GBC053 - Gerenciamento de Bancos de Dados

Aula 2
Arquivos

Humberto Razente
humberto.razente@ufu.br

Arquivos físicos e lógicos

- Arquivo em uma mídia
  - coleção de bytes armazenados na mídia
- Quando a palavra ”arquivo” é usada nesse sentido, existe fisicamente
- Um disco pode conter milhares de arquivos

Arquivos físicos e lógicos

- Do ponto de vista de uma aplicação, a noção de arquivo é diferente
  - analogia com linha telefônica conectada a rede de telefonia:
    - um programa pode receber e enviar bytes pela linha telefônica, sem saber realmente de onde vem e para onde vão esses bytes
    - o programa conhece apenas a sua ponta da linha telefônica

Abrindo arquivos

- Um arquivo tem um identificador lógico
  - ligado ao arquivo físico ou dispositivo
- Opções:
  - abrir um arquivo existente
  - criar um novo arquivo

Abrindo arquivos

- Biblioteca Unix → C
  - fd = open(filename, flags, [pmode]);
  - fd → descritor do arquivo
    - usado pelo programa para manipular o arquivo
  - filename → nome do arquivo
  - flags → O_APPEND, O_CREATE, O_RDONLY, O_RDRW, …
  - pmode → se O_CREATE, pmode é necessário

r	w	e		r	w	e		r	w	e
4	2	1		4	2	1		4	2	1
owner				group				other		

Abertura de arquivos

- C
  - Abrir um arquivo para leitura e escrita, ou criá-lo se necessário
    - int fd = open(”x.txt”, O_RDWR | O_CREATE, 751);

Fechamento de arquivos

- Fechar um arquivo é como desligar o telefone
- Após o fechamento, a linha está disponível para uma nova ligação
  - o manipulador pode ser usado para abrir um novo arquivo
- O fechamento garante a persistência do arquivo na mídia
  - antes do fechamento, dados podem estar em cache em memória principal
- C
  - close(fd);

Leitura e escrita

- Operações de Input/Output (I/O)
  - read (fd, destino, tamanho);
  - write (fd, destino, tamanho);

Exemplo

```
int main () {
    char c; int fd; char filename[30];
    printf("Entre com o nome do arquivo: ");
    gets(filename);
    fd = open(filename, O_RDONLY);
    if (fd < 0) {
       printf("Arquivo não encontrado\n\n");
       return 0;
    }
    while (read(fd, &c, 1) != 0)
       printf("%c", c);
    close(fd);
    return 0;
}
```

Movimentação (seeking)

- Com ”read” pode-se ler um arquivo byte a byte a partir do seu início, até atingir o seu final
  - a cada byte lido, o sistema operacional movimenta o ponteiro de I/O à frente
- Entretanto, se um programa precisa de uma informação que esteja 1.000.000 de bytes à frente
  - ”pular” até a posição é mais interessante

Movimentação (seeking)

- C
  - pos = lseek (fd, byte_offset, origin)
    - pos → long, posição do ponteiro após movimentação
    - fd → descritor do arquivo
    - byte_offset → número de bytes a mover a partir da origem
      - pode ser um valor negativo ou positivo
        - origin
      - 0 → a partir do início do arquivo
      - 1 → a partir da posição corrente
      - 2 → a partir do final do arquivo

Linguagem C

- open, read, write, lseek, close
  - POSIX (Portable Operating System Interface)
    - padrão IEEE para sistemas UNIX

- fopen, fread, fwrite, fseek, fclose
  - Standard C Library
    - mais portável que POSIX

fopen

- FILE * fopen(char * filename, char * mode);

C string containing a file access modes. It can be:

"r"	Open a file for reading. The file must exist.
"w"	Create an empty file for writing. If a file with the same name already exists its content is erased and the file is treated as a new empty file.
"a"	Append to a file. Writing operations append data at the end of the file. The file is created if it does not exist.
"r+"	Open a file for update both reading and writing. The file must exist.
"w+"	Create an empty file for both reading and writing. If a file with the same name already exists its content is erased and the file is treated as a new empty file.
"a+"	Open a file for reading and appending. All writing operations are performed at the end of the file, protecting the previous content to be overwritten. You can reposition (fseek, rewind) the internal pointer to anywhere in the file for reading, but writing operations will move it back to the end of file. The file is created if it does not exist.

Exemplo em Standard C Library

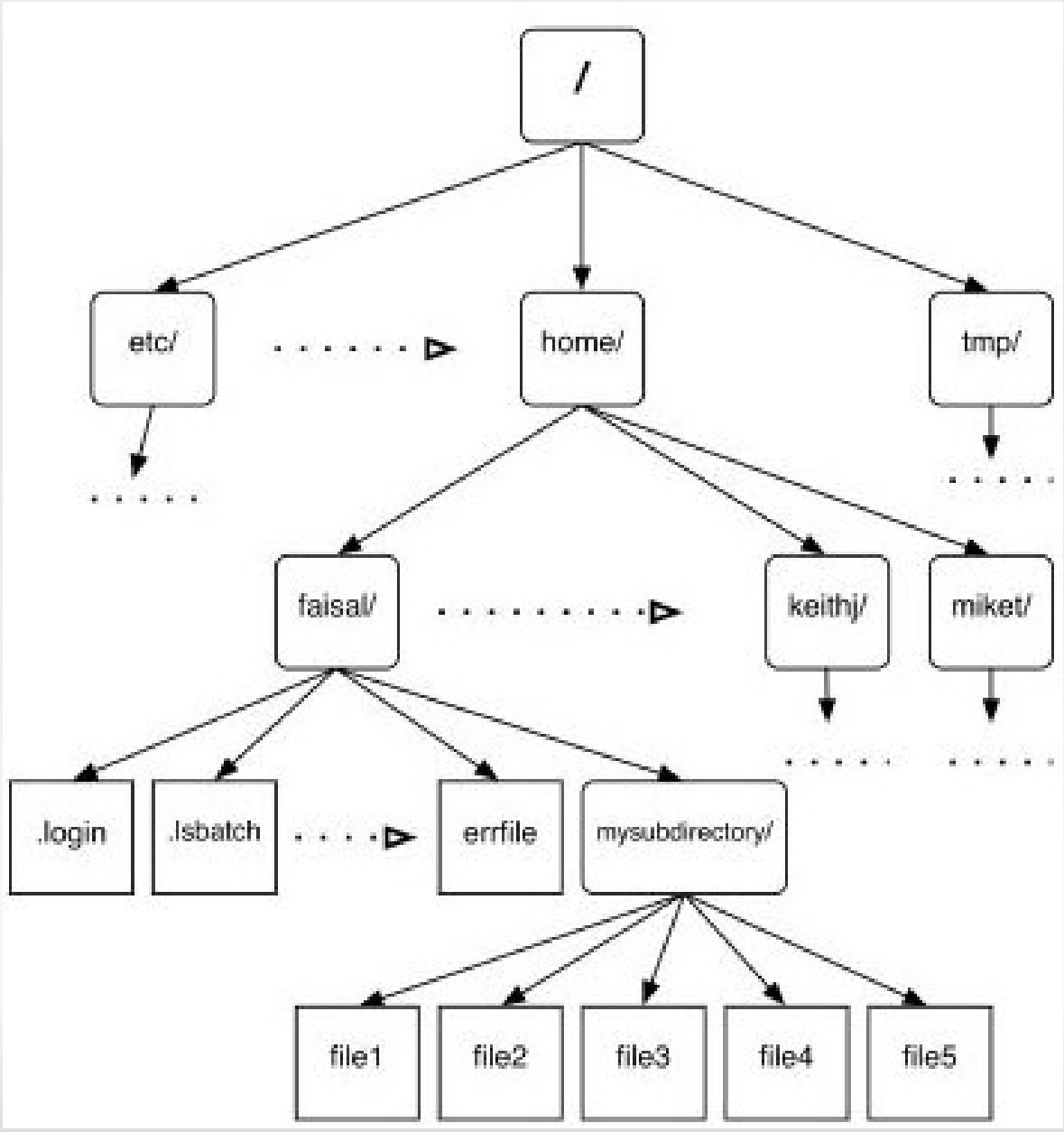
```
#include <stdio.h>

int main () {
   char c; FILE *fd; char filename[30];
   printf("Entre com o nome do arquivo: ");
   gets(filename);
   fd = fopen(filename, "rt");
   if (fd == NULL) {
      printf("Arquivo não encontrado\n\n");
      return 0;
   }
   while (fread(&c, 1, 1, fd) != 0)
      printf("%c", c);
   fclose(fd);
   return 0;
}
```

Sistema de arquivos Unix / Linux

- Leitura sugerida
  - Capítulo 1 do livro ”The Linux Kernel”
    - ”Linux is not Unix”
    - http://kernelbook.sourceforge.net/
    - http://kernelbook.sourceforge.net/pdf/ch-intro.pdf

- Cada arquivo é parte do sistema de arquivos
  - inicia com a raiz → /
  - cada arquivo pode ser identificado univocamente pelo seu caminho absoluto
  - diretórios e dispositivos são representados como arquivos

Sistema de arquivos Unix / Linux



Sistema de arquivos Unix / Linux

- Arquivo
  - em Unix, um arquivo é uma sequência de bytes, sem qualquer implicação de como ou onde estes bytes estão armazenados ou de onde originam
    - visão conceitual que define em poucas operações tarefas que requerem muitas diferentes operações em outros sistemas operacionais
      - dispositivos também são tratados como arquivos
        - teclado, modem, rede, discos, mídias diversas

Sistema de arquivos Unix / Linux

- Redirecionamento de I/O e pipes
  - permite especificar em tempo de execução arquivos alternativos para entrada e saída
    - < arquivo
      - redireciona STDIN para ”arquivo”
    - > arquivo
      - redireciona STDOUT para ”arquivo”
  - se, ao invés de armazenar a saída de um programa em um arquivo, desejar usá-lo imediatamente em outro programa, um ”pipe” permite essa operação
    - programa1 | programa2

Redirecionamento e pipes

- O comando cat escreve o conteúdo de um arquivo na saída
- Usa-se o símbolo > para redirecionar a saída de um comando. Por exemplo, para criar um arquivo contendo uma lista:

  cat > lista1

- Em seguida, entre com a lista:

  pera

  banana

  laranja

  ^D (Control D para parar)

Redirecionamento e pipes

- O que acontece é que o comando cat lê a entrada padrão (teclado) e o > redireciona a saída do cat, que normalmente é a tela, para um arquivo denominado lista1

  Para ler o conteúdo

  ```
  cat lista1
  ```

- >> redireciona para o final:

  ```
  cat >> lista1
  pera
  abacaxi
  ^D
  ```

Redirecionamento e pipes

```
cat lista1 lista1 > lista2
sort < lista2
sort < lista2 > lista3
```

- Pode-se também conectar a saída de um programa com a entrada de outro:

```
who | sort
```

Sistema de arquivos Unix / Linux

- Principais comandos:
  - cat → imprime conteúdo de um arquivo
  - tail → imprime últimas linhas de um arquivo
  - cp → copia arquivos
  - mv → move arquivos
  - rm → remove arquivos
  - chmod → altera permissões rwe-rwe-rwe
  - ls → lista arquivos
  - mkdir → cria diretório
  - rmdir → remove diretório

Leitura complementar

- Capítulo ”Fundamental file processing operations” do livro
  - Folk et al. ”File Structures: An Object-Oriented Approach with C++”, Editora Pearson, 3ª edição, 1998
- Sintaxe comandos das bibliotecas ”stdio.h” e ”string.h”:
  - http://www.cplusplus.com/reference/clibrary/cstdio/
  - http://www.cplusplus.com/reference/clibrary/cstring/