



# Sistemas embarcados com Linux - primeiros passos -



Eng. Dr. Marcelo Barros de Almeida  
[marcelobarrosalmeida@gmail.com](mailto:marcelobarrosalmeida@gmail.com)

**smar**

Smar Equipamentos Industriais LTDA



# Direitos de cópia



## Créditos - ShareAlike 2.0

### Você é livre para

- copiar, distribuir, apresentar e executar trabalhos
- fazer trabalhos derivados
- fazer uso comercial deste trabalho

### Sob as seguintes condições

Créditos. Você deve dar crédito ao autor original.



**Compartilhe do mesmo modo.** Se você alterar, mudar, ou realizar trabalhos usando este como base, você deve redistribuir o trabalhos resultante sob uma licença idêntica a esta.



- Para qualquer reuso ou distribuição você deve deixar claro os termos de licença deste trabalho.
- Qualquer uma destas condições podem ser abandonadas se você obtiver uma permissão do detentor dos direitos autorais.

**Faça uso justo e o direitos dos outros não serão afetados de forma alguma pelas restrições acima.**

Texto da licença:

<http://creativecommons.org/licenses/by-sa/2.0/legalcode>



© Copyright 2008 - Marcelo Barros  
[marcelobarrosalmeida@gmail.com](mailto:marcelobarrosalmeida@gmail.com)

Documentos originais, atualizações e traduções:

<http://linuxabordo.com.br/>

Correções, sugestões e traduções são bem vindas!



# Marcelo Barros ?



- Formação:

- Engenheiro eletrônico (EFEI, 1996)
- Mestre (UFMG, 1998)
- Doutor (UFMG, 2002)

- Atualmente:

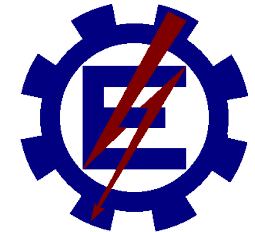
- Engenheiro (Smar Equip. Industriais LTDA)
- Professor do Barão de mauá

- Detalhes, currículo, blogs:

[http://linuxabordo.com.br/wiki/index.php?title=Marcelo\\_Barros](http://linuxabordo.com.br/wiki/index.php?title=Marcelo_Barros)

<http://jedizone.wordpress.com>

<http://twitter.com/marcelobarros>



**smar**



# Sistemas embarcados com Linux



- **Introdução**
  - **Definição e exemplos de sistemas embarcados**
  - **Mercado**
  - **Linux tradicional x Linux embarcado**
  - **Vantagens e cuidados**
- Pré-requisitos
- Criando sistemas embarcados
- Créditos, agradecimentos e links



# Definição de sistema embarcado



- O que exatamente significa “sistema embarcado” ?
  - Definição da Wikipedia: “*um computador de propósito especial, que é completamente encapsulado pelo dispositivo que controla*”.
- Definição muito abrangente. Algumas dicas:
  - Propósito específico
  - Microprocessado/Microcontrolado
  - Aplicação em ROM/Flash
  - Restrições de consumo ou tamanho são freqüentes
  - Requisitos de tempo real também



# Exemplos de sistemas embarcados



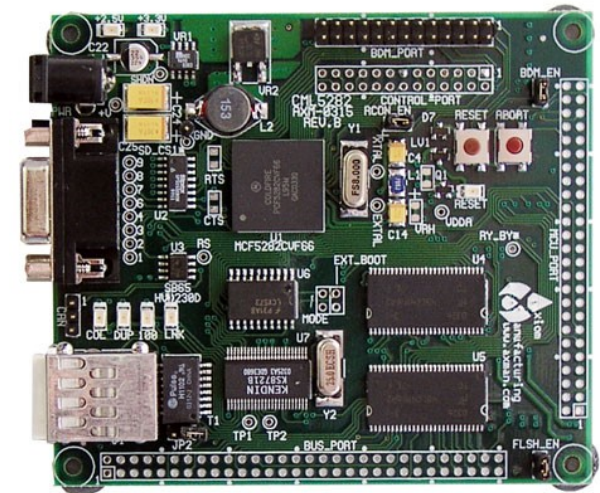
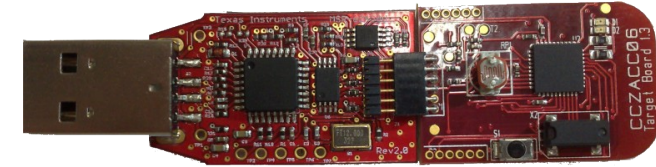
- Estamos rodeados de sistemas embarcados:
  - Alarmes automotivos, sistemas de airbag e ABS, computadores de bordo, injeção eletrônica, GPS
  - Máquinas de lavar, microondas, DVD/Media players, TV Digital
  - Calculadoras, videogames, PDAs
  - Semáforos, câmeras de vigilância, radares
  - Equipamentos para rede (roteadores, switches, modems)
  - Telefones celulares, centrais telefônicas
  - Equipamentos de controle industriais
- Existem muito mais processadores sendo usado em sistemas embarcados do que em PCs



# Exemplos de sistemas embarcados



- pequenos: com apenas alguns kb de RAM e poucas dezenas de kb de memória não volátil (programa).
  - Ex: processadores MSP430 (Texas Instruments), com 1 kb de RAM e 60Kb de flash.
- médios: algumas centenas de kb de memória para programa, dezenas de kb de RAM.
  - Ex: inúmeros processadores, como a linha Coldfire (Freescale) ou mesmo AVR (Atmel).
- grandes: memória não volátil já em megabytes e RAM na casa das centenas.
- extra-large: dezenas de MB de flash e de RAM.
  - Vários fabricantes, principalmente baseados em processadores ARM, PowerPC e x86.



# Exemplos de sistemas embarcados com Linux



PDA Sharp Zauro SL-C3100



Multimídia Archos PMA400



Telefone VoIP WiFi Accton VM1188T



Relógio



Tablets



GPS



DVDs



Robôs



Telefones celulares



Roteador Linksys WTR54G



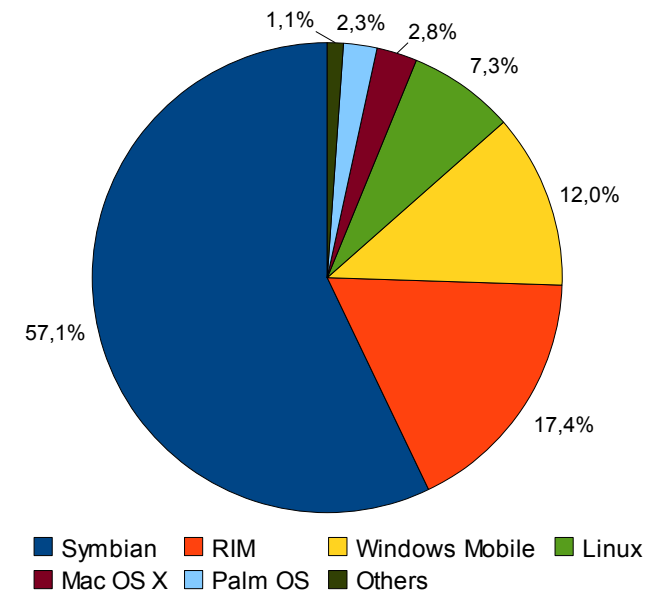
# Linux embarcado na telefonia celular



- Mercado aquecido, com grandes empresas apostando em Linux
  - Google/HTC (Android)
  - Motorola
  - Nokia (Maemo/Trolltech)
  - NEC
  - Panasonic
- Fundações privadas (LiMo) e abertas (Open Moko) gerando especificações e SDKs



Mercado de Smartphones



fonte: <http://www.linuxdevices.com/news/NS8289089946.html>



**Sistemas Embarcados com Linux – primeiros passos**

© Copyright 2008, Marcelo Barros de Almeida

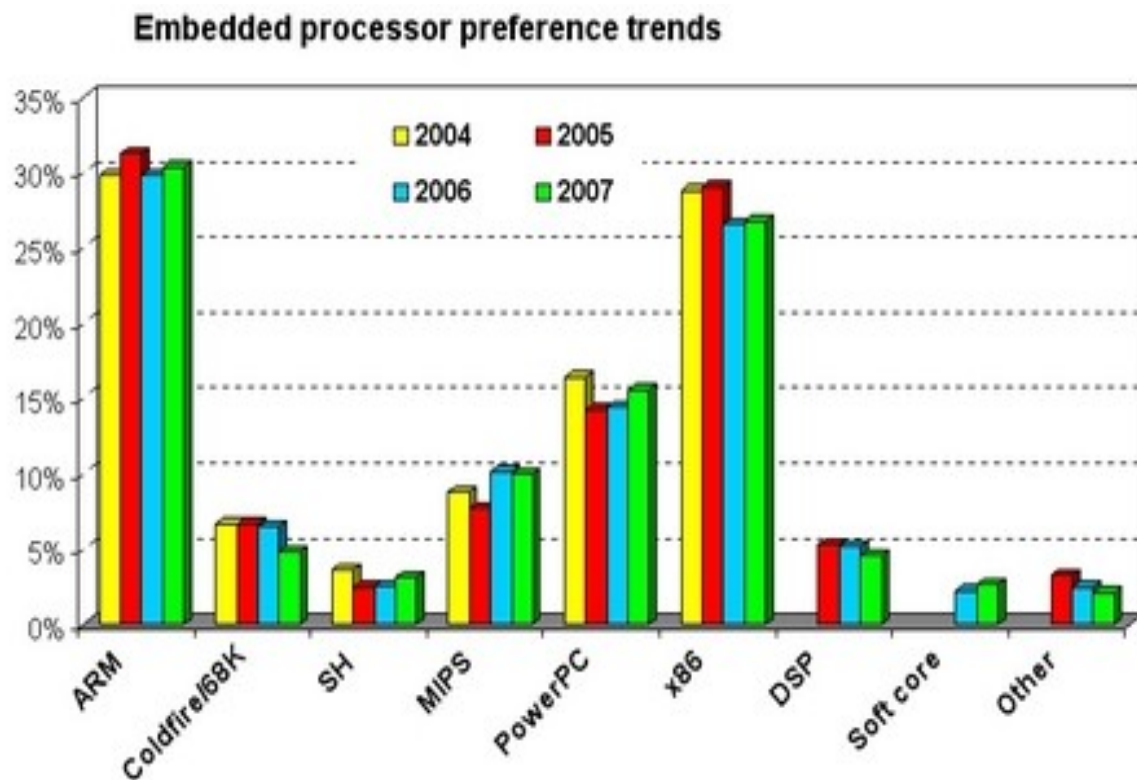
Licença Creative Commons Attribution-ShareAlike 2.0

<http://www.smar.com.br> <http://www.linuxabordo.com.br>

# Nem tudo é x86 no mundo de sistemas embarcados ...



Quais foram as CPU usada nos seus projetos recentes ?



Fonte: pesquisa espontânea realizada por linuxdevices.com (<http://linuxdevices.com/news/NS5319577519.html>)

**Sistemas Embarcados com Linux – primeiros passos**

© Copyright 2008, Marcelo Barros de Almeida

Licença Creative Commons Attribution-ShareAlike 2.0

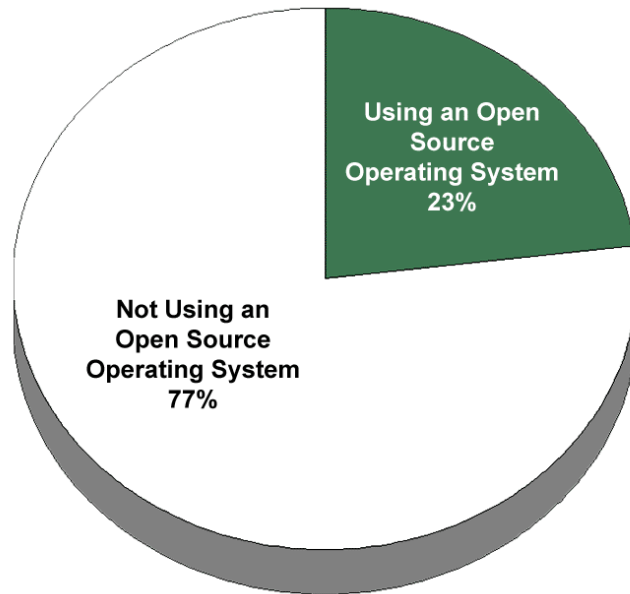
<http://www.smar.com.br> <http://www.linuxabordo.com.br>



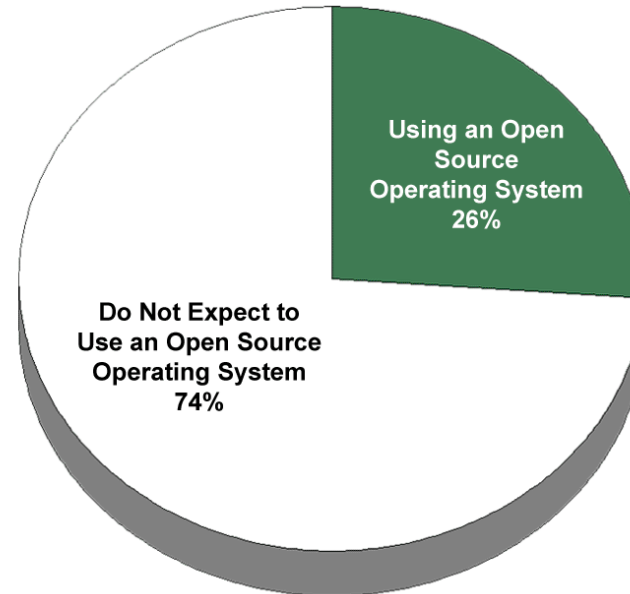
# Nem tudo é fechado no mundo de sistemas embarcados ...



**Percent of Respondents Reporting the Use of an Open Source Operating System on their Current Project**



**Percent of Respondents Expecting to Use an Open Source Operating System on their Next Project**



- Linux: 18%
- eCos, BSD, FreeRTOS, and TinyOS: 5%

Fonte: <http://www.linuxdevices.com/news/NS4920597981.html>



# Mas ser aberto não significa ser sempre de graça ...



My current embedded project uses ...

My next embedded project will likely use ...

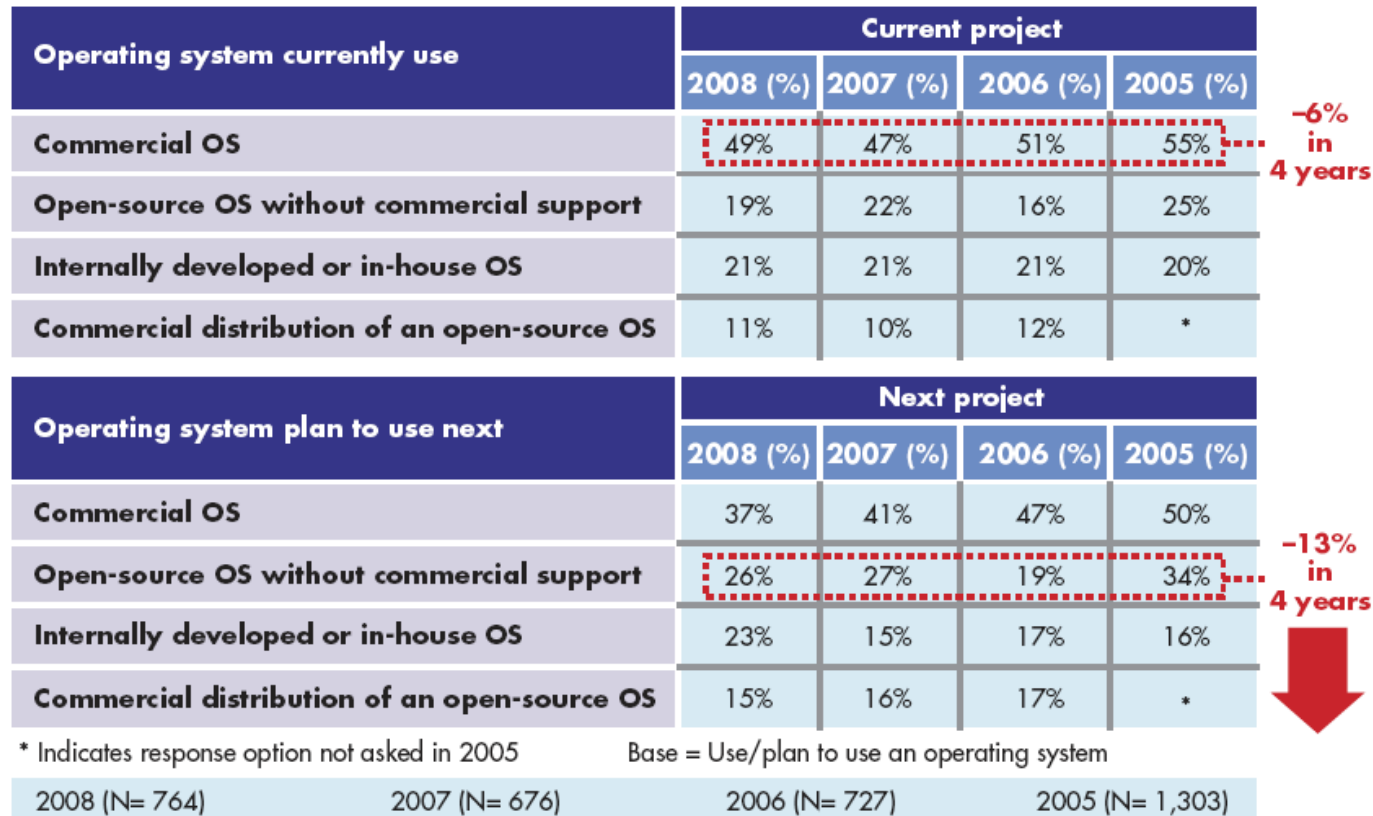


Figure 7

fonte: <http://www.embedded.com/products/softwaretools/210200580>



Sistemas Embarcados com Linux – primeiros passos

© Copyright 2008, Marcelo Barros de Almeida

Licença Creative Commons Attribution-ShareAlike 2.0

<http://www.smar.com.br>    <http://www.linuxabordo.com.br>

# Linux tradicional x Linux Embarcado



## GNU Tradicional / Sistema Linux



Navegador web, escritório, multimídia...



ls, vi, wget, ssh, httpd, gcc...

libjpeg, libstdc++, libxml, libvorbis...



Biblioteca GNU C



Kernel Linux

Kernel completo com a maioria das características e com *drivers* para todo tipo de hardware de PC do planeta!!

## Sistema Linux embarcado

Interface personalizada



busybox  
(ls, vi, wget, httpd...)  
dropbear (ssh)...

Gráficos,  
navegador web,  
servidor de web.

Implementações  
muito mais leves!  
Sem ferramentas de  
desenvolvimento.

libjpeg, libstdc++, libxml, libvorbis...

uClibc

Muito mais leve  
do que a  
biblioteca C GNU!



Kernel Linux / uClinux (sem MMU)

Kernel leve, somente  
com as características  
necessárias e *drivers*

Interface  
com o usuário

Utilitários de  
linha de  
comando

Bibliotecas  
compartilhadas

Biblioteca C

Kernel



# Vantagens de Linux embarcado



- Qualidade do código
- Footprint *relativamente* pequeno:
  - Kernel: 0,5 – 2MB de flash
  - Sistema de arquivos: variável
  - Mesmo assim, pode ser grande para algumas aplicações
- Portabilidade e escalabilidade
- Grande número de aplicativos disponíveis
- Possibilidade de custos reduzidos
- Suporte
  - Fórum, listas, email, FAQs, exemplos, suporte comercial disponível.



*Use the sources you must, Luke*



# Cuidados com Linux embarcado



- Linux é um sistema de propósito geral !
- Existem muitas opções de aplicativos, requer análise refinada
- Sistema em evolução constante
- Faça análises imparciais, evite o deslumbramento
- O formato das licenças deve ser verificado sempre



# Licenças e Linux embarcado



## GPL2:

*You may copy and distribute the Program (or a work based on it ...) ... provided that you ... accompany it with the complete corresponding machine-readable source code ...*

## Kernel Linux é GPL2, mas ...

*This copyright does **\*not\*** cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does **\*not\*** fall under the heading of "derived work" ... (Linus Torvalds)*

## Lesser GPL (LGPL):

*We use this license for certain libraries in order to permit linking those libraries into non-free programs ... As an exception ... you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications ...*



# Licenças e Linux embarcado



- GPL3

- Fortemente contra o DRM (Digital Rights Management)
- Inibe o uso de patentes
- Contra a “Tivoization”
- Até o momento, o Linux é GPLv2

- Alternativas:

- Uma lista interessante em:  
<http://debianlinux.net/os.html>
- NetBSD
- eCos (médio)
- freeRTOS (pequeno)



FREE SOFTWARE IS FREEDOM



St. IGNUcius



# Sistemas embarcados com Linux



- Introdução
- **Pré-requisitos**
  - Conhecimentos necessários
  - Equipamento necessários
  - Fazendo escolhas
- Criando sistemas embarcados
- Créditos, agradecimentos e links



# Conhecimentos necessários: Kernel Linux



- Linux
  - Operação e funcionamento do sistema Linux
  - Inicialização na plataforma desejada
  - Conhecimento da estrutura do Kernel
  - Compilação e instalação do Kernel
- Alguns detalhes do Linux\*:
  - Versão 2.6.12.5:
    - ~ 189MB
    - ~ 4,686 milhões de linhas
    - ~ 468 desenvolvedores

\* A study of Linux kernel evolution, Oded Koren, ACM SIGOPS Operating Systems Review, Volume 40 , Issue 2, pag. 110-112



# Conhecimentos necessários: ferramentas



- **Compiladores cruzados (toolchains)**

- Portes do GCC para a plataforma desejada (binutils/glibc/GCC/GDB,...)
- Download de toolchains pré-compilados ou compilação (pode ser facilitada com buildroot, CrossTool, OpenEmbedded, Scratchbox, T2 Project ...)

- **Emuladores**

- Qemu (x86/ARM/PPC/MIPS/Sparc)
- Específicos: Skyeye/Softgun/SWARM (ARM), Coldfire emulator

- **Outros**

- Conhecimentos de redes (Configuração, TFTP, NFS, ...)
- Uso de aplicativos como minicom (console serial), hexdump, conversores
- Controle de versão (CVS/Subversion/Git), patches, diffs, Makefiles, etc



# Conhecimentos necessários: programação



- Programar em C é obrigatório
- Assembly para a plataforma desejada pode ser necessário
- Um pouco de shell script não faz mal a ninguém
- Desenvolvimento de módulos e device drivers pode ser necessário
- Bônus track:
  - HTML, servidores HTTP
  - CGI e scripts (Python, PHP, Perl, etc)
  - Java



# Equipamentos necessários



- Estação de trabalho Linux
- Plataforma embarcada
- Equipamento para debug
- Cabos (serial e ethernet)
- Switches



Imagem: nbpfaus.net/~pfau/pictures/MyWorkstation.jpg





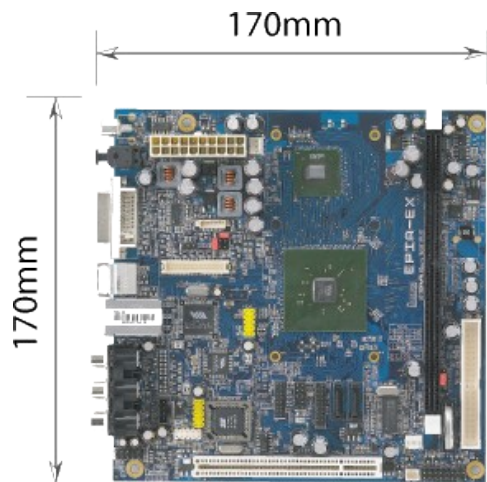
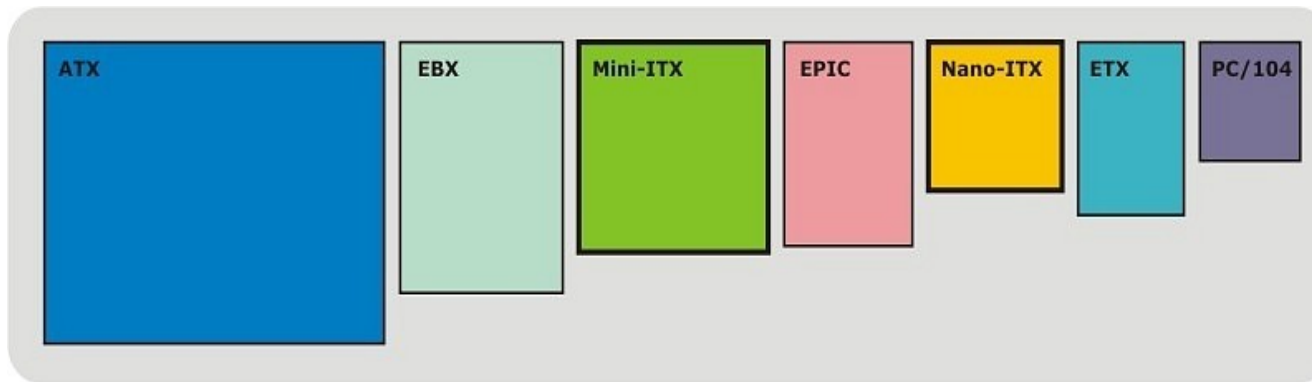
# Equipamentos necessários: plataforma embarcada



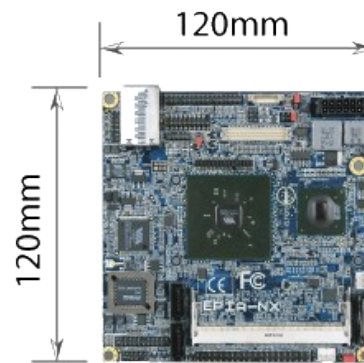
- Principais alternativas para a plataforma embarcada
  - Desenvolver tudo antes no desktop e recompilar depois para o sistema desejado
  - Emulação: Qemu ou outro emulador específico
  - PC (ATX, mini-ITX, nano-ITX)
  - PC/104
  - SBC (Single Board Computers)
  - Hardware hackeado (PDAs, celulares, MP3 players, roteadores, set top boxes, vídeo games, relógios, etc)
  - Hardware proprietário



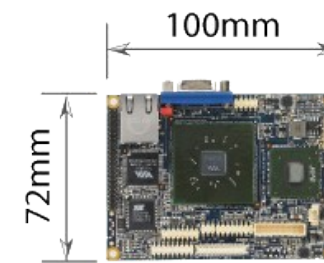
# Alguns form-factors para plataforma embarcada



Mini-ITX



Nano-ITX



Pico-ITX

<http://www.linuxdevices.com/articles/AT2614444132.html>

<http://www.via.com.tw/en/products/mainboards/>



# Exemplo de SBC: Atmel NGW100



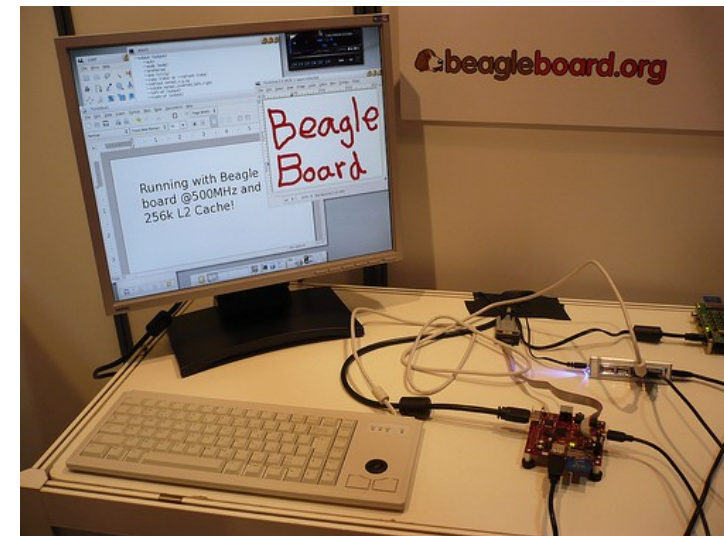
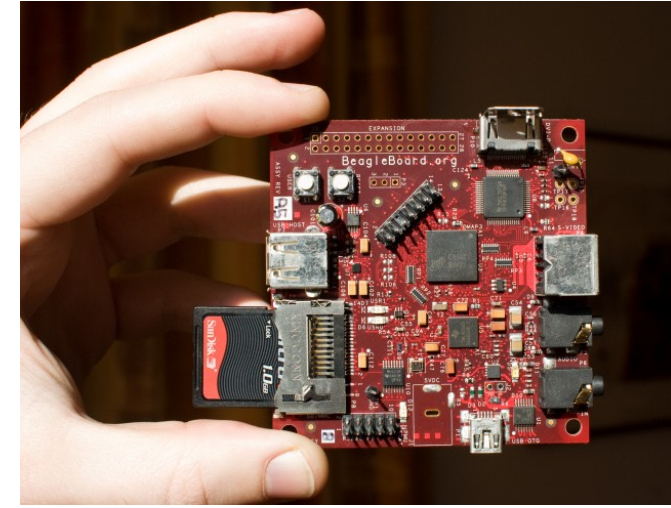
- Processador Atmel AVR32, 32AP7000, industrial, 150Mhz máximo e 210DMIPS.
- Dois controladores ethernet (34Mbits/s de performance)
- 32MB de SDRAM, 16MB de flash (8MB paralela e 8MB serial)
- Conector para cartões SD/MMC
- USB, JTAG, conectores para expansão
- Toolchain e kernel Linux com suporte a todos os periféricos
- Esquemáticos, gerber files
- Aproximadamente 80 dólares



# Exemplo de SBC: Beagleboard



- Core ARM Cortex-A8 (1200 DMIPS) e cache L2 256KB com acelerador gráfico 2D/3D, da Texas Instruments
- DSP TMS320C64x+ (HD video e processamento de sinais até 430MHz)
- 128MB Ram e 256MB Flash
- I2C, I2S, SPI, MMC/SD (via conector de expansão)
- DVI-D, JTAG, S-Video, SD/MMC+
- Stereo Out, Stereo In
- USB 2.0 HS OTG, Serial RS-232
- Aproximadamente 150 dólares



# Equipamentos necessários: plataforma embarcada



- Hardware proprietário pode ter custo elevado:
  - Projeto do circuito eletrônico
  - Layout da placa
  - Confeção do PCB (Printed Circuit Board) e montagem
  - Instrumentação (osciloscópios, analisadores lógicos, multímetros, estações de soldagem)
  - Integração hardware x software
  - Mecânica (caixas, fixação, conectores, etc)
  - Certificação, quando necessário
- Avaliar sempre o custo do investimento



# Equipamentos necessários: debug



- Se o seu kernel roda sem problemas, o GDB remoto resolve a depuração dos aplicativos. Caso contrário ...
- JTAG (Joint Test Action Group)
  - Permite debug da plataforma através de uma interface simples, geralmente via porta paralela, USB ou ethernet (*ICE-In Circuit Emulation*), desde que o chip tenha suporte ao JTAG
  - Custo baixo, alguns podem ser feitos em casa
  - Breakpoints, inspeção de memória, execução passo a passo, acesso a registros, etc
- BDM (Background Debug Mode)
  - Funcionalidade ICE similar ao JTAG, empregado pela Motorola
- Emuladores (via hardware)
  - Equipamentos que emulam o processador/microcontrolador. Caros.



# Fazendo escolhas: plataforma e suporte



- Várias plataformas existentes:
  - ARM (vários fabricantes, longa busca...)
  - Coldfire (Freescale)
  - PowerPC (Freescale/IBM/Apple)
  - x86 fanless (Geode/Alchemy da AMD, Celeron M da Intel, Eden da VIA)
  - MIPS (MIPS)
  - AVR/AVR32 (Atmel)
- Suporte
  - Comercial x comunitário



# Fazendo escolhas: distribuições



- Distribuições comerciais

- Montavista
- TimeSys
- LinuxWorks
- WindRiver
- SnapGear
- SysGo
- Koan
- ...

- Distribuições livres

- uClinux
- Emdebian
- Embedded Gentoo
- Embedded/Mobile Ubuntu
- OpenEmbedded
- Familiar
- LTIB (Freescale)
- T2 SDE
- Intel Moblin
- <Ponha seu nome aqui>



# Fazendo escolhas: suporte a tempo real



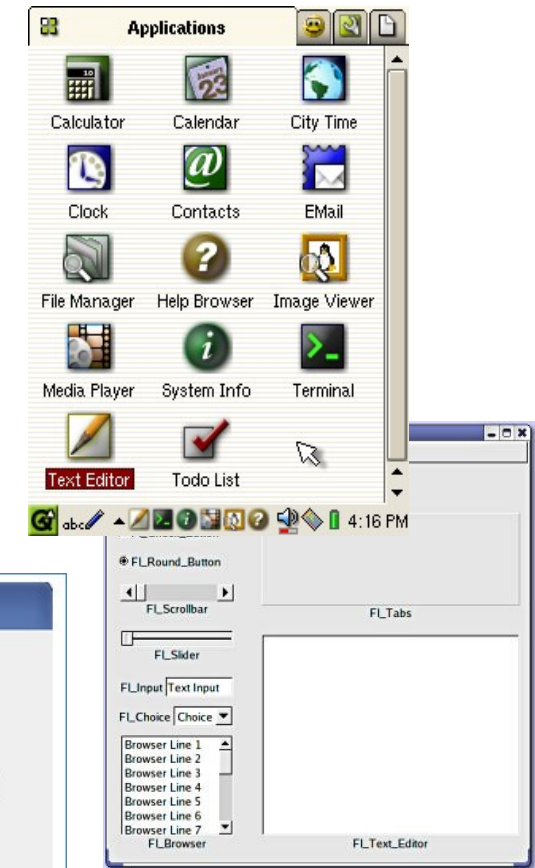
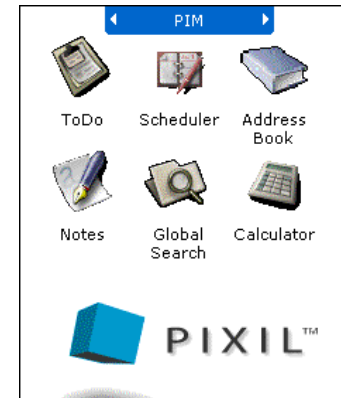
- Suporte a tempo real:
  - Comercial:
    - RTLinuxPro (Antes FMSLabs, agora Wind River)
    - MontaVista
  - Open source:
    - RTLinux Free (Wind River, licença dual)
    - RTAI (Usando Adeos, livre de patentes agora)



# Fazendo escolhas: interfaces gráficas



- Várias sistemas gráficos:
  - Qtopia
  - DirectFB
  - Matchbox
  - SDL
  - FLTK
  - MiniGUI
  - Nano-X (Microwindows)



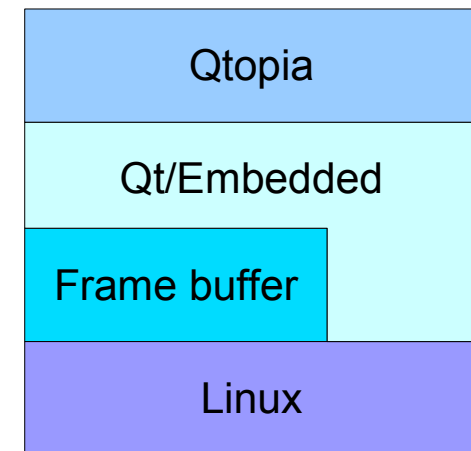
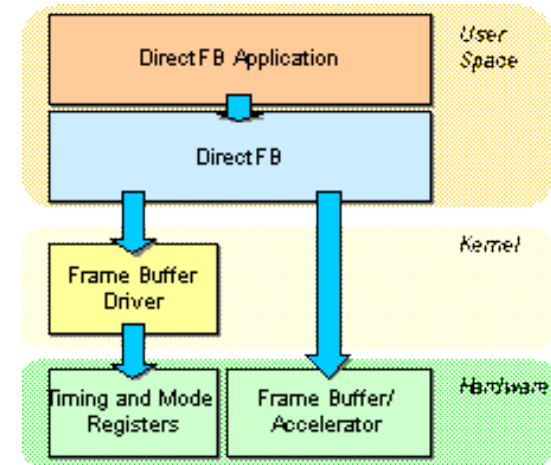
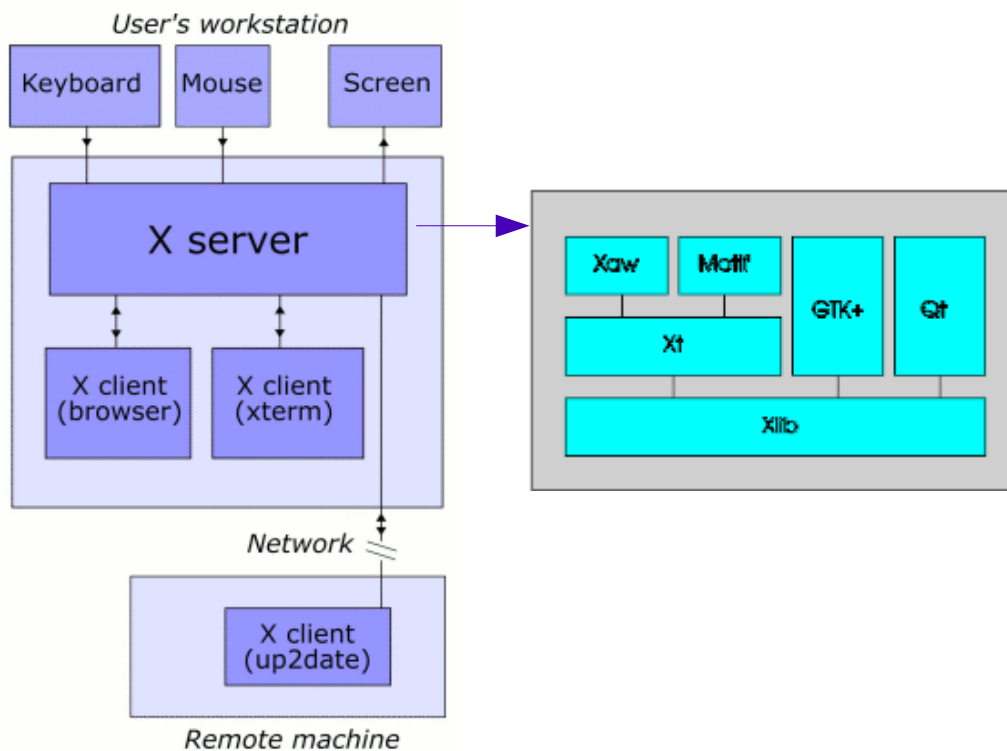
Referências: <http://www.linuxdevices.com/articles/AT9202043619.html>  
<http://www.linuxjournal.com/article/9403>



# Fazendo escolhas: interfaces gráficas



## Frame buffer x Xlib:



# Em teoria, não existe diferença entre teoria e prática...



- **I'm too young to die**
  - Target igual (x86), toolchain e kernel prontos para a CPU/placa
- **Hey, not too rough**
  - Target diferente, toolchain e kernel prontos para a CPU/placa (kits)
- **Hurt me please**
  - Target diferente, toolchain portado, porte do kernel para a CPU mas com novos periféricos na placa (drivers)
- **Ultra-violence**
  - Target diferente, toolchain e porte do Linux “quase” prontos (CPU parecida)
- **Nightmare**
  - Target diferente, sem toolchain portado nem Linux para a CPU/placa (de preferência SoPC/FPGA)



Imagem: [www.idsoftware.com/](http://www.idsoftware.com/)



# Sistemas embarcados com Linux



- Introdução
- Pré-requisitos
- **Criando sistemas embarcados**
- Créditos, agradecimentos e links



# Criando sistemas embarcados



Com tudo definido, é hora de aprender novas lições:

- Conceito 1: execução em RAM
- Conceito 2: bootloader
- Conceito 3: rootfs
- Conceito 4: init



# Conceito 1: execução em RAM



- O programa, armazenado em flash, em geral é descompactado para a RAM. Depois, a execução é transferida para a RAM.
  - O custo da memória flash por megabyte é maior
  - A velocidade da RAM é bem superior
  - O programa pode ser armazenado compactado em flash
  - É necessário um **bootloader** (programa de carga)
- Atenção com processadores sem MMU:
  - A memória é compartilhada por aplicativos e pelo próprio kernel. A falha em um aplicativo pode comprometer o kernel em processadores sem unidade de gerenciamento de memória (MMU)
- Executando direto da flash com XIP (eXecution In Place):
  - [http://elinux.org/Kernel\\_XIP](http://elinux.org/Kernel_XIP)
  - <http://axfs.sourceforge.net/>



# Conceito 2: bootloader



- Usado para inicializar o sistema operacional e também a configuração inicial da plataforma em uso
- Geralmente agrega outras tarefas, como descompactação, boot remoto via rede ou serial, operações com a flash, etc
- Alguns exemplos:
  - **Das U-Boot:** PPC, ARM, AVR32, MIPS, Coldfire, ...
  - **MicroMonitor:** ARM, ColdFire, SH2, 68K, MIPS, PowerPC, XScale ...
  - redBoot: ARM,x86,MIPS,PPC, SHx ....
  - Grub/Lilo: x86, boot do Linux, Windows, etc.
  - Outros: blob, SmartLoader, colilo, etc.



Imagem: <http://itpro.nikkeibp.co.jp/article/COLUMN/20060908/247572/zu1.jpg>



# Conceito 3: rootfs



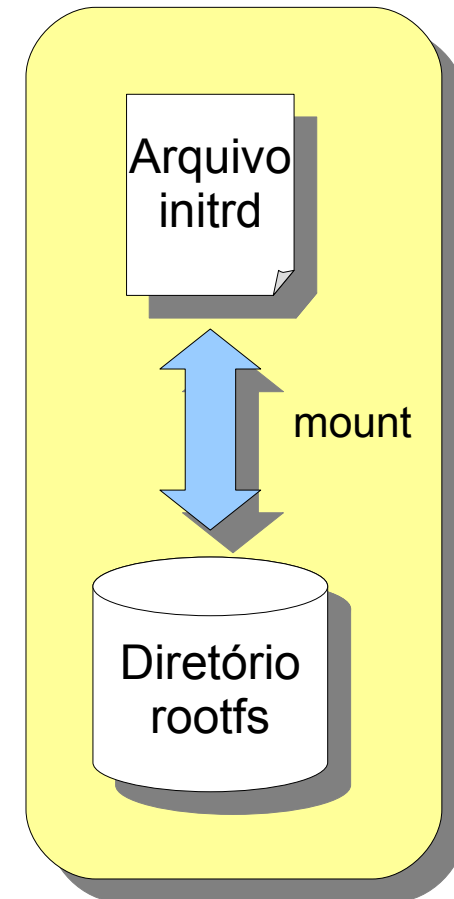
- O kernel está pronto, mas e o restante das aplicações ? De onde elas serão lidas ? Rootfs !
- O rootfs é o sistema de arquivo inicial do Linux. Pode ser um arquivo cpio/etx2 (comum em aplicações embarcadas), uma partição (geralmente sistemas não embarcados) ou ainda via rede (NFS, por exemplo).
- No momento do boot, o parâmetro “root=” é passado para o Linux, indicando ele irá encontrar o sistema de arquivo inicial. Exemplos:
  - root=/dev/hda1
  - root=/dev/ram0 rootfstype=ramfs
  - root=/dev/mtdblock1 rootfstype=jffs2
- O busybox pode ser uma boa alternativa para popular o rootfs, emulando vários aplicativos tradicionais do Linux. Pode usar a biblioteca uClibc, bem menor que a glibc.



# Conceito 4: init



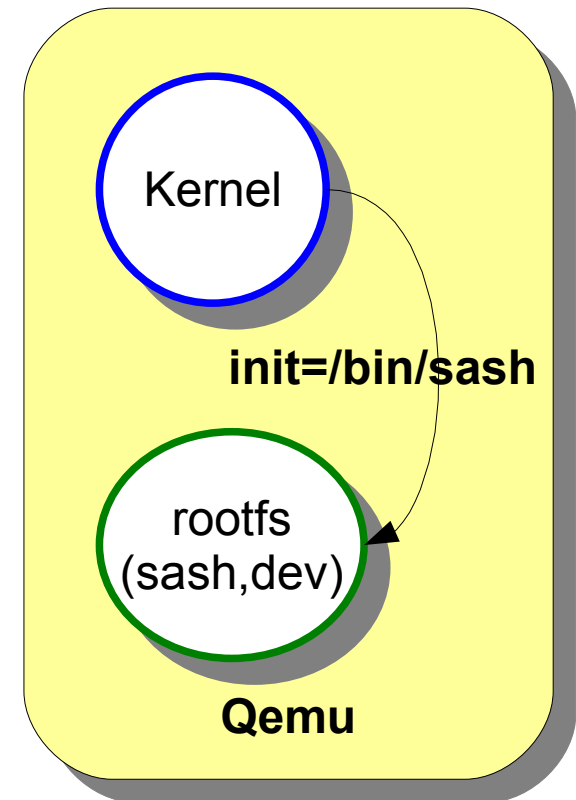
- Após montar o rootfs, o kernel executa o programa **init** (ou outro, caso seja usado o parâmetro “init=”). Também procura por **linuxrc**.
- Este é o primeiro programa executado e irá fazer a inicialização do sistema. O arquivo `/etc/inittab` dá as diretrizes de como isto deve ser feito.
- Máquinas com versão completa do Linux podem ter esquemas diferentes quando são usados módulos externos que precisam ser carregados antes do rootfs (esquema kernel → `initrd` → rootfs).



# Criando um sistema mínimo com sash e Linux



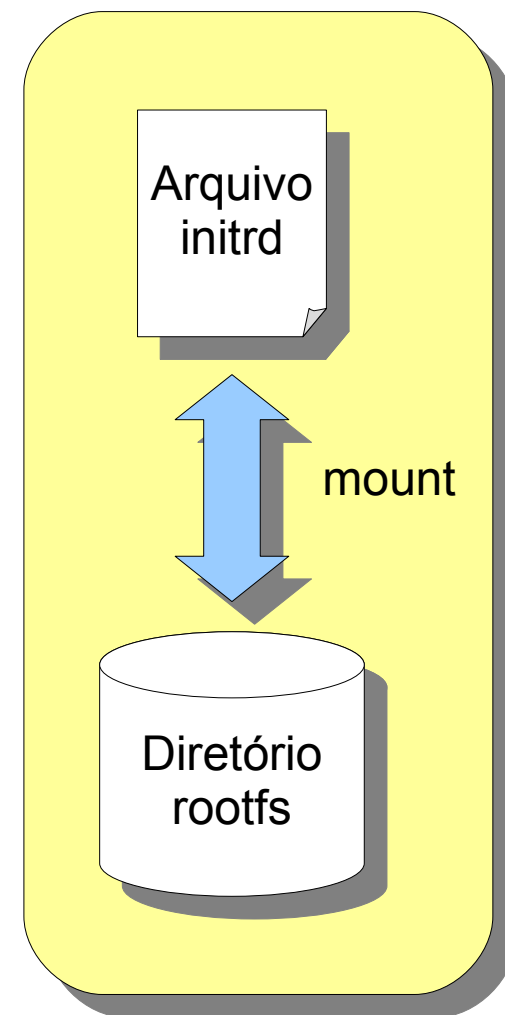
- O que é necessário num sistema mínimo com Linux ?
  - O sistema operacional (kernel Linux)
  - Sistema raiz (rootfs), com aplicativo(s) e bibliotecas
- Sistema mínimo com sash
  - sash = static linked shell (sem libc!)
  - /dev/console (necessário para o kernel)
  - /dev/hda (necessário para o qemu/kernel)
  - /bin/sash: processo a ser executado



# Criando um sistema mínimo com sash e Linux



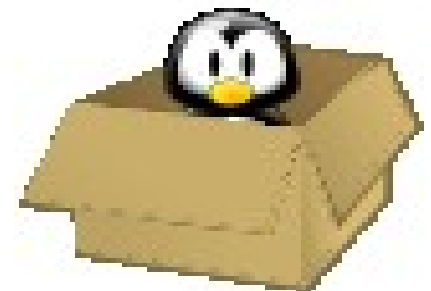
- Compile um kernel monolítico
- Crie um arquivo vazio e formate-o como ext2
  - `dd if=/dev/zero of=rootfs.img bz=1024k count=1`
  - `/sbin/mkfs.ext2 -i 1024 -m 0 -F rootfs.img`
- “Monte” este arquivo num diretório temporário
  - `mount -o loop rootfs.img rootfs`
- “Popule” este diretório:
  - Dispositivo do terminal: `/dev/console`
  - Prompt de comandos (shell) e suas dependências
  - Script de partida: `/bin/sash`
- Execute o qemu para iniciar a emulação



# Criando um sistema mínimo com sash e Linux: melhorias



- uClibc:
  - Restrições menores de licença (LGPL 2)
  - Footprint pequeno com quase a mesma funcionalidade da glibc
  - <http://www.uclibc.org>
- Busybox
  - Versões reduzidas de utilitários Unix em um único executável, bastante modular e configurável
  - Aceita também uClibc
  - Suporta outras plataformas via GCC cross compiler
  - <http://busybox.net/>
- Deve-se ter um toolchain construído com a uClibc



# Sistemas embarcados com Linux



- Introdução
- Pré-requisitos
- Criando sistemas embarcados
- **Créditos, agradecimentos e links**



# Projetos de construção do sistema



- buildroot: <http://buildroot.uclibc.org/>
- Scratchbox: <http://www.scratchbox.org/>
- Croostool: <http://www.kegel.com/croostool/>
- T2 Project: <http://www.t2-project.org>
- Podem gerar do zero, usando a uClibc:
  - Cross compiler
  - Rootfs
  - Linux kernel
- Várias plataformas suportadas (x86, ARM, PPC, MIPS ...)



# Links



- Linux a Bordo: <http://linuxabordo.com.br>
- Linux Devices: <http://linuxdevices.com>
- Free Electrons: <http://free-electrons.com>
- ucdot: <http://ucdot.org>
- Qemu: <http://fabrice.bellard.free.fr/qemu/>
- Busybox: <http://www.busybox.net/>
- Code sourcery: <http://www.codesourcery.com/>
- avr32fs: <http://code.google.com/p/avr32fs/wiki/MainPage>
- Das U-Boot: <http://www.denx.de/wiki/UBoot>



# Agradecimentos



- Ao Michael Opdnacker, por deixar público excelente material sobre Linux embarcado, algumas parte usadas aqui: <http://free-electrons.com>
- Ao projeto Open Cliparts: <http://openclipart.org>
- Ao projeto Open Office pelos excelente programas

