MC-102 — Aula 22
Arquivos

Instituto de Computação – Unicamp

Segundo Semestre de 2011

Roteiro

Arquivos

Características

- Podem armazenar grande quantidade de informação.
- Dados são persistentes (gravados em disco).
- Acessso aos dados pode ser não seqüencial (acesso direto a registros em um banco de dados).
- Acesso à informação pode ser concorrente (mais de um programa ao mesmo tempo).

Nomes e extensões

- Arquivos são identificados por um nome.
- O nome de um arquivo pode conter uma extensão que indica o conteúdo do arquivo.

Algumas extensões

arq.txt	arquivo texto simples
arq.c	código fonte em C
arq.pdf	*portable document format*
arq.html	arquivo para páginas WWW *(hypertext markup language)*
arq*	arquivo executável (UNIX)

Tipos de arquivos

Arquivos podem ter o mais variado conteúdo, mas do ponto de vista dos programas existem apenas dois tipos de arquivo:

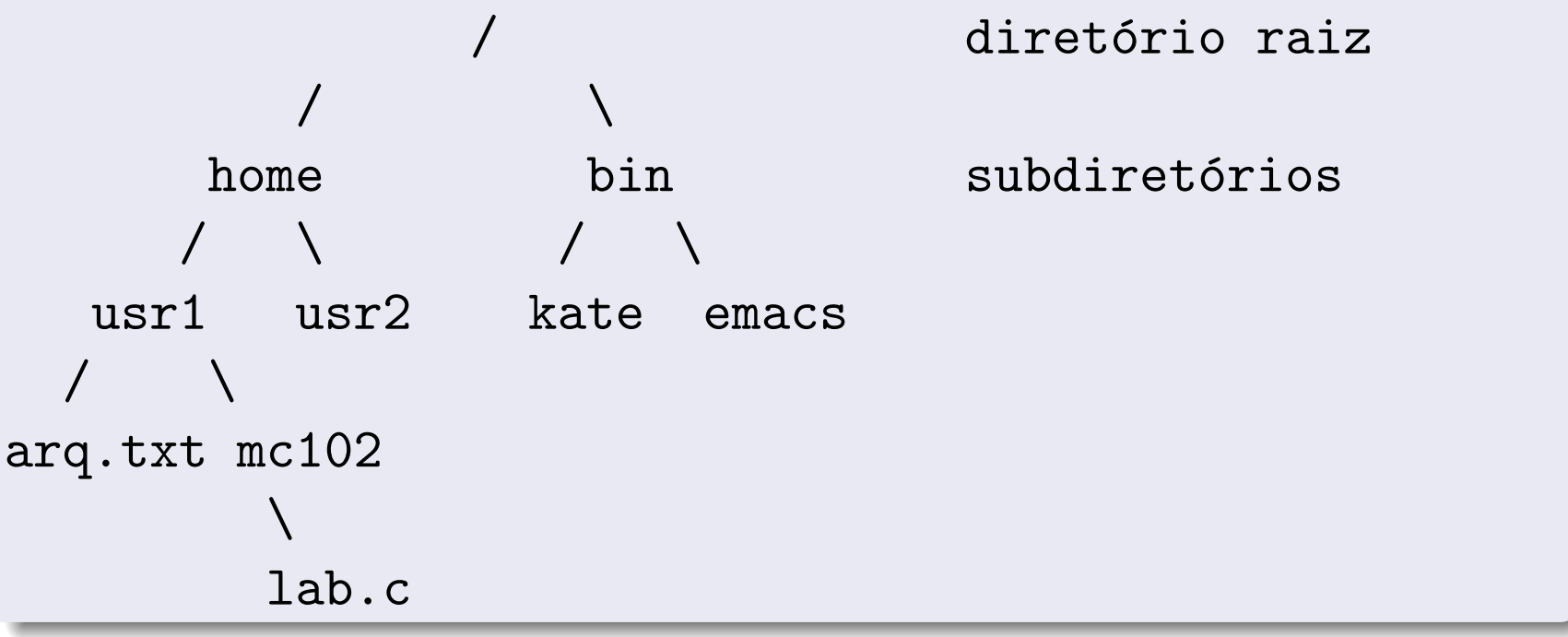
Arquivo texto: Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de textos simples. Exemplos: código fonte C, documento texto simples, páginas HTML.

Arquivo binário: Seqüência de bits sujeita às convenções dos programas que o gerou, não legíveis diretamente. Exemplos: arquivos executáveis, arquivos compactados, documentos do Word.

Diretório

- Também chamado de pasta.
- Contém arquivos e/ou outros diretórios.

Uma hierarquia de diretórios



Caminhos absolutos ou relativos

O nome de um arquivo pode conter o seu diretório, ou seja, o caminho para encontrar este arquivo a partir da raiz. Os caminhos podem ser especificados de duas formas:

Caminho absoluto: descrição de um caminho desde o diretório raiz.

```
/bin/emacs
/home/usr1/arq.txt
```

Caminho relativo: descrição de um caminho desde o diretório corrente.

```
arq.txt
mc102/lab.c
```

Para ver qual é o diretório corrente, use o comando `pwd`. Para mudar de diretório, use o comando `cd`.

Atributos de arquivos

Além do nome, arquivos possuem vários outros atributos:

- Nome do arquivo
- Proprietário do arquivo
- Datas de criação, alteração e acesso
- Tamanho em bytes
- Permissão de acesso

Para ver estes atributos, use os comandos `ls -l` e `stat`.

Permissão de acesso

Existem três níveis de controle: proprietário, grupo e todos.

```
$ ls -l
-rw-r----- 1 jose alunos  545 Nov  8  2005 cp.c
drwxr-xr-x 2 jose alunos 4096 Jun  6 14:54 mc102/
```

- r: leitura
- w: escrita
- x: execução para arquivos, permissão de entrada para diretórios

Abrindo um arquivo para leitura

- Antes de acessar um arquivo, devemos abri-lo com a função `fopen()`.
- Em caso de erro a função retorna `NULL`.
- A função `perror()` obtém e exibe uma mensagem explicativa.

Abrindo o arquivo `teste.txt`

```
if (fopen("teste.txt", "r") == NULL)
  perror("Erro ao abrir o arquivo.\n");
else
  printf("Arquivo aberto para leitura.\n");
```

Veja o exemplo em `fopen-r.c`.

Lendo dados de um arquivo

- Não basta chamar a função `fopen()`, temos que pegar o seu valor de retorno (um apontador para *stream*).
- Para ler dados do arquivo, usamos a função `fscanf()`, semelhante à função `scanf()`.
- Para fechar o arquivo usamos a função `fclose()`.

Lendo dados do arquivo `teste.txt`

```
FILE *f = fopen ("teste.txt", "r");
while (fscanf(f, "%c", &c) != EOF)
  printf("%c", c);
fclose(f);
```

Veja o exemplo em `fscanf.c`.

Escrevendo dados em um arquivo

- Para escrever em um arquivo, ele deve ser aberto de forma apropriada.
- Usamos a função `fprintf()`, semelhante a função `printf()`.

Copiando dois arquivos

```
FILE *fr = fopen ("teste.txt", "r");
FILE *fw = fopen ("saida.txt", "w");
while (fscanf(fr, "%c", &c) != EOF)
  fprintf(fw,"%c", c);
fclose(fr);
fclose(fw);
```

Veja o exemplo em `fprintf.c`.

fopen

Um pouco mais sobre a função `fopen()`.

```
FILE* fopen(const char *caminho, char *modo);
```

Modos de abertura de arquivo

modo	operações	ponto no arquivo
r	leitura	início
r+	leitura e escrita	início
w	escrita	início
w+	leitura e escrita	início
a	escrita	final
a+	leitura	início
	escrita	final

Motivação

- Variáveis `int` ou `float` têm tamanho fixo na memória. Por exemplo, um `int` ocupa 4 bytes.
- Representação em texto precisa de um número variável de dígitos (`10`, `5.673`, `100.340`), logo de um tamanho variável.
- Armazenar dados em arquivos de forma análoga a utilizada em memória permite:
  - Reduzir o tamanho do arquivo.
  - Realizar busca não seqüencial.

fread e fwrite

- As funções `fread` e `fwrite` permitem a leitura e escrita de blocos de dados.
- Devemos determinar o número de elementos a serem lidos ou gravados e o tamanho de cada um.

```
unsigned fread(void *buffer, int Num_bytes,
               int cont, FILE *stream);

unsigned fwrite(void *buffer, int Num_bytes,
               int cont, FILE *stream);
```

fread e fwrite

```
unsigned fread(void *buffer, int Num_bytes,
               int cont, FILE *stream);
```

- buffer: região de memória que armazena dados lidos de fp
- num bytes: tamanho da unidade lida
- cont: quantas unidades lidas
- fread retorna número de unidades lidas (pode ser ¡ count se final de arquivo)

```
unsigned fwrite(void *buffer, int Num_bytes,
               int cont, FILE *stream);
```

- fwrite: "companheira"de fread, porém escreve no arquivo
- fwrite retorna número de ítens escritos

Veja o exemplo em fwrite-fread.c

Registros

- Um arquivo pode armazenar registros (como um banco de dados).
- Isso pode ser feito de forma bem fácil se lembrarmos que um registro, como qualquer variável em C, tem um tamanho fixo.
- A leitura ou escrita do registro pode ser feita usando as funções `fread` e `fwrite`.

Veja o exemplo em `esc-registroArquivo.c` e `le-registroArquivo.c`..

Exercício

- Escreva um programa que abra um arquivo texto e conte o número de caracteres presentes nele e imprima na tela o número de caracteres.
- Escreva um programa que some os números de um arquivo chamado numero.txt. O resultado deve ser impresso na tela.
- Desafio: Escreva um programa que leia dois arquivos de inteiros aleatórios ordenemos e escreva um arquivo cuja saída é um único arquivo ordenado.