
- Développé à l'origine pour uClinux. Maintenant un projet indépendant.
- La Debian Woody complète a été récemment portée... Vous pouvez être certain qu'elle subviendra à tous vos besoins.
- Exemple de taille (ARM) : approx. 400 Ko (libuClibc : 300 Ko, libm : 55Ko)
- Exemple d'un programme “hello world” : 2 Ko (lié dynamiquement), 18 Ko (lié statiquement).

# Limitations de uClinux

Rapidement

- Mémoire virtuelle = Mémoire physique
- Mémoire fixe pour les processus, impossible de fragmenter la mémoire : plus de consommation de mémoire
- Pas de protection de la mémoire

Détails très utiles (par David McCullough)
http://www.linuxjournal.com/article.php?sid=7221




# Pas de gestion de la mémoire

- Pas de mémoire virtuelle
  Les adresses des programmes doivent être traitées («resituées») avant l'exécution pour obtenir des espaces d'adressage unique.
- Pas de protection de la mémoire
  Tout programme peut bloquer un autre programme ou le noyau. La corruption peut ne pas être apparente et se révéler bien plus tard... difficile à tracer ! Concevez votre code avec attention. Méfiez-vous des données provenant de l'extérieur !
- Swapping impossible
  Pas vraiment un problème dans le cas des petits périphériques embarqués

# Meilleure performance

uClinux peut être significativement plus rapide que Linux sur le même processeur !

- Les opérations MMU peuvent représenter une surcharge significative. Même lorsqu'une MMU est disponible, elle est souvent désactivée dans les systèmes temps-réel.
- Le basculement de contexte peut être beaucoup plus rapide sur uClinux. Sur l'ARM9, par exemple, le cache basé en VM doit être vidé à chaque changement de contexte. Inutile lorsque tous les processus gèrent le même espace d'adressage.
  Voir un comparatif intéressant de H.S. Choi et H.C. Yun :
  http://opensrc.sec.samsung.com/document/uc-linux-04_sait.pdf

# Différences au niveau du noyau

- Format exécutable différent

  Les formats standard sont liés à la MV.

  *format plat («flat»)*: un format exécutable condensé, ne stockant que le code exécutable et les données, plus les replacements nécessaires pour charger l'exécutable à n'importe quel emplacement en mémoire.

- Implémentation mmap() différente

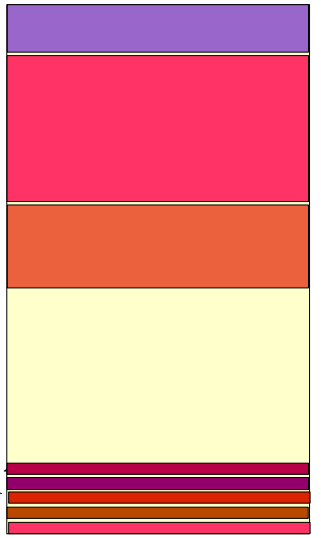
  Nécessaire de garantir que les données du fichier sont stockées de façon continue. Besoin d'allouer de la mémoire pour garantir ceci, sauf si c'est garanti par le système de fichiers (seul romfs le fait).

# Allocation mémoire

- Système d'allocation standard de Linux
  Alloue des blocs d'une taille de $2^n$.
  S'il y a une demande de mémoire pour une application de 65 Ko, 128 Ko seront alloués, et les 63 Ko restant ne seront pas réutilisables dans uClinux.
- Système d'allocation mémoire uClinux :

  `kmalloc2` (alias `page_alloc2`)

  - Alloue des blocs d'une taille de $2^n$ jusqu'à 4 Ko
  - Utilise des pages de 4 Ko
    pour des demandes plus importantes
  - Ne stocke pas plus de 8 Ko au début de la mémoire,
    et des parts plus importantes à la fin. Réduit la fragmen

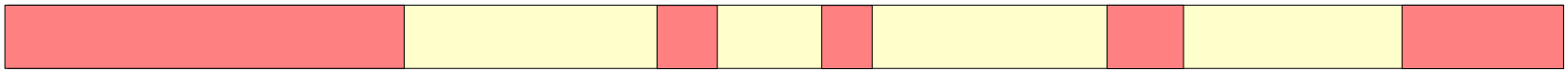







# Pas de pile dynamique

- La taille de la pile doit être allouée au moment de la compilation.
  4 Ko par défaut. Vérifiez si c'est suffisant en cas de problèmes étranges :
  - Soit vous recompilez : lancez `export FLTFLAGS=-s <stacksize>` pour le Makefile
  - Soit vous lancez `flthdr -s <stacksize> <executable>`
- Pas de taille dynamique pour le tas
  - Un processus fou peut prendre toute la mémoire disponible
  - Il peut être impossible d'allouer assez de mémoire contiguë
  - De telles situations peuvent être détectées via `/proc/mem_map` (kmalloc2)



Impossible d'allouer la mémoire pour ce bloc supplémentaire alors qu'il s'agit de moins que la moitié de la mémoire libre





# Pas de fork()

- `fork()` traditionnel

  Le processus fils est un clone du parent, utilisant le même espace mémoire virtuel. Les nouvelles allocations de mémoire n'interviennent pour le fils uniquement lorsqu'il modifie une page (“copy on write”).

- uClinux implémente seulement `vfork()`
  - l'exécution du processus parent est stoppé tant que le fils n'appelle pas `exec()` ou `exit()`
  - La mémoire est créée avant que le processus fils ne soit exécuté
  - Besoin de remplacer tous les `fork()` par des `vfork()`
  - Pas d'impact significatif sur le multi-tâche.

# Execute In Place (XIP)

- Permet de lancer une application sans la charger en RAM.
- S'applique aussi aux multiples instances du même programme. Sauve beaucoup de RAM !
- Supporté seulement par romfs (a besoin d'une stockage contigu, non compressé)
- Supporté seulement par la méthode PIC (Position-Independent Code) du format plat (doit être supporté par le compilateur)
- Attention : XIP pourrait être bien plus lent si le temps d'accès au stockage est important.

# Bibliothèques partagées

- Vraiment différent sous uClinux
- Différentes options de compilation... pas très familier.
- Créées au format “flat”, tout comme les applications
- Doit être compilé pour XIP. Sans XIP, les bibliothèques partagées résultent en une copie complète de la bibliothèque pour chaque application l'utilisant, ce qui est pire que l'édition de liens statiques pour vos applications.

# uClinux et Linux 2.6

- La plupart du code de uClinux est maintenant intégré avec les sources de Linux.
- La partie m68k de uClinux est maintenant disponible directement : `arch/m68knommu` (pour les processus embarqués m68k de Motorola)
- Autres architectures supportés : la série H8/300 d'Hitachi, le processeur NEC v850
- `arch/armnommu` non intégré pour l'instant. Toujours disponible en tant que correctif séparé.
- Linux 2.6 peut être construit sans le système de mémoire virtuelle.

# uClinux sur les ARM sans MMU

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

- Correctifs pour le noyau Linux 2.6 standard disponible chez Hyok S. Choi (correctifs « -hsc »)
- Processeurs supportés
  - ARM7TDMI : Atmel AT91xxx, Samsung S3C3410X, S3C4510B, S3C44B0
  - ARM920T : S5C7375

# uClinux sur plateformes m68knommu

- Disponible avec le noyau Linux standard (`arch/m68knommu`)
- Processeurs supportés : très longue liste ! (voir `arch/m68knommu/Kconfig`)

# Résumé

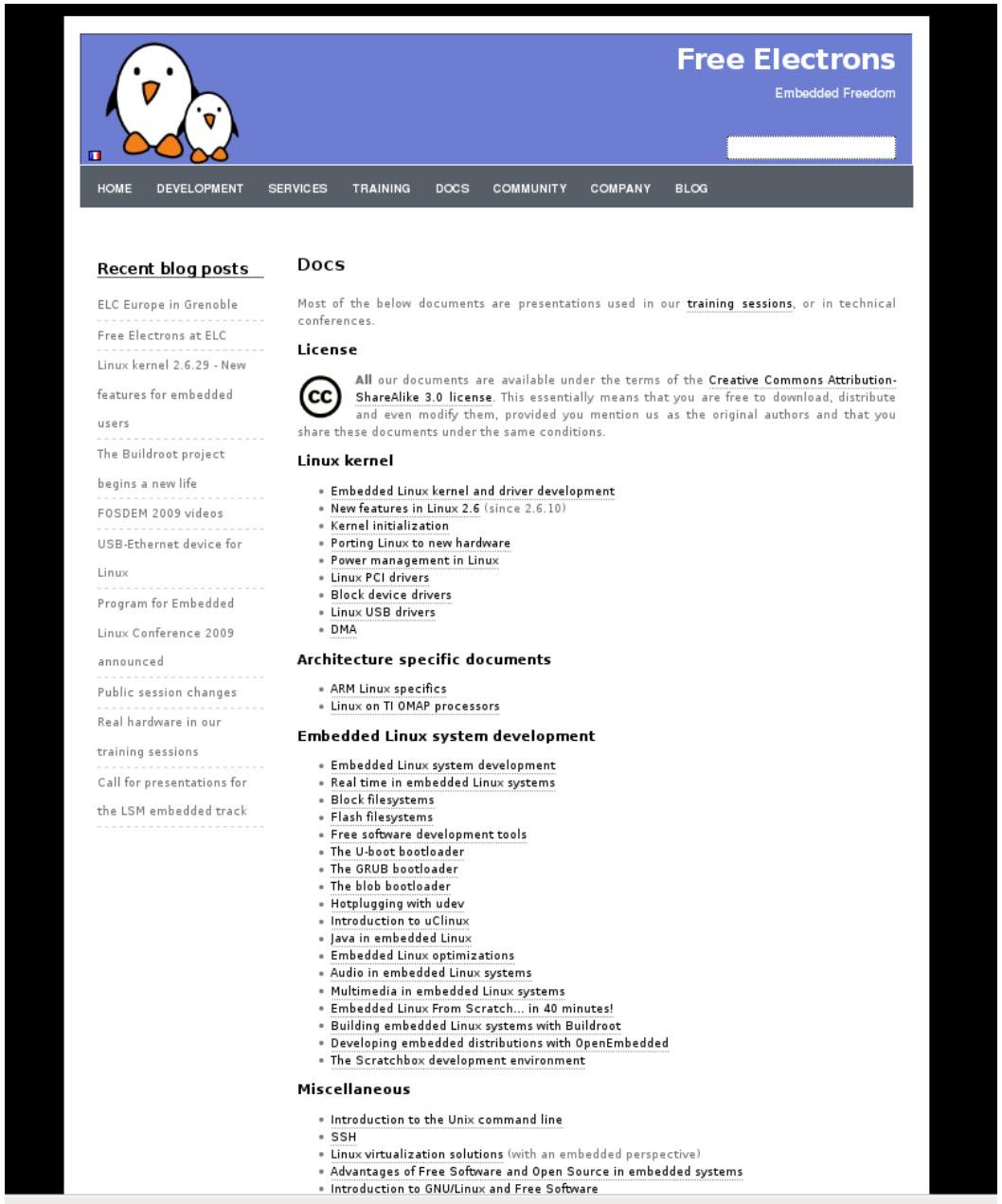
- Utilisation très intéressant de uClinux (plutôt qu'autre chose) sur les processeurs sans MMU.
- Dans certains cas, il est préférable d'utiliser uClinux plutôt que Linux sur les processeurs avec MMU, pour des raisons de performance.
- Bien que beaucoup de travail ait déjà été fait et que la plupart des applications ont déjà été portées, une expérience de uClinux ou une formation est nécessaire lorsque vous lancez un nouveau projet.

# Ressources

- http://ucdot.org/
  Beaucoup de ressources, FAQs, forums pour les développeurs sur uClinux !
  N'oubliez pas la FAQ très complète :
  http://www.ucdot.org/faq.pl

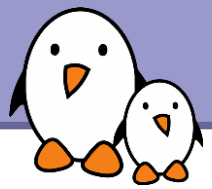



# Related documents




All our technical presentations on http://free-electrons.com/docs

- Linux kernel
- Device drivers
- Architecture specifics
- Embedded Linux system development



# How to help

You can help us to improve and maintain this document...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order development, consulting and training services performed by the authors of these documents (see http://free-electrons.com/).
- By sharing this document with your friends, colleagues and with the local Free Software community.
- By adding links on your website to our on-line materials, to increase their visibility in search engine results.

## Linux kernel

Linux device drivers
Board support code
Mainstreaming kernel code
Kernel debugging

## Embedded Linux Training

***All materials released with a free license!***

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

# Free Electrons

## Our services

### Custom Development

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

### Consulting and technical support

Help in decision making
System architecture
System design and performance review
Development tool and application support
Investigating issues and fixing tool bugs