Meios Eletrônicos Interativos II

Aula 03 Linux Embarcado I

2o. Semestre de 2011

Sobre esta apresentação

- Trata-se de uma introdução a Linux Embarcado, mostrando a geração de um Linux embarcado simples para plataforma embarcado x86. Esta transparência é complementada com os seguintes materiais:
  - aula3a, aula3b, aula3c, aula3d, aula3e e aula3f, disponíveis para consulta no site da disciplina. A sequência dos tópicos pode não estar necessariamente na ordem.
- Trata-se da geração de uma imagem de sistema operacional Linux em armazenamento USB (Pen Drive USB) “bootável” (Live USB) para uma placa Intel ATOM. Como os CPUS da estação de desenvolvimento e da plataforma alvo são x86, não utilizaremos nesta aula um toolchain de compilação cruzada
- A parte prática é apresentada nas transparências AULA03 – PARTE PRÁTICA
- Na próxima aula será feita a geração de uma imagem de sistema operacional Linux para uma placa ARMv9 utilizando portanto um toolchain de compilação cruzada

Objetivos de Aula

- Gerar uma versão de Linux Embarcado para uma placa Intel ATOM
- Configurar e compilar o kernel do Linux
- Criação do sistema de arquivos raiz (rootfs)
- Configurar, compilar e instalar o Busybox
- Criação de dispositivos (/dev)
- Geração da imagem e configuração do bootloader
- Gerar um Live USB e executar o Linux Embarcado
- Testar a versão no emulador QEMU – X86
- Testar na Placa ATOM

Linux Embarcado

Porque Linux Embarcado?

Uma das vantagens é a grande quantidade de bibliotecas e aplicações de terceiros:

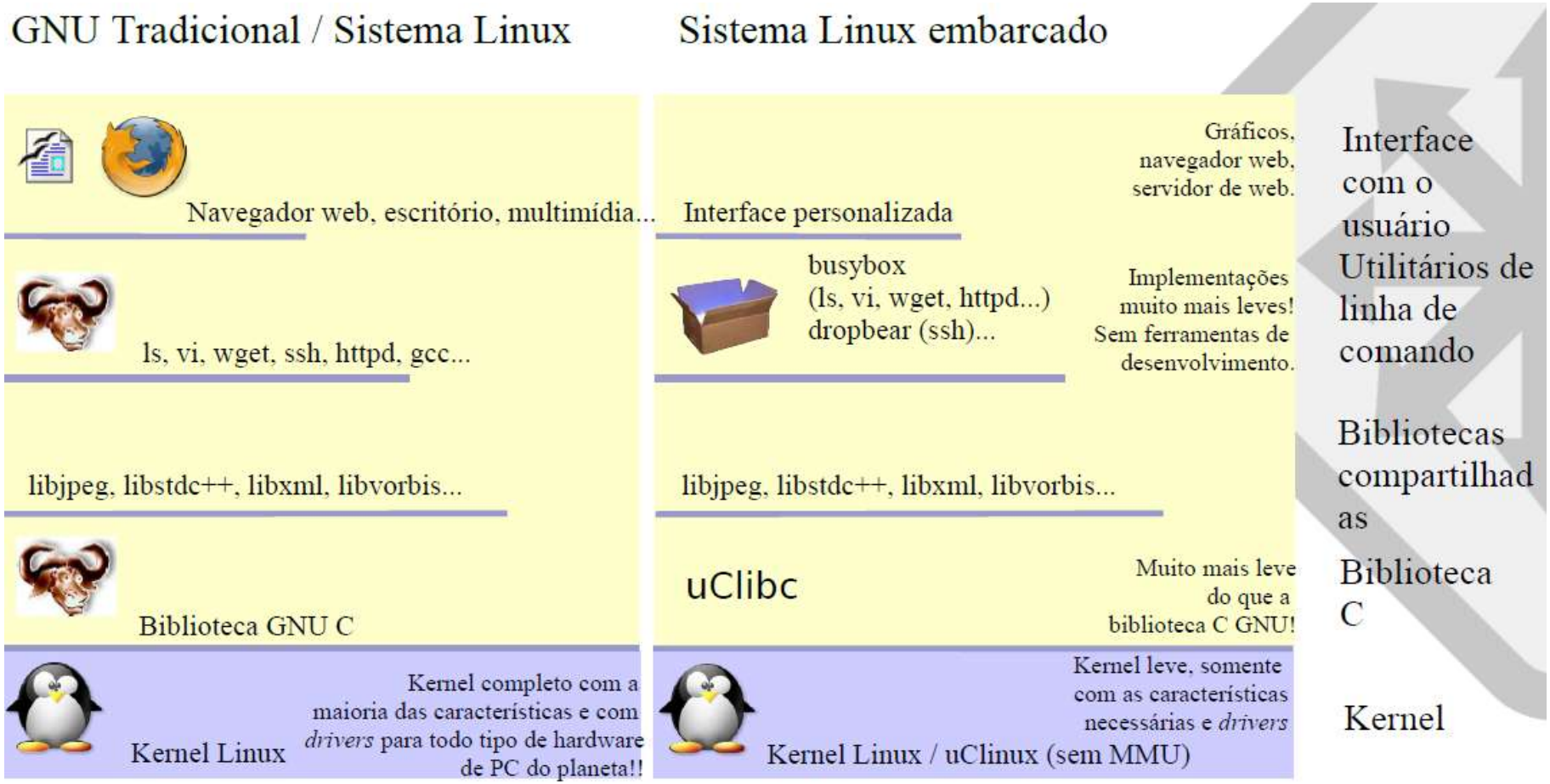
- Eles são disponíveis livremente, distribuíveis livremente, e graças ao seu caráter código aberto,eles podem ser analisados e modificados de acordo com as necessidades do projeto
- Porém o reuso eficiente desses componentes nem sempre é fácil. Deve-se:
  - Encontrar estes componentes
  - Escolher os mais apropriados
  - Compilar cruzadamente
  - E então integrá-los no sistema embarcado junto com as outras aplicações

O que é um Linux Embarcado?

- Um Linux “enxuto” para rodar dentro das limitações de HW do dispositivo embarcada
  - Limitações de memória RAM
  - Limitações de armazenamento
- O Linux não foi feito pensando em micro-controladores de recurso muito limitados, por exemplo, com memória RAM e armazenamento FLASH da ordem de dezenas ou centenas de Kilobytes. Opções:
  - Não usar um sistema operacional
  - Sistemas minimalistas, como o FreeRTOS, BRTOS etc

Universo de aplicações de Linux Embarcado Multimídia

Linux: Tradicional versus Embarcado



Fonte: Marcelo Barros de Almeida

Requisitos de Hardware

- Um CPU suportado pelo GCC e pelo Linux Kernel
  - CPU 32 ou 64 bits
  - CPU’s sem MMU (unidade de gerenciamento de memória, responsável pelo mapeamento entre os espaços de memória virtual e física) são suportados pelo Linux através do Projeto uCLinux

Software: Como encontrar componentes existentes

- Freshmeat, um website de projetos opensource
  - http://www.freshmeat.net
- Free Software Directory
  - http://directory.fsf.org
- Examinar outros produtos embarcados Linux, e ver quais componentes são usados
- Olhar a lista de pacotes de software empacotados por sistemas embarcados Linux
- Entrar nos sites de comunidades de embarcados Linux, pesquisar e perguntar
- Por fim, pesquisar na literatura existente, especialmente através de sistemas de busca como o Google

Escolhendo componentes

Nem todos os componentes de software são necessariamente bons para reuso. Alguns pontos que merecem atenção:

- **Vitalidade do desenvolvedor e comunidades de usuários. Esta vitalidade assegura** manutenção de longo prazo do componente, e suporte relativamente bom.
  - Pode ser avaliado olhando-se o tráfego das listas de correio eletrônico e a atividade do sistema de controle de versão.
- **Qualidade do componente. Tipicamente, se um componente está já** disponível em **embedded buid systems**, e tem uma comunidade de usuários dinâmica, é provável que a qualidade seja relativamente boa.
- **Licença. A licença do componente deve casar com suas restrições de licença**. Por exemplo, bibliotecas GPL não podem ser usadas em aplicações proprietárias.
- **Requisitos técnicos. Evidentemente, o componente deverá casar com seus requisitos técnicos**. Mas não se esqueça que você terá de aperfeiçoar os componentes existentes se alguma característica estiver faltando!

Componentes de software em um sistema Linux embarcado simples

- Em muitas aplicações embarcadas, o sistema faz apenas a monitoração e controle de um dispositivo
- Os componentes típicos são:
  - Kernel
  - Busybox
  - Biblioteca C
  - Aplicações rodando sobre a biblioteca C, às vezes usando algum recurso de processamento Tempo-Real do kernel
  - Às vezes um servidor web para controle remoto, ou outro tipo de servidor usando um protocolo proprietário

Dispositivo Multimídia Embarcado: Moldura de Fotografia digital

Referência: Matt Porter, Embedded Alley at ELC 2008

- Hardware: ARM SoC com DSP, áudio, 800x600 LCD, MMC/SD (Secure Digital), NAND, botões, speakers
- A moldura de fotos dever ser capaz de:
  - Mostrar a imagem no LCD
  - Detectar a inserção de cartão SD, notificar as aplicações de que houve inserção, de modo que as aplicações possam construir um catálogo de figuras presentes no cartão SD
  - Interface Humano Máquina 3D moderna, com belas transições
  - Navegação através de botões
  - Suporte a áudio playback (MP3, playlists, ID3 tag)
  - Redimensionamento e rotação JPEG

“Leveraging Free and Open Source Software in a Product Development Environment”

Presentation:

*http://elinux.org/images/a/a2/**Elc-foss**.pdf*

Video:

http://free-electrons.com/pub/video/2008/elc/elc2008-matt-porter-product-development.ogg

Componentes

Sistema Base:

- Componentes presentes em virtualmente todos os sistemas Linux embarcados
- O bootloader U-Boot
- Linux Kernel
  - Com Drivers para: dispositivos de entrada, som framebuffer, armazenamento SD/MMC
- Busybox
- Build system: foi escolhido o OpenEmbedded
- Componentes: **U-Boot, Linux, Busybox**

Componentes, cont.

- Manipulação de eventos para detecção da inserção do cartão SD
  - udev, que recebe eventos do, cria device nodes, e envia eventos ao HAL
  - HAL, que mantêm o base de dados dos dispositivos disponíveis e provê um API D-Bus (atualmente, em 2011, o HAL está em processo de obsolescência, havendo uma tendência de ser integrada ao udev)
  - D-Bus para conectar HAL com a aplicação. A aplicação assina evento HAL através do D-Bus e é notificado quando eles são acionados
  - Componentes:
    - **udev, hal, d-bus**

Componentes, cont.

- Display JPEG
  - **libjpeg para decodificar as figuras**
  - **jpegtran para redimensionar e rotacioná-las**
  - FIM (Fbi Improved) para *dithering*
    - *http://savannah.nongnu.org/projects/fbi-improved/*
- Suporte a MP3
  - **libmad para tocar**
  - **libid3 para ler tags ID3**
  - **libm3u para suportar playlists**
  - Componentes providos for fornecedores para permitir que o uso do DSP para tocar MP3

Componentes, cont.

- Interface 3D
  - Vincent, uma implementação opensource do OpenGL ES
  - Clutter, API de alto nível para desenvolvimento de aplicações 3D
- A aplicação
  - Gerencia eventos de mídia
  - Usa as bibliotecas JPEG para decofificar e renderizar figuras
  - Recebe eventos de entrada Linux dos botões e desenha a Interface de Usuário (UI) baseado em OpenGL, desenvolvida com o Clutter
  - Gerencia uma Configuração definida pelo usuário
  - Toca a música com as bibliotecas relacionadas com MP3
  - Mostra as fotografias em modo *slideshow*

Outros exemplos de dispositivos multimídia com Linux Embarcado

- Tocador MP3
- Rádio Digital
- Câmera de vigilância
- Telefone Celular e Smartphone

Cuidado: Licenças

- Todos os softwares que estão sob uma licença de software livre oferece quatro liberdades aos usuários
  - Liberdade de uso
  - Liberdade de estudo
  - Liberdade de cópia
  - Liberdade de modificar e distribuir as cópias modificadas
- Consultar http://www.gnu.org/philosophy/freesw.html para uma definição de Software Livre
- Softwares Open Source, segundo a definição da “Open Source Initiative”, são tecnicamente similares a Software Livre em termos de liberdades
- Consultar http://www.opensource.org/docs/osd para uma definição de Open Source Software

Tipos de Licenças

- Licenças de Software Livre caem em duas categorias principais
  - *copyleft licenses*
  - *noncopyleft licenses*
- O conceito de « copyleft » tem a ver com a reciprocidade nas liberdades oferecidas ao usuário.
- O resultado é que quando você recebe um software sob um “copyleft free software license” e distribui versões modificadas dele, você deve fazê-lo sob a mesma licença
  - Mesmas liberdade aos novos usuários
  - É um incentivo de você retribuir com as suas alterações, em vez de mantê-los secretos
- “Noncopyleft licenses” não tem tais requisitos, e versões modificadas podem ser mantidas proprietárias, mas eles ainda requerem atribuição (?)

GPL

- GNU General Public License
- Cobre ~55% dos projetos de software livre, incluindo o Linux kernel, Busybox, e muitas aplicações
- É uma licença *copyleft*
- Requer que trabalhos derivados sejam liberados sob a mesma licença
- Programas ligados com uma biblioteca liberada sob uma licença GPL devem também ser liberados sob a mesma GPL
- Alguns programas cobertos pela versão 2 (Linux kernel, Busybox e outros)
- Mais e mais programas cobertos pela versão 3, lançada em 2007
- Impacto no mercado de embarcados: o requisito que o usuário deve ser capaz de rodar as versões modificadas do dispositivo, se o dispositivo é um dispositivo « consumer » (?)

GPL, cont.

- Nenhuma obrigação, se o software não for distribuído
- Você pode manter suas modificações secretas até a entrega do produto
- É autorizado a distribuição de versões binárias de software, se uma das seguintes condições for atingida:
  - Acompanhar o binário com uma cópia da fonte em uuma mídia física
  - Acompanhar o binário com uma declaração escrita válida por 3 anos que indica como o código fonte pode ser buscado
  - Acompanhar o binário com um endereço de rede da locação onde o código pode ser encontrado
- Ver a seção 6. da licença GPL
- Em todos os casos, a atribuição e a licença devem se preservados
- Ver seções 4. e 5.

LGPL

GNU Lesser General Public License

- Representa ~10% dos projetos de software livre
- Licença copyleft
- Versões modificadas devem ser entregues sob a mesma licença
- Porém, programas ligados com uma biblioteca LGPL não precisam ser liberados sob a mesma LGPL e podem ser mantidas proprietárias
- Porem, o usuário deve manter a capacidade de atualizar a biblioteca independentemente do programa, de modo que deve ser usado ligação dinâmica (*dynamic linking*)
- Usado em vez do GPL para a maioria das bibliotecas, incluindo as bibliotecas C
- Algumas exceções: MySQL, ou Qt <= 4.4
- Também disponível em duas versões, v2 e v3

Licenciamento, exemplos

- Você faz modificações no Linux kernel (para adicionar drivers ou adaptá-lo à sua placa), Busybox, Uboot ou outro software GPL
  - Você deve liberar as versões modificadas sob a mesma licensa, e estar pronto para distribuir o código fonte para seus clientes
- Você faz modificações na biblioteca C ou qualquer outra biblioteca LPGL
  - Você deve liberar as versões modificadas sob a mesma licença
- Você cria uma aplicação ligada com bibliotecas LGPL
  - Você pode manter a sua aplicação proprietária, mas deve fazer ligação dinâmica com as bibliotecas LGPL
- Você faz modificações a um software *noncopyleft*
  - Você pode manter suas modificações proprietárias, mas deve manter o crédito aos autores

Licenças não Copy-left

- Alguns exemplos
  - Apache license (~ 4%)
  - BSD license (~ 6%)
  - MIT license (~ 4%)
  - X11 license
  - Artistic license (~9 %)

BSD

Copyright (c) <year>, <copyright holder>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the <organization> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

[...]

Respeito às licenças de Software Livre

- Cuidado: Free Software não é software de domínio público, os distribuidores tem obrigações devidos às licenças
- **Antes de usar um componente de software, assegure-se que a licença** casa com as suas restrições de projeto
- Mantenha uma lista completa dos pacotes de software que você usou, a versão original que você usou e mantenha as suas modificações e adaptações bem separadas da versão original
- Verifique a conformidade aos requisitos de licença **antes de enviar o produto** aos clientes
- Mais informações:
  - GPLviolations.org, http://www.gplviolations.org
  - Software Freedom Law Center, http://www.softwarefreedom.org/

Manter em separado as alterações

- Quando estiver integrando componentes opensource existentes no seu projeto, pode ser necessário fazer modificações nele visando:
  - Melhor integração, redução de tamanho, eliminação de bugs, novas características, etc.
- Em vez de misturas estas modificações, é melhor mantê-los separados da versão original do componente
  - Se o componente necessitar de atualização, é util saber quase modificações foram feitas no componentes
  - Se for solicitado suporte pela comunidade, é importante saber quão diferente o componente que voce está usando é da versão original
  - Torna possível contribuir as modificações para a comunidade
- É ainda melhor manter em separado estas várias mudanças feitas em um dado componente
  - É mais fácil rever e atualizar para novas versões

Quilt

- A solução mais simples é usar o **Quilt**
  - **Quilt** é uma ferramenta que permite manter uma pilha de **patch**s sobre o código fonte
  - Facilita adicionar, remover modificações de um **patch**, para adicionar ou remover **patch**s da pilha e atualizá-los
  - A pilha de **patch**s pode ser integrada ao seu sistema de controle de versões
  - https://savannah.nongnu.org/projects/quilt/
- Outra solução é o uso de um sistema de controle de versões
  - Importar a versão original do no seu sistema de controle de versões
  - Manter suas mudanças em uma linha separada

Linux Básico

O Linux

Linux foi o nome dado ao núcleo (kernel) de sistema operacional criado por Linus Torvalds. Por extensão, sistemas operacionais que usam o núcleo Linux são chamados genericamente de Linux.

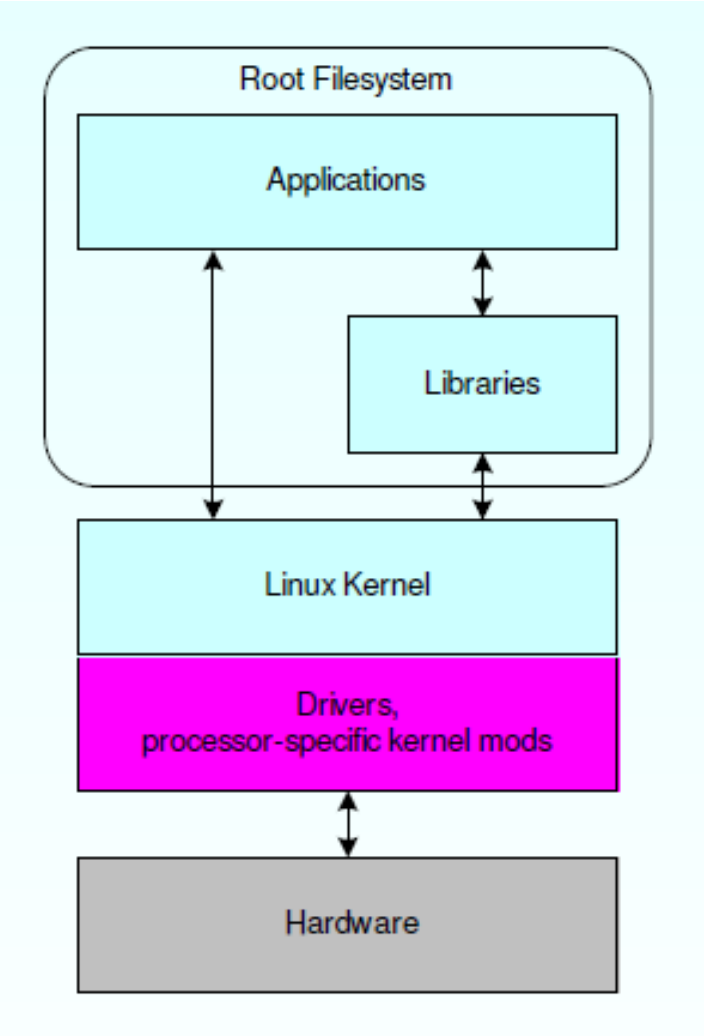
Entretanto, a *Free Software Foundation* afirma tais sistemas operacionais são, na verdade, sistemas GNU, e o nome mais adequado para tais sistemas é *GNU/Linux*, uma vez que grande parte do código-fonte dos sistemas operacionais baseados em Linux são ferramentas do projeto GNU.

Richard Stallman. *Linux and the GNU Project*

Distribuições Linux

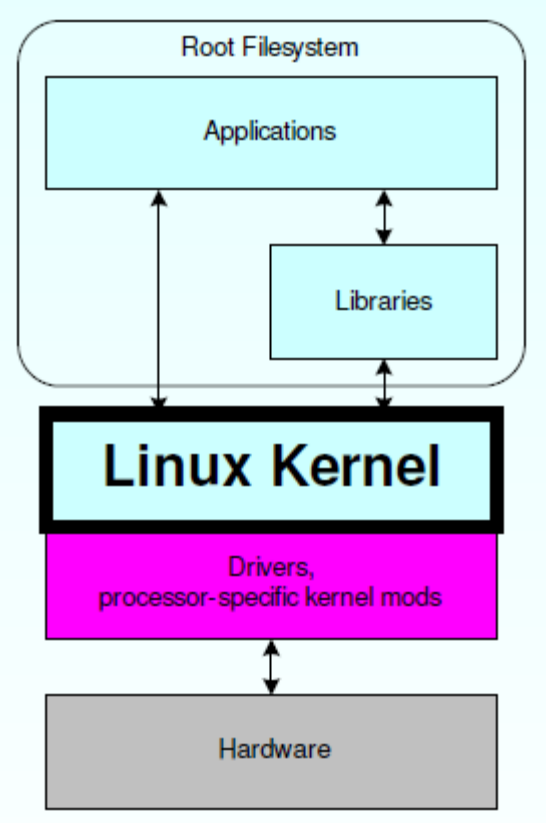
- Uma distribuição **Linux é um membro da família de sistemas operacionais** Unix-like construído sobre o Linux kernel.
- Tais distribuições são Sistemas Operacionais com uma grande coleção de aplicações de software, como processadores de texto, planilhas, tocadores de mídia, e banco de dados.
- O Sistema Operacional consiste do Linux kernel e um conjunto de bibliotecas e utilitários do GNU project, com graphics suportado pelo X Window System.
- Distribuições otimizadas para tamanjo podem não conter X e tendem a usar alternativas mais compactas aos utilitários GNU, como o Busybox, uClibc, ou dietlibc.
- Há atualmente várias centenas de Linux distributions, muitas em estado ativo de desenvolvimento.

Componentes de um sistema LInux

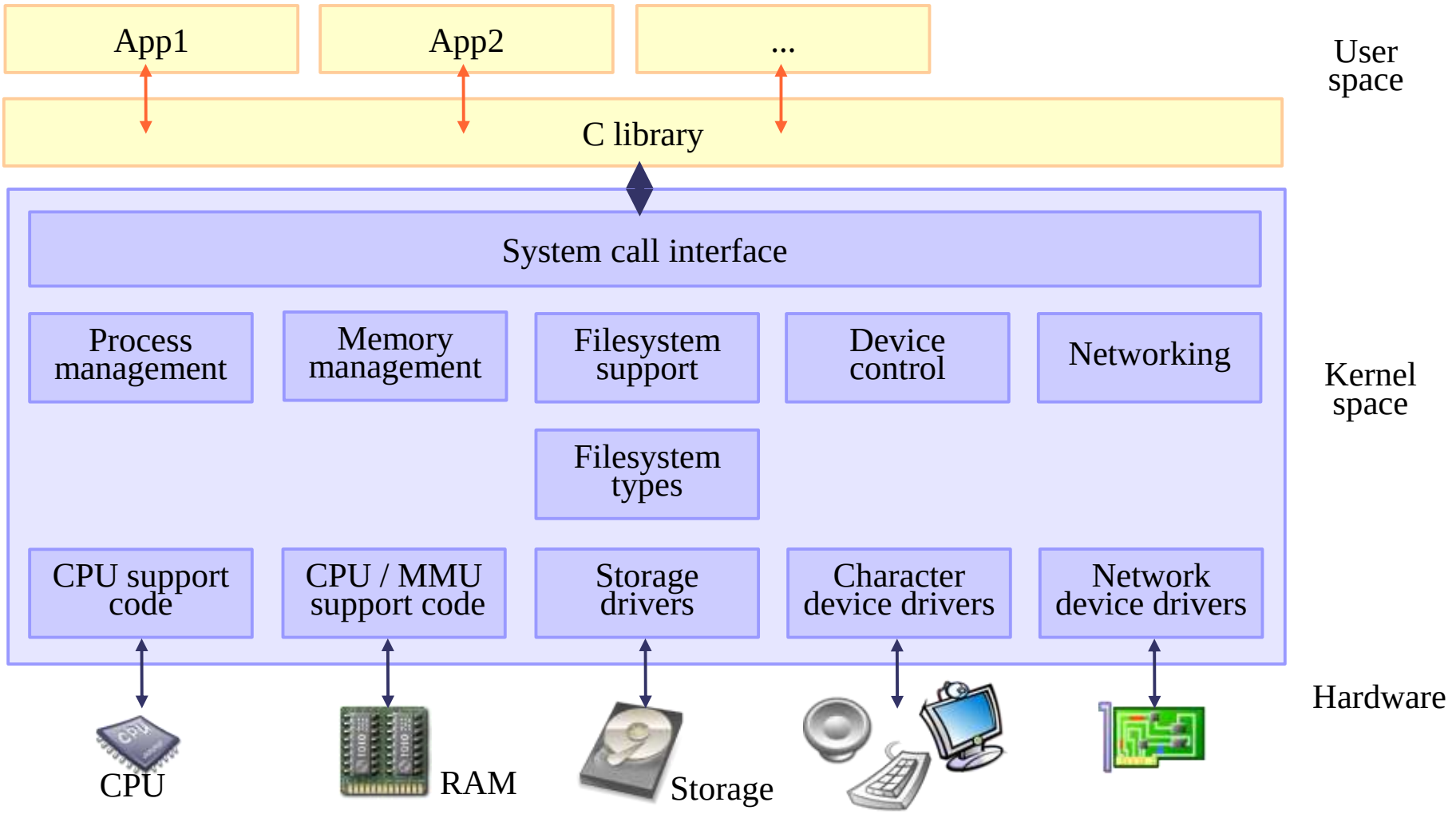


Kernel

- Série atual do kernel do LInux: 2.6, indo para 3.0
  - suporte a várias arquiteturas de processadores
- Configurável
  - Pode-se configurar para o Hwe aplicação desejados
- Múltiplos modos possíveis de execução
  - Execute-in-place (XIP)
  - Compressed/loadable



Kernel do Linux

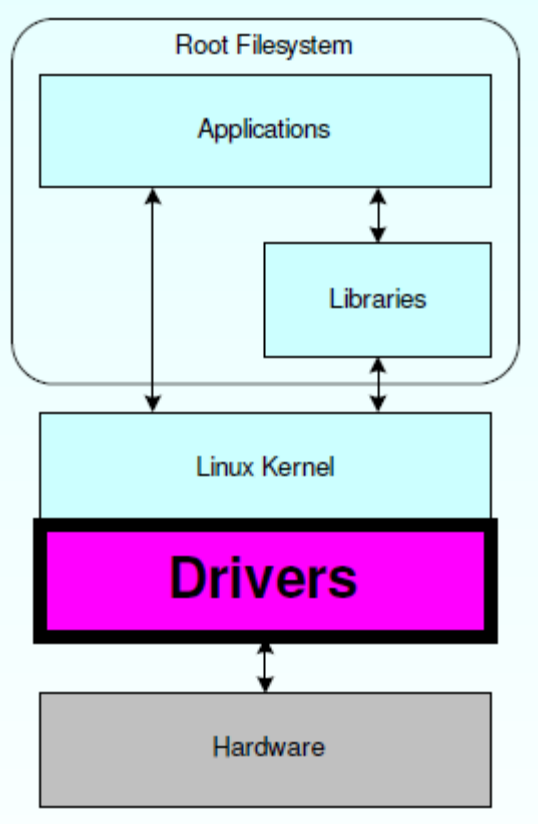


Kernel do Linux

- Portabilidade. Roda em muitas arquiteturas.
- Escalabilidade
  Pode rodar em sistemas de diversos portes, desde sistemas de alto desempenho a dispositivos pequenos portáteis.
- Aderência a padrões e interoperabilidade.
- Suporte exaustivo de rede.
- Segurança
  Não pode esconder suas lacunas de segurança. Seu código é revisto por muitos especialistas.
- Estabilidade e confiabilidade.
- Modularidade
  Pode incluir apenas o que o sistema necessita em um dado momento de execução.
- Suporte a programação.

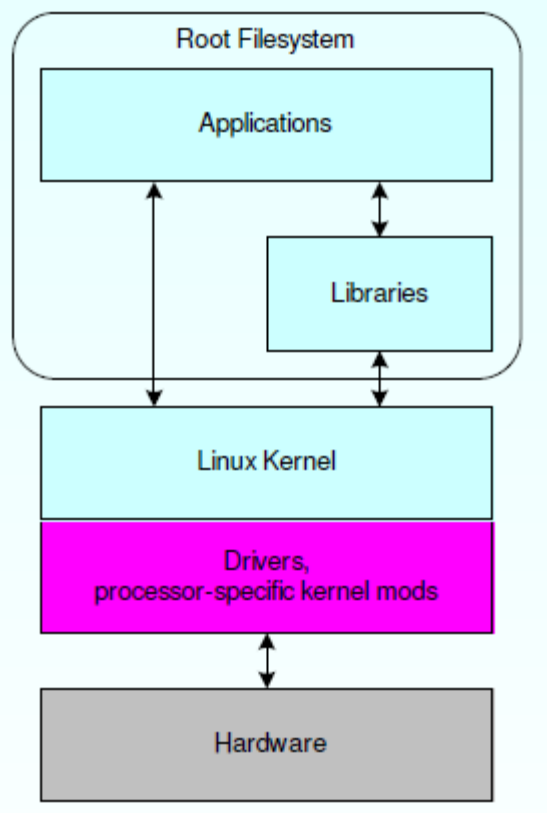
Drivers

- Gerencia recursos de HW (periféricos)
- Suporte a grande número de periféricos
- Podem ser embutidos no kernel ou carregáveis em tempo de execução (run-time)
- Processo bem documentado para criação de novos drivers

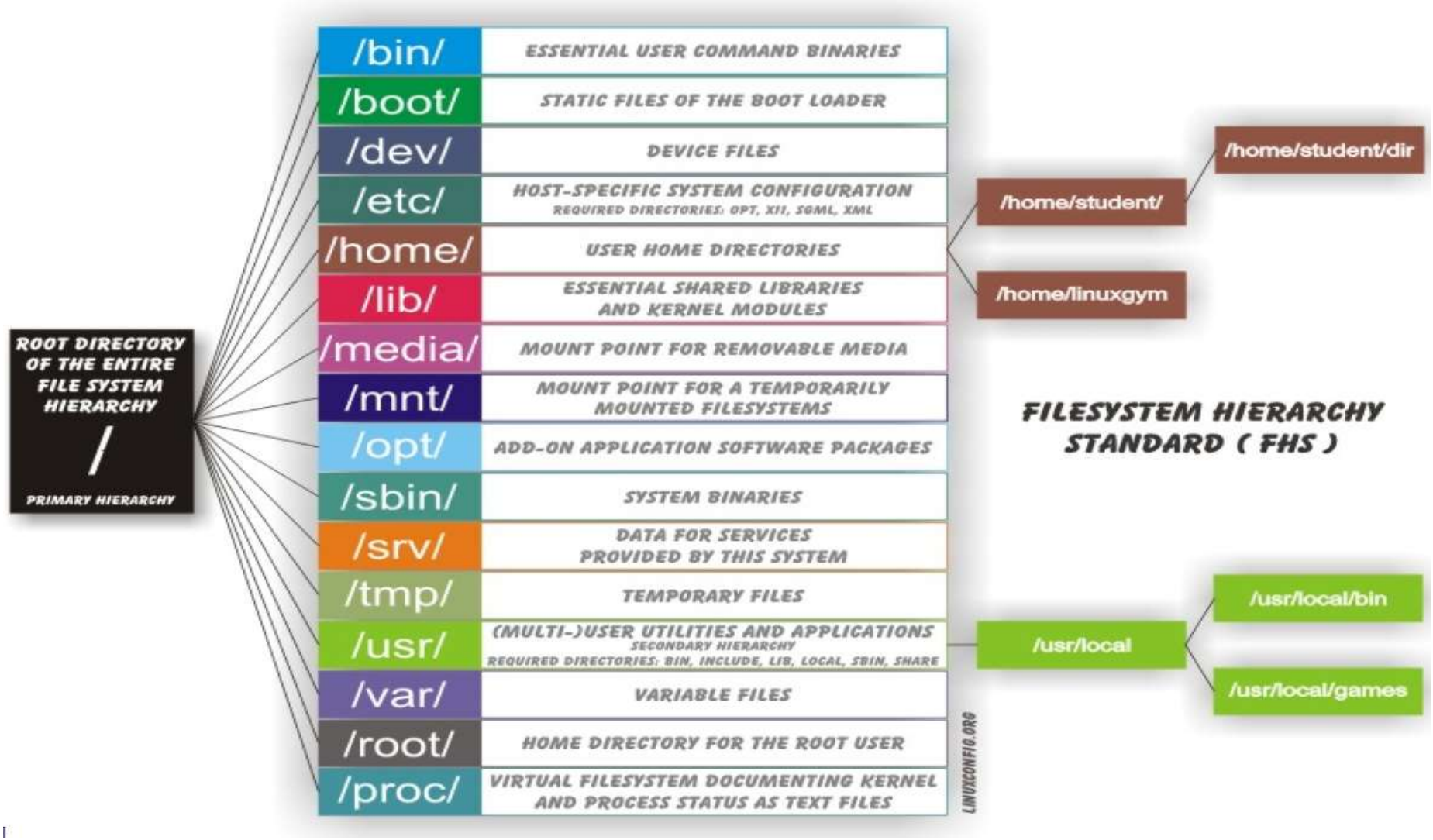


Sistema de arquivos raiz

- Árvore de diretório contendo bibliotecas, scripts, aplicações
  - Organização geralmente segue as convenções de sisteam de arquivo do Linux (/bin, /sbin, /etc, etc.)
- Armazenado na forma de um tipo de sistema de arquivos padrão Linux
  - sistema de arquivo cramfs ou jffs2 comprimido no caso de Flash
  - Ext2/3 para disco

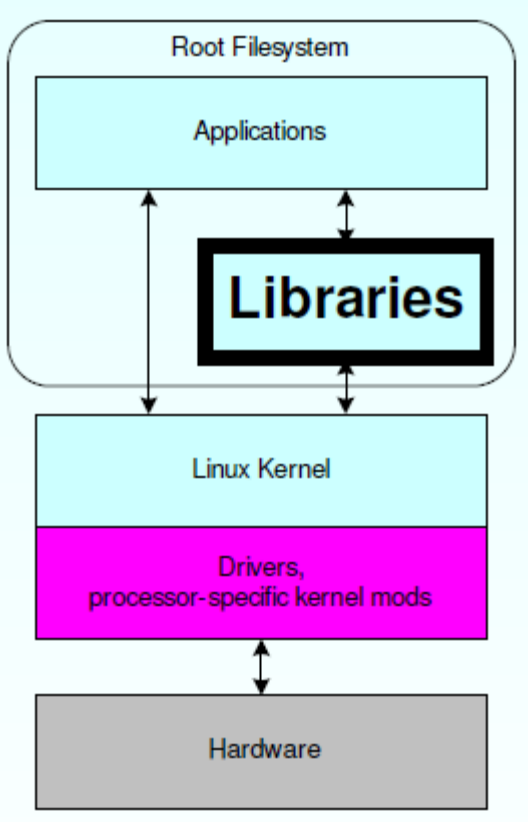


Diretórios do Sistema de Arquivo



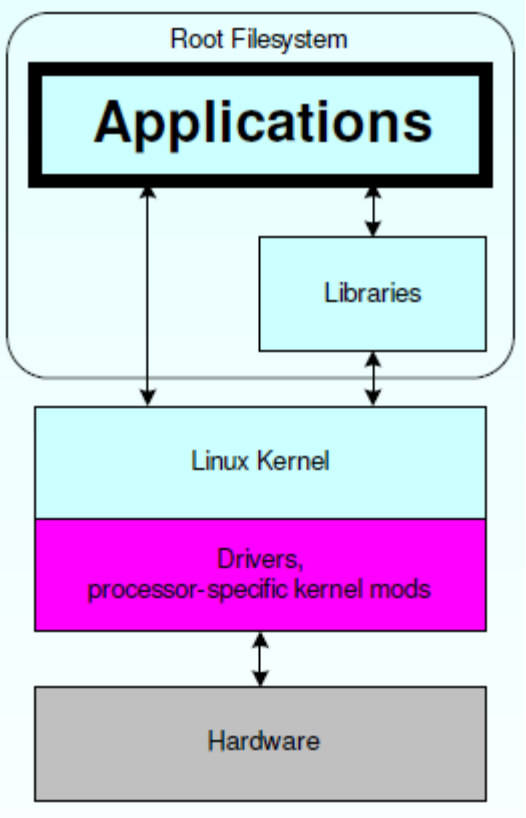
Bibliotecas

- Biblioteca C
  - Funções de utilidade padrão, interface para a funcionalidade do kernel
  - Variantes:
    - Glibc: grande e completo
    - uClibc: pequeno,configurável e voltado para sistemas embarcados
- Outros
  - Pthreads
  - ALSA
  - Suporte a GUI



Aplicações

- padrão Posix/Unix
- Armazenados no sistema de arquivos e carregado em RAM para execução
- Aplicações padrão
  - Busybox
    - Utilitários padrão UNIX em um único pacote
    - Configurável
- Aplicações personalizadas
  - Aplicações de GUI
  - Qualquer aplicação específica de sistema (background network applications, etc.)



Scripts

- Usado para iniciar o sistema de inicialização/shut down
- Controle de acesso, configuração
- Armazenados no diretório /etc do sistema de arquivos raiz

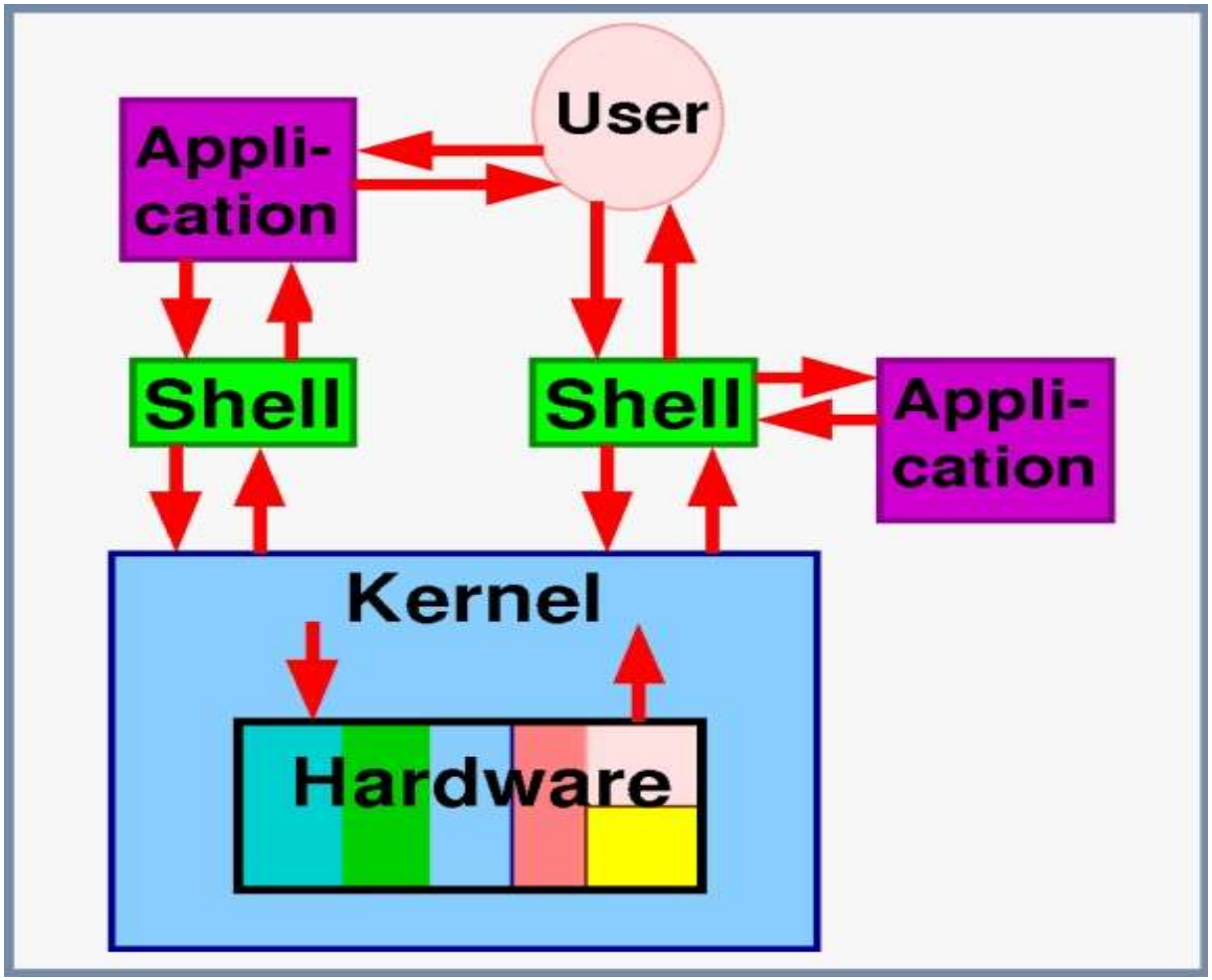
GUI

- Fornece o ambiente do desktop
  - Ambiente de janelas para criação e gerenciamento de aplicações GUI
  - Muitas aplicações disponíveis (produtividade, multimídia, etc.)
- Sistemas portáteis móveis
  - Qtopia Phone Edition

Runtime System

- Console Serial
- Aplicações disparadas na inicialização do sistema
- Daemons (serviços que ficam sempre rodando)
- Kernel threads (por exemplo, coleta de lixo (garbage collection) do JFFS2)

Shell


User
Appli-
cation
Shell
Shell
Appli-
cation
Kernel
Hardware

Shell

- Uma interface entre o sistema Linux e o usuário
- Usado para chamar comandos e programas
- Um interpretador de comandos
- Linguagem de programação poderosa
  - “Shell scripts” = .bat .cmd EXEC REXX
- Grande variedade (bsh; ksh; csh; bash; tcsh)
- Baseado em caracteres em sistema gráfico

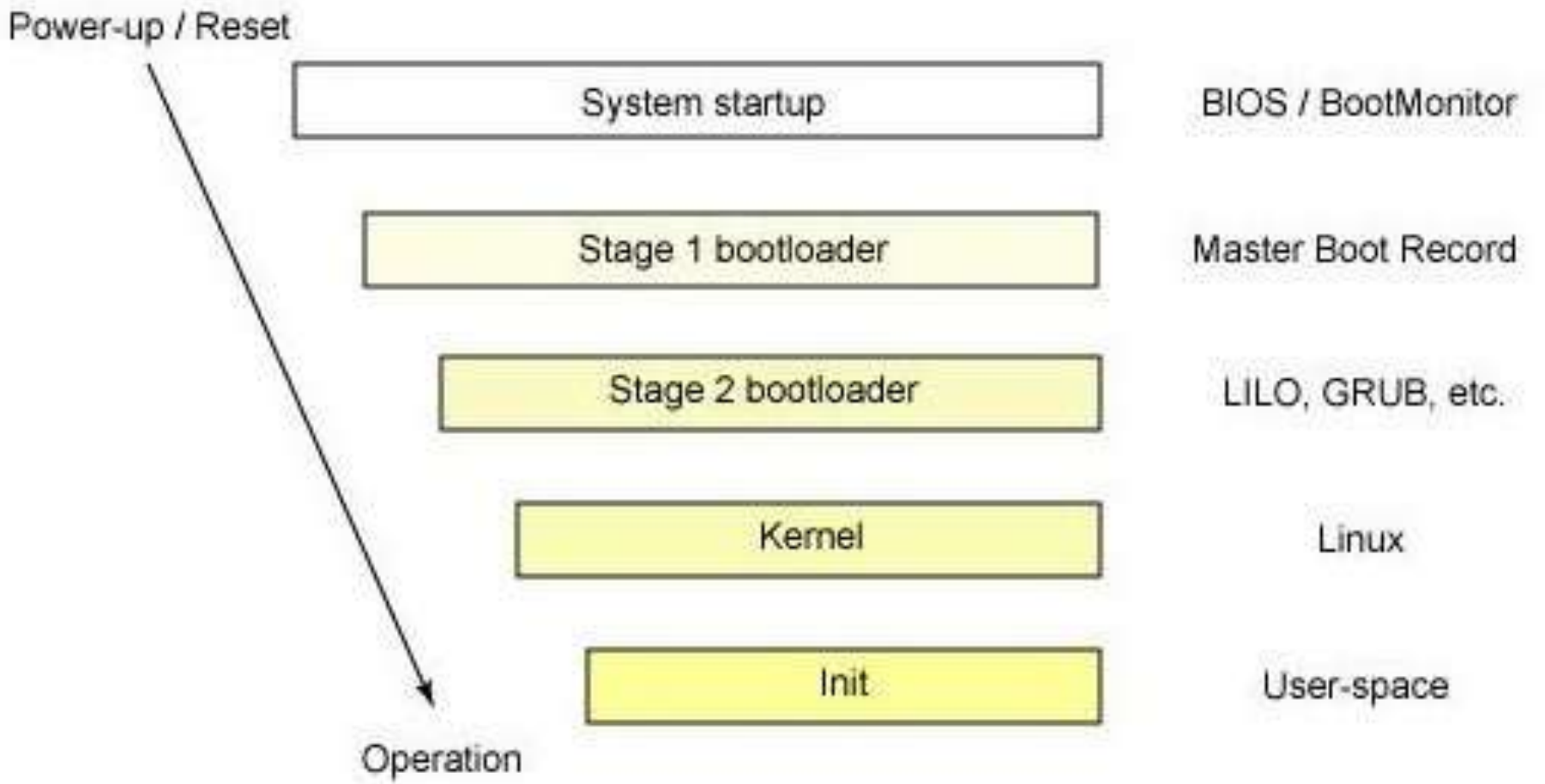
Comandos básicos LINUX

comando	Descrição (preencha a tabela)
ls	
cp	
mv	
vi	
chmod	
Mkdir, rmdir	
cd	
rm	
man	

Shell scripting

```
% cat  > hello.sh << MEU_PROGRAMA
#!/bin/sh
echo “Hello, world”
MEU_PROGRAMA
% chmod +x hello.sh
% hello.sh
Hello, world
```

O Boot de Linux em um PC tradicional



O initrd e initramfs

- O initrd (Initial Ram Disk) é um sistema de arquivos raiz temporário que é montado durante o boot do sistema, para dar suporte ao esquema de boot em duas fases.
- O initrd vem desde as primeiras versões do kernel do Linux (1.xx), e existe para carregar módulos que são requeridos pelo kernel no instante de boot, mas que não foram compilados no kernel por motivo de tamanho de código.
- **A partir do kernel 2.6 kernels o initrd evoluiu para o initramfs.**
- Hoje as distribuições Linux utilizam o initramfs, embora o nome initrd tenha sido mantido em algumas distribuições para o arquivo dentro do diretório /boot. **Exceto em situações em que comparamos a antiga solução initrd e a nova, initramfs, utilizaremos o termo initrd e initramfs de forma indistinta para o sistema de arquivos raiz temporário initramfs**
- O initrd contém vários diretórios e executáveis, como o insmod para carregar módulos no kernel, que permitem que o sistema de arquivos real possa ser montado, após o qual o disco RAM initrd é desmontado e a memória liberada. Em muito sistemas embarcados, o initrd/initramfs é o sistema de arquivos raiz final.

Initrd

- Inicialmente o initrd era construído usando um dispositivo loop. Em seguida, passou a usar um arquivo cpio comprimido
- Este arquivo (cpio) contém uma hierarquia de diretórios simples, similar à hierarquia de de sistema de arquivos do Linux
- Geração

  $ gzip -dc initrd.img | cpio -idv
- /linuxrc e /init contêm o script que é executado para fazer o boot do sistema

/linuxrc ou /init

- /linuxrc ou /init é o primeiro executável depois que o initrd ou initramfs é carregado
- /linuxrc é usado pelo antigo initrd
- /init é usado pelo initramfs
- Este executável é geralmente um shell script; Debian usa /bin/sh mas Red Hat usa /bin/nash

Initrd x Initramfs

- Initramfs é usado pelos kernels 2.6 enquanto initrd foi usado nos kernels 2.4 (ou anteriores)
- initramfs usa um sistema de arquivo RAM dinamicamente-alocado; o antigo initrd usa um disco RAM estaticamente alocado

O initiramfs é obrigatório?

- Não. Por exemplo, se não houver necessidade do uso de módulos correspondentes a dispositivos, por exemplo, de blocos, o initramfs pode não ser necessário

Debian

- O utilitário do Debian para criação de um initramfs é o mkinitramfs
- Outro utilitário é o updateinitramfs, que permite atualizar o initramfs
- yaird tambem gera um initramfs
  - yaird gera um initramfs menor por default; initramfs-tools não
  - initramfs-tools sempre cria initramfs. yaird não gera um initramfs caso não tenha certeza de que irá funcionar
  - Um initramfs gerado por initramfs-tools tem um shell de emergencia caso falhe; yaird não inclui a menos que voce explicitamente especifique

Criação manual do initramfs

- Depende da distribuição:
  - Debian: initramfs-tools
  - Fedora: drakut

    http://fedoraproject.org/wiki/Dracut
  - Gentoo: genkernel, manualmente
- http://ntworks.net/initramfs-simple.html

Configurando o bootloader

- GRUB – trecho extraído de um arquivo grub.conf

```
title Debian GNU/Linux, kernel 2.6.18-4-686
root (hd0,0)
kernel /vmlinuz-2.6.18-4-686 root=/dev/hda3 ro
initrd /initrd.img-2.6.18-4-686
```

- Kernel e initrd apontam para arquivos dentro do diretório /boot/grub

Toolchain

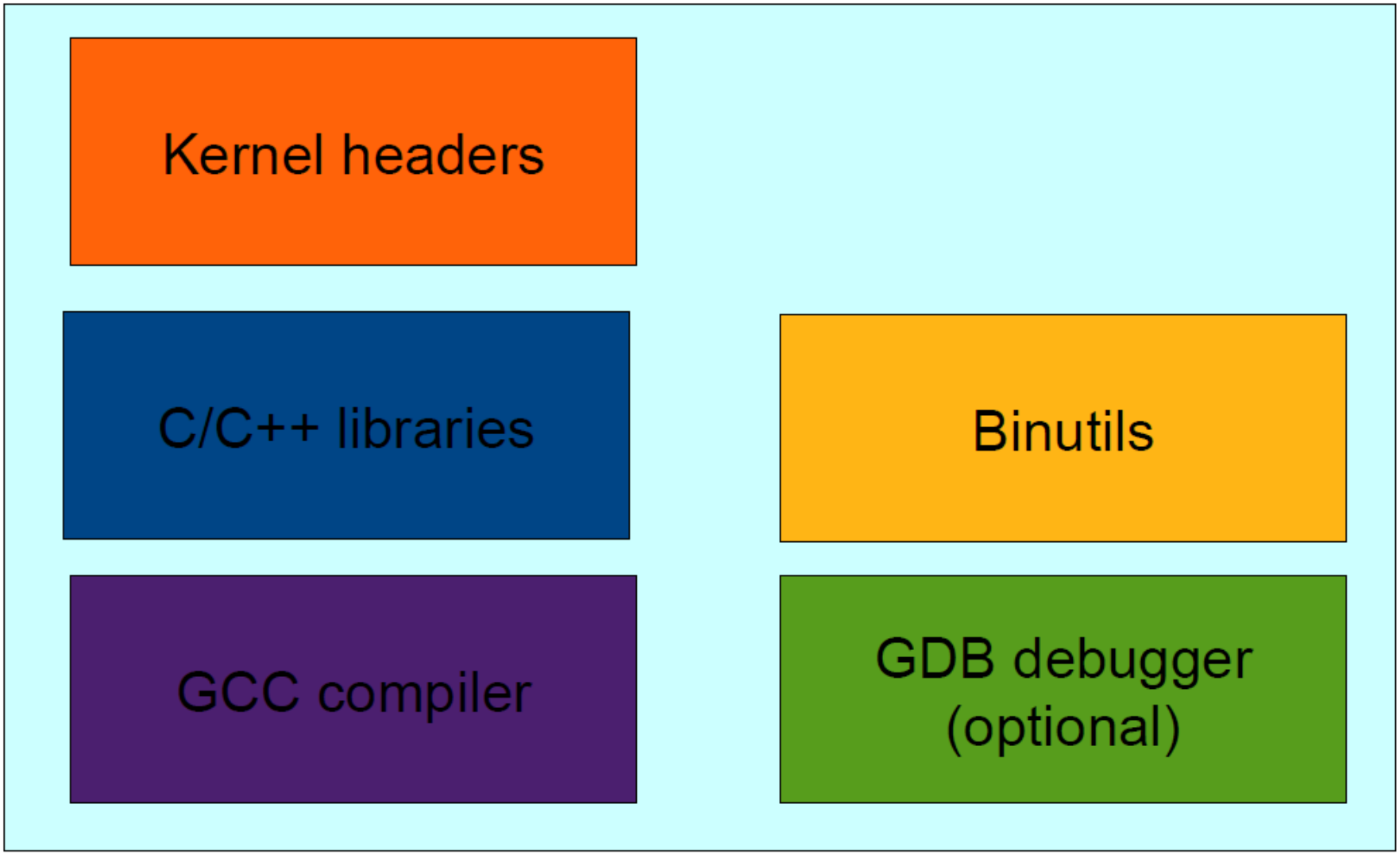
- Trata-se de uma série de ferramentas para geração de código de sistema e aplicações (basicamente o compilador, montador, ligador e o depurador)
- Como o tanto o processador hospedeiro (Host) como o processador (Target) são x86, não iremos empregar um Toolchain de Compilação Cruzada. (verifique se o Linux da estação de desenvolvimento é x86 mesmo ou se é AMD64)
- Usaremos o toolchain nativo para a geração da imagem do Linux embarcado

GNU Tolchain

Projetos incluidos no GNU toolchain são:

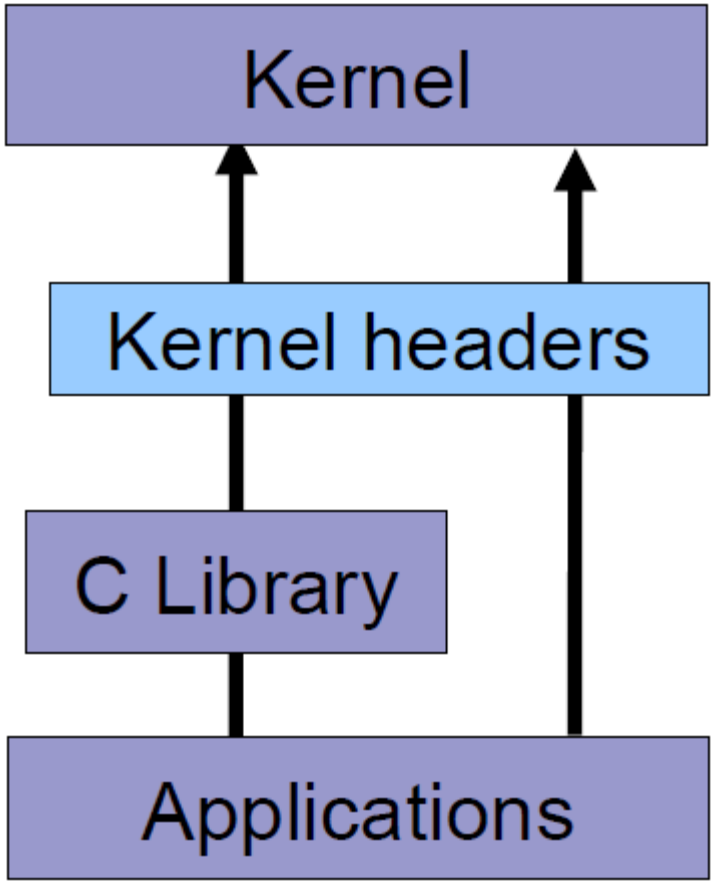
- GNU make: Ferramenta de automação para compilação e Ferramenta de automação para compilação e **build**;
- GNU Compiler Collection (GCC): Conjunto de compiladores para diversas linguagens de programação;
- GNU Binutils: Conjunto de ferramentas incluindo ligador (linker), montador (assembler) e outros;
- GNU Bison: Parser generator
- GNU m4: m4 macro processor
- GNU Debugger (GDB): Ferramenta de depuração de código;
- GNU build system (autotools):
  - Autoconf
  - Autoheader
  - Automake
  - Libtool

Componentes do Toolchain



Kernel headers

- Os programas compilados e a biblioteca C precisam, para interagir como kernel do linux:
  - Chamadas de sistema dispoíveis e seu números
  - Definições de constantes
  - Estruturas de dados
- Os kernel headers provêm tais informações
- Disponíveis em <linux/...> e <asm/...> e alguns outros diretórios dentro de include/ nas fontes do kernel



Binutils

- São um conjunto de ferramentas para gerar e manipular binários para uma dada arquitetura
  - as
  - ld
  - ar, ranlib
  - objdump, readlf, size, nm, strings, strip

GNU Compiler Collection

- O **GNU Compiler Collection: c**hamado usualmente por **GCC,** é um conjunto de compiladores de linguagens de programação produzido pelo projecto GNU. É software livre distribuído pela Free Software Foundation (FSF) sob os termos da GNU GPL, e é um componente-chave do conjunto de ferramentas GNU. É o compilador padrão para sistemas operativos UNIX e Linux e certos sistemas operacionais derivados tais como o Mac OS X.
- O **GNU C Compiler**: Originalmente suportava somente a linguagem de programação C e era designado GNU C Compiler (compilador C GNU). Com o tempo ganhou suporte às linguagens C++, Fortran, Ada, Java e Objective-C, entre outras
- Mais informações: http://gcc.gnu.org/

Biblioteca C

- A biblioteca C é um componente essencial de um sistema C
  - Interface entre a as aplicações e o kernel
  - Provê a API C padrão para facilitar o processo de desenvolvimento
- Bibliotecas C disponíveis:
  - Glibc, uClibc, eglibc, dietlibc, newlibc etc.
- A escolha da biblioteca C deve ser feita no momento de geração do toolchain, uma vez que o compilador GCC é compilado para uma biblioteca C específica

Sistemas de Arquivos: Bloco versus Flash

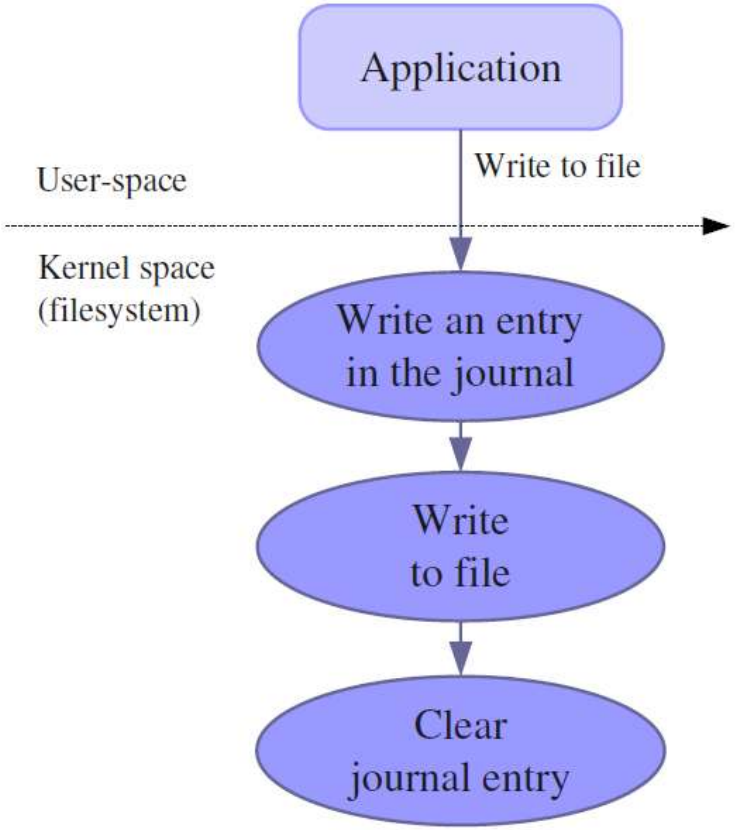
- Dispositivos de armazenamento são classificados em dois tipos principais: **dispositivos de bloco** e **dispositivos flash**
  - Eles são manipulados por subsistemas diferentes e sistemas de arquivos diferentes
- **Dispositivos de bloco podem ser lidos e escritos em blocos**, sem necessidade de apagamento, e **não se desgastam** com o uso quando muito usado
  - Discos rígidos, discos flexíveis, discos em RAM
  - Pen Drives USB, Compact Flash, cartão SD, são baseados em memória flash, mas possuem um controlador integrado que emula um dispositivo de bloco
- **Dispositivos Flash podem ser lidos, mas a escrita requer apagamento,** em geral com o tamanho maior do que o do bloco
  - NOR flash, NAND flash

Sistemas de arquivo de bloco tradicionais

- Podem ficar em um estado não consistente após uma falha de sistema (system crash) ou após um desligamento repentino do sistema, o que acaba exigindo que todo o sistema de arquivos seja verificado após o reinício.
  - ext2: sistema de arquivos tradicional do Linux (reparar com fsck.ext2)

Journaled File System

- Projetado para manter um estado coerente mesmo após uma falha de sistema ou desligamento repentino
- Todas as operações de escritas são descritas no “journal” (log) antes de serem realizadas em arquivo



Journaled Block Filesystems

Journaled filesystems

- ext3: ext2 com suporte a journaling

  ext4: a nova geração, com novos aprimoramentos.

  Pronto para produção. São os sistemas de arquivos default de sistemas Linux.
- O Linux kernel suporta muitos outros sistemas de arquivo: reiserFS, JFS, XFS, etc. Cada um deles tem suas próprias características (*workloads* científicos, servidores etc)
- btrfs (“Butter F S”)

  A próxima geração

Criando volumes ext2/ext3

- Para criar um sistema de arquivos ext2/ext3 vazio em um dispositivo de bloco ou dentro de um arquivo imagem existente
  - mkfs.ext2 /dev/hda3
  - mkfs.ext3 /dev/sda2
  - mkfs.ext2 disk.img
- Para criar uma imagem de sistema de arquivo a partir de um diretório contendo todos os arquivos e instruções
  - Use a ferramenta genext2fs, do pacote do mesmo nome
    genext2fs –d rootfs/ rootfs.img
  - Sua imagem está então pronta para ser transferida para um dispositivo de bloco

Squashfs

- Squashfs: http://squashfs.sourceforge.net
- Readonly, sistema de arquivo comprimido para dispositivos de bloco

  Bom para partes do sistema de arquivo que podem ser readonly (kernel, binaries...)
- Grande taxa de compressão e desempenho de acesso tipo leitura
- Usado na maioria das distribuições live CDs e live USB
- Suporta compressão LZO para melhor desempenho em sistemas embarcados com CPUs mais lentas (às custas de uma taxa de compressão menor)
- Disponível no Linux desde a versão 2.6.29.

Benchmarks: (aprox. 3 vezes menor que ext3, e 24 vezes mais rápido)

http://elinux.org/Squash_Fs_Comparisons

Squashfs – como usar

- Instalar o pacote squashfs-tools
- Criação da imagem
- Na estação de trabalho, criar o sistema de arquivo imagem:
  mksquashfs rootfs/ rootfs.sqfs
- Cuidado: se a imagem já existir, remova-o primeiro,
  ou use a opção no-append.
- Instalação da imagem
- Vamos assumir que a sua partição está em /dev/sdc1
- No alvo, copie a imagem do sistema de arquivo no dispositivo (CUIDADO: não rode na sua estação de trabalho: pode destruir partições de sistema críticos)
  dd if=rootfs.sqfs of=/dev/sdc1
- Monte seu sistema de arquivo:
  mount –t squashfs /dev/sdc1 /mnt/root

tmpfs

Útil para armazenar dados temporários em RAM: arquivos de log de sistema, dados de conexão, arquivos temporários...

- É uma alternativa ao uso de ramdisks! Ramdisks tem muitos limitações: fixo em tamanho, espaço restante não usável como RAM, arquivos duplicados em RAM (no dispositivo de blocos e no cache de arquivos)!
- Configuração do tmpfs configuration: File systems -> Pseudo filesystems

  Vive no cache de arquivos do Linux. Não gasta RAM: cresce e encolhe para acomodar os arquivos armazenados. Salva RAM: não há duplicação; pode swap out pages para disco quando necessário.
- Como usar: escolher um nome para distinguir as várias instâncias de tmpfs que houver. Exemplos:

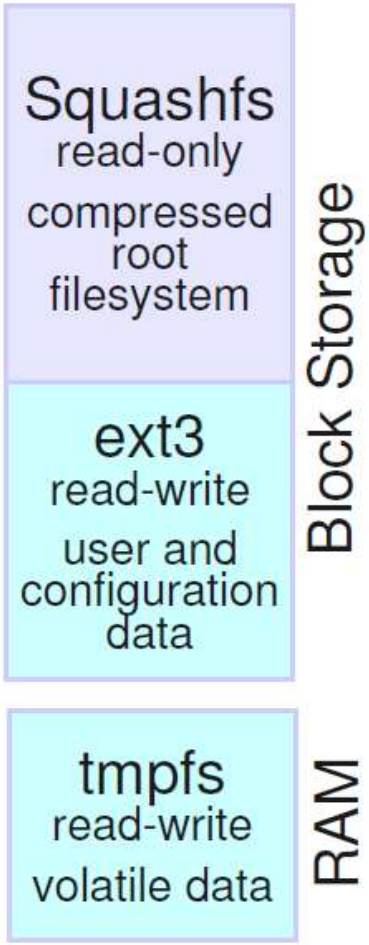
  mount –t tmpfs varrun /var/run

  mount –t tmpfs udev /dev

Misturando sistemas de arquivo read-only com read-write

Boa idéia para dividir o seu armazenamento de bloco em:

- Uma partição comprimida readonly (Squashfs)
- Tipicamente usado para o root filesystem (binários, kernel...).
- Compressão salva espaço. Acesso readonly protege seu sistema de enganos e corrompimento de dados.
- Uma partição readwrite com um sistema de arquivos journaled (como o ext3)
- Usado para armazenar dados de configuração ou de usuário.
- Garante integridade do sistema de arquivos após quedas de força ou panes.
- Armazenamento Ram para arquivos temporários (tmpfs)



Ext2 ou Ext3 em pendrive USB?

- Duas visões:
  - Problema de Wearing-out em pendrives USB:
    - Não usar sistemas com journaling, como o ext3 em pendrives USB
  - Necessidade de evitar problemas de inconsistência do sistema de arquivo em sistemas embarcados
    - Usar sistemas com journaling, como o ext3
  - Questões de tamanho, desempenho...
    - Usar o Ext2, devido ao menor overhead
- Resposta?
  - Há uma tendência de melhoria dos dispositivos pendrive em relação ao problema de wearing-out
    - Hoje(até 2011 ?): usar ext2
    - Amanhã (2012 ?): ext3

Gerando o Linux Embarcado

System Building

- Objetivo
  - Integrar todos os componentes de sofware, próprios e de terceiros, em um sistema de arquivos
  - Envolve o download, extração, configuração, compilação de todos os componentes, além de possíveis consertos, e adaptação dos arquivos de configuração
- Diversas soluções
  - Manualmente
  - System building tools
  - Distribuições

Manualmente

- Fazer o building de um sistema alvo envolve downloading, configuração, compilação e instalação de todos os componentes do sistema.
- Todas as bibliotecas e dependências devem ser configuradas, compiladas e instaladas na ordem certa.
- Às vezes, o sistema de build usado pela biblioteca ou aplicações não é muito “amigável” com a compilação cruzada, de forma que algumas adaptações se tornam necessárias.
- Não há infraestrutura para reproduzir o build do zero, o que pode causar problemas se algum componente necessitar ser trocado, se alguma outra pessoa assume o projeto, etc.

Fundamentos do Sistema

- Um sistema de arquivo raiz precisa pelo menos de:
  - Uma hierarquia tradicional de diretório, com /bin, /etc, /lib, /root, /usr/bin, /usr/lib, /usr/share, /usr/sbin, /var, /sbin
  - Um conjunto de utilitários básicos, provendo pelo menos o programa **init**, um shell e outras ferramentas de linha de comandos UNIX tradicionais. É geralmente provido pelo
  - A biblioteca C e bibliotecas relacionadas (thread, math, etc.) instaladas em /lib
  - Alguns arquivos de configuração, como /etc/inittab, e scripts de inicialização em /etc/init.d
- No topo desta base, comum à maioria dos sistemas Linux Embarcados, podemos adicionar componentes próprios ou de terceiros

Espaços de Build e Alvo

Além do espaço do hospedeiro (estação de trabalho), devemos distinguir dois espaços importantes de desenvolvimento: o build e o alvo

- Busybox e biblioteca C, constituem o núcleo do sistema de arquivos raiz do alvo
- No building de outros componentes, deve-se distinguir dois diretórios
  - O espaço « target », que contém o sistema de arquivo do alvo, tudo que é necessário para a execução da aplicação
  - O espaço « build », que contém muito mais arquivos que o espaço «target», uma vez que é usado para manter tudo que é necessário para **compilar bibliotecas e aplicações.** Isto inclui os **headers,** documentação e outros arquivos de configuração

Build Systems

- Cada componente open-source vem com um mecanismo para configurar, compilar e instalar
  - Um simples Makefile
    - Deve-se estudar o código e ver como modificar para compilação cruzada
  - Um build system baseado no Autotools
  - CMake, http://www.cmake.org/
    - Mais recente e mais simples que o Autotools. Usado por grandes projetos como o KDE ou Second Life
  - Scons, http://www.scons.org/
  - Waf, http://code.google.com/p/waf/
  - Outros

Autotools

- Uma família de ferramentas, que associados, formam um sistema de build completo e extensível
  - **autoconf** é usado para manipular a configuração do pacote de software
  - **automake** é usado para gerar os Makefiles necessários para o build do pacote de software
  - **pkgconfig** é usado para facilitar a compilação contra as bibliotecas compartilhadas já instaladas
  - **libtool** é usado para manipular a geração da bibliotecas compartilhadas em uma forma independente de sistema
- A maioria destas ferramentas são antigas e relativamente complicadas de usar, mas são usadas pela maioria dos pacotes de sofware hoje existentes. Deve-se portanto ter compreender que eles são e como eles funcionam de forma básica.
- http://www.gnu.org/s/hello/manual/automake/Autotools-Introduction.html#Autotools-Introduction

Utilizando uma placa mãe X86 como placa de desenvolvimento embarcado

- Placa Mãe Intel com processador ATOM 230 dual-threaded, 32/64 bits
- Modelo: D945GCLF

Intel D945CGLF - Especificações

- Connectors (rear) 2x PS/2
  1x Parallel
  1x D-SUB15 (VGA)
  1x RS232

  1x LAN 10/100MBit (Realtek)
  3x Sound (Line Out, Line In, Microphone)
  Audio
  Realtek High Definition Audio (ALC)

  Connectors (internal) PCI (with support for 1>2 PCI riser)
  AUDIO
  2 x USB
  2 x SATA (3GBs)

  Power
  20 Pin ATX, P4-connector

  1 x IDE
  Included
  D945GCLF Main board (with 1.6Ghz Atom CPU)
  I/O ATX rear plate

  SATA cable
  IDE cable
  Drivers CD

  Form Factor
  Mini-ITX (17x17cm)

- **Processor**
  Intel Atom 230 @ 1x 1.60GHz (Diamondville 45nm)

  Chipset
  Intel 82945G (ICH7)

  L2 Cache
  512kb

  System Memory
  1x DDR2 533/667 RAM (max 2GB)

  Graphics
  Intel GMA 950

- Memory

  instalado: 1GBytes DDR2, 667

- USB Keyboard and Mouse

- External 1024 768 Display

Características peculiares da placa D945 para aplicações embarcadas

- A D945 é uma placa-mãe para desktops x86, empregando um processador ATOM
- Como placa de desenvolvimento de sistemas embarcados, ele apresenta as seguintes vantagens e desvantagens:
- Vantagens:
  - Fácil disponibilidade
  - Baixo custo
- Desvantagens:
  - Não possui memória Flash NAND para dados/programas embutida na placa
  - Não possui interface para cartão SD
  - Não possui interface serial para depuração
  - Não possui dispositivos de suporte ao desenvolvimento e depuração, como displays, LEDs, teclados ou chaves embutidos
- Outros:
  - Possui uma memória flash com Bios, portanto o U-Boot ou outro bootloader típico de sistemas embarcados
  - Apesar de possuir interface para disco rígido (SATA), não instalaremos disco rígido, disco flexível etc.
  - O dispositivo de armazenamento será um pendrive USB (USB 2.0), tipicamente de 4 Gbytes. Apesar de ser implementado com memória flash, o controlador interno do pendrive irá oferecer uma interface de dispositivo de armazenamento de bloco (squashfs, ext3) , de forma que os sistemas de arquivo empregados serão os de blocos, e não os de armazenamento flash.

Parte Prática I

Atencão: Seguir o roteiro detalhado nas transparências da Parte Prática

Configurar e compilar o kernel

Trata-se de gerar uma versão “enxuta” do kernel do Linux, que atenda às características e restrições da plataforma alvo

- Puxar o código fonte do Linux do site do linux kernel (http://kernel.org)
- Descompactar
- Caso haja um arquivo de configuração pronta, copiar para o arquivo .config
- Executar **make menuconfig** e examinar as opções selecionadas para se certificar que os driver de dispositivos estão corretamente contemplados (verfique a especificação técnica da placa)
- Executar **make.** O kernel ficará em /sources/2.6.XX arch/i386/boot/bzImage

Criação do sistema de arquivos raiz

- dd if=/dev/zero of=rootfs.img bs=1024k count=1
- /sbin/mkfs.ext2 -i 1024 -F rootfs.img

Como popular o rootfs

- Arquivos do Busybox
- Diretórios adicionais
  - /dev (devfs)
  - /proc (procfs)
  - /sys (sysfs – nas versões mais modernas do kernel o sysfs agregar algumas informações que estavam no procfs e do devfs))
- Scripts de inicialização

Busybox

- Baixar o código fonte (http://busybox.net)
- Configure, escolhendo uClib como a biblioteca C

  make menuconfig
- Compile e instale (no diretório _install do busybox)

  make

  make install

Criação e Inicialização do sistema de arquivos raiz

- Vamos criar dispositivos dentro do seu próprio diretório home

  mkdir ~/mnt
- Crie um ponto de montagem no seu diretório home e monte o sistema de arquivos, utilizado o dispositivo loop

  mkdir ~/mnt/rootfs

  mount -o loop rootfs.img /mnt/rootfs

- Copie o busybox para dentro do sistema de arquivos

  rsync -a ~/busybox/_install/ /mnt/rootfs/

  chown -R root:root ~/mnt/rootfs/

  sync

Outros diretórios, não presentes no Busysbox

- /dev
- /proc
- /sys

Criando /dev estaticamente

- 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 (*major number*) e menores (*minor number*):
  - ls –l /dev/console
  - ls –l /dev/null

Montando /proc e /sys

- Geralmente feito pelo init executando os scipts dentro de /etc/init.d
- Montar /proc:

  mount –t proc none /proc
- Montar /sys:

  mount –t sysfs none /sys
- Onde:

  Sysfs: Tipo do Sistema de Arquivos

  Opção none: Raw device ou filesystem image. No caso de virtual filesystems, pode-se colocar qq. string

  /sys: Ponto de Montagem

Arquivo de configuração e inicialização: /etc/inittab

- Criando o arquivo /etc/inittab requerido pelo busybox init, exemplo extraído 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
```

Script de inicialização /etc/init.d/rcS

```
#!/bin/sh
mount –t proc none /proc
mount –t sysfs none /sys
```

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!

Bootloader

- Em sistemas Linux usa-se bootloaders para carregar o sistema operacional e executá-lo. Exemplos:
  - Em desktops: LILO, GRUB etc.
  - Em embarcados: U-Boot
- Como a nossa placa de desenvolvimento é uma placa-mão de PC ATOM, com bastante memória, vamos utilizar o GRUB
- Uma alternativa ao processo de BOOT através do BIOS tradicional é usar:
  - coreboot
  - http://www.coreboot.org/Welcome_to_coreboot
  - O coreboot substitui o BIOS tradicional presente nas placas mão X86

Criação de um sistema “bootável”em PEN DRIVE USB

Gerando uma imagem ISO

- Já se dispõe até o presente do kernel comprimido, do sistema de arquivos raiz, dos binários para popular o sistema de arquivos raiz e dos scripts de inicialização
- Vamos gerar uma imagem ISO
- Geração de uma imagem ISO:

 # dd if=/dev/cdrom
of=/home/usuário/nome_da_imagem.iso

if = onde está o arquivo que se quer criar a iso
of = destino, para onde será criada a imagem iso.

Geração de Live USB

- Manualmente

- Ferramentas automáticas rodando sobre o Linux
  - Unetbootin (Ubuntu, Fedora, LinuxMint, ...)
  - Ubuntu Live USB Creator (Ubuntu)
  - Fedora Live USB Creator (Fedora)
  - Multisystem LiveUSB MultiBoot

Passos típicos para geração de Live USB

- Um USB flash drive precisa ser conectado ao sistema, e ser detectado
- Pode se preciso criar uma ou mais partições no USB flash drive
- O flag "bootable" deve ser ativado na partição primária do USB flash drive
- Um MBR deve ser escrito na partição primária do USB flash drive
- A partição deve ser formatada
- Um bootloader deve ser instalado na partição
- Um arquivo de configuração bootloader deve ser escrito
- Os arquivos necessários do sistema operacional e as aplicações default devem ser copiadas para o USB flash drive

O flash USB drive

- Um dispositivo de armazenamento flash USB pode ser utilizado de duas maneiras:
  - Como dispositivo de blocos
  - Como dispositivo emulador de CD/DVD ou floppy, compatível com o padrão “el-torito”
- Para que o Pen Drive USB seja visto como um CD (no caso de um LIVE CD) , ele deve ser formatado segundo o padrão ISO9660. Para ser “bootável”, ele deve seguir o padrão “el-torito”
- Em ambos, os caso, para ser “bootável:
  - o dispositivo precisa ter determinadas locações gravadas com arquivo o de inicialização e carga
  - A partição onde se encontra o sistema deve ser “bootável”
  - O BIOS deve suportar “boot”a partir do USB e do tipo de armazenamento escolhido (blocos ou “el torito”)

Geração manual de um Live USB

- Geração de um Live USB com duas partições:
  - Squashfs: read-only, para alocação do kernel e do rootfs
  - Ext2: R/W, para os dados do usuário e aplicações (home)

Passos da geração manual para Ubuntu Live

- http://edoceo.com/liber/ubuntu-live-usb
- Particularidades do UBUNTU:
  - O ubuntu Live utiliza o CASPER, que está sendo substituido em outras distribuições pelo LIVE-INITRAMFS),
  - O Ubuntu também utiliza o RAMDISK INITRD em vez do INITRAMFS
- Re-particionamento e formatação do PEN DRIVE USB.
  - Escolher um PEN DRIVE USB, padrão USB 2.0, tamanho 4 ou 8 GBytes
  - Garanta que a 1a. Partição tem pelo menos 1GBytes e é marcada como “bootável”
  - Formatar o restante como ext2
- reparticionar o disco e marcar a 1ª. Partição como “bootável”
  cfdisk /dev/sda

- Formatar as partições como ext2 (sem journaling)
  mkfs.ext2 /dev/sda1
  mkfs.ext2 /dev/sda2

Cont...

- Montar o dispositivo correspondente à 1ª. Partição e copiar o kernel e demais arquivos e diretórios
- criar o diretório usb_disc

  mkdir /mnt/usb_disc
- montar a 1ª. Partição do PEN DRIVE

  mount /dev/sda1 /mnt/usb_disc
- criar o diretório do Live USB

  mkdir /mnt/live_usb \
- criar um arquivo com um sistema de arquivo dentro (device loop)

mount -o loop (caminho)\linux-x86.iso /mnt/live_cd

cd /mnt/live_usb/

cp -aR casper disctree dists install pool preseed .disk \ /mnt/usb_disc

Cont...

- Editar o arquivo de configuração /mnt/usb_disc/extlinux/extlinux.conf

cd / mkdir /mnt/usb_disc/extlinux extlinux -i /mnt/usb_disc/extlinux

DEFAULT /casper/vmlinuz

APPEND file=preseed/kubuntu.seed boot=casper initrd=/casper/initrd.gz ramdisk_size=1048576 root=/dev/ram0 rw quiet splash --

Método usando Ferramentas: UNetbootin

- UNetbootin tem suporte para automaticamente gravar as seguintes distribuições Linux: Ubuntu, Debian, Fedora, PCLinuxOS, Linux Mint, Sabayon Linux, Gentoo, MEPIS, openSUSE, Zenwalk, Slax, Dreamlinux, Arch Linux, Elive, CentOS, Damn Small Linux, Mandriva, SliTaz, FaunOS, Puppy Linux, FreeBSD, gNewSense, Frugalware, entre outros.
- Cria uma unidade de armazenamento USB “bootável”, tendo duas partições: o bootloader e o SO.
- Suporta persistência

Unetbootin – como funciona?

- O Unetbootin tem três fases: download, extração e instalação do bootloader
- Download
  - Descarregar pacotes de sofware para gravação
- Extração
- Instalação do bootloader:
  - Para o modo de criação Live USB, o UNetbootin gera um arquivo de configuração syslinux apropriado em /syslinux.cfg, e torna o seu USB drive “bootável”.
  - Se, em sistemas Linux, extlinux estiver instalado e o USB drive for ext2 ou ext3, então extlinux é usado; o arquivo de configuração é instalado em /extlinux.conf. A partição no qual ele foi instalado é também marcado como ativo

Suporte a persistência

- Persistência de dados em LIVE USB é obtido através de um segundo sistema de arquivos, read-write, que possa salvar mudanças no sistema (customizações como layout de teclado, numlock, resolução de tela, preferências, instalação de pacotes adicionais etc.), além de dados do usuário

O casper e o live-initramfs

- Casper
  - Utilizado pelo Ubuntu para a criação de Live-USB (ou Live-CD/DVD) com suporte a persistência
  - Exige um volume de nome casper-rw para que o sistema identifique que há um diretório onde os dados podem ser mantidos de forma persistente.
  - Na fase de boot, o usuário deve fornecer a opção de execução com persistência

- Live-initramfs
  - Utilizado pelo Debian para a criação de Live-USB (ou Live-CD/DVD) com suporte a persistência
  - Exige um volume de nome live-rw para que o sistema identifique que há um diretório onde os dados podem ser mantidos de forma persistente.
  - Na fase de boot, o usuário deve ser fornecer a opção de execução com persistência

Usando o UNetbootin

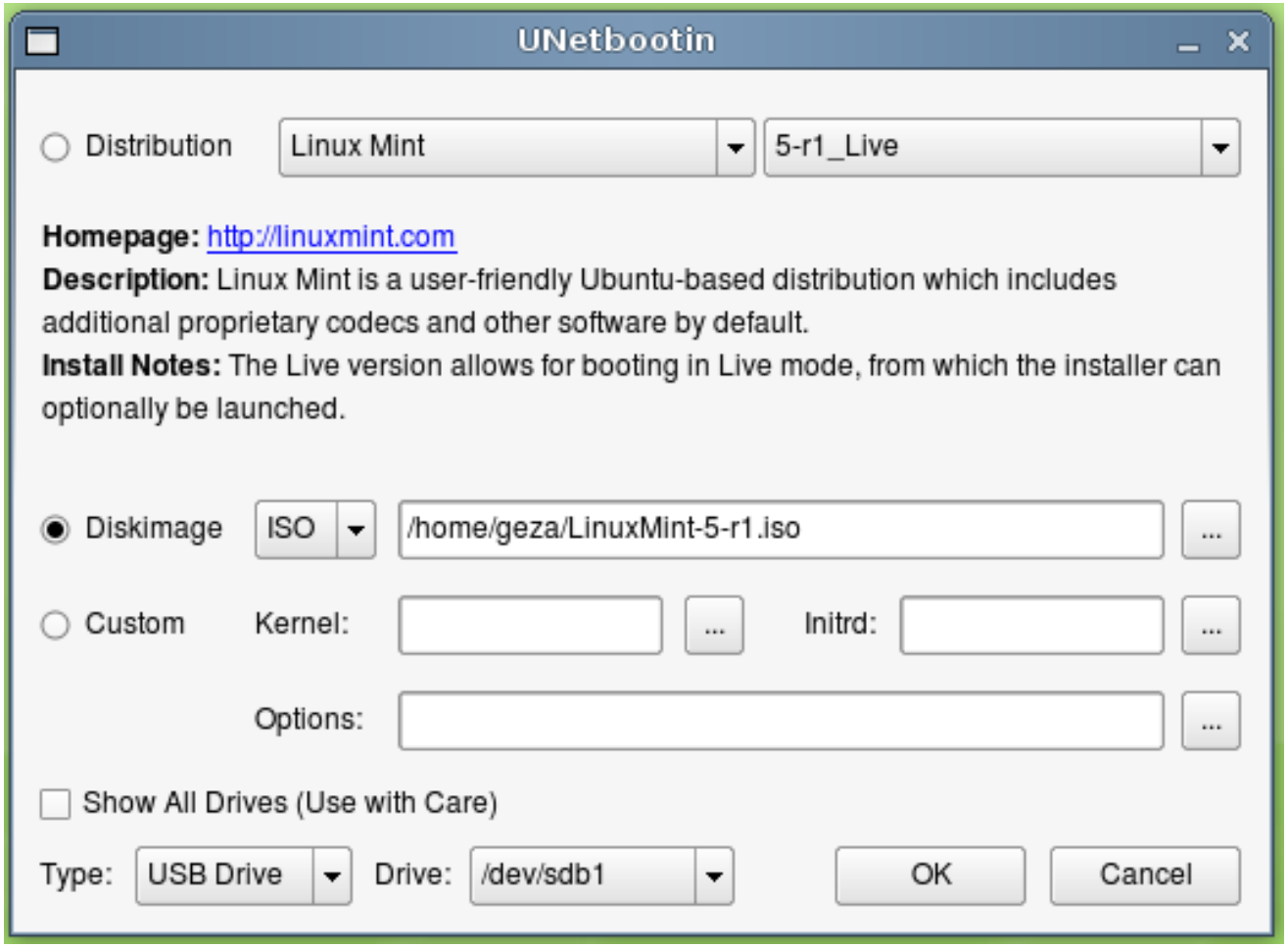
- Para gravar a imagem ISO no pendrive pode-se utilizar o programa UNetbootin
- Faça a descarga do programa em:

http://ufpr.dl.sourceforge.net/sourceforge/unetbootin/unetbootin-linux-248

- Torne o programa executável, mudando suas permissões

$ chmod + x Unetbootin-linux

UNetbootin


UNetbootin
Distribution
Linux Mint
5-r1_Live
Homepage: http://linuxmint.com
Description: Linux Mint is a user-friendly Ubuntu-based distribution which includes additional proprietary codecs and other software by default.
Install Notes: The Live version allows for booting in Live mode, from which the installer can optionally be launched.
Diskimage
ISO
/home/geza/LinuxMint-5-r1.iso
...
Custom
Kernel:
...
Initrd:
...
Options:
...
Show All Drives (Use with Care)
Type:
USB Drive
Drive:
/dev/sdb1
OK
Cancel

UNetbootin – Linux suporte a persistência

- Na versão Windows, o Unetbootin provê modo persistência para distribuições Live Ubuntu. O usuário pode selecionar o tamanho do espaço reservado para persistência.

Executando no emulador QEMU

QEMU

- O QEMU pode ser utilizado para simular o sistema alvo
- Temos duas alternativas:
  - Especificar na execução do QEMU diretamente o rootfs e a imagem do kernel
  - Especificar na execução do QEMU o boot a partir de um Live USB com o Linux Embarcado gerado
- Vamos testar estas duas formas.
- Sempre que se gerar um Live USB é interessante utilizar o QEMU para testá-lo.

QEMU: executando diretamente a imagem do kernel e o rootfs

- Qemu –m 32 – hda rootfs.img –kernel linux-2.6.38/arch/i386/boot/bzImage –append "root=/dev/hda clock=pit"

- Onde:

  -m 32: Quantidade de memória no sistema emulado

  -hda rootfs.img: Conteúdo do sistema de arquivos

  -kernel linux-2.6.38/arch/i386/boot/bzImage: Kernel

  -append "root=/dev/hda clock=pit": linha de comando do kernel

QEMU: executando o LIVE USB

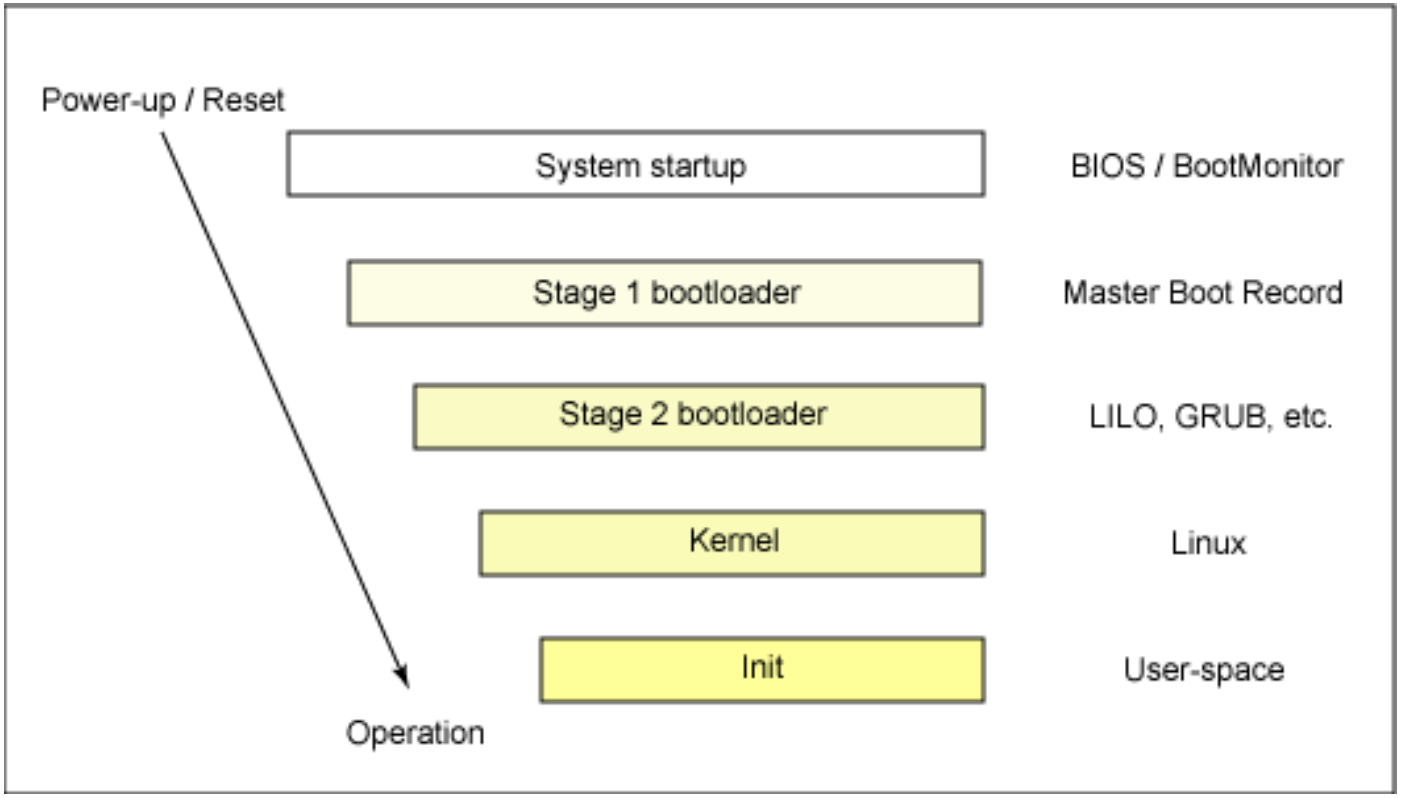
- Pesquise na internet a linha de comandos para ordenar a execução do QEMU a partir de um LIVE USB

Executando na placa ATOM

Rodando na placa Alvo ATOM

- Placa Mãe Intel ATOM, com 1GBytes de RAM, sem disco rígido e sem leitor de DVD/CD
- Dois métodos de transferência de código, selecionáveis pelo BIOS
  - Método 1: Interface de Rede Ethernet (desde que seja por PXE)
  - Método 2: Live USB – Pendrive USB
    - Deve ser um dispositivo USB da classe armazenamento de massa (USB boot não suporta booting a partir de um CD padrão El Torito em drive USB).
    - Procure utilizar dispositivos USB novos, que possuem atualmente (2011) mais confiabilidade, mais ciclos de escrita, maior velocidade (use USB 2.0 ou superior), tecnologia NAND
- Vamos utilizar o método 2, escolhendo a opção “boot from USB device” do Bios

Relembrando: etapas do boot do Linux no PC



Mais detalhes do boot

- Ligar a energia da placa
- O Bios assume o controle
- O Bios inicializa algum hardware
- O Bios carrega o Bootloader
- O Bootloader carrega o sistema operacional
- O Kernel faz uma busca do hardware
- O Kernel encontra e monta o root filesystem
- O Kernel executa o init

O que difere de um sistema embarcado típico?

- Em sistemas embarcados típicos não há um BIOS.
- Um bootloader, como o U-Boot deve ser gravado em um dispositivo de armazenamento não volátil para a inicialização do HW e disparo do kernel.

Boot do Linux embarcado na placa Intel ATOM

- Ao contrário dos sistemas embarcados típicos a nossa placa tem um BIOS, pois a placa de desenvolvimento é uma placa-mãe padrão PC
- Portanto, o nosso boot irá envolver o BIOS
- Apesar do nosso Linux ser uma versão “enxuta’, pois foi gerada tendo em vista sistemas embarcados, ela não difere conceitualmente de uma versão de Linux conhecida como Live-USB.

Testando na placa Intel ATOM

- monte o aparato experimental (placa, fonte, monitor, teclado, mouse, rede)
- Conecte o pendrive USB com a imagem do SO
- Ligue a fonte e verifique se o BIOS está configurado para boot a partir de armazenamento USB
- Acompanhe o processo de boot de 2 estágios

Testando com imagens Linux prontas

- Caso se deseje testar o procedimento de geração do pendrive USB bootável com imagens ISO de Linux prontas, como o Ubuntu, LinuxMint, é só repetir o processo descarregando da Internet a distribuição desejada.
- Como a plataforma alvo é uma motherboard PC Intel ATOM, e portanto x86, o sistema deverá rodar sem problemas.
- Como essas distribuições não são voltadas geralmente para sistemas embarcados, elas possuem um conjunto maior de recursos e aplicações

Próxima aula

Linux no mini6410

- Instalação e teste usando um cartão de memória flash SD de:
  - Uma imagem Linux pronta, fornecida pelo fabricante
  - Uma imagem de Linux embarcado ajustada pelo aluno para a placa mini6410/mini2440
  - O cartão SD será empregado no lugar no pendrive USB, e deverá usar um sistema de arquivos para dispositivos Flash.
  - Será empregado o U-Boot. A placa não contém BIOS.

Duvidas? Obrigado!

Bibliografia

- Linux From Scratch.
- www.free-eletrons.com
- Marcelo Barros. http://www.linuxabordo.com.br/wiki/index.php?title=Main_Page
- Sergio Prado. http://www.sergioprado.org/2011/01/30/como-se-tornar-um-desenvolvedor-de-linux-embarcado/