

# SISTEMAS OPERACIONAIS

## Gerenciamento de Processos - Projeto 1

### Turma: 04G11

Nome: Erik Samuel Viana Hsu  
TIA: 32265921

Nome: Mateus Kenzo lochimoto  
TIA: 32216289

Nome: Rodrigo Machado de Assis Oliveira de Lima  
TIA: 32234678

Nome: Thiago Shihan Cardoso Toma  
TIA: 32210744

## Relatório

### Cenário 1:

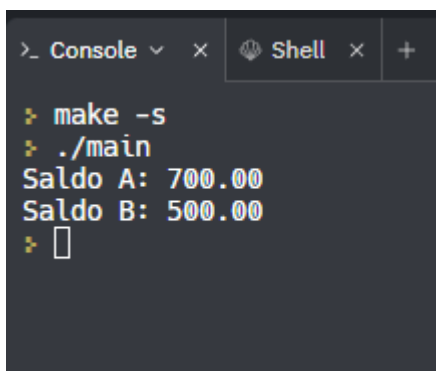
```
float valorAB = 50; //50 reais de A para B  
float valorBA = 200; //200 reais de B para A
```

Valores de transferência utilizados de A para B e de B para A

### Testes para saldo de A e B = 500

Saldo final esperado: A = 650; B = 350

Saldo final obtido:



```
>_ Console x Shell x +  
✚ make -s  
✚ ./main  
Saldo A: 700.00  
Saldo B: 500.00  
✚
```

Saldo final esperado: A = 650; B = 350

Saldo final obtido:

```
> Console x Shell x +
> make -s
> ./main
Saldo A: 500.00
Saldo B: 550.00
> 
```

### Teste para saldo de A e B = 300

Saldo final esperado: A = 450; B = 150

Saldo final obtido:

```
> Console x Shell x
> make -s
> ./main
Saldo A: 300.00
Saldo B: 350.00
> 
```

### Testes para saldo de A e B = 200

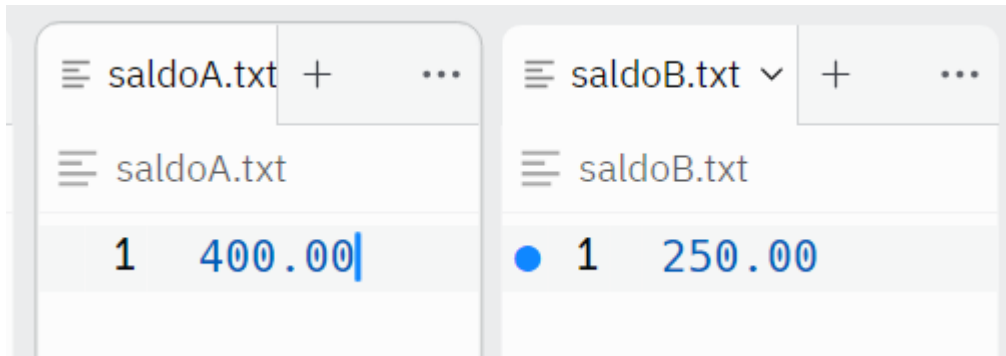
Saldo final esperado: A = 350; B = 50

Saldo final obtido:

saldoA.txt	+	...	saldoB.txt	+	...
saldoA.txt			saldoB.txt		
1	150.00		1	250.00	

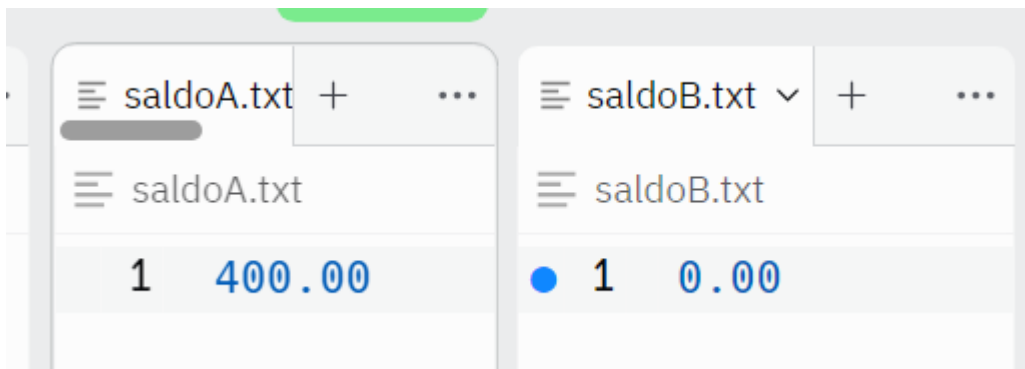
Saldo final esperado: A = 350; B = 50

Saldo final obtido:



Saldo final esperado: A = 350; B = 50

Saldo final obtido:



Explicação:

As duas threads acessam os dois arquivos em momentos diferentes, o que causa um desencontro das informações, ou seja, quando o cliente A acessa o seu saldo e o altera, o cliente B também faz a mesma coisa, ou até em tempos diferentes, o que pode fazer com que o cliente A grave o novo saldo de B quando ele ainda está lendo e o alterando, e assim gerando alterações erradas e desatualizadas.

**Cenário 2:**

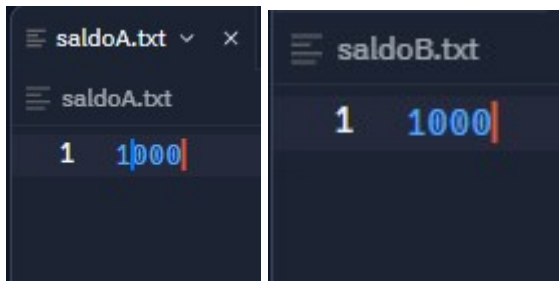
**Valores de transferências:**

```
int main(void) {  
    //declarando valores de transferencias  
    float valorAB = 50; //50 reais de A para B  
    float valorBA = 200; //200 reais de B para A
```

**Testes para saldo de A e B = 1000**

Saldo final esperado: A = 1150; B = 850

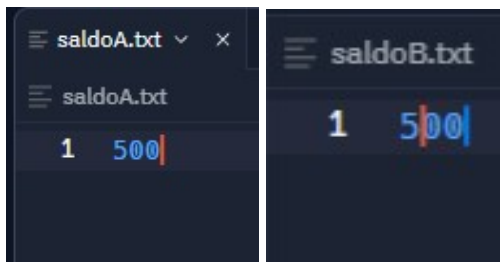
Saldo final obtido:



### Testes para saldo de A e B = 500

Saldo final esperado: A = 650; B = 350

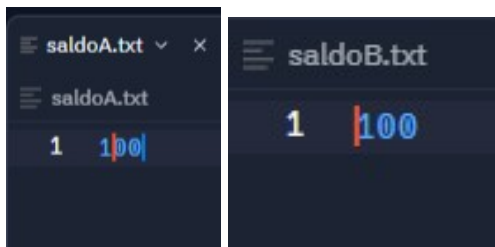
Saldo final obtido:



### Testes para saldo de A e B = 100

Saldo final esperado: A = 250; B = -50

Saldo final obtido:



Explicação:

Nesse cenário ocorre um deadlock, ou seja, ambas as transferências acabam travadas, pois cada uma delas precisa utilizar um recurso que está simultaneamente sendo travado pela outra. No caso do nosso exemplo o cliente A precisa do acesso ao saldo do cliente B para que a transferência AB ocorra, porém o mesmo está sendo travado pelo próprio cliente B, que por sua vez espera pelo acesso ao saldo do cliente A (que só seria liberado ao final da transferência AB), dessa forma ambos os processos ficam travados e não ocorre mudança nos saldos.

### Cenário 3:

Valores de transferências:

```

v int main(void) {
    //declarando valores de transferencias
    float valorAB = 50; //50 reais de A para B
    float valorBA = 200; //200 reais de B para A

```

### Testes para saldo de A e B = 1000

Saldo final esperado: A = 1150; B = 850

Saldo final obtido:

saldoA.txt	saldoB.txt
1 1150.00	1 850.00

### Valores de transferências:

```

//declarando valores de transferencias
float valorAB = 300; //valor em reais de A para B
float valorBA = 100; //valor em reais de B para A

```

### Testes para saldo de A e B = 1000

Saldo final esperado: A = 800; B = 1200

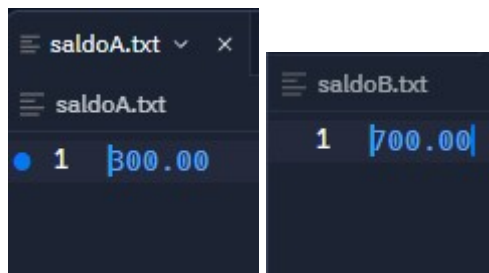
Saldo final obtido:

saldoA.txt	saldoB.txt
1 800.00	1 1200.00

### Testes para saldo de A e B = 500

Saldo final esperado: A = 300; B = 700

Saldo final obtido:



Explicação:

Para resolver o problema de deadlock encontrado no cenário 2, criamos um semáforo auxiliar que é travado assim que qualquer uma das threads se inicia e é destravado apenas ao final da execução das threads. Isso faz com que a outra thread espere a conclusão da primeira thread evitando assim o problema do deadlock e do desencontro de informações e obtendo o resultado esperado das transferências.