Sistemas Operacionais

Prof. Robson de Souza

Aulas 33 e 34

Conteúdo: Sistemas de arquivos

Todo aplicativo de computador precisa armazenar e recuperar informações. Um processo pode armazenar uma quantidade limitada de informações em sua execução, de acordo com o espaço de memória virtual. Um problema que acontece é que nem sempre este espaço é suficiente dependendo da aplicação.

Outro problema é que quando os dados são mantido no espaço de endereçamento de um processo, uma vez que ele termina a sua execução, todas as informações salvas por ele nesse espaço são perdidas. Para muitos aplicativos, as informações precisam ser mantidas por um período longo de tempo.

Um terceiro problema é o fato de que em alguns casos, é necessário que múltiplos processos acessem os mesmos dados ao mesmo tempo. Se todas as informações são salvas baseadas no endereçamento do processo, apenas aquele processo terá acesso às informações.

A solução geral para esse problema é armazenar os dados em mídias externas e dispositivos de armazenamento em unidades chamadas **arquivos**.

As informações salvas em arquivos devem ser **persistentes**, ou seja, não devem ser afetadas pela criação e pela finalização de processos. Um arquivo deve desaparecer apenas quando seu proprietário explicitamente der uma ordem para o S.O removê-lo. Os arquivos são gerenciados pelo Sistema Operacional e a parte do S.O que lida com arquivos é chamada de **sistema de arquivos**.

Do ponto de vista do usuário, o aspecto mais importante de um sistema de arquivos é como aparece para ele, isto é, o que constitui um arquivo, como os arquivos são nomeados e protegidos, que operações são permitidas em arquivos e assim por diante.

Nomes de arquivo

Os arquivos são um mecanismo de abstração. Oferecem uma maneira de armazenar as informações no disco e de lê-las de volta mais tarde. Isso deve ser feito de tal maneira que esconda do usuário os detalhes de como e onde as informações são armazenadas e de como os discos realmente trabalham.

Provavelmente, a mais importante característica de qualquer mecanismo de abstração é a maneira como são nomeados os objetos que estão sendo gerenciados. Quando um processo cria um arquivo, ele lhe dá um nome. Quando o processo termina, o arquivo continua a existir e a poder ser acessado por outros processos, utilizando seu nome.

As regras para nomeação de arquivos variam de sistema para sistema, mas normalmente são permitidos vários caracteres e muitos sistemas de arquivos suportam nomes com até 255 caracteres. Alguns sistemas de arquivos distinguem nomes escritos com letras maiúsculas e nomes escritos com letras minúsculas, enquanto outros não o fazem.

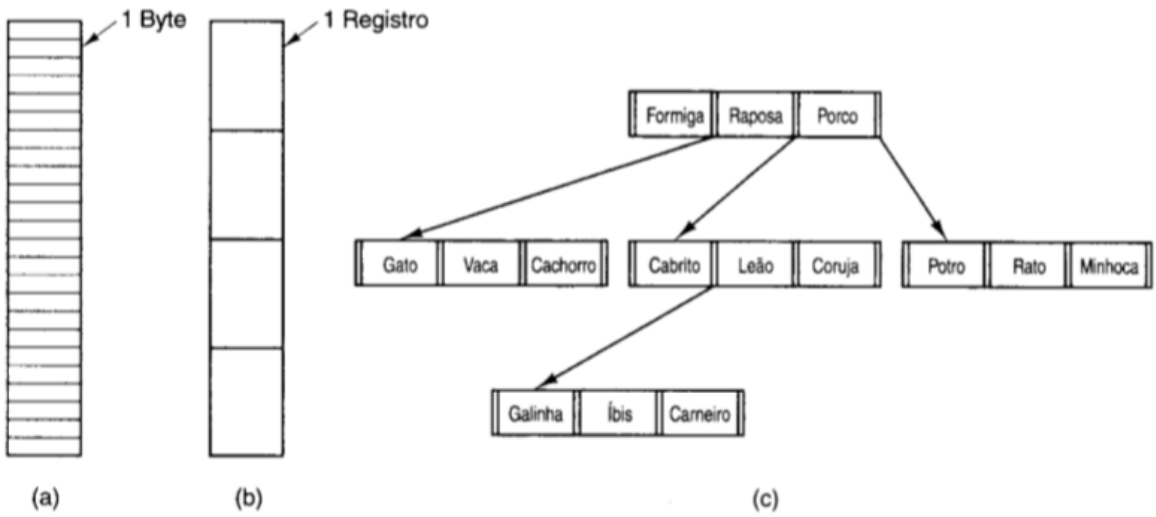
Muitos sistemas operacionais suportam nomes de arquivos escritos em duas partes, ambas separadas por um ponto. A parte que segue ao ponto é chamada de extensão do arquivo. Por exemplo, no MS-DOS os nomes de arquivos tinham de 1 a 8 caracteres seguidos de uma extensão opcional de 1 a 3 caracteres.

Em alguns casos, as extensões de arquivos são apenas convenções e não são necessariamente impostas. Por exemplo, um arquivo.txt é provavelmente algum arquivo de texto, mas essa extensão serve mais para lembrar o proprietário do que para carregar quaisquer informações específicas para o computador. Por outro lado, um compilador C pode realmente exigir que os arquivos sejam terminados em .c e recusar a compilá-los se essa exigência não for seguida.

Estruturas de Arquivos

Os arquivos podem ser estruturados de várias maneiras. Existem três possibilidades muito comuns:

- Sequência de bytes → O sistema operacional não sabe e nem se importa com o que está no arquivo, tudo que ele vê são bytes. Qualquer significado deve ser imposto por programas no nível do usuário. Ter o sistema operacional considerando arquivos como nada mais que sequencia de bytes oferece um máximo de flexibilidade, pois os programas de usuário podem colocar qualquer coisa que quiserem em arquivos e nomeá-los de qualquer maneira que lhe seja conveniente. O sistema operacional não ajuda, mas também não atrapalha.
- Sequência de registros → Um arquivo é uma sequência de registros de comprimento fixo, cada um com alguma estrutura interna. Nesse caso, a operação de leitura retorna um registro e as operações de gravação sobrescrevem ou anexam um registro.
- Árvore → Nessa organização, um arquivo consiste em uma árvore de registros, não necessariamente todos do mesmo comprimento, cada um contendo um **campo-chave** em uma posição fixa no registro. A árvore é classificada pelo campo chave, permitindo localizar rapidamente uma chave em particular. A operação básica aqui não é obter o "próximo" registro, embora isso também seja possível, mas obter o registro com uma chave específica.



Três tipos de arquivos. (a) Seqüência de bytes. (b) Seqüência de registros. (c) Árvore.

Tipos de arquivo

Muitos sistemas operacionais suportam vários tipos de arquivos. **Arquivos comuns** são os que contém informações do usuário. **Diretórios** são arquivos de sistema para manter a estrutura do sistema de arquivos. **Arquivos especiais de caractere** relacionam-se com a entrada/saída e são utilizados para modelar dispositivos de E/S seriais como terminais, impressoras e redes. **Arquivos especiais de bloco** são utilizados para modelar discos.

Arquivos comuns são geralmente arquivos ASCII ou arquivos binários. Os arquivos ASCII consistem em linhas de texto. A grande vantagem de arquivos ASCII é que podem ser exibidos e impressos como são e podem ser editados com um editor de texto comum. Além disso, se um grande número de programas utiliza arquivos ASCII para entrada e saída, é fácil conectar a saída de um programa à entrada de outro.

Os arquivos binários não são arquivos ASCII. Imprimi-los resulta em uma lista incompreensível cheia de, aparentemente, lixo aleatório. Normalmente eles tem alguma estrutura interna.

Todo sistema operacional deve reconhecer um tipo de arquivo, seu próprio arquivo executável, mas alguns reconhecem mais.

Acesso a arquivos

Os sistemas operacionais antigos ofereciam somente um tipo de acesso a arquivos: **acesso sequencial**. Nesses sistemas, um processo poderia ler todos os bytes ou registros de um arquivo em ordem, iniciando no começo, mas não poderia pular e lê-los fora de ordem. Arquivos sequenciais podem ser retrocedidos. Isso era popular quando se utilizava fita magnética ao invés de discos.

Quando se começou a utilizar discos para armazenar arquivos, tornou-se possível ler os bytes ou registros de um arquivo fora da ordem, ou acessar registros por chave em vez de por posição. Os arquivos cujos bytes ou registros podem ser lidos em qualquer ordem são chamados **arquivos de acesso aleatório**.

Arquivos de acesso aleatório são essenciais para muitos aplicativos como, por exemplo, sistemas de banco de dados. Se um cliente de uma companhia aérea ligar e quiser reservar um assento em um determinado vôo, o programa de reserva deve ser capaz de acessar o registro desse vôo sem primeiro ler os registros de milhares de outros vôos.

Atributos de arquivos

Cada arquivo tem um nome e dados. Além disso, todo sistema operacional associa outras informações com cada arquivo, por exemplo, a data e a hora em que o arquivo foi criado, o tamanho do arquivo, etc. Esses itens extras podem ser chamados de atributos do arquivo e a lista de atributos varia consideravelmente de sistema para sistema.

Exemplos de possíveis atributos de um arquivo:

- Senha → O usuário deve apresentar uma senha para acessar um arquivo.
- Sinalizadores → Bits ou campos curtos que controlam ou ativam alguma propriedade específica, por exemplo, os arquivos ocultos.
- Comprimento de um registro → Número de bytes em um registro
- Comprimento da chave → Número de bytes no campo-chave
- Tempo de criação → Data e hora em que o arquivo foi criado

Operações com arquivos

Os arquivos existem para armazenar informações e permitir que estas informações sejam recuperadas mais tarde. Sistemas diferentes oferecem operações diferentes para permitir armazenamento e recuperação. Algumas operações mais comuns em arquivos são:

- CREATE → O arquivo é criado sem dados, o propósito da chamada é anunciar que o arquivo está vindo e configurar alguns atributos.
- DELETE → Quando o arquivo não é mais necessário, ele precisa ser excluído para liberar espaço em disco. Há sempre uma chamada de sistema para esse propósito.
- OPEN → Antes de utilizar um arquivo, um processo deve abri-lo. O propósito da chamada OPEN é permitir que o sistema transfira os atributos e a lista de endereços de disco para a memória principal, permitindo acesso rápido em chamadas posteriores.
- CLOSE → Quando todos os acessos terminaram, os atributos e os endereços de disco não são mais necessários, então o arquivo deve ser fechado para liberar mais espaço na tabela.
- READ → Os dados são lidos do arquivo. Normalmente os bytes provêm da posição atual.
- WRITE → Os dados são gravados no arquivo, novamente, em geral, na posição atual. Se a posição atual for o fim do arquivo, o tamanho aumentará. Se a posição atual estiver no meio do arquivo, os dados existentes serão sobrescritos e perdidos para sempre.
- APPEND → Essa chamada é uma forma restringida do WRITE, ela somente pode colocar os dados no fim do arquivo.
- RENAME → Essa chamada torna possível renomear um arquivo caso o usuário precise alterar o

nome de um arquivo existente.

Referências bibliográficas:

TANENBAUM, Andrew. 2ª ed. **Sistemas Operacionais Modernos,** Editora Pearson, 2003.

SILBERSCHATZ, Abraham. **Sistemas Operacionais com JAVA,** 6ª ed. Editora Campus

MACHADO, Francis B. **Arquitetura de Sistemas Operacionais,** 4ª ed, LTC, 2007.