

Sistemas Operacionais

Gestão de arquivos - uso de arquivos

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Agosto de 2020

Conteúdo

- 1 Interfaces de acesso
- 2 Acesso a arquivos
- 3 Compartilhamento de arquivos
- 4 Controle de acesso
- 5 APIs de acesso a arquivos

Interfaces de acesso

Existem dois níveis de interface de arquivos.

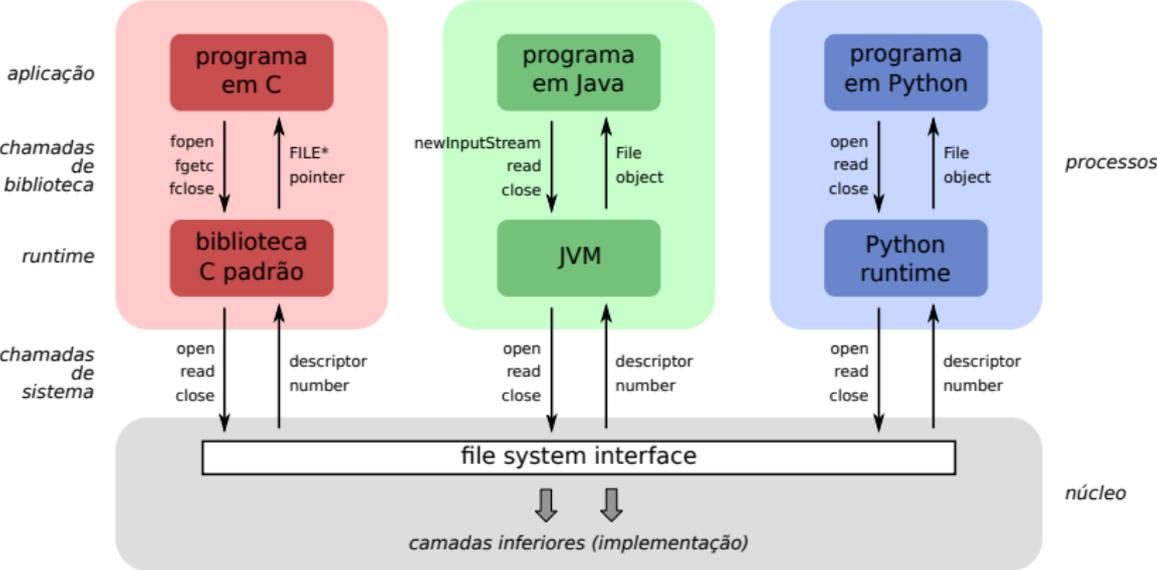
Interface de **alto nível**:

- Chamadas de biblioteca (ou *runtime*)
- **Depende da linguagem**, independente do SO

Interface de **baixo nível**:

- Chamadas de sistema
- **Depende do SO**, independente da linguagem

Interface de acesso



Funções de bibliotecas

Usadas pela aplicação para manipular arquivos.

Dependem da linguagem de programação.

| Operação | C (padrão C99) | Java (classe File) |
|-----------------|------------------------------|------------------------------|
| Abrir arquivo | <code>fd = fopen(...)</code> | <code>obj = File(...)</code> |
| Ler dados | <code>fread(fd, ...)</code> | <code>obj.read()</code> |
| Escrever dados | <code>fwrite(fd, ...)</code> | <code>obj.write()</code> |
| Fechar arquivo | <code>fclose(fd)</code> | <code>obj.close()</code> |
| Remover arquivo | <code>remove(...)</code> | <code>obj.delete()</code> |
| Criar diretório | <code>mkdir(...)</code> | <code>obj.mkdir()</code> |

Chamadas de sistema

Usadas pelo processo ao interagir com o núcleo.

Dependem do sistema operacional.

| Operação | Linux | Windows |
|-----------------|--------------|-------------------------|
| Abrir arquivo | OPEN | NtOpenFile |
| Ler dados | READ | NtReadRequestData |
| Escrever dados | WRITE | NtWriteRequestData |
| Fechar arquivo | CLOSE | NtClose |
| Remover arquivo | UNLINK | NtDeleteFile |
| Criar diretório | MKDIR | NtCreateDirectoryObject |

Descritor de arquivo

Estrutura que representa um **arquivo aberto**.

Descritor de **alto nível**:

- Provido pela biblioteca ou *runtime*
- C: variável dinâmica do tipo FILE*
- Java: objetos da classe File

Descritor de **baixo nível**:

- Provido pelo núcleo
- Windows: *file handle* (*struct* opaco)
- Linux: *file descriptor* (*int*)

Acesso ao conteúdo do arquivo

Passos:

- 1 Abrir o arquivo
- 2 Acessar o conteúdo do arquivo:
 - Acesso sequencial
 - Acesso direto (ou aleatório)
 - Mapeamento em memória
 - Acesso indexado
- 3 Fechar o arquivo

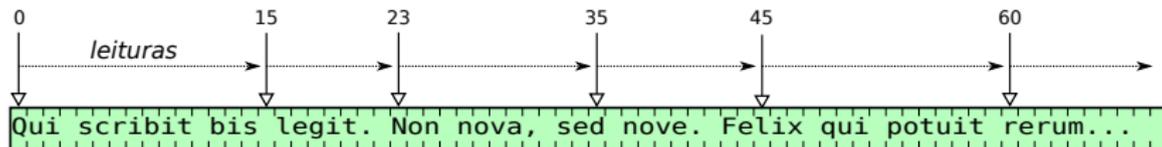
Passos da abertura de um arquivo

- No processo:
 - 1 a aplicação pede a abertura do arquivo (`fopen()`)
 - 2 o *runtime* da linguagem invoca uma *syscall*
- No núcleo:
 - 3 o núcleo localiza o arquivo no dispositivo físico
 - 4 verifica as permissões de acesso do arquivo
 - 5 cria um descritor para o arquivo aberto no núcleo
 - 6 insere o descritor em uma lista de arquivos abertos
 - 7 devolve ao processo uma referência ao descritor
- No processo:
 - 8 o *runtime* recebe o descritor de baixo nível
 - 9 o *runtime* cria um descritor de alto nível
 - 10 a aplicação recebe o descritor de alto nível

Acesso sequencial

Forma de acesso mais usual:

- dados acessados em seqüência do início ao fim do arquivo
- Para cada arquivo aberto há um *ponteiro de acesso*
- A cada leitura ou escrita, o ponteiro é incrementado
- Um *flag (End-of-File)* indica o fim do arquivo



Acesso direto (ou aleatório)

Acesso direto a posições específicas do arquivo:

- Indica-se a posição do arquivo onde fazer o acesso
- `read (file, position, size, buffer)`
- Importante em SGBDs e aplicações similares
- Raramente implementada “pura” nos SOs

Geralmente feita com o reposicionamento do ponteiro:

- chamada de sistema `lseek()`
- chamada de biblioteca C `fseek()`

Mapeamento em memória

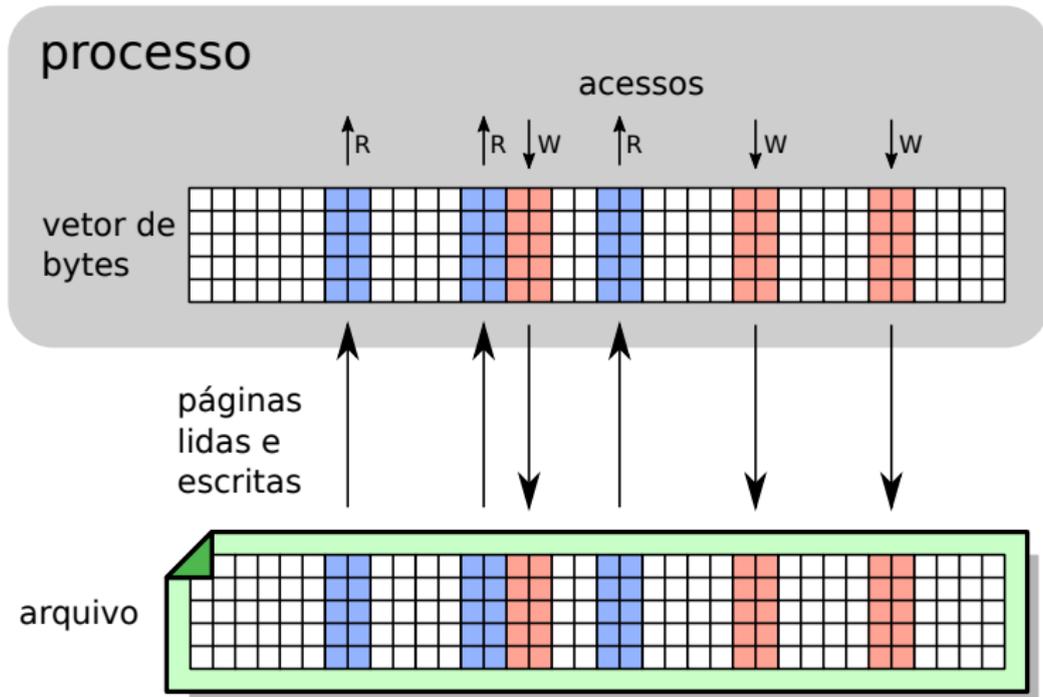
Usa os mecanismos de memória virtual (páginação).

O arquivo é associado a um vetor de bytes em RAM.

Quando uma posição do vetor é lida, é gerada uma falta de página e o conteúdo correspondente no arquivo é lido.

Podem ser mapeadas apenas partes do arquivo.

Mapeamento em memória



Acesso indexado

Implementado em alguns sistemas, como o OpenVMS.

Estrutura interna do arquivo é vista com tabela indexada.

Operações de acesso usando pares chave/valor:

- `value = read (key)`
- `write (key, value)`

Acesso muito rápido (indexação implementada no núcleo).

Pouco usado nos SOs atuais, mas disponível por bibliotecas.

Travas (*locks*)

Controle de concorrência (*mutex*):

- Necessário ao compartilhar arquivos entre processos
- Arquivos ou trechos podem ser travados
- Vários tipos de travas estão disponíveis

Várias travas são oferecidas:

- pelo sistema operacional
- por bibliotecas

Tipos de travas

Trava obrigatória (*mandatory lock*)

- Incontornável, imposta pelo núcleo aos processos
- Processos são suspensos aguardando liberar a trava
- *Default* nos sistemas Windows

Trava recomendada (*advisory lock*)

- Gerenciada pelo suporte de execução (biblioteca)
- Pode ser ignorada pela aplicação
- Útil para gerenciar concorrência dentro de uma aplicação
- O programador deve usar as travas conforme necessário
- *Default* nos sistemas UNIX

Tipos de travas

Trava exclusiva (*write lock*)

- Garante acesso exclusivo ao arquivo

Trava compartilhada (*read lock*)

- Impede travas exclusivas sobre o arquivo
- Permite mais travas compartilhadas

Equivalente ao problema dos “leitores/escritores”.

Semânticas de compartilhamento

Em acessos concorrentes a um arquivo compartilhado:

- Como cada processo vê os dados do arquivo?
- Como escritas concorrentes se comportam?

Várias possibilidades (“semânticas”):

- Semântica imutável
- Semântica UNIX
- Semântica de sessão
- Semântica de transação

Semânticas de compartilhamento

Semântica imutável:

- Arquivos compartilhados não podem ser alterados.
- Solução simples para garantir a consistência dos dados.
- Usada em alguns sistemas de arquivos distribuídos.
- Modo default nos sistemas Windows.

Semântica UNIX:

- Modificações são visíveis imediatamente a todos
- Usual em sistemas de arquivos locais UNIX

Semânticas de compartilhamento

Semântica de sessão:

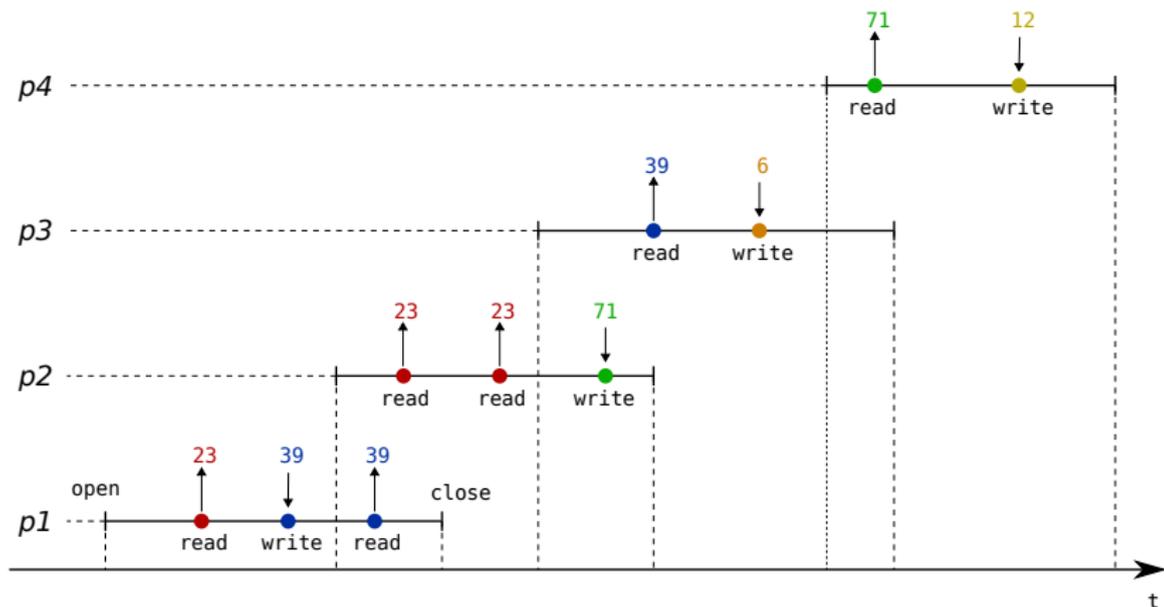
- Cada processo usa um arquivo em uma sessão.
- A sessão vai da abertura ao fechamento do arquivo.
- Modificações só são visíveis aos demais no fim da sessão.
- Sessões concorrentes podem ver conteúdos distintos.
- Normalmente usada em sistemas de arquivos de rede.

Semântica de transação:

- Similar à sessão, mas mais curta (`begin ... end`).
- Usa o conceito de *transação* dos SGBDs.

Semântica de sessão

Arquivo compartilhado inicialmente contém o valor 23.



Controle de acesso

Definir e controlar que usuários podem acessar que arquivos.

Atributos relevantes:

- **Proprietário:** usuário dono do arquivo
- **Permissões:** operações permitidas aos usuários

Implementação:

- Definição de usuários e grupos
- Listas de controle de acesso (ACL)
- O núcleo autoriza ou nega cada acesso
- Modelos mais complexos (SELinux, AppArmor, MIC, etc)

ACL

Access Control List associada a cada arquivo.

Indica as ações que os usuários podem fazer sobre ele.

Exemplo:

```
1  arq1.txt  : (João: ler), (José: ler, escrever), (Maria: ler, remover)
2  video.avi : (José: ler), (Maria: ler)
3  musica.mp3: (Daniel: ler, escrever, apagar)
```

Permissões em UNIX

- **Usuários:**
 - **user** : o proprietário do arquivo
 - **group**: um grupo de usuários associado ao arquivo
 - **other**: os demais usuários
- **Permissões: read, write, execute**

```

1 $ ls -l
2 drwx----- 2 maziero prof 4096 2008-09-27 08:43 figuras
3 -rwxr-x--- 1 maziero prof 7248 2008-08-23 09:54 hello-unix
4 -rw-r--r-- 1 maziero prof 54 2008-08-23 09:54 hello-unix.c
5 -rw----- 1 maziero prof 59 2008-08-23 09:49 hello-windows.c
6 -rw-r--r-- 1 maziero prof 89580 2008-09-26 22:08 main.pdf
7 -rw----- 1 maziero prof 40494 2008-09-27 08:44 main.tex
  
```

API de acesso a arquivos em C

- Abertura e fechamento de arquivos:
 - `fopen (const char *fname, ...)`
 - `fclose (FILE *f)`
- Leitura e escrita de caracteres e *strings*:
 - `fputc (int c, FILE *f)`
 - `fgetc (FILE *f)`
- Uso do ponteiro do arquivo:
 - `ftell (FILE *f)`
 - `fseek (FILE *f, long offset, ...)`
 - `feof (FILE *f)`
- Uso de travas:
 - `flockfile (FILE *f)`
 - `funlockfile (FILE *f)`

API de acesso a arquivos em C

```

1  int main (int argc, char *argv[])
2  {
3      FILE *arq ;           // descritor de arquivo
4      char c ;
5
6      arq = fopen ("infos.dat", "r") ; // abre arquivo para leitura
7
8      if (! arq) {         // descritor inválido
9          perror ("Erro ao abrir arquivo") ;
10         exit (1) ;
11     }
12
13     c = getc (arq) ;      // le um caractere do arquivo
14     while (! feof (arq)) { // enquanto não chegar ao fim do arquivo
15         putchar (c) ;    // imprime o caractere na tela
16         c = getc (arq) ; // le um caractere do arquivo
17     }
18
19     fclose (arq) ;       // fecha o arquivo
20 }
  
```