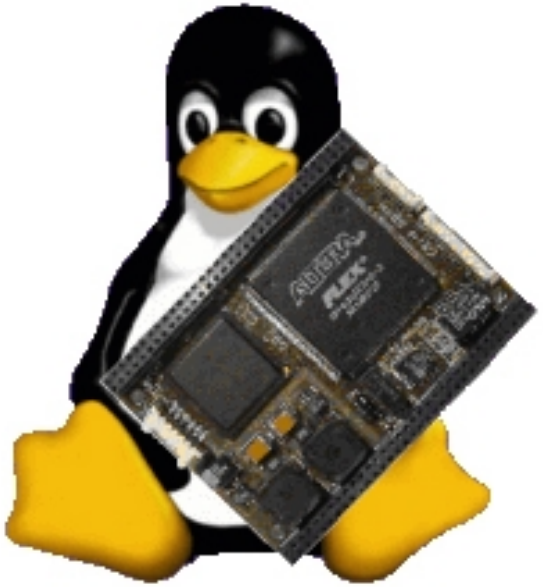
Criação de uma instalação

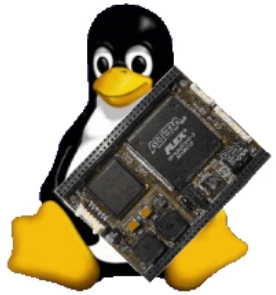


Linux Embedded

Sérgio Julião N. Mec.: 29976
Sérgio Soldado N. Mec.: 31397

Podem ser classificados através de:

- **Tamanho**:
  - **Físico**: limita à partida as capacidades do hardware do sistema.
  - **Dos Componentes**:
    - CPU
    - RAM
    - Armazenamento (disco rígido, flash, etc.)
- **Interacção com o utilizador**:
  - Sistemas com base na interacção: Telemóveis, PDAs, etc.
  - Sistemas sem interacção: Processos de controlo industriais, autopiloto, etc.
- **Networkability**:
  - Ultimos standards de comunicações (Bluetooth, Ethernet, 802.11g, 802.11n).
  - Nokia N770, N800, N810 Internet tablet's são exemplos de dispositivos embedded com sistema linux.
- **Requisitos Temporais**:
  - **Real-time systems**: sistemas que reagem de forma determinística, geralmente ligados a operações sensíveis (por exemplo os travões de um veículo, e até mesmo um leitor mp3 ou um telemóvel).



Porquê linux?

- **Grande base de código qualificado:** é *open source* (Tendo no entanto standards de qualidade, portanto o código é no geral fiável).

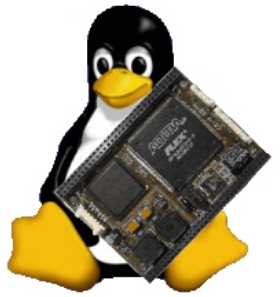
- **É vastamente suportado:** move uma comunidade *open source* a nível global que contribui a níveis de suporte e desenvolvimento.

- **Suporte de hardware:**
  - **Driver's Comunitários:** Grande disponibilidade.
  - **Arquitecturas diversas:** ARM, AVR32, Intel x86, M32R, MIPS, Motorola 68000, PowerPC, etc.

- **Custo:** Outras soluções comerciais podem ter custos iniciais de desenvolvimento, custos de ferramentas adicionais, e custos relacionados com *runtime royalties* (por exemplo um custo por unidade de produto).

- **Grande número de ferramentas disponíveis:**
  - Freshmeat.net
  - SourceForge
  - Google, etc.

- **Licenciamento:** Liberdade de modificar e redistribuir (GPL, LGPL, BSD,MPL, etc.).

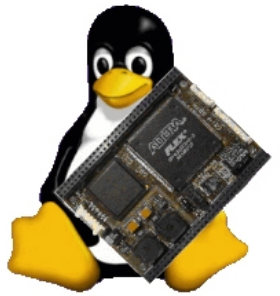


Hardware para correr Linux:

- 32bit CPU com MMU.
- Ram suficiente (8mb no mínimo).
- Input / Output (debugging e troubleshooting).
- Armazenamento (para guardar o kernel): disco rígido, flash...

Exemplo: Relógio de pulso da IBM com base em linux (2001).

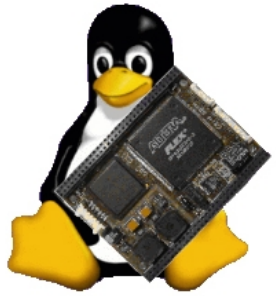
- Kernel: 2.2.1
- ARM processor equivalent to a 100 Mhz Pentium.
- Touch sensitive display
- 8MB Flash
- 8MB DRAM
- IrDA
- Radio Frequency Wireless connectivity
- Rechargeable Lithium Polymer battery



Porque não usar uma distribuição?

- **Usar uma distribuição VS. desenvolver o sistema por conta própria**
  - Depende do grau de **controlo** que se requer do sistema.
  - Desenvolver um sistema de raíz requer muito **tempo** (escolher pacotes e suas versões assegurando e mantendo compatibilidade entre estes).

- **Quem desenvolve linux embedded?**
  - **Comunidade *open source*.**
  - **Indústria (empresas comerciais):** por exemplo o MontaVista MobiLinux, na qual a motorola baseia por inteiro uma vasta gama dos seus produtos.



Como criar um sistema Embedded Linux?

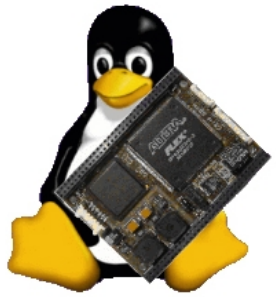
Passos principais:

- **Determinar os componentes do sistema:** focar nos objectivos do sistema para evitar uma implementação irrealista.
- **Configurar e compilar o kernel.**
- **Construir o root file-system.**
- **Setup do boot software e configuração deste.**

KERNEL: O que é isso? O que faz?

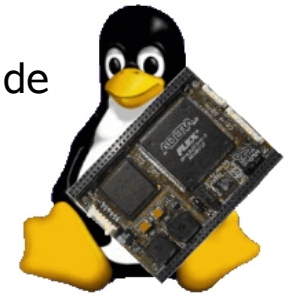
- **Gere hardware fornecendo uma abtracção de alto nível para uso do software de utilizador (APIs como o POSIX e outras).**
- **Process Scheduling.**
- **Gere acessos I/O.**
- **Gere partilha de memória.**
- **Outras tarefas administrativas.**
- **Gere mensagens de erro ;)**

É importante mencionar a existência de vários patches para o kernel. Linux começou como um GPOS, mas existe um patch para que este se converta num RTOS, o RTLinux patch. Desta forma as latências tornam-se previsíveis e pode-se usar o so no contexto de industrias como a aeronáutica.



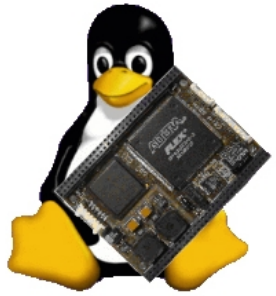
Como criar um sistema Embedded Linux?

- **Determinar os componentes do sistema:**
  - focar nos objectivos do sistema para evitar uma implementação irrealista.
- **Configurar e compilar o kernel:**
  - escolher o núcleo estável mais recente possível, caso contrário pode não ter suporte ou sentir necessidade de corrigir bug's já detectados e corrigidos pela versão mais recente.
  - maioria dos sistemas embutidos ainda com kernel 2.4 porque os engenheiros se sentem mais confortáveis com esta versão.
  - devido à restante dependência dos processos desta fase, é importante decidir e configurar adequadamente todos os componentes do kernel.
- **Construir o root file-system.**
  - contém apenas um conjunto mínimo de aplicações, bibliotecas e ficheiros considerados essenciais para o funcionamento do sistema.
- **Setup do boot software e configuração.**
  - o processo de boot é dependente da arquitectura.
  - na mesma arquitectura, o processo de boot difere ainda devido aos dispositivos de armazenamento



Como criar um sistema Embedded Linux?

- **Arranque do sistema**
  - **Bootloader:**
    - muito dependente do hardware do sistema embutido;
    - responsável pela inicialização de baixo nível do hardware;
    - carrega o código de arranque do kernel;
  - **Kernel**
    - código inicial de arranque varia consoante a arquitectura
    - prepara um ambiente que permite correr código C
    - invoca o start_kernel() - independente da arquitectura – que inicializa as funcionalidades de alto nível do kernel, monta os sistemas de ficheiros e inicia o processo init();
  - **Init()**
    - responsável pelo restante processo de arranque já no espaço do utilizador;



Como criar um sistema Embedded Linux?

- **Tipos de configuração de desenvolvimento host/target (host/target development setup)**

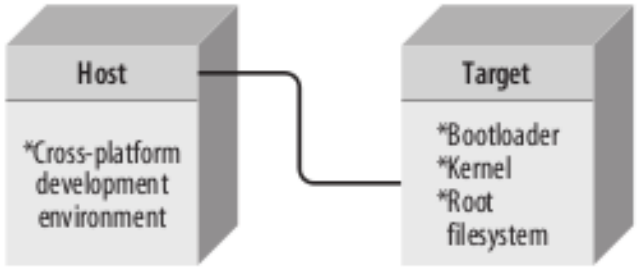


Figura 1 - linked setup

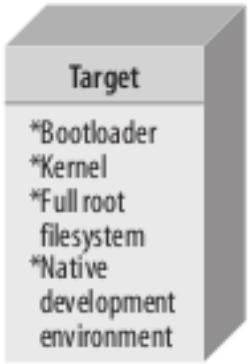


Figura 3 - standalone setup

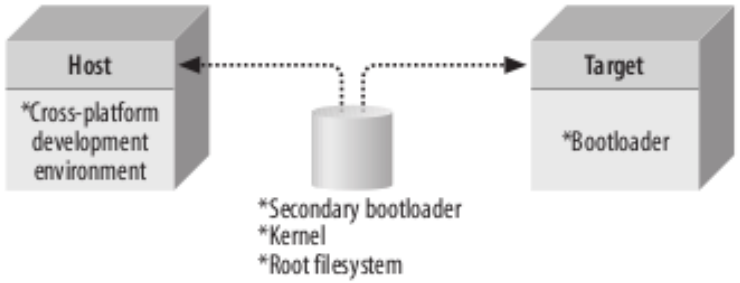
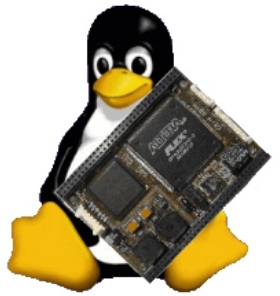
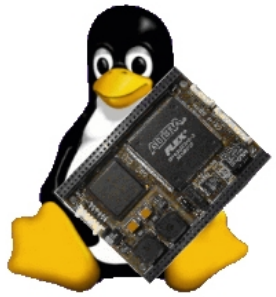


Figura 2 – Removable Storage Setup



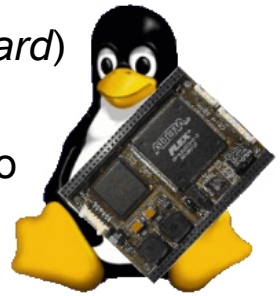
Como criar um sistema Embedded Linux?

- **Tipos de configuração de teste host/target (host/target debug setup)**
  - **3 tipos de interface usados para a ligação entre o host e o target para efeitos de teste**
    - **ligação série:**
      - limitação de velocidade;
      - se o sistema tiver uma única interface série, não permite interagir e testar simultaneamente o sistema;
    - **rede**
      - maior velocidade;
      - permite várias conexões sobre a mesma ligação física;
      - todavia é necessária uma networking stack!!!
    - **hardware específico de teste**
      - a ligação série e rede necessitam de software básico que reconheça as primitivas básicas I/O do hardware...
      - existe muito hardware específico de teste...mas a maioria é muito dispendioso e complexo
      - actualmente utilizam-se frequentemente as interfaces BDM e JTAG que recorrem às funcionalidades BDM e JTAG do CPU.



Como criar um sistema Embedded Linux?

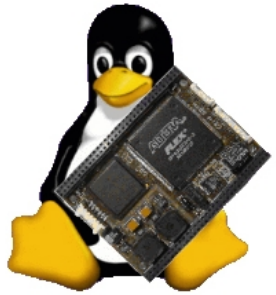
- **Ferramentas de desenvolvimento**
  - **cross-platform developtment tools / cross-development tools**
  - **toolchains**
    - podem ser obtidas através de:
      - download do código fonte e posterior compilação;
      - download do código binário;
- **Configuração do Kernel**
  - onde obter o kernel pode ser influenciado pela arquitectura do sistema embutido; não optar por um kernel oficial pode limitar o suporte!
  - adicionar ou remover as funcionalidades conforme as necessidades do sistema embutido
  - File System
    - configurar a estrutura de directórios pretendida, permissões, utilizadores, directorias temporárias para arranque, etc.
    - directorias essenciais: /bin, /dev, /etc, /lib, /proc, /sbin, /sys e /usr (*de facto standard*)
    - pode ser definido de uma forma totalmente diferente, o que implica configurar o kernel...mas não é aconselhável devido à inconsistência com a maioria do código *open source* ou pacotes *free software*



Como criar um sistema Embedded Linux?

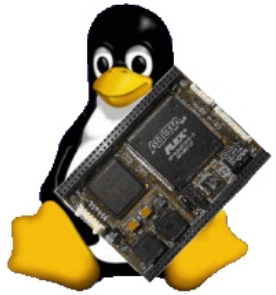
- **Configuração do Root FileSystem**
  - **Um FileSystem pode ser caracterizado segundo:**
    - suporte de escrita/actualização
    - persistência
    - confiança e segurança ao desligar o cabo de energia
    - compressão
  - **Tipos de FileSystem's**
    - linux suporta 50 filesystem's diferentes mas apenas um subconjunto é normalmente utilizado em sistemas embutidos:
      - **Ext2:** suporta escrita e persistência; não suporta confiança e segurança;
      - **Ext3:** adiciona a confiança às características do ext2;
      - **Cramfs e Squashfs:** simples, suporta compressão e é readonly;
      - **JFFS2 e YAFFS2:** suporta escrita, presistência, compressão e confiança/segurança;
      - **Tmpfs:** suporta escrita mas não persistência;

Como escolher? Depende das necessidades.

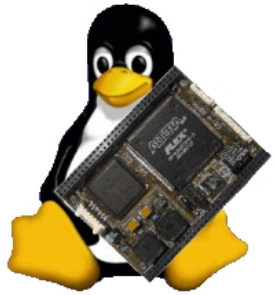


Como criar um sistema Embedded Linux?

- **Configuração do Boot loader**
  - **boot loader é responsável por ler o kernel do sistema operativo e a sua infraestrutura para a memória;**
  - **nos sistemas embutidos assume as funções do extensive system firmware (por ex. BIOS, UEFI, OpenFirmware, etc);**
  - **assim é responsável por:**
    - ler a imagem do kernel;
    - programar os controladores de memória do sistema;
    - inicializar as caches do processador;
    - activar vários dispositivos de hardware;
    - implementar directamente o suporte para a infraestrutura de arranque em rede;
    - entre muito mais...
  - **existem milhares e milhares de boot loader's para placas embutidas, muitos dos quais para linux (com imensas configurações possíveis para uma mesma board).**
  - **normalmente, é fornecido pelo fabricante.**



QUESTÕES?



Como criar um sistema Embedded Linux?

- **Bibliografia**
  - Building Embedded Linux System's, 2nd edition, Karim Yaghmour, Jon Masters, Gilad Ben-Yossef & Philippe Gerum
  - Build an embedded Linux distro from scratch, IBM (paper, 1994)
  - www.wikipedia.com