

Embedded Linux From Scratch

Embedded Linux From Scratch

em 40 minutes!

Michael Opdenacker

Free Electrons

<http://free-electrons.com/>

nada + 40 min =



Traduzido por Brivaldo Jr

Criado com [OpenOffice.org](http://openoffice.org) 2.x



Direitos de Cópia



Attribution – ShareAlike 2.5

Você é livre para

- Copiar, distribuir, mostrar, e adaptar o trabalho
- Para fazer trabalhos derivados
- Para fazer uso comercial do trabalho

Seguindo certas condições

Atribuição. Você deve dar os devidos créditos ao autor original.



Compartilhar. Se você altera, transformar ou construir em cima deste trabalho, você deverá distribuir o trabalho resultante somente sobre uma licença idêntica a esta.



- Para qualquer reuso ou distribuição, você deve deixar claro aos outros os termos de licença deste trabalho.
- Qualquer destas condições podem ser modificadas se você tiver permissão do autor original.

Se uso e outros direitos não são afetados pelas regras acima.

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

© Copyright 2005-2007

Free Electrons

feedback@free-electrons.com

Document sources, updates and translations:

<http://free-electrons.com/articles/elfs>

Corrections, suggestions, contributions and translations are welcome!



Melhor visualizado com...

Este documento é melhor visto com um leitor de PDF recente ou com o próprio OpenOffice.org!

- ▶ Tire vantagens dos hyperlinks internos e externos. Então, não exite, clique neles!
- ▶ Ache páginas rapidamente graças a busca automática.
- ▶ Use os thumbnails para navegar pelo documento de uma forma rápida.

Se você está lendo um paper ou uma cópia em HTML, você poderá adquirir uma cópia em PDF ou no formato do OpenOffice.org em <http://free-electrons.com/articles/elfs!>



Objetivos do Tutorial

Construir um sistema mínimo embarcado totalmente do zero (*scratch*), em 40 minutos.

- ▶ Configuração e compilação do kernel Linux kernel
- ▶ Criação do sistema de arquivos raiz (*root*)
- ▶ Compilação e instalação do Busybox
- ▶ Criação dos arquivos de dispositivos
- ▶ Scripts de inicialização do sistema: sistema de arquivos virtual, rede
- ▶ Configuração de uma interface simples em HTTP para o sistema

Mostrar a vocês como isso pode ser muito simples!



Abordagem Top-down

A abordagem de construção do sistema embarcado Top-down

- ▶ Iniciar com um desktop completo do GNU/Linux (Debian, Fedora...) e remover as funcionalidades desnecessárias.
- ▶ Trabalho muito tedioso: necessita conhecer um número enorme de pacotes e arquivos. Necessita saber o que cada arquivo de cada pacote faz antes de tentar removê-lo.
- ▶ Manutenção de um complexo de scripts e arquivos de configuração desnecessários.
- ▶ O resultado final é um tanto grande, como um conjunto de ferramentas de um desktop padrão que são usados. Várias bibliotecas compartilhadas são utilizadas também.



Abordagem Bottom-up

A abordagem de construção do sistema embarcado Bottom-up

- ▶ Começa com um sistema de arquivos raiz vazio ou minimalista, adicionando somente as coisas que você precisa.
- ▶ Muito fácil de fazer! Você só precisa despendar tempo nas coisas que você precisa.
- ▶ Muito fácil de controlar e manter: você constroi e entende sobre as ferramentas que você usa.
- ▶ Você somente precisa de scripts de configuração muito simples.
- ▶ O resultado final pode ser extremamente pequeno.



Embedded Linux From Scratch

Ferramentas utilizadas neste tutorial
Slides explicativos serão mostrados enquanto
compilando



qemu

<http://fabrice.bellard.free.fr/qemu/>
Emulador de processador *rápido*
usando um tradutor dinâmico portátil.



2 modos de operação

- ▶ Emulação de sistema completa: processador e vários periféricos
Suportado: **x86**, **x86_64**, **ppc**, **arm**, **sparc**, **mips**, **m68k**
- ▶ Emulação User mode (hosts **Linux** somente): pode rodar aplicações compiladas para outra CPU.
Suportado: **x86**, **ppc**, **arm**, **sparc**, **mips**, **m68k**



Exemplos qemu

Emulação User Mode

- ▶ Fácil de rodar o Busybox para arm no i386 GNU / Linux:
`qemu-arm -L /usr/local/arm/3.3.2 \`
`/home/bart/arm/busybox-1.00-pre8/busybox ls`
- ▶ `-L`: caminho para as bibliotecas binárias do C (aqui o caminho das ferramentas de compilação cruzada)

Emulação do Sistema

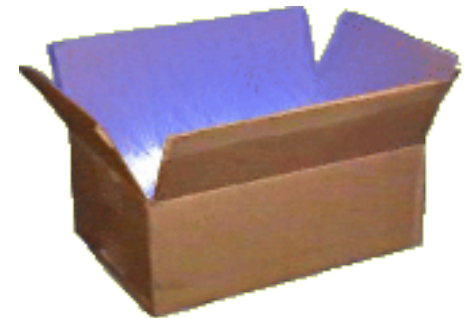
- ▶ Fácil de rodar:
`qemu linux.img`
- ▶ `linux.img`: imagem de partição completa incluindo o kernel.
Muitas imagens livres de sistemas operacionais podem ser encontradas em
<http://free.oszoo.org>!



Ferramenta de propósito geral: busybox

<http://www.busybox.net/> da empresa Codepoet Consulting

- ▶ A maioria dos utilitários de linha de comando em um único executável! Incluindo também um servidor web!
- ▶ Tamanho inferior a 1 MB (compilado estaticamente com a [glibc](#)) menor que 500 KB (compilado estaticamente com a [uClibc](#))
- ▶ Fácil de configurar com opções inclusas
- ▶ A melhor escolha para
 - ▶ Initrds com scripts complexos
 - ▶ Qualquer sistema embarcado!



Comandos do Busybox!

addgroup, adduser, adjtimex, ar, arping, ash, awk, basename, bunzip2, bzcat, cal, cat, chgrp, chmod, chown, chroot, chvt, clear, cmp, cp, cpio, crond, crontab, cut, date, dc, dd, dealloct, delgroup, deluser, devfsd, df, dirname, dmesg, dos2unix, **dpkg**, dpkg-deb, du, dumpkmap, dumpleases, echo, egrep, env, expr, false, fbset, fdflush, fdformat, fdisk, fgrep, find, fold, free, freeramdisk, fsck.minix, ftpget, ftpput, getopt, getty, grep, gunzip, gzip, halt, hdparm, head, hexdump, hostid, hostname, **httpd**, hush, hwclock, id, ifconfig, ifdown, ifup, inetd, init, insmod, install, ip, ipaddr, ipcalc, iplink, iproute, iptunnel, kill, killall, **klogd**, lash, last, length, linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread, losetup, ls, lsmod, makedevs, md5sum, msg, mkdir, mkfifo, mkfs.minix, mknod, mkswap, mktemp, modprobe, more, mount, msh, mt, mv, nameif, nc, netstat, nslookup, od, openvt, passwd, patch, pidof, ping, ping6, pipe_progress, pivot_root, poweroff, printf, ps, pwd, rdate, readlink, realpath, reboot, renice, reset, rm, rmdir, rmmod, route, **rpm**, rpm2cpio, run-parts, rx, sed, seq, setkeycodes, shasum, sleep, sort, start-stop-daemon, strings, stty, su, sulogin, swapoff, swapon, sync, sysctl, syslogd, tail, tar, tee, telnet, **telnetd**, test, tftp, time, top, touch, tr, traceroute, true, tty, **udhcpc**, **udhcpd**, umount, uname, uncompress, uniq, unix2dos, unzip, uptime, usleep, uudecode, uuencode, vconfig, **vi**, vlock, watch, watchdog, wc, **wget**, which, who, whoami, xargs, yes, zcat

Embedded Linux From Scratch ... in 40 minutes!

© Copyright 2005-2007, Free Electrons

Creative Commons Attribution-ShareAlike 2.5 license

<http://free-electrons.com>

Sep 15, 2009



glibc

<http://www.gnu.org/software/libc/>

- ▶ Bibliotecas de C do projecto **GNU**
- ▶ Desenhado para performance, compatibilidade com padrões e portabilidade
- ▶ Encontrado em todos os sistemas **GNU / Linux**
- ▶ Um pouco grande para sistemas embarcados: em torno de 1.7MB na **Familia Linux** iPAQs (**libc**: 1.2 MB, **libm**: 500 KB)
- ▶ Exemplo de tamanho de um programa “hello world” tamanho do programa: 12 KB (linkada dinamicamente), 350 KB (statically linked)



uClibc

<http://www.uclibc.org/> from CodePoet Consulting

- ▶ Biblioteca leve de C para sistemas pequenos e embarcados, com a maioria das funcionalidades.
- ▶ O **Debian Woody** foi recentemente todo portado para ele... Nós podemos assumir que satisfaz todas as necessidades!
- ▶ Exemplo de tamanho (**arm**): aprox. 400KB
(**libuClibc**: 300 KB, **libm**: 55KB)
- ▶ O tamanho de um programa exemplo “hello world” tem tamanhos: 2 KB (linkado dinamicamente), 18 KB (linkado estaticamente).



Interface no espaço de usuário do Kernel

Alguns exemplos:

- ▶ `/proc/cpuinfo`: informação de processador
- ▶ `/proc/meminfo`: estado da memória
- ▶ `/proc/version`: informação de versão e construção
- ▶ `/proc/cmdline`: linha de comando do kernel
- ▶ `/proc/<pid>/environ`: ambiente de chamadas
- ▶ `/proc/<pid>/cmdline`: linha de comando do processo

... e muito mais! Todos os detalhes nos fontes do kernel:

`Documentation/filesystems/proc.txt`



Embedded Linux From Scratch

O que nós faremos



Compilando o kernel do Linux

- ▶ Baixe os fontes do Linux em <http://kernel.org>
- ▶ Comece com uma configuração do kernel minimalista:
`make allnoconfig`
- ▶ Adicionando configurações específicas para sistemas embarcados:
`make xconfig` or `make menuconfig`
(o arquivo de configuração do kernel que nós usaremos
http://free-electrons.com/doc/embedded_lfs/linux-2.6.15.config)
- ▶ Compilando:
`make`
- ▶ Resultado: imagem de kernel comprimido `arch/i386/boot/bzImage`



Criando o sistema de arquivos raiz

- ▶ Criando um arquivo fazio com 320K de tamanho:
`dd if=/dev/zero of=rootfs.img bs=320k count=1`
- ▶ Formatando o arquivo com o sistema de arquivos `ext2`:
`mkfs.ext2 -i 1024 -F rootfs.img`
1 inode every 1024 bytes -> 320 files
instead of 1 inode every 8192 bytes -> only 40 files!



Compilando o busybox

- ▶ Pegando os fontes de <http://busybox.net>
- ▶ Configurando o busybox:
`make menuconfig`
Escolha construir estaticamente, compilando o executável estaticamente.
- ▶ Compilando o busybox:
`make`
- ▶ Pre-instalando o busybox (no subdiretório `_install/`):
`make install`
- ▶ Resultado: um arquivo executável de **500K** implementando todos os comandos que nós precisamos!



Re-compilando o busybox

500K tem um tamanho perfeito para sistemas embarcados!



- ▶ Configurando o busybox denovo

`make menuconfig`

Escolhendo a construção estática, um executável “cross-compilado”, usando as ferramentas `uClibc` ao invés da padrão `glibc`.

- ▶ Compilando o busybox:

`make`

- ▶ Pre-instalando o busybox (no subdiretório `_install/`):

`make install`

- ▶ Resultando: um executável com 250K implementando todos os comandos que nós precisamos!



Populando o sistema de arquivos raiz

Logando como `root`:

- ▶ Criando um ponto de montagem:

```
mkdir /mnt/rootfs
```

- ▶ Montando a imagem do sistema de arquivos:

```
mount -o loop rootfs.img /mnt/rootfs
```

- ▶ Copiando a estrutura de arquivos do busybox para a imagem montada:

```
rsync -a busybox/_install/ /mnt/rootfs/  
chown -R root:root /mnt/rootfs/
```

- ▶ Efetivando as mudanças na imagem do arquivo montado:

```
sync
```



Inicializando o sistema virtual

Usando o emulador **qemu** como um bootloader
(não é necessário copiar o kernel para o alvo de armazenamento)

```
qemu \
  -m 32 \      Quantidade de memória (MB) na máquina emulada
  -hda rootfs.img \  Conteúdo do disco emulado
  -kernel linux-2.6.12/arch/i386/boot/bzImage \
                                     Imagem do Kernel
  -append "root=/dev/hda clock=pit"
                                     Linha de comando do Kernel
```



Criando arquivos de dispositivo

- ▶ Criando arquivos de dispositivos para o sistema:

```
mkdir /mnt/rootfs/dev
```

```
mknod /mnt/rootfs/dev/console c 5 1
```

```
mknod /mnt/rootfs/dev/null c 1 3
```

- ▶ Usando o sistema GNU/Linux base como exemplo para encontrar os números maiores e menores:

```
ls -l /dev/console
```

```
ls -l /dev/null
```



Montando o sistema de arquivos virtual

Tornando o `/proc` e o `/sys` disponíveis
(necessário por diversos programas de linha de comando como o `ps`)

▶ Montando `/proc`:

```
mount -t proc none /proc
```

▶ Montando `/sys`:

```
mount -t sysfs none /sys
```

Tipo do Sistema de Arquivos

Raw device
or filesystem image
In the case of virtual
filesystems, any string is fine

Ponto de Montagem



Arquivo /etc/inittab para o início do busybox

Criando o arquivo `/etc/inittab` requerido pelo busybox `init`

Tirando um exemplo da documentação do busybox
(não do host GNU/Linux... faltam funcionalidades!)

```
# Este é o primeiro script
::sysinit:/etc/init.d/rcS
# Comece com o shell "askfirst" no console
::askfirst:-/bin/sh
# O que fazer quando restartar um processo init
::restart:/sbin/init
# O que fazer antes de reiniciar
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r
```



Ativando a rede

▶ Ativando o TCP/IP e o driver da placa de rede no kernel

▶ Levantando a interface de rede:

```
ifconfig eth0 172.20.0.2
```

▶ Usando o host GNU/Linux como gateway:

```
route add default gw 172.20.0.1
```

▶ Testando a rede:

```
ping -c 3 172.20.0.1
```

-c 3: útil quando o [Ctrl][C] não funciona
(configuração de tty perdidas)

▶ Testando a rota:

```
ping -c 3 <endereço externo>
```



Inicializando o servidor http

- ▶ Copiando as páginas HTML em `/www` (por exemplo)
- ▶ Criando scripts CGI em `/www/cgi-bin/`
- ▶ Inicializando o servidor http do busybox:
`/usr/sbin/httpd -h /www/ &`



Script de inicialização /etc/init.d/rcS

```
#!/bin/sh
mount -t proc none /proc
mount -t sysfs none /sys
ifconfig eth0 172.20.0.2
route add default gw 172.20.0.1
/usr/sbin/httpd -h /www/ &
/bin/sh
```

Veja como isso pode ser simples!



Erros comuns /etc/init.d/rcS

- ▶ Não esqueça o `#!/bin/sh` no início de cada shell script! Sem o caractere `#!`, o kernel Linux não tem como saber que isso é um shell script e irá tentar executar como um binário!
- ▶ Em nosso exemplo, não esqueça de iniciar o shell no final do script. Por outro lado, a execução irá parar sem pedir que você digite nenhum comando!



Um script CGI simples

```
/www/cgi-bin/uptime:
```

```
#!/bin/sh
```

```
echo "Content-type: text/html"
```

```
echo ""
```

```
echo "<html><header></header><body>"
```

```
echo "<h1>Uptime information</h1>"
```

```
echo "Your embedded device has been  
running for:<pre><font color=Blue>"
```

```
echo `uptime`
```

```
echo "</font></pre></u>"
```

```
echo "</body></html>"
```



Limitações

Algumas pequenas limitações

- ▶ CGI scripts: não podem implementar scripts não triviais
Precisam de codificação em C para suportar a análise de passagens de parâmetro em URL.
- ▶ Software específico de sistema: não podem fazer parte do busybox.
Precisam de mais executáveis em C. E como consequência precisam incluir a biblioteca **uClibc** e compilar os executáveis com suporte a compartilhamento de bibliotecas.

Eles são simples e baratos de fazer!



Sistemas embarcados Reais

Este tutorial foi testado em placas de desenvolvimento real!

- ▶ Precisa instalar e configurar um bootloader (se perdido)
- ▶ Precisa transferir o kernel e uma imagem do sistema raiz para o alvo. Um modo eficiente é fazer o alvo inicializar em um sistema NFS exportado do diretório no host GNU/Linux.



Obrigado

▶ As pessoas que enviaram correções e sugestões:
Florent Peyraud, Fabrice Menard, Robert P. J. Day.

▶ Aos tradutores:

Brivaldo Jr (Portuguese)

Guillaume Lelarge (French)





Related documents



Free Electrons

Embedded Freedom

HOME DEVELOPMENT SERVICES TRAINING DOCS COMMUNITY COMPANY BLOG

Recent blog posts

ELC Europe in Grenoble

Free Electrons at ELC

Linux kernel 2.6.29 - New features for embedded users

The Buildroot project begins a new life

FOSDEM 2009 videos

USB-Ethernet device for Linux

Program for Embedded Linux Conference 2009 announced

Public session changes

Real hardware in our training sessions

Call for presentations for the LSM embedded track

Docs

Most of the below documents are presentations used in our [training sessions](#), or in technical conferences.

License

 All our documents are available under the terms of the [Creative Commons Attribution-ShareAlike 3.0 license](#). This essentially means that you are free to download, distribute and even modify them, provided you mention us as the original authors and that you share these documents under the same conditions.

Linux kernel

- [Embedded Linux kernel and driver development](#)
- [New features in Linux 2.6](#) (since 2.6.10)
- [Kernel initialization](#)
- [Porting Linux to new hardware](#)
- [Power management in Linux](#)
- [Linux PCI drivers](#)
- [Block device drivers](#)
- [Linux USB drivers](#)
- [DMA](#)

Architecture specific documents

- [ARM Linux specifics](#)
- [Linux on TI OMAP processors](#)

Embedded Linux system development

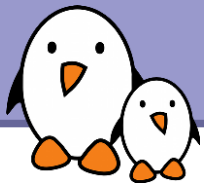
- [Embedded Linux system development](#)
- [Real time in embedded Linux systems](#)
- [Block filesystems](#)
- [Flash filesystems](#)
- [Free software development tools](#)
- [The U-boot bootloader](#)
- [The GRUB bootloader](#)
- [The blob bootloader](#)
- [Hotplugging with udev](#)
- [Introduction to uClinux](#)
- [Java in embedded Linux](#)
- [Embedded Linux optimizations](#)
- [Audio in embedded Linux systems](#)
- [Multimedia in embedded Linux systems](#)
- [Embedded Linux From Scratch... in 40 minutes!](#)
- [Building embedded Linux systems with Buildroot](#)
- [Developing embedded distributions with OpenEmbedded](#)
- [The Scratchbox development environment](#)

Miscellaneous

- [Introduction to the Unix command line](#)
- [SSH](#)
- [Linux virtualization solutions](#) (with an embedded perspective)
- [Advantages of Free Software and Open Source in embedded systems](#)
- [Introduction to GNU/Linux and Free Software](#)

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

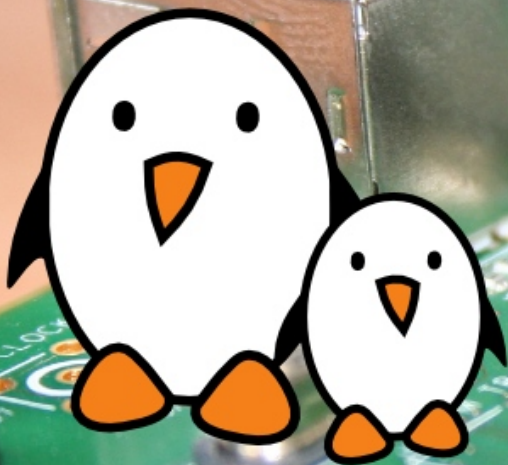
Our services

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



Free Electrons
Embedded Linux Experts

<http://free-electrons.com>