Linux a Bordo

Criando sistemas embarcados com Linux

Marcelo Barros de Almeida
marcelobarrosalmeida@yahoo.com.br

Criando sistemas embarcados com Linux

- **Introdução**
  - **Definição de sistema embarcado**
  - **Linux tradicional x Linux embarcado**
  - **Exemplos**
- Motivação
- Pré-requisitos
- Criando sistemas embarcados
- Créditos, agradecimentos e links

Introdução

- 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

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

Gráficos, navegador web, servidor de web.

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

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

Introdução

Alguns exemplos de sistemas embarcados com Linux:

PDA Sharp Zauro SL-C3100

Celular Haier N60 (WiFi, GSM, câmera, touchscreen, MP3 player)

Multimídia Archos PMA400

Roteador Linksys WTR54G

Telefone VoIP WiFi Accton VM1188T

Thinclients

Tablets

Relógio

DVDs

Robôs

Criando sistemas embarcados com Linux

- Introdução
- **Motivação**
  - Mercado
  - Vantagens
  - Principais cuidados
- Pré-requisitos
- Criando sistemas embarcados
- Créditos, agradecimentos e links

Mercado de Linux embarcado

- Grande número de concorrentes:
  - Sistemas proprietários (home-brew)
  - VxWorks
  - QNX
  - Windows
- Principal concorrente (2005): Windows Embedded

Cool devices, by embedded OS -- Round Four Scorecard

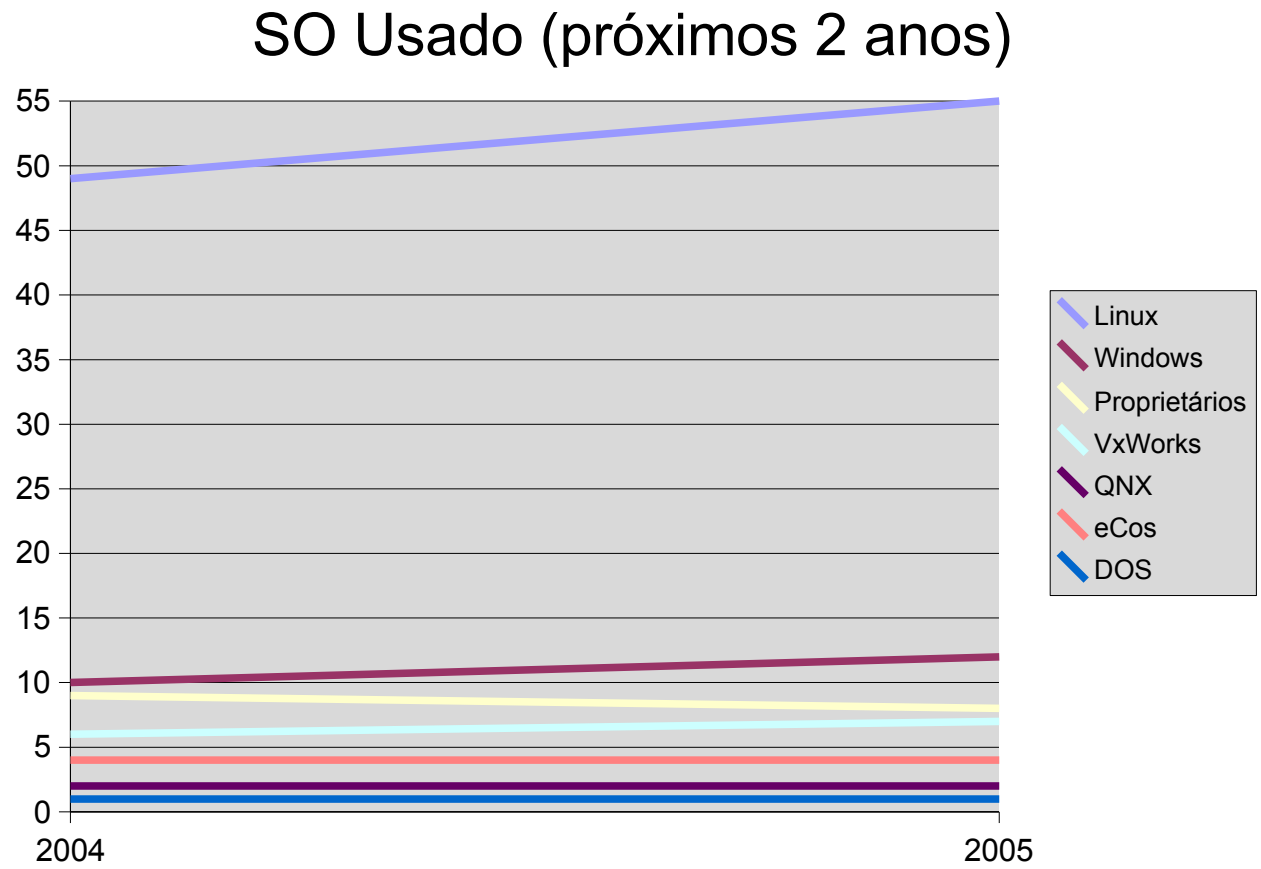
(click each quantity to view devices)

Device Category	Windows Embedded	Embedded Linux
PDAs, handhelds	120 devices	41 devices
Mobile phones	55 devices	30 devices
VoIP phones/devices	13 devices	15 devices
Robots	(included in other)	11 devices
Audio/video devices	23 devices	68 devices
Thin client devices	45 devices	28 devices
Tablets/webpads	40 devices	14 devices
Gateways, servers, APs	(included in other)	91 devices
Other	52 devices	69 devices
TOTAL:	**348 devices**	**367 devices**

Fonte: http://linuxdevices.com/articles/AT6743418602.html http://linuxdevices.com e http://windowsfordevices.com)

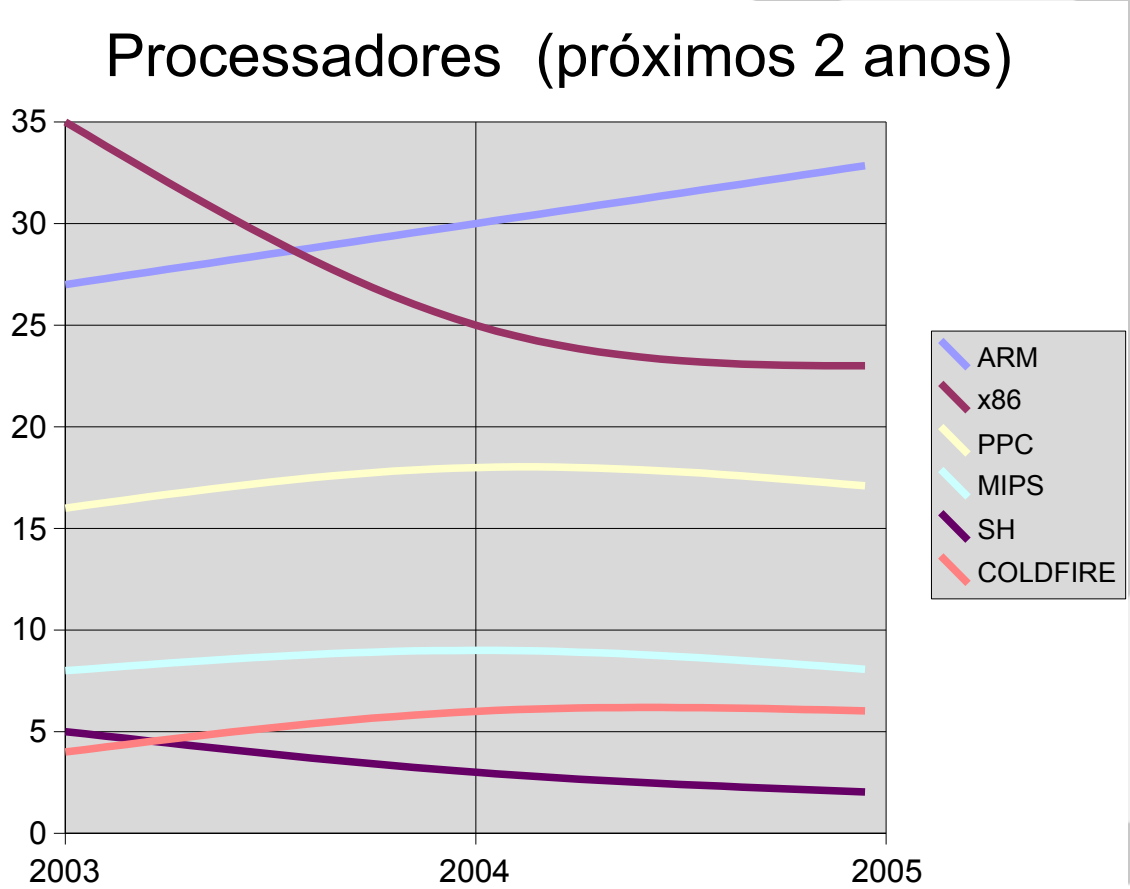
Mercado de Linux embarcado

- Pesquisa espontânea realizada em www.linuxdevices.com
- Números mais modestos (~25% para Linux) em pesquisas realizadas pela www.vdc-corp.com



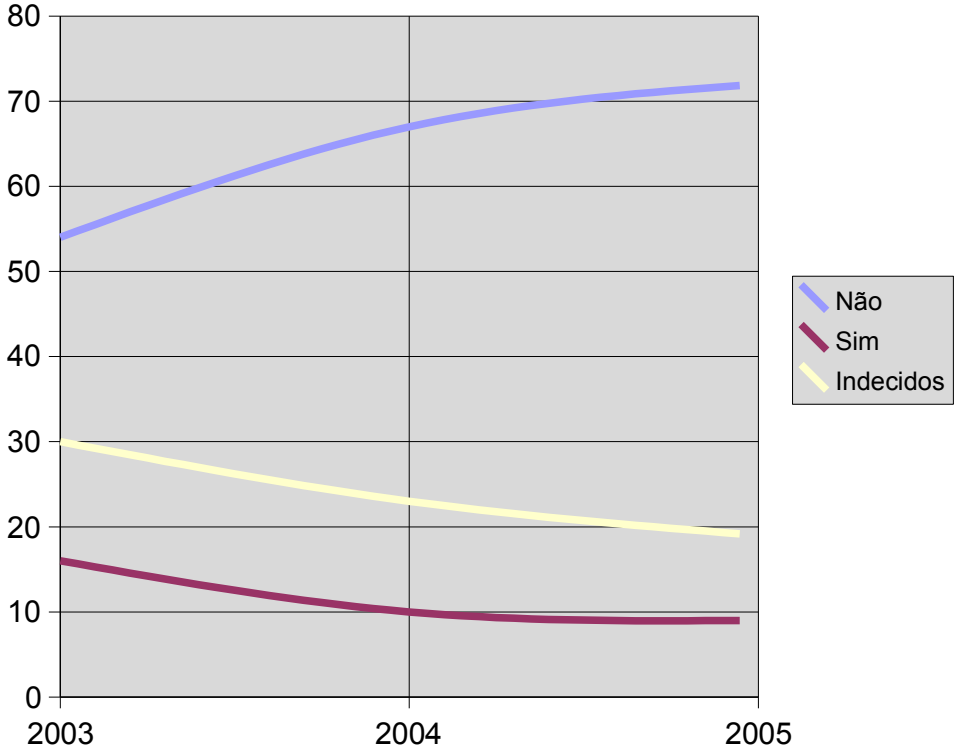
Mercado de Linux embarcado

- Grande número de processadores suportados
- Processadores ARM em alta atualmente. Nova linha VIA (x86) pode se tornar competitiva
- Pesquisa espontânea realizada em www.linuxdevices.com

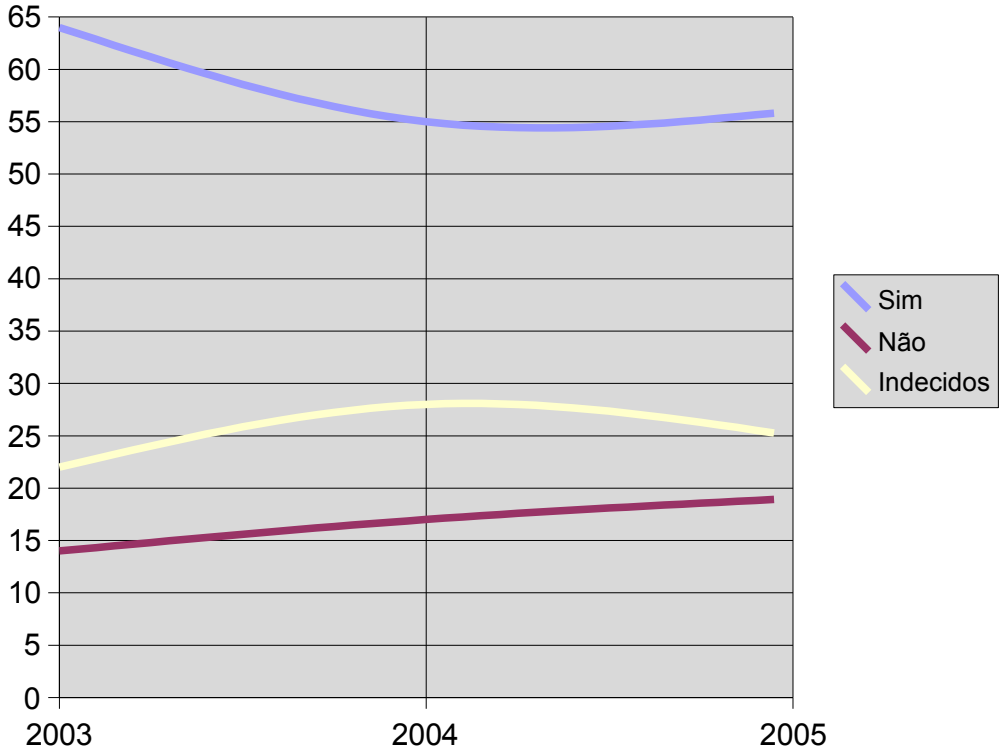


Mercado de Linux embarcado

Você pagaria royalties por unidade ?



Você pagaria por suporte ?



Pesquisa espontânea realizada em www.linuxdevices.com

Vantagens 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
- Custos reduzidos
- Suporte
  - Fórum, listas, email, FAQs, exemplos, suporte comercial disponível.

Cuidados com Linux embarcado

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

Criando sistemas embarcados com Linux

- Introdução
- Motivação
- **Pré-requisitos**
  - Conhecimentos necessários
  - Equipamento necessários
  - 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

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 ou scratchbox)
- 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, 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 (device drivers)
- Bônus track:
  - HTML, servidores HTTP
  - CGI e scripts (Python, PHP, Perl, etc)
  - Java

Conhecimentos necessários: eletrônica digital

- Microcontroladores/Microprocessadores
- Padrões de memórias:
  - SDRAM
  - Flash
  - EEPROM
- Padrões de barramentos pode ajudar caso precise desenvolver drivers
- Outros:
  - Timers, PWM, SPI, I2C, interrupções, RTCs, etc

Equipamentos necessários

- Estação de trabalho Linux
- Plataforma embarcada
- Equipamento para debug

Equipamentos necessários: plataforma embarcada

- Principais alternativas para a plataforma embarcada
  - Emulação: Qemu ou outro emulador específico
  - PC (ATX, mini-ATX, nano-ATX)
  - PC/104
  - SBC (Single Board Computers)
  - BSP (Board Support Package)
  - Hardware hackeado (PDAs, celulares, MP3 players, roteadores, set top boxes, vídeo games, relógios, etc)
  - Hardware proprietário

Equipamentos necessários: plataforma embarcada

- Hardware proprietário pode ter custo elevado:
  - Projeto do circuito eletrônico
  - Layout da placa
  - Confecção 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 versus a fabricação por terceiros

Equipamentos necessários: debug

- 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. Bem mais caros.

Criando sistemas embarcados com Linux

- Introdução
- Motivação
- Pré-requisitos
- **Criando sistemas embarcados**
  - Fazendo escolhas
  - Conceitos
  - Criando dispositivos
- Créditos, agradecimentos e links

Fazendo escolhas: plataforma e suporte

- Várias plataformas existes:
  - ARM (vários fabricantes, longa busca...)
  - Coldfire (Motorola)
  - PPC (Motorola/IBM/Apple)
  - x86 fanless (Geode/Alchemy da AMD, Celeron M da Intel, Eden da VIA)
  - MIPS (MIPS)
- Suporte
  - Comercial x comunitário

Fazendo escolhas: distribuições

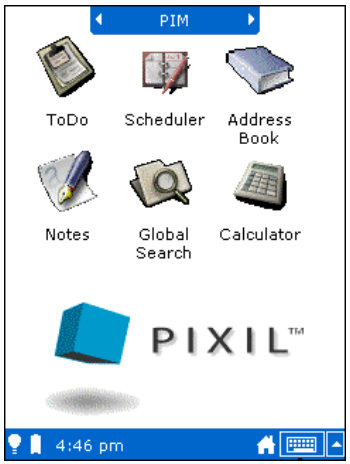
- Distribuições comerciais
  - Montavista
  - TimeSys
  - LinuxWorks
  - WindRiver
  - Freescale
  - SnapGear
  - SysGo
- Distribuições livres
  - uClinux
  - Emdebian
  - Embedded Gentoo
  - Embedded Ubuntu
  - OpenEmbedded
  - Familiar
  - <Ponha seu nome aqui>

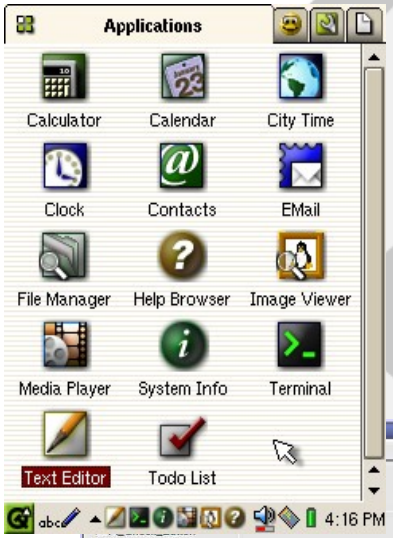
Fazendo escolhas: suporte a tempo real

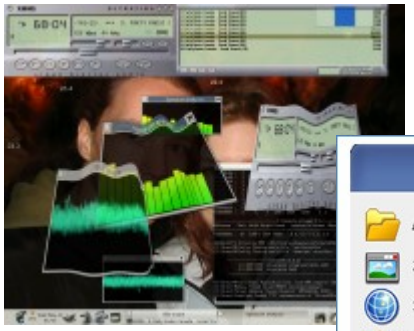
- Suporte a tempo real:
  - Comercial:
    - RTLinuxPro
    - MontaVista
  - Open source:
    - RTAI
    - RTLinux (Patentes envolvidas)

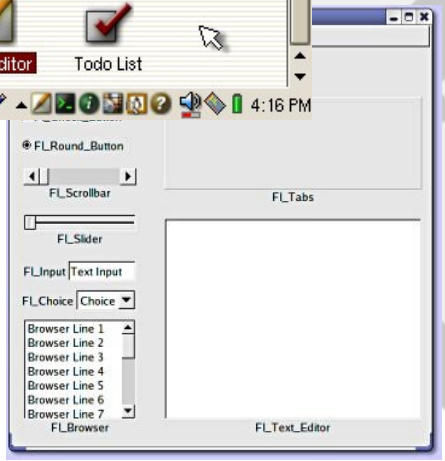
Fazendo escolhas: interfaces gráficas

- Várias interfaces gráficas:
  - MicroWindows
  - Qtopia
  - DirectFB
  - FLTK
  - MiniGUI









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 copiado (ou descompactado) para a RAM dinâmica (SDRAM). Depois, a execução é transferida para a RAM.
- Algumas vantagens:
  - O custo da memória flash por megabyte é maior
  - A velocidade da SDRAM é bem superior
  - O programa pode ser armazenado em flash, compactado
- 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)
- É necessário um bootloader (programa de carga)

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:
  - u-boot: PPC, ARM e MIPS para boot do Linux.
  - MicroMonitor: ARM, ColdFire, SH2, 68K, MIPS, PowerPC, XScale e vários O.S., como VxWorks, Linux, pSOS, Nucleus, CMX, uC/OS, eCos, etc.
  - redBoot: ARM,x86,MIPS,PPC, Shx, etc para boot do Linux e eCos.
  - Grub/Lilo: x86, boot do Linux, Windows, etc.
  - Outros: blob, SmartLoader, colilo, etc.

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 (comum em aplicações embarcadas), uma partição (geralmente sistemas não embarcados) ou ainda via rede (NFS).
- No momento do boot, o parâmetro “root=” é passado para o Linux, indicando ele irá encontrar o sistema de arquivo inicial.
- 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 /sbin/init (ou outro, caso seja usado o parametro “init=”).
- 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 externosl que precisam ser carregados antes do rootfs (esquema kernel➔initrd➔rootfs).

Colocando a mão na massa

- Duas abordagens:
  - Top-down: parte da distribuição completa e remove o desnecessário
  - Bottom-up: parte com um sistema mínimo e adiciona o desejado
- Etapas (bottom-up):
  - Configuração e compilação do kernel
  - Criação do sistema de arquivos (rootfs)
  - Compilação do busybox e instalação
  - Criação de dispositivos (/dev/)
  - Configuração dos scripts de inicialização
  - Geração da imagem e configuração do bootloader

Exemplo mínimo com emulação x86

1) Compile o kernel
   - Baixe o código fonte (http://kernel.org)
   - Altere a variável EXTRA_VERSION no Makefile
   - make menuconfig
   - make
   - O kernel ficará em arch/i386/boot/bzImage
2) Crie o sistema de arquivos raiz e formate para ext2
   - dd if=/dev/zero of=rootfs.img bs=320k count=1
   - mkfs.ext2 -i 1024 -F rootfs.img

Exemplo mínimo com emulação x86

3) Compile o busybox

- Baixe o código fonte (http://busybox.net)
- Configure, escolhendo uClib como a biblioteca C
  - make menuconfig
- compile e instale (inicialmente em _install/)
  - make
  - make install

4) Inicializando o sistema de arquivos

- Crie um ponto de montagem e monte o sistema de arquivos
  - mkdir /mnt/rootfs
  - mount -o loop rootfs.img /mnt/rootfs

Exemplo mínimo com emulação x86

- Copie o busybox para dentro do sistema de arquivos
  - rsync -a busybox/_install/ /mnt/rootfs/
  - chown -R root:root /mnt/rootfs/
  - sync

5) Rodando via qemu como bootloader

```
qemu \
 -m 32 \              Quantidade de memória no sistema emulado
 -hda rootfs.img \    Conteúdo do sistema de arquivos
 -kernel linux-2.6.12/arch/i386/boot/bzImage \  Kernel
 -append "root=/dev/hda clock=pit" linha comando do kernel
```

Criando sistemas embarcados com Linux

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

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
- A equipe programa FreeMind, usado para projetar a apresentação http://freemind.sourceforge.net

À toda comunidade de Software Livre: "Idéias são à prova de balas."

Do filme V de vingança

Links

- Linux Devices: http://linuxdevices.com

  Muita informação sobre Linux embarcados, lançamentos, placas de desenvolvimento, análises, etc.

- Linux a Bordo: http://linuxabordo.com.br

  Conteúdo em português sobre Linux embarcado. Notícias, novidades, artigos, etc.

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

  Apresentações sobre Linux embarcado.

- ucdot: http://ucdot.org

  Conteúdo voltado para uClinux

Direitos de cópia

creative commons
COMMONS DEED

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

Marcelo Barros
marcelobarrosalmeida@yahoo.com.br

Documentos originais, atualizações e traduções:

http://linuxabordo.com.br/

Correções, sugestões e traduções são bem vindas!