Sistemas Operacionais

Sistemas de Arquivos

Interface de Sistemas de Arquivos

Sistema de Arquivos

- Arquivos
  - Atributos
  - Operações
  - Tipos
  - Estrutura
  - Métodos de acesso
  - Arquivos Mapeados em Memória
- Diretórios
  - Estrutura Lógica
  - Operações
- Proteção

Interface de Sistemas de Arquivos

- Para a maioria dos usuários, o Sistema de Arquivos é o aspecto mais visível do SO
- É a parte do SO responsável pelo armazenamento e acesso de dados e programas do SO e dos usuários
- Consiste de:
  - conjunto de arquivos
  - estrutura de diretórios
  - partição

Arquivos

- Computadores armazenam informação em diferentes tipos de dispositivos físicos:
  - fitas magnéticas
  - discos magnéticos
  - discos óticos, ...
- SO abstrai as propriedades físicas do dispositivo definindo uma unidade de armazenamento lógica:

O arquivo

- O SO associa cada arquivo a um dispositivo físico

Arquivos

- Arquivo é um conjunto nomeado de informações que são gravadas em memória secundária
  - Programas: código-fonte ou objeto
  - Dados: numéricos , alfabéticos, alfanuméricos ou binários
- Formato:
  - Livre: arquivos-texto
  - Rigidamente Formatado
- Geralmente é uma seqüência de bits, bytes, linhas ou registros, cujo significado é definido pelo criador do arquivo ou pelo usuário.

Atributos

- Os arquivos são nomeados para consumo humano ➔ é referenciado pelo nome
- Nomes são normalmente seqüências de caracteres (alguns SOs diferenciam letras maiúsculas e minúsculas, outros não)
- Ao receber um nome, um arquivo se torna independente do processo, do usuário e mesmo do sistema que o criou.
- Além do nome, arquivos possuem outros atributos, isto é, informações de controle associadas a cada arquivo (variam de SO para SO)

Atributos

- Tipicamente:
  - **Nome**: informação para consumo humano
  - **Tipo**: necessário quando o SO diferencia tipos
  - **Localização**: ponteiro para dispositivo físico e localização dentro dele
  - **Tamanho**: quantidade de bytes, palavras ou blocos do arquivo (e, possivelmente, o tamanho máximo permitido)
  - **Proteção**: informação de controle de acesso ao arquivo (leitura, escrita, execução)
  - **Dono**: dono do arquivo
  - **Data, hora e usuário**: usado para (1) criação, (2) última modificação e (3) último uso → informação útil para proteção, segurança e monitoramento de uso
  - **Senha**: utilizada no acesso

Operações

- O SO fornece uma série de *system calls* (chamadas de sistema) para manipulação de arquivos
- Operações básicas:
  - Criar arquivo:
    (1) encontrar espaço em disco
    (2) criar uma entrada para o novo arquivo no diretório (atributos)
  - Escrever:
    (1) a partir do nome, localizar o arquivo
    (2) escrever o conteúdo desejado na posição atual de gravação, indicada por um ponteiro
    (3) atualizar o ponteiro

Operações

- Ler:
  - (1) a partir do nome, localizar o arquivo
  - (2) ler a quantidade desejada a partir da posição atual de leitura, indicada por um ponteiro (normalmente o mesmo da gravação)
  - (3) atualizar o ponteiro
- Reposicionar o ponteiro de um arquivo (seek):
  - (1) localizar o arquivo
  - (2) posicionar o ponteiro
- Remover (excluir) arquivo:
  - (1) localizar o arquivo
  - (2) liberar seu espaço alocado
  - (3) eliminar sua entrada no diretório

Operações

- Truncamento de arquivo:
  - (1) localizar o arquivo
  - (2) liberar seu espaço alocado

- Outras operações: adição de dados (append), renomeação, cópia, etc
- A maioria das operações envolve pesquisar no diretório para localizar a entrada associada ao arquivo ➔ para evitar esta constante pesquisa, vários sistemas abrem (open) o arquivo antes de usá-lo e fecham-no (close) ao final. Esta operação apenas inclui o arquivo na tabela de arquivos abertos

Operações

- Em ambientes multiusuários, normalmente o SO utiliza dois tipos de tabelas para implementar as operações **open** e **close**:
  - Uma para cada processo, contendo todos os arquivos abertos por ele (apontador de arquivo)
  - Outra global, contendo dados dos arquivos abertos do sistema, independentes de processos (contador de uso do arquivo, localização do arquivo no disco, etc)

Operações

- Unix *system calls* :
  - `fd`: descritor de arquivo = referência para tabela de arquivos
  - `fd = creat(nome, modo)`
  - `fd = open(arq, como)`
  - `s = close(fd)`
  - `n = read(fd, buffer, nbytes)`
  - `n = write(fd, buffer, nbytes)`
  - `pos = lseek(fd, offset, inicio/atual/fim)`
  - `s = stat(nome, &buf)`
  - `s = chmod(nome, modo)`

Tipos

- O SO deve reconhecer o tipo de arquivo?
  - ➔ impressão de arquivo executável
- Técnica para implementar tipo de arquivo:
  - incluir tipo como parte do nome ➔ nome + extensão

indica tipo

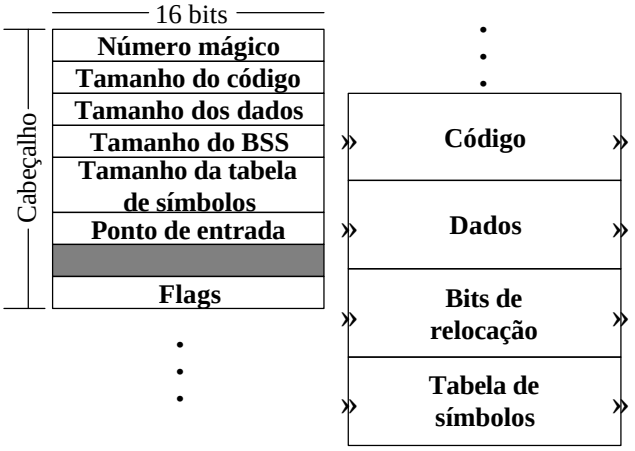
- Macintosh (Apple): um atributo contém o nome do programa que o criou
- Unix: permite o uso de extensões, mas não são usadas pelo SO (possui um número mágico para apenas alguns tipos de arquivos)

Tipos

Tipo do arquivo	Extensão usual	Função
Executável	Exe, com, bin ou nenhuma	Programa executável
Objeto	Obj, o	Prog. compilado, mas não ligado
Arquivos de comandos	C, p, pas, f77, asm	Código-fonte em diversas linguagens
Texto	Txt	Dados textuais/documentos
Processador de texto	Doc, tex	Formatos usados por processadores de texto
Biblioteca	Lib, a	Bibliotecas de rotinas
Impressão, visualização	Ps, dvi, gif	Arquivos para impressão ou exibição na tela
Arquivamento	Arc, zip, tar	Arquivos agrupados para arquivamento

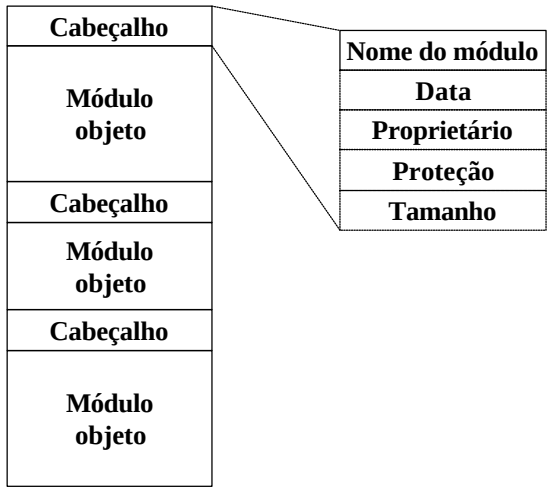
Tipos

Um arquivo binário executável no Unix



Tipos

Um arquivo binário não executável (biblioteca de módulos) no Unix



Estrutura

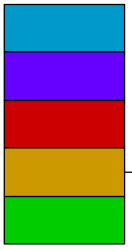
- O modo como os dados de um arquivo estão internamente armazenados pode variar em função do tipo de informação contida no arquivo
- A forma mais simples de organização é através de uma seqüência não estruturada de bytes

- Vantagem: máxima flexibilidade
- Desvantagem: mínimo suporte, cada aplicação deve incluir o código para interpretar o arquivo

⇒ Unix e DOS

Estrutura

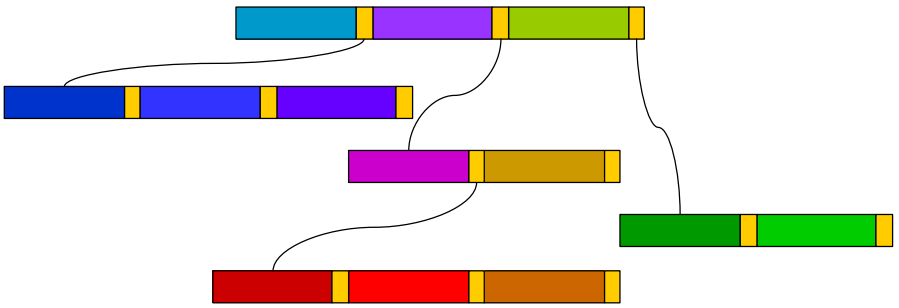
- Seqüência de registros:



- Quando se usava cartões perfurados de 80 colunas, muitos sistemas de arquivo implementavam arquivos com registros de 80 caracteres

Estrutura

- Árvore:
- Os registros são organizados em uma árvore de registros (tamanho fixo ou variável)



Estrutura

- Alguns SOs adotam diferentes formas de organização

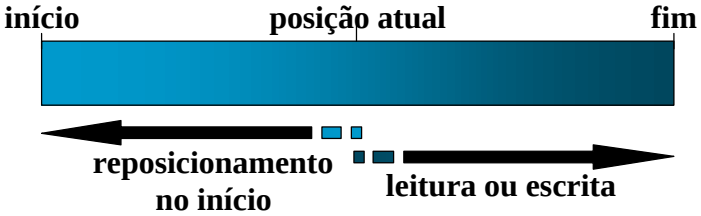
- Vantagem:
  - o SO fornece instruções especiais para manipulação de arquivos de acordo com sua estrutura
- Desvantagem:
  - aumento da complexidade (código para cada tipo de arquivo)
  - introdução de novos tipos causa problema
    - ➔ exemplo: se o SO suporta ASCII (com <CR> e <LF>) e binários executáveis, como introduzir arquivo criptografado (não é ASCII e nem executável !)

Método de Acesso

- Quando um arquivo é usado, sua informação deve ser acessada e lida para memória
- Os sistemas podem dar suporte a um tipo de acesso ou a vários

Acesso Seqüencial

- Método mais simples
- A informação é processada em ordem, um registro após o outro.
- É o mais comum (editores e compiladores fazem acesso desta forma)



- Sistema baseado em fitas magnéticas

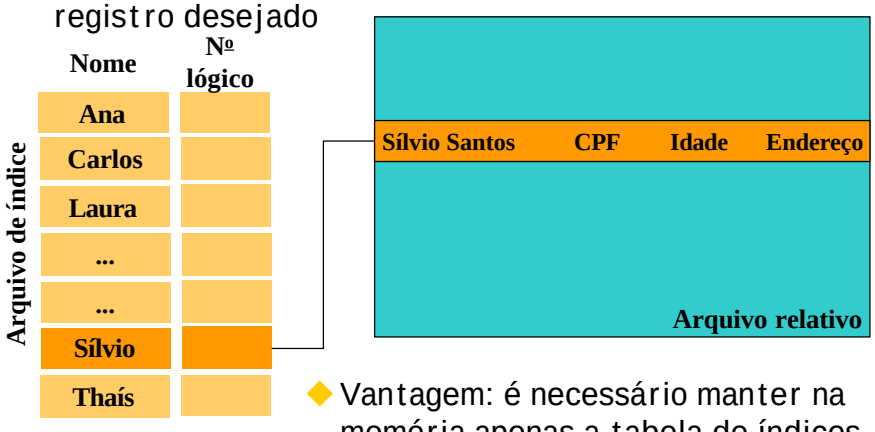
Acesso Direto (relativo)

- Arquivo composto por registros de tamanho fixo, permitindo ao programador ler e escrever registros aleatoriamente

- Sistema baseado em discos magnéticos

Acesso Indexado

- Usa um índice que identifica o ponteiro para o registro desejado



- Vantagem: é necessário manter na memória apenas a tabela de índices

Arquivos Mapeados em Memória

- Para aumentar a eficiência, alguns SOs mapeiam arquivos na memória, no espaço de endereçamento do processo
- Assim, leituras/escritas provocam uma falta de página, carregando a respectiva página do arquivo e, então, realizando-se a operação de leitura/escrita na memória
- Quando o processo termina sua execução, o arquivo é levado para o disco
- O mapeamento de arquivos funciona melhor em sistemas que suportam segmentação (paridade de endereços do arquivo com o segmento)

Arquivos Mapeados em Memória

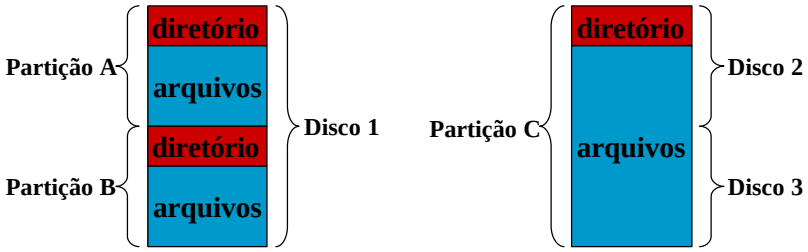
- Pequenos problemas:
  - dificuldade de se saber o tamanho exato do arquivo que foi criado/escrito (saber quantos bytes de uma página foram escritos)
  - manter consistência da informação no caso do arquivo ser mapeado por um processo e aberto por outro (alterações só são atualizados no disco quando a página modificada é removida da memória)
  - arquivos podem ser menores do que o segmento alocado ou mesmo maiores do que todo o espaço de endereçamento virtual, implicando na necessidade de se mapear apenas pedaços e não ele inteiro, o que é menos conveniente

Diretórios

- O sistema de arquivo pode ser grande ➔ para se gerenciar os dados, torna-se necessários organizá-los
- A organização é feita em duas partes:
  - O sistema é dividido em partições (volumes):
    - cada disco contém pelo menos uma partição
    - usadas para fornecer áreas separadas em disco → cada uma é tratada como um dispositivo de armazenamento diferente
    - alguns sistemas permitem que uma partição agrupe vários discos em uma única estrutura lógica

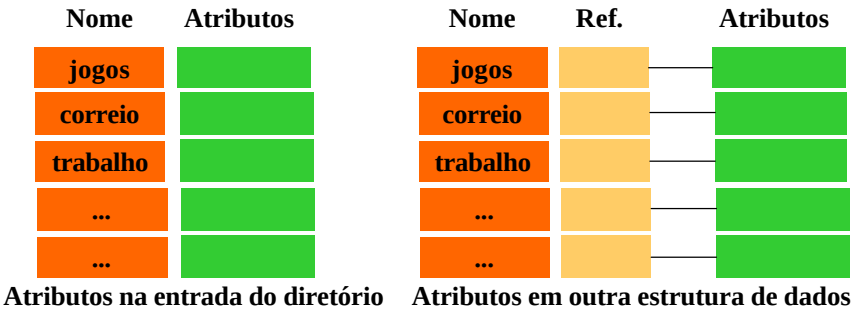
Diretórios

- Cada partição contém informação sobre os arquivos que ela armazena:
  - mantida em diretório de dispositivo ou tabela de conteúdo do volume (mais conhecido como *diretório*)
  - o diretório registra informações como nome, localização, tamanho, tipo, etc...



Sistemas de Diretório Hierárquico

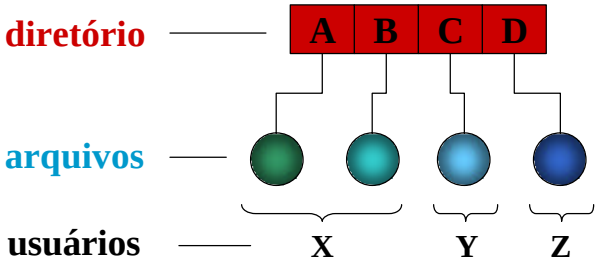
- Um diretório contém tipicamente um conjunto de entradas, uma por arquivo.



- Quando um arquivo é aberto, todas as informações sobre o arquivos são transferidas para a memória principal

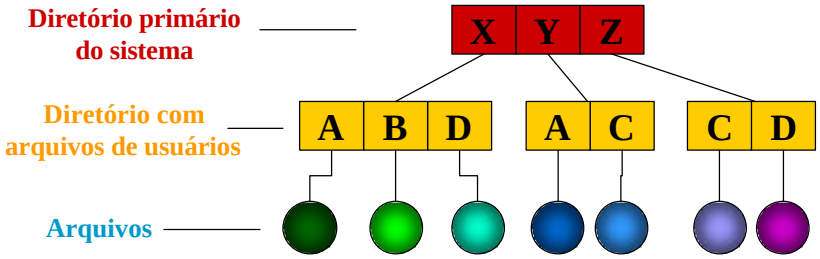
Estrutura Lógica: Diretório de Nível Único

- Estrutura mais simples
- Um único diretório contém todos os arquivos
- Problema: um arquivo deve ter nome único
  - e se o diretório for compartilhado com vários usuários?



Estrutura Lógica: Diretório de Dois Níveis

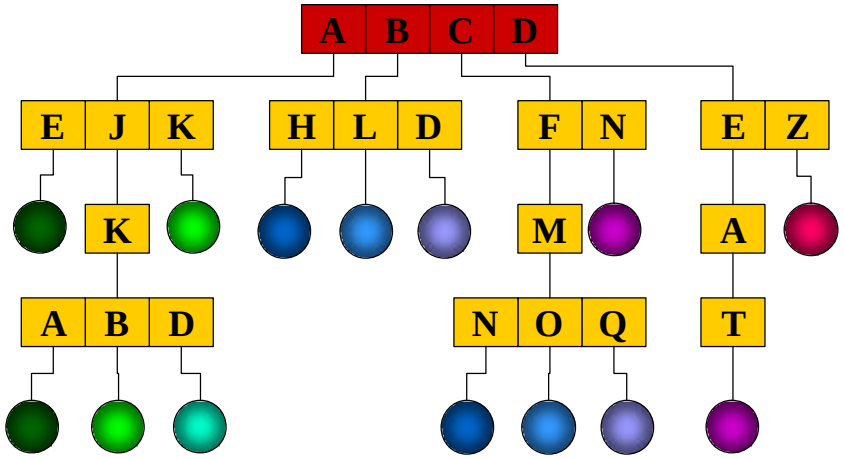
- Um diretório por usuário ➔ Usuários diferentes podem criar arquivos com o mesmo nome
- É necessário criar diretórios de usuários
- Conceito de "caminho", caso um usuário necessite usar arquivos de outro usuário



Estrutura Lógica: Árvore por Usuário

- É o mais comum
- O usuário organiza seus arquivos (pode criar subdiretórios)
- Um diretório é um arquivo tratado de forma especial
- Conceito de diretório atual "." (diretório corrente) e de diretório pai ".."
- Caminho pode ser absoluto ou relativo (relativo ao diretório atual)

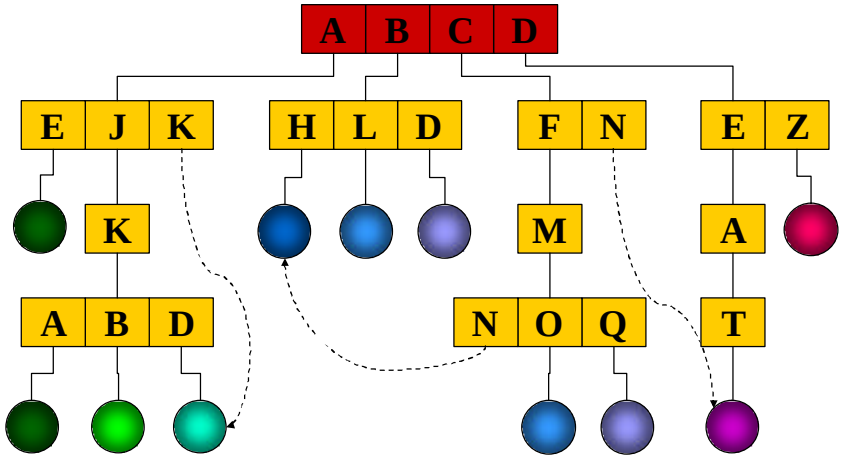
Estrutura Lógica: Árvore por Usuário



Estrutura Lógica: Grafo Acíclico

- Como tratar o caso de usuários que compartilham arquivo(s) ?
  - Árvore ➔ 2 cópias do arquivo
  - Grafo acíclico: diretórios diferentes compartilham o mesmo arquivo físico (ou o mesmo subdiretório)
- Unix: *link*
  - Absoluto: entradas idênticas nos 2 diretórios
  - Simbólico (relativo): quando é feita uma referência ao arquivo, busca-se no diretório original
- Problema do *link* absoluto: qual é o original e qual é a cópia?

Estrutura Lógica: Grafo Acíclico



Estrutura Lógica: Grafo Acíclico

- Estrutura mais flexível, porém mais complexa
- Remoção de diretórios e arquivos
  - *link* simbólico: quando se detecta o *link* para um arquivo, ele (o link) é apagado. Se o arquivo for apagado o *link* fica inconsistente (a próxima tentativa de referência gera erro)
  - *link* absoluto: mantém o arquivo até que todas as referências sejam apagadas (contador de referências)

Operações

- Operações que devem poder ser realizadas sobre um diretório:
  - Criar e excluir um arquivo
  - Alterar o nome de um arquivo
  - Pesquisar por um arquivo
  - Criar e remover um diretório
  - Listar o conteúdo de um diretório
  - Alterar o nome de um diretório
  - Percorrer o sistema de arquivos
  - Criar e remover um link

Proteção

- Sistemas de arquivos precisam de mecanismos para que os protejam contra danos físicos (confiabilidade) e contra acesso indevido (segurança)
- Confiabilidade é geralmente obtida por meio de cópias *BACKUP*
- A proteção contra acessos indevidos pode ser obtida através de acesso controlado (permissão de acesso). Existem várias abordagens:
  - Tipo de acesso: leitura, escrita, execução, concatenação, etc
  - Lista de grupo e lista de acesso: proprietário/grupo/universo
  - Proteção de diretórios: controlar operações sobre diretórios
  - Senha de acesso a arquivos: associação de senhas a arquivos