Uso de arquivos

PROFESSOR: Jheymesson A. Cavalcanti

DISCIPLINA: SISTEMAS OPERACIONAIS II

1. Introdução

A grande pergunta desta aula é: **“Como os processos fazem uso dos arquivos?”**

Para usar arquivos, os processos têm à sua disposição uma interface de acesso;

Essa interface de acesso depende de alguns aspectos, como:

- A linguagem utilizada;
- O sistema operacional subjacente;
- Etc.

Nesta aula, veremos alguns aspectos relativos ao uso dos arquivos.

2. Interface de acesso

A interface de acesso a um arquivo normalmente é composta por:

- Uma **representação lógica do arquivo**, denominada descritor de arquivo (file descriptor ou file handle);
- Um conjunto de **funções para manipular o arquivo**.

Através dessa interface, um processo pode localizar o arquivo no dispositivo físico, ler e modificar seu conteúdo, entre outras operações.

2. Interface de acesso

Existem dois níveis de interface de acesso:

- Uma interface de **baixo nível**, oferecida pelo sistema operacional aos processos através de chamadas de sistema;
- Uma interface de **alto nível**, composta de funções na linguagem de programação usada para implementar cada aplicação.

2. Interface de acesso

A interface de baixo nível é definida por chamadas de sistema, pois os arquivos são abstrações criadas e mantidas pelo núcleo do sistema operacional;

Por essa razão, essa interface é dependente do sistema operacional subjacente;

Exemplos de chamadas de sistema oferecidas pelos sistemas operacionais Linux e Windows para o acesso a arquivos:

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

2. Interface de acesso

Em contrapartida, a interface de alto nível é específica para cada linguagem de programação e normalmente não depende do sistema operacional subjacente;

Qual a diferença?

Esta independência auxilia a portabilidade de programas entre sistemas operacionais distintos;

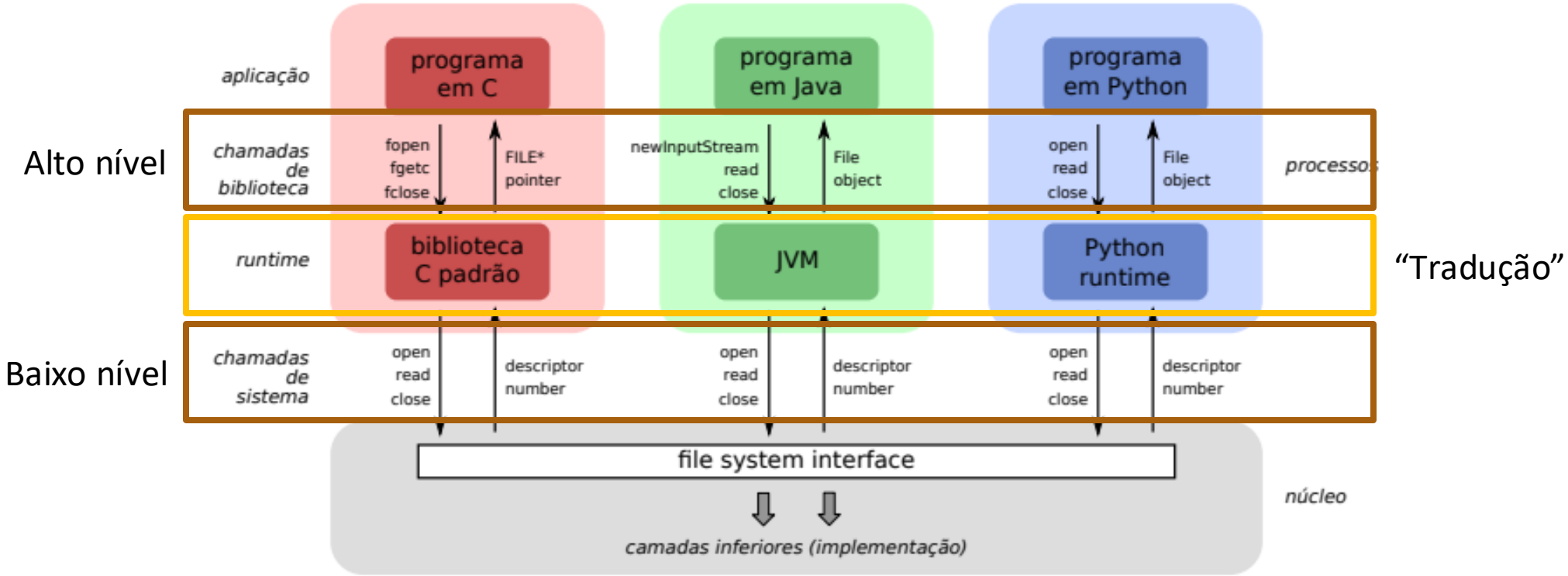
A interface de alto nível é implementada sobre a interface de baixo nível, geralmente na forma de uma biblioteca de funções e/ou um suporte de execução (runtime);

Alguns exemplos de funções/métodos para acesso a arquivos das linguagens C e Java:

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

2. Interface de acesso

Os dois níveis de abstração:



3. Conceitos importantes (Descritores)

Um descritor de arquivo é uma **representação lógica** de um arquivo em uso por um processo;

O descritor é criado no momento da abertura do arquivo e serve como uma referência ao mesmo nas operações de acesso subsequentes;

Obs:

- Os descritores de arquivo usados nos dois níveis de interface geralmente são distintos!

3. Conceitos importantes (Descritores)

Descritores de alto nível representam arquivos dentro de uma aplicação;

Eles dependem da linguagem de programação escolhida, pois são implementados pelas bibliotecas que dão suporte à linguagem;

Por outro lado, independem do sistema operacional subjacente, facilitando a portabilidade da aplicação entre SOs distintos;

Exemplos:

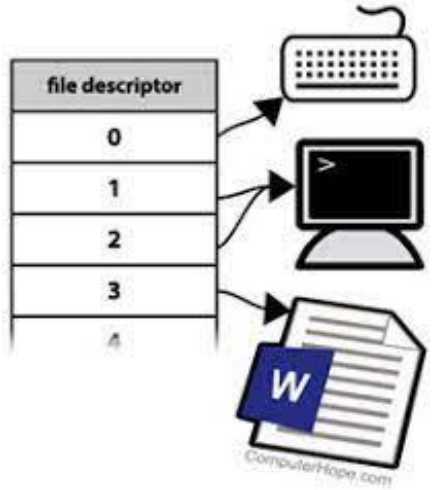
- Descritores de arquivos usados na linguagem C são ponteiros para estruturas do tipo FILE (ou seja, FILE*), mantidas pela biblioteca C;
- Descritores de arquivos abertos em Java são objetos instanciados a partir da classe File.

3. Conceitos importantes (Descritores)

Os descritores de baixo nível não são ligados a uma linguagem específica e variam conforme o sistema operacional;

Exemplos:

- Em sistemas POSIX/UNIX, o descritor de arquivo de baixo nível é um número inteiro positivo;
- Ele indica a posição do arquivo correspondente em uma tabela de arquivos abertos mantida pelo núcleo.



3. Conceitos importantes (Descritores)

Exemplos:

- Em sistemas Windows, os arquivos abertos por um processo são representados pelo núcleo por referências de arquivos (*file handles*);
- Estas referências são estruturas de dados criadas pelo núcleo para representar cada arquivo aberto.

Dessa forma, cabe às bibliotecas e ao suporte de execução de cada linguagem de programação mapear a representação de arquivo aberto fornecida pelo núcleo do sistema operacional na referência de arquivo aberto usada por aquela linguagem.

3. Conceitos importantes (Abetura de arquivo)

Para poder ler ou escrever dados em um arquivo, cada aplicação precisa “abri-lo” antes;

A abertura de um arquivo consiste basicamente em preparar as estruturas de memória necessárias para acessar os dados do arquivo;

Na abertura de um arquivo, os seguintes passos são realizados:

1. No processo:
   - (a) a aplicação solicita a abertura do arquivo (fopen(), se for um programa C);
   - (b) o suporte de execução da aplicação recebe a chamada de função, trata os parâmetros recebidos e invoca uma chamada de sistema para abrir o arquivo.

3. Conceitos importantes (Abetura de arquivo)

2. No núcleo:
   - (a) o núcleo recebe a chamada de sistema;
   - (b) localiza o arquivo no dispositivo físico, usando seu nome e caminho de acesso;
   - (c) verifica se o processo tem as permissões necessárias para usar aquele arquivo da forma desejada;
   - (d) cria uma estrutura de dados na memória do núcleo para representar o arquivo aberto;
   - (e) insere uma referência a essa estrutura na relação de arquivos abertos mantida pelo núcleo, para fins de gerência;
   - (f) devolve à aplicação uma referência a essa estrutura (o descritor de baixo nível), para ser usada nos acessos subsequentes ao arquivo.

3. No processo:
   - (a) o suporte de execução recebe do núcleo o descritor de baixo nível do arquivo;
   - (b) o suporte de execução cria um descritor de alto nível e o devolve ao código da aplicação;
   - (c) o código da aplicação recebe o descritor de alto nível do arquivo aberto, para usar em suas operações subsequentes envolvendo aquele arquivo.

3. Conceitos importantes (Abetura de arquivo)

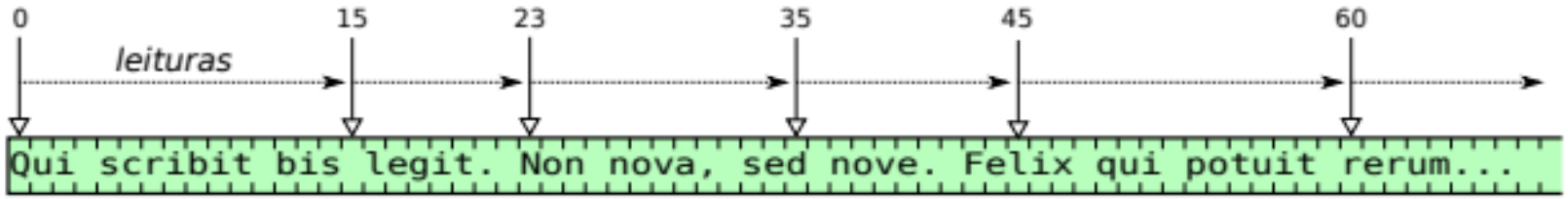
Assim que o processo tiver terminado de usar um arquivo, ele solicita ao núcleo o fechamento do arquivo;

Isso implica em concluir as operações de escrita eventualmente pendentes e remover da memória do núcleo as estruturas de dados criadas durante sua abertura.

3. Conceitos importantes (Formas de acesso)

3.1 Acesso sequencial:

- Os dados são sempre lidos e/ou escritos em sequência, do início ao final do arquivo;



- Para cada arquivo aberto por uma aplicação é definido um ponteiro de acesso, que inicialmente aponta para a primeira posição do arquivo;
- A cada leitura ou escrita, esse ponteiro é incrementado e passa a indicar a posição da próxima leitura ou escrita;
- Quando esse ponteiro atinge o final do arquivo, as leituras não são mais permitidas, mas as escritas podem sê-lo, permitindo acrescentar dados ao final do mesmo;
- A chegada do ponteiro ao final do arquivo é normalmente sinalizada ao processo através de um flag de fim de arquivo (EoF – End-of-File).

3. Conceitos importantes (Formas de acesso)

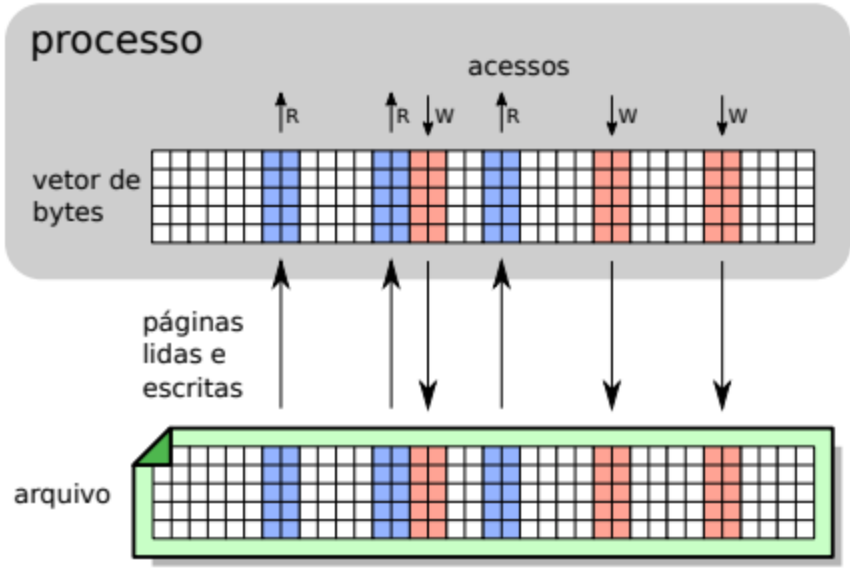
3.2 Acesso aleatório (ou direto):

- Pode-se indicar diretamente a posição no arquivo onde cada leitura ou escrita deve ocorrer, sem a necessidade de um ponteiro de posição corrente;
- Essa forma de acesso é muito importante em gerenciadores de bancos de dados e aplicações congêneres.

3. Conceitos importantes (Formas de acesso)

3.3 Acesso mapeado 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.



3. Conceitos importantes (Formas de acesso)

3.3 Acesso mapeado em memória:

- O acesso mapeado em memória é extensivamente usado pelo núcleo para carregar código executável na memória;
- Esse procedimento é usualmente conhecido como paginação sob demanda (demand paging), pois os dados são lidos do arquivo para a memória em páginas.

3. Conceitos importantes (Formas de acesso)

3.4 Acesso indexado:

- O sistema implementa arquivos cuja estrutura interna pode ser vista como uma tabela de pares chave/valor;
- Os dados do arquivo são armazenados em registros com chaves (índices) associados a eles, e podem ser recuperados usando essas chaves, como em um banco de dados relacional;
- O armazenamento e busca de dados nesse tipo de arquivo costuma ser muito rápido, dispensando bancos de dados para a construção de aplicações mais simples;
- Isso ocorre porque o próprio núcleo do sistema implementa os mecanismos de acesso e indexação do arquivo.

4. Compartilhamento de arquivos

Em um sistema multitarefas, é frequente ter arquivos acessados por mais de um processo, ou mesmo mais de um usuário;

O acesso simultâneo a recursos compartilhados pode gerar condições de disputa (ou corrida) que levam à inconsistência de dados e outros problemas;

O acesso concorrente em leitura a um arquivo não acarreta problemas;

No entanto, a possibilidade de escritas e leituras concorrentes pode levar a condições de disputa, precisando ser prevista e tratada de forma adequada.

4. Compartilhamento de arquivos

A solução mais simples e mais frequentemente utilizada para gerenciar o acesso concorrente a arquivos é o uso de travas de exclusão mútua (mutex locks);

A maioria dos sistemas operacionais oferece algum mecanismo de sincronização para o acesso a arquivos, na forma de uma ou mais travas (locks) associadas a cada arquivo aberto;

A sincronização pode ser feita sobre o arquivo inteiro ou sobre algum trecho específico dele, permitindo que dois ou mais processos possam trabalhar em partes distintas de um arquivo sem conflitos;

Várias travas são oferecidas:

- Pelo sistema operacional;
- Por bibliotecas.

5. 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.

5. Semânticas de compartilhamento

5.1 Semântica imutável:

- Arquivos compartilhados não podem ser alterados (apenas lidos);
- Solução simples para garantir a consistência dos dados;
- Usada em alguns sistemas de arquivos distribuídos.
- Modo padrão nos sistemas Windows;

5.2 Semântica UNIX:

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

5. Semânticas de compartilhamento

5.3 Semântica de sessão:

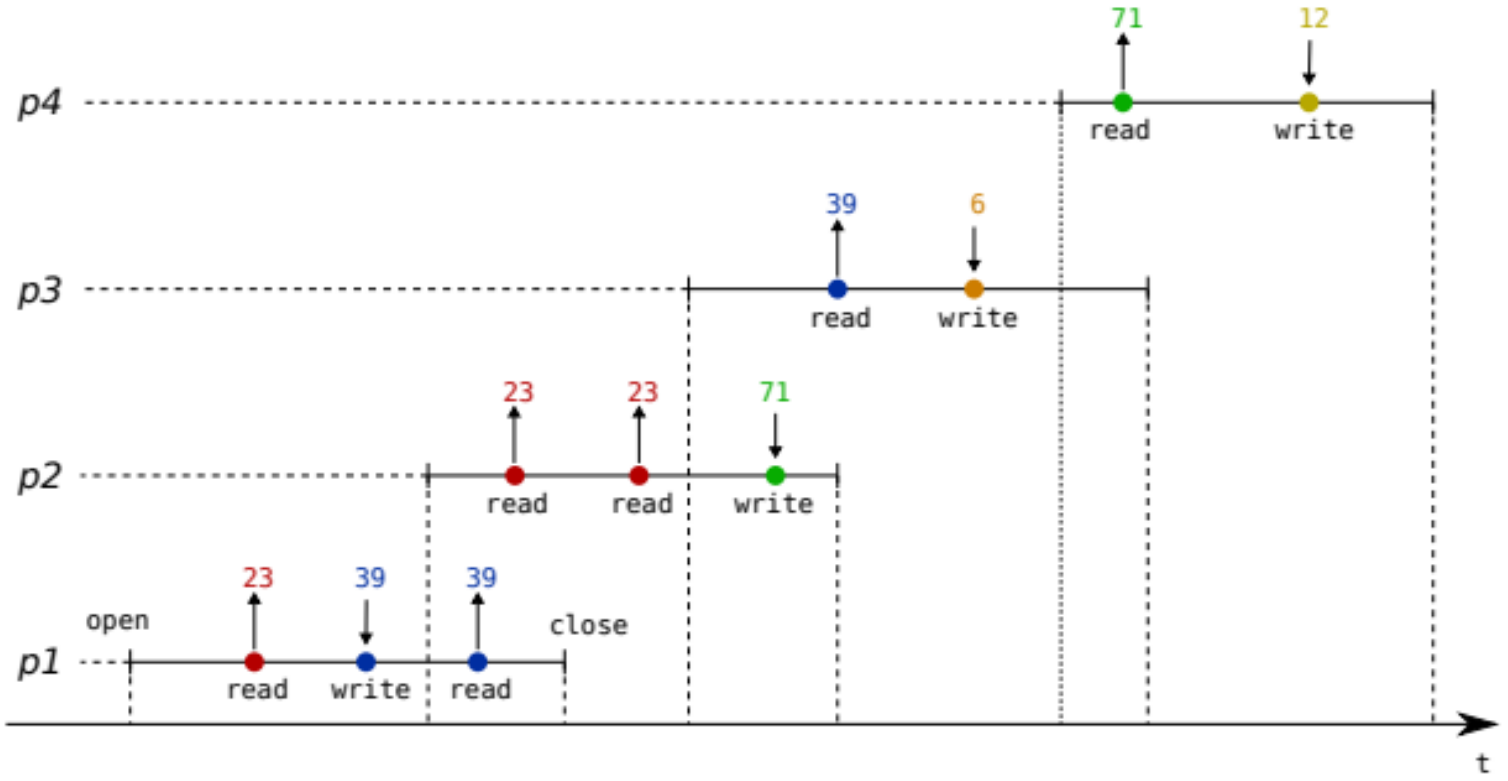
- Cada processo usa um arquivo em uma sessão;
- A sessão vai da abertura ao fechamento do arquivo;
- Modificações só são visíveis aos demais no fim da sessão;
- Sessões concorrentes podem ver conteúdos distintos (Problema!);
- Normalmente usada em sistemas de arquivos de rede.

5.4 Semântica de transação:

- Usa o conceito de transação dos SGBDs;
- É similar à semântica de sessão, mas aplicada a cada transação (sequência de operações) e não ao período completo de uso do arquivo (da abertura ao fechamento).

5. Semânticas de compartilhamento

Exemplo de semânica de sessão:



6. Controle de acesso

Trata de definir e controlar que usuários podem acessar que arquivos;

Atributos relevantes:

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

Implementação:

- Se dá pela definição de usuários e grupos;
- A definição é feita com listas de controle de acesso (ACL);
- O núcleo autoriza ou nega cada acesso.

6. Controle de acesso

Uma lista de controle de acesso (ACL) é basicamente uma lista indicando que usuários estão autorizados a acessar o arquivo, e como cada um pode acessá-lo;

Um exemplo conceitual de listas de controle de acesso a arquivos seria:

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

No entanto, essa abordagem se mostra pouco prática caso o sistema tenha muitos usuários e/ou arquivos, pois as listas podem ficar muito extensas e difíceis de gerenciar.

6. Controle de acesso

O UNIX usa uma abordagem mais simplificada para controle de acesso, que considera basicamente três tipos de usuários e três tipos de permissões:

- Usuários: o proprietário do arquivo (User), um grupo de usuários associado ao arquivo (Group) e os demais usuários (Others);
- Permissões: ler (Read), escrever (Write) e executar (eXecute).

Dessa forma, no UNIX são necessários apenas 9 bits para definir as permissões de acesso a cada arquivo ou diretório.

Exemplos:

```
1 host:~> ls -l
2 - rwx r-x --- 1 maziero prof    7248 2008-08-23 09:54 hello-unix
3 - rw- r-- r-- 1 maziero prof      54 2008-08-23 09:54 hello-unix.c
4 - rw- r-- r-- 1 maziero prof  195780 2008-09-26 22:08 main.pdf
5 - rw- --- --- 1 maziero prof   40494 2008-09-27 08:44 main.tex
```

7. Material para leitura

https://www.marcobehler.com/guides/java-files

https://www.baeldung.com/java-concurrent-locks