Sistemas Operacionais

Linux

2ª edição

Capítulo 9

Revisão: Fev/2003

Introdução: histórico, distribuições e versões

- Oriundo da decepção do estudante finlandês Linus Torvalds com o sistema operacional minix
  - Após conclusão de uma versão inicial disponibilizou código na Internet
- Derivado em grande parte da família UNIX BSD

Distribuições

- É composto pelo núcleo do sistema operacional Linux, um conjunto de software aplicativos e uma interface de instalação
- Existem várias distribuições
  - Diferenciam-se por alguns softwares aplicativos e na interface instalação
  - Questão de gosto pessoal
- Os softwares aplicativos possuem uma base comum grande
  - Desenvolvida pela GNU (GNU is Not Unix) da Free Software Foundation
- O que se paga em uma distribuição?
  - Não é o Linux, nem GNU, mas sim uma série de serviços disponibilizados
    - e.g: cdrom, manuais de instalação, *hot-line*, software proprietário,etc

Versões

- Existe núcleo de desenvolvimento, núcleo estável e *releases*
- Sistema de númeração de versões é baseado em três números:
  - Versão principal: existem três até hoje 0, 1 e 2
  - Indicativo se o núcleo é estável (par) ou desenvolvimento (impar)
  - *Release*

O ciclo de vida de processos

- Criação é feita sempre a partir de um processo pai
  - Chamada de sistema *fork-exec*
- Durante sua execução umprocesso Linux assume 5 estados:
  - task_running: equivale aos estados pronto e apto
  - task_interruptible: equivale ao estado bloqueado
  - task_uninterruptible: equivale ao estado bloqueado (situações críticas)
  - task_stopped: parado esperando por sinalização externa (*signal*)
  - task_zombie: equivale ao estado zumbi
- Término

O conceito de *threads*

- Suporte a *thread*s nível de sistema (modelo 1:1)
- Implementadas através da chamada de sistema `clone()`
- O núcleo não distingue *threads* de processos
  - Empregam a mesma estrutura de dados (PCB)
    - Threads tem ponteiros para o PCB do processo que as porta

Escalonamento

- Duas classes em função do tipo de processos (*threads*)
  - Processos interativos e *batch*
  - Processos de tempo real
- Políticas de escalonamento do linux (padrão POSIX)
  - SCHED_FIFO: FIFO com prioridade estática
    - Válido apenas para processos de tempo real
  - SCHED_RR: Round-robin com prioridade estática
    - Válido apenas para processos de tempo real
  - SCHED_OTHER: Filas multinível com prioridades dinâmicas (*time-sharing*)
    - Processos interativos e *batch*

Escalonamento linux tempo real

- Linux implementa dois tipos de prioridade
  - Estática: exclusivamente processos de tempo real
  - Dinâmica: processos interativos e batch
- Prioridade é definida pelo usuário e não é modificada pelo escalonador
  - Somente usuários com privilégios especiais
- Seleciona sempre processos de mais prioridade para executar
  - Executa segundo a política selecionada: SCHED_FIFO ou SCHED_RR
- Processo em tempo real tem preferência (prioridade) sobre processos interativos e batch

Escalonamento linux *timesharing*

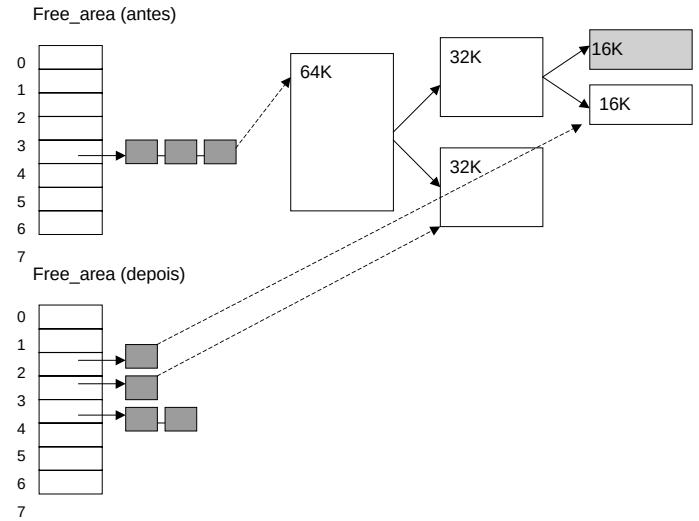
- Baseado no uso de créditos e prioridade
- Sistema de créditos:
  - Cada processo executa um certo número de créditos
  - O processo com maior crédito é o selecionado
  - Cada interrupção de tempo o processo em execução perde um crédito
  - Processo que atinge zero créditos é suspenso (escalonado médio prazo)
  - Se no estado apto não existir processos com créditos é realizado uma redistribuição de créditos para todos os processos (qualquer estado)

$$Créditos = \frac{Créditos}{2} + prioridade$$

Gerência de memória

- Implementa memória virtual através de paginação por demanda
- Paginação a três níveis
  - Portabilidade a diferentes plataformas
  - Adapta, via *software*, a dois níveis em função do processador
- Algoritmo de substituição de páginas é o algoritmo do relógio com segunda chance
- Algoritmo de alocação é conhecido como *Buddy*

Algoritmo *Buddy*



Sistema de arquivos

- Organizado de forma hierarquizada via grafo acíclico
  - Suporte a hard links e a software links
- Oferece suporte a diferentes sistemas de arquivos
  - e.g: ext2, ext3, xia, minix, iso9669, fat, smb, ufs, entre outros
- Baseado no conceito de partições e montagem
- Implementa camada VFS (Virtual File System)
- Sistema nativo do Linux é o ext2
  - Versões mais recentes 2.4.x disponibilizam o ext3

O sistema de arquivos ext2fs

- Estruturas básicas são os blocos e os i-nodes
- Utiliza como método de alocação o indexado (combinado)
  - Com VFS suporta arquivos de até 4 Tbytes
- "Cartografia" de blocos e i-nodes livres/ocupados é mantido através de *bitmaps*
- Emprega cache para melhorar o desempenho

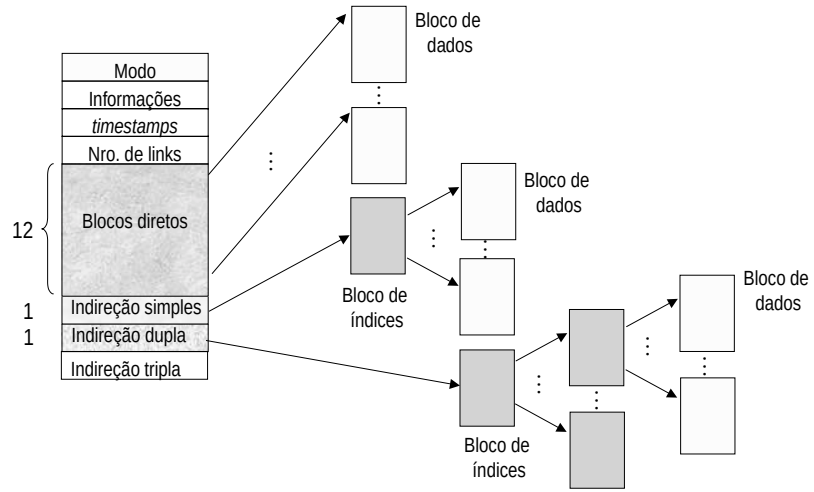
Bloco

- Um arquivo é composto por uma quantidade inteira de blocos
  - Fragmentação interna no último bloco
- Bloco tem tamanho configurável (por partição)
  - Valores típicos são 1024 bytes, 2048 bytes, 4096 bytes (*default*)
- Bloco armazena:
  - Dados
  - Certas informações de controle do sistema operacional
    - e.g.: blocos de índices, blocos com bitmaps, etc...

i-node

- Cada arquivo é representado por uma estrutura (i-node)
- Cada i-node possui:
  - Tipo do arquivo (modo)
  - Direito de acesso
  - Proprietário
  - Tamanho do arquivo
  - *timestamps*
  - Contador de links
  - Ponteiros para bloco de dados
- i-node possui 128 bytes

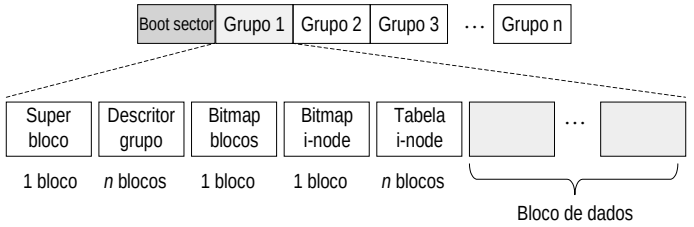
Estrutura interna de *i-node*



Estrutura física do disco em ext2fs

- Fortemente influenciada pelo sistema de arquivos UNIX BSD
- Sistema de arquivos é organizado como um grupo de blocos
  - Cilindros na terminologia do UNIX BSD
- Objetivo é favorecer uma localização espacial dos blocos e das estruturas de gerência do sistema de arquivos
  - Reduzir tempo de *seek*
  - Tentativa de alocar blocos para dados é sempre a partir do grupo de blocos mais próximo da posição atual do cabeçote
    - Se não tem disponível, tenta nos vizinhos mais próximos

Layout do sistema de arquivos ext2fs



- Super bloco: descritor do tamanho e formato do sistema de arquivos
- Descritor grupo: organização do grupo (tamanho e formato)
- Bitmap blocos: indicação se um bloco do grupo está livre/ocupado
- Bitmap i-nodes: indicação se um i-node do grupo está livre/ocupado
- Tabela i-nodes: associação de blocos de dados aos i-nodes do grupo

Tipos de arquivos

- Regular
- Diretório
- Link simbólico
- Caracter/bloco
- Fifo (named pipe)
- Socket

Regular

- Tipo mais comum de arquivo
- Seqüência de bytes cuja interpretação é dada pela aplicação
- Utiliza blocos para armazenamento de informação
  - Exceção obvia: blocos só são alocados quando o arquivo possui dados

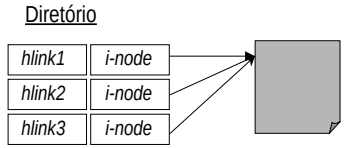
Diretório

- Estruturado em uma árvore hierárquica
- Cada diretório possui arquivos e subdiretórios
- Diretório é um tipo especial de arquivo, onde cada entrada possui:
  - i-node: descreve localização dos blocos que compõem o arquivo
  - tamanho da entrada: tamanho em bytes dessa entrada no diretório
  - tamanho do nome: tamanho em bytes do nome do arquivo
  - tipo: indicação do tipo do arquivo
  - nome do arquivo: *string* de máximo 256 caracteres

i-node	tamanho entrada	tamanho nome	tipo	nome do arquivo

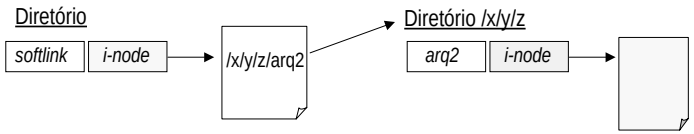
Hard link

- A cada link criado o contador de links é incrementado
  - Remover um link implica em decrementar o contador de links e liberar o i-node e blocos associados SE o contador for zero
- Só existe no interior de uma mesma partição porque a referencia é número do *i-node* na partição
- Não é permitido (para evitar ciclos) links para diretórios
- Tecnicamente todo arquivo é um *hard link*
  - Não existe tipo de arquivo *hard link*



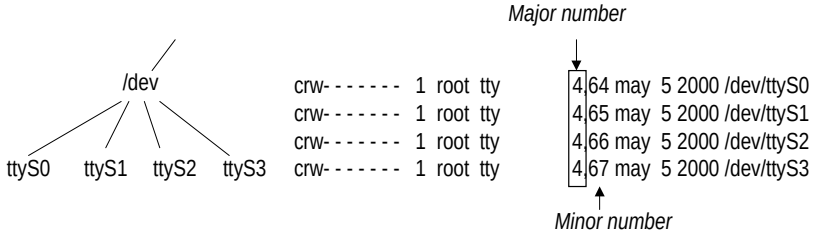
Soft link

- Link simbólico
- Arquivo que contém o nome (*path*) de um arquivo
  - Na realidade existe ainda os *fast symbolic links*
    - O *path* do arquivo é armazenado diretamente no i-node
      - limita o tamanho do *path* a 64 caracteres
- Pode ser utilizado entre partições



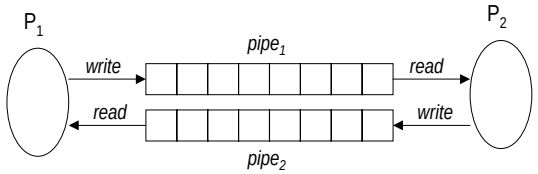
Arquivos de dispositivos

- Periféricos são acessados através de arquivos especiais
  - Modo bloco e modo caracter
- Não ocupam blocos de dados, apenas i-node
- São identificados por um:
  - *major number*: tipo do dispositivo
  - *minor number*:unidade do dispositivo



Fifo (*named pipes*)

- Mecanismo de IPC (*Inter Process Communication*)
- Não ocupa bloco de dados, apenas i-node
- Cria um “canal” de comunicação unidirecional entre dois processos



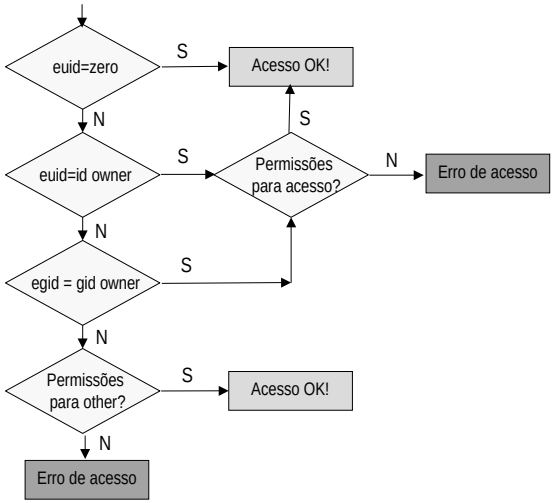
Sockets

- Mecanismo de IPC (*Inter Process Communication*)
- Tipo especial de arquivo destinado a realização de serviços de rede
  - Maneira pela qual TCP e UDP identificam os processos de uma comunicação
- Não ocupa bloco de dados, apenas i-node
- Um *socket* é um descritor que possui:
  - Protocolo : TCP ou UDP
  - Endereço da máquina : endereço IP
  - Porta : identificador do processo
- Comunicação é feita através de chamadas de sistemas *read* e *write*

Proteção

- Cada processo possui 4 identificadores associados
  - *Real user* e *real group identifier* (*uid* e *gid*)
  - *Effective user* e *effective group identifier* (*euid* e *egid*)
- Cada arquivo possui os seguintes atributos
  - *User id* e *group id*
  - Bits de proteção rwx
  - *Set user id* e *set group id* (bit s)
    - Permite que durante a execução seja trocado o *userid e o gid de* um processo *(effective user/group)*
- Proteção de acesso é realizado em função desses identificadores

Verificação de acesso (proteção)



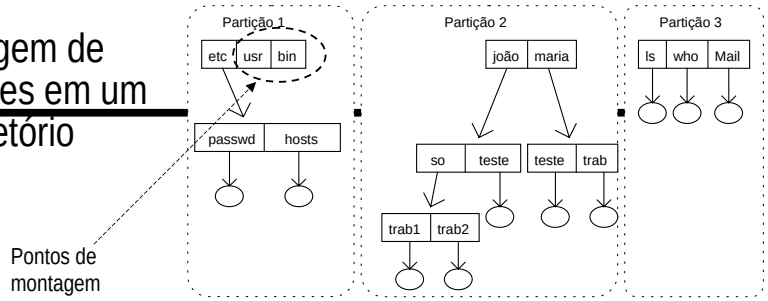
Interpretação dos bits *rwx* em diretórios

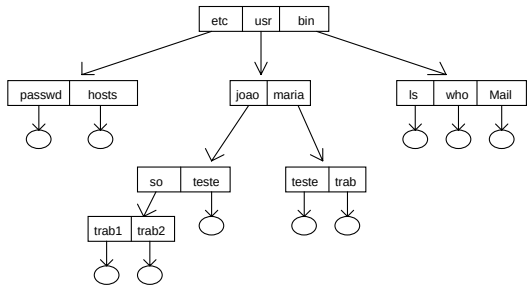
- Bit *r*
  - Permite acesso ao contéudo do diretório (i.e., comando ***ls***)
- Bit *w*
  - Alteração do conteúdo do diretório (i.e., inclusão/remoção de arquivos)
- Bit *x*
  - Possibilitar selecionar como diretório corrente de trabalho (i.e.., comando ***cd***)
- Bit s
  - Arquivos criados dentro do diretório herdam o grupo do diretório e não do usuário que criou o arquivo

O conceito de montagem

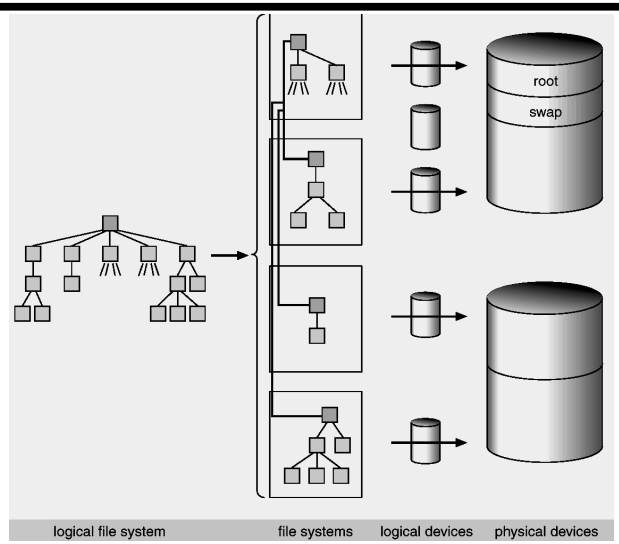
- Sistema de arquivos é individualizado por meio de armazenamento
  - e.g.; discos rígidos, disquetes, cd-rom, fitas, etc
- Montar um sistema de arquivo significa integrá-lo a uma hierarquia ja existente de um outro sistema de arquivos
  - Ponto de montagem
- Sistema de arquivos pode estar em outra máquina (montagem remota)
  - e.g.: Network File System (NFS)
    - Conceito de exportação e de importação

Montagem de partições em um subdiretório





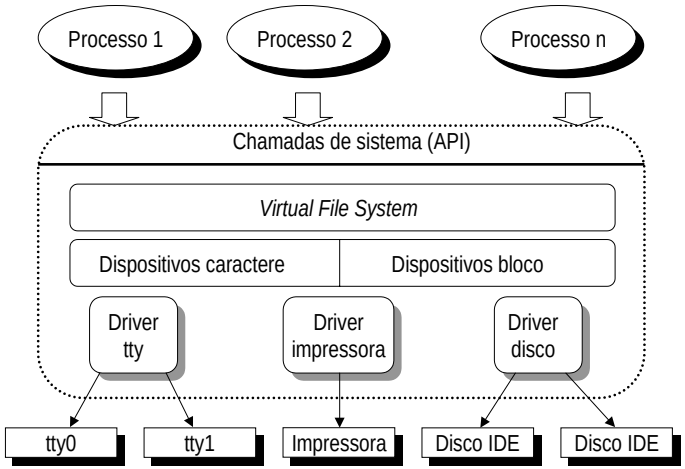
Exemplo de montagem



Gerência de entrada e saída

- Baseado em uma organização em camadas:
  - E/S a nível de usuário
  - Subsistema de entrada e saída
  - *Drivers* de dispositivos
- Por questões de desempenho os drivers podem ser compilados junto ao kernel ou carregados dinamicamente (módulos)

Organização em camadas da gerência de E/S



Leituras complementares

- R. Oliveira, A. Carissimi, S. Toscani; Sistemas Operacionais. Editora Sagra-Luzzato, 2001.
  - Capítulo 9
- A partir do site *Linux Documentation Project* (LDP) se tem acesso a várias referências
  - http://www.ibiblio.org/mdv/index.html