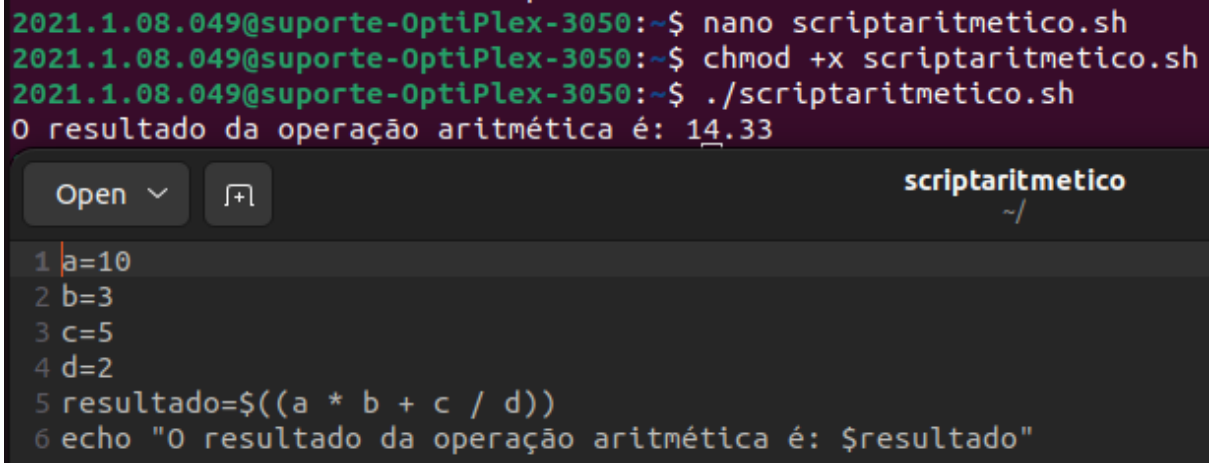


1) Crie um script chamado `scriptaritmetico`, com uma operação aritmética arbitrária usando pelo menos 4 variáveis, realizando uma operação de divisão cujo resultado não seja um número inteiro. Execute o script e mostre o resultado. Qual o recurso a ser utilizado caso você queira que o valor não inteiro apareça no resultado? Qual variável eu uso para isso?

```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano scriptaritmetico.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x scriptaritmetico.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./scriptaritmetico.sh
O resultado da operação aritmética é: 14.33
```



```
1 a=10
2 b=3
3 c=5
4 d=2
5 resultado=$((a * b + c / d))
6 echo "O resultado da operação aritmética é: $resultado"
```

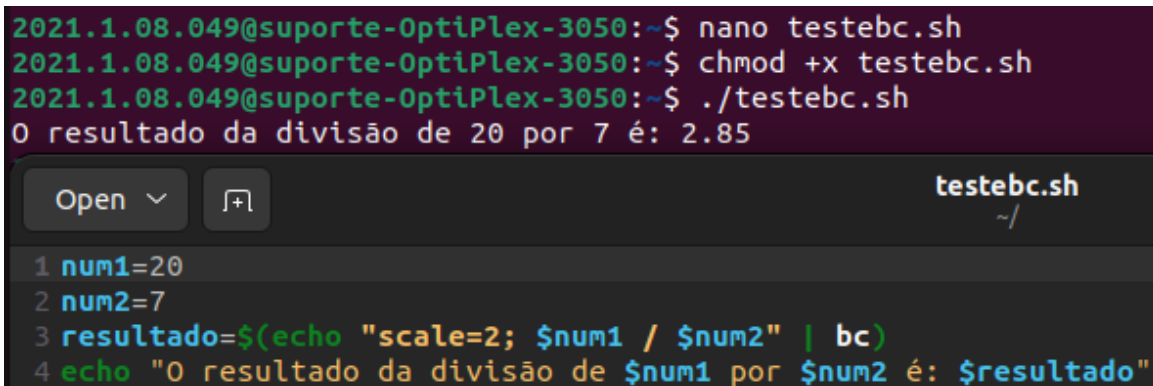
Para que o valor não inteiro apareça no resultado é necessário utilizar o comando `bc` com a opção `-l` para especificar a precisão decimal usando a variável `scale`.

2) Ponha em execução a calculadora `bc`. Mostre o uso da variável `scale`, exibindo um resultado de operação aritmética com 6 casas decimais.

```
2021.1.08.049@suporte-OptiPlex-3050:~$ echo "scale=6; 10 / 3" | bc
3.333333
2021.1.08.049@suporte-OptiPlex-3050:~$
```

3) Crie um script simples chamado `testebc`, em que você utilize a calculadora `bc` dentro dele, envolvendo o uso de algumas variáveis e a operação de divisão, com o direcionamento via pipe. Execute o script, mostrando o resultado.

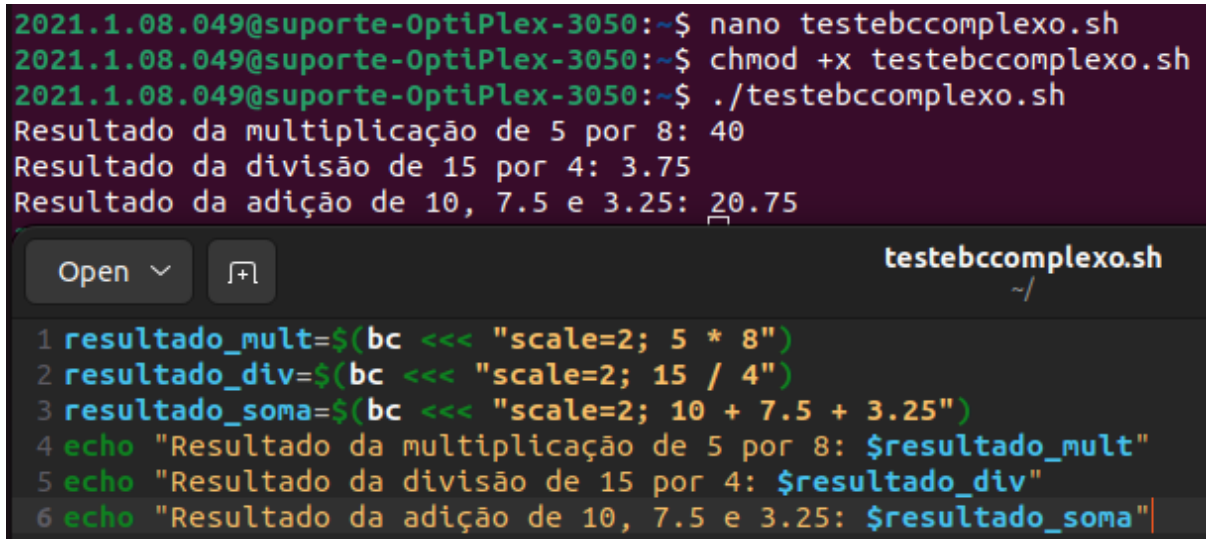
```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano testebc.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x testebc.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./testebc.sh
O resultado da divisão de 20 por 7 é: 2.85
```



```
1 num1=20
2 num2=7
3 resultado=$(echo "scale=2; $num1 / $num2" | bc)
4 echo "O resultado da divisão de $num1 por $num2 é: $resultado"
```

4) Crie um script chamado `testebccomplexo`, em que você utilize operações aritméticas diversas com a calculadora `bc` (pelo menos duas), armazenando os resultados em variáveis, como mostrado na aula. Neste caso, utilize a técnica de redirecionamento de entrada inline. Execute o script, mostrando o resultado.

```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano testebccomplexo.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x testebccomplexo.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./testebccomplexo.sh
Resultado da multiplicação de 5 por 8: 40
Resultado da divisão de 15 por 4: 3.75
Resultado da adição de 10, 7.5 e 3.25: 20.75
```



```
1 resultado_mult=$(bc <<< "scale=2; 5 * 8")
2 resultado_div=$(bc <<< "scale=2; 15 / 4")
3 resultado_soma=$(bc <<< "scale=2; 10 + 7.5 + 3.25")
4 echo "Resultado da multiplicação de 5 por 8: $resultado_mult"
5 echo "Resultado da divisão de 15 por 4: $resultado_div"
6 echo "Resultado da adição de 10, 7.5 e 3.25: $resultado_soma"
```

5) O que consiste o status de saída de um programa? Mostre um exemplo de execução de dois comandos (um com sucesso e outro desconhecido) e verifique esse status. Mostre em tela.

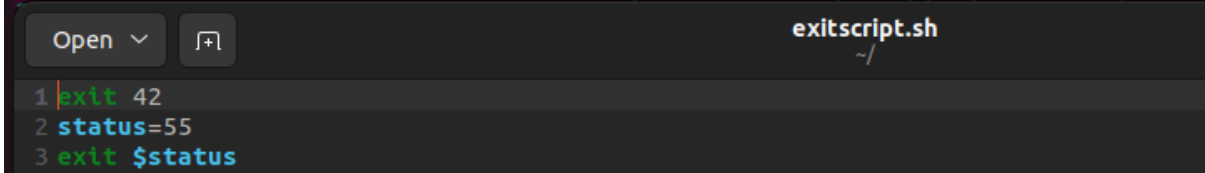
O status de saída de um programa é um valor numérico retornado pelo programa ao final de sua execução. Esse valor é uma forma de comunicação entre o programa e o ambiente em que foi executado, indica se a execução ocorreu com sucesso ou se ocorreu algum erro durante o processo.

```
2021.1.08.049@suporte-OptiPlex-3050:~$ ls
arquivo.txt  donuteptrato.blend  ls_output_20240430_122911.txt  numeros.txt  R  script.sh  testcrases.sh  testebc.sh
Desktop      Downloads            ls_output_20240430_122914.txt  Pictures     R  scriptaritmetico  snap  teste  testevariaveisambiente.sh
Documents    ls_output_20240430_122907.txt  Music                          Public       R  scriptaritmetico.sh  Templates  testebccomplexo.sh  Videos
2021.1.08.049@suporte-OptiPlex-3050:~$ echo "O status de saída do comando 'ls' é: $?"
O status de saída do comando 'ls' é: 0
2021.1.08.049@suporte-OptiPlex-3050:~$ comando_desconhecido
comando desconhecido: command not found
2021.1.08.049@suporte-OptiPlex-3050:~$ echo "O status de saída do comando desconhecido é: $?"
O status de saída do comando desconhecido é: 127
```

6) Qual a função do comando `exit`? Mostre um exemplo do uso do comando `exit` dentro de um script, mudando o valor padrão do status de saída. Mostre tanto o uso do `exit` exibindo um número qualquer até 255, quanto o valor de uma variável que você utilize no script. Execute o script e mostre o valor do status de saída em cada caso.

O comando `exit` é usado para encerrar a execução de um script shell e definir o status de saída que será retornado ao ambiente em que o script foi chamado.

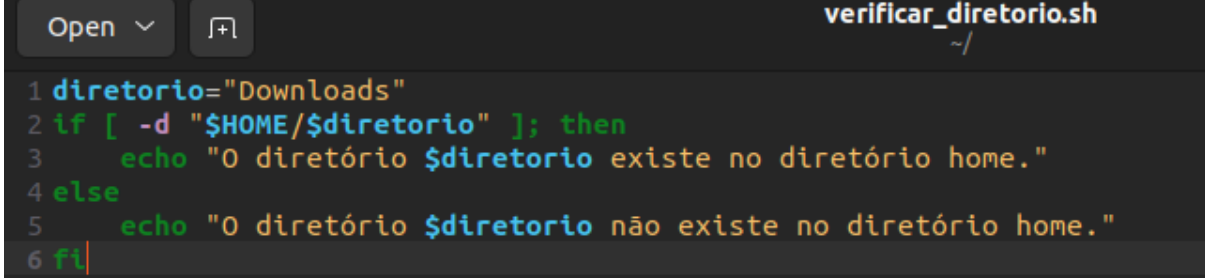
```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano exitscript.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x exitscript.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./exitscript.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ echo "O status de saída do script é: $?"
O status de saída do script é: 42
```



```
1 exit 42
2 status=55
3 exit $status
```

7) Crie um script simples envolvendo comandos condicionais if then else, para verificar a existência de um diretório específico no seu home. Primeiro procure um diretório inexistente, depois um diretório existente e exiba as mensagens específicas de acordo com o resultado. Execute o script e mostre em tela.

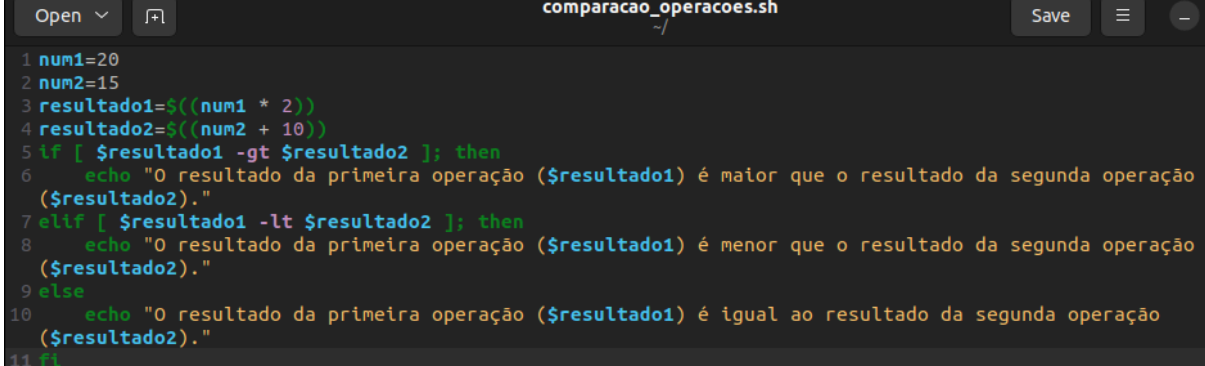
```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano verificar_diretorio.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x verificar_diretorio.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./verificar_diretorio.sh
O diretório diretorio_inexistente não existe no diretório home.
2021.1.08.049@suporte-OptiPlex-3050:~$ ./verificar_diretorio.sh
O diretório Downloads existe no diretório home.
```



```
1 diretorio="Downloads"
2 if [ -d "$HOME/$diretorio" ]; then
3     echo "O diretório $diretorio existe no diretório home."
4 else
5     echo "O diretório $diretorio não existe no diretório home."
6 fi
```

8) Crie um script envolvendo várias condicionais usando a estrutura if then elif else, fazendo duas operações aritméticas arbitrárias, verificando o valor das variáveis que armazenam essa operação, checando se o valor da primeira é maior, menor ou igual ao valor da segunda. Execute o script e mostre o resultado em tela.

```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano comparacao_operacoes.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x comparacao_operacoes.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./comparacao_operacoes.sh
O resultado da primeira operação (40) é maior que o resultado da segunda operação (25).
```



```
1 num1=20
2 num2=15
3 resultado1=$((num1 * 2))
4 resultado2=$((num2 + 10))
5 if [ $resultado1 -gt $resultado2 ]; then
6     echo "O resultado da primeira operação ($resultado1) é maior que o resultado da segunda operação ($resultado2)."
7 elif [ $resultado1 -lt $resultado2 ]; then
8     echo "O resultado da primeira operação ($resultado1) é menor que o resultado da segunda operação ($resultado2)."
9 else
10    echo "O resultado da primeira operação ($resultado1) é igual ao resultado da segunda operação ($resultado2)."
11 fi
```

9) Crie um script envolvendo condicionais usando a estrutura if then else, criando duas variáveis string arbitrárias e verificando seus valores, checando se o conteúdo das variáveis é igual. Execute o script e mostre o resultado em tela.

```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano verificar_strings.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x verificar_strings.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./verificar_strings.sh
0 conteúdo das variáveis é diferente.
```

```
1 string1="bom"
2 string2="dia"
3 if [ "$string1" = "$string2" ]; then
4     echo "0 conteúdo das variáveis é igual."
5 else
6     echo "0 conteúdo das variáveis é diferente."
7 fi
```

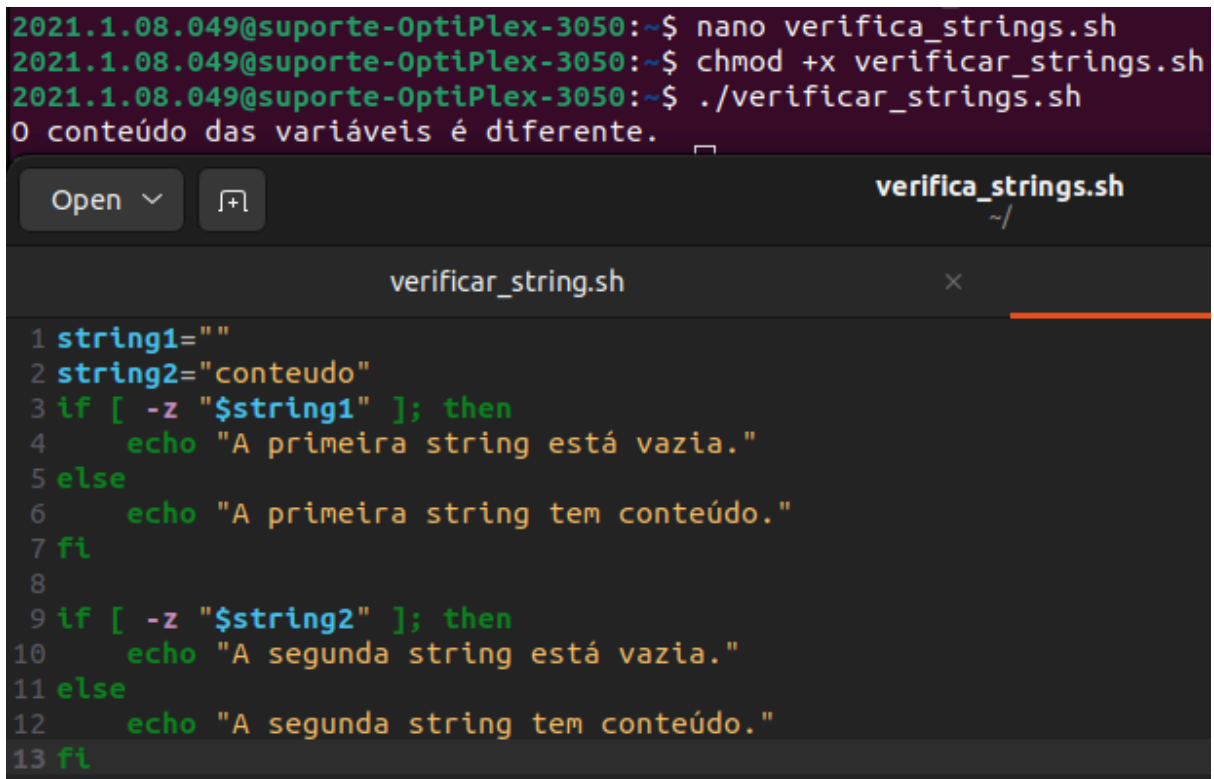
10) Crie um script envolvendo condicionais usando a estrutura if then else, criando uma string com um conteúdo, verificando se seu valor é “fruta”. Execute o script e mostre o resultado em tela.

```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano verificar_string.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x verificar_string.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./verificar_string.sh
0 valor da variável é 'fruta'.
```

```
1 palavra="fruta"
2 if [ "$palavra" = "fruta" ]; then
3     echo "0 valor da variável é 'fruta'."
4 else
5     echo "0 valor da variável não é 'fruta'."
6 fi
```

11) Crie um script envolvendo condicionais usando a estrutura if then else, criando duas strings, uma vazia, outra com conteúdo e verificando estes resultados (se tem conteúdo em ambos os casos).

```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano verifica_strings.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x verificar_strings.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./verificar_strings.sh
0 conteúdo das variáveis é diferente.
```

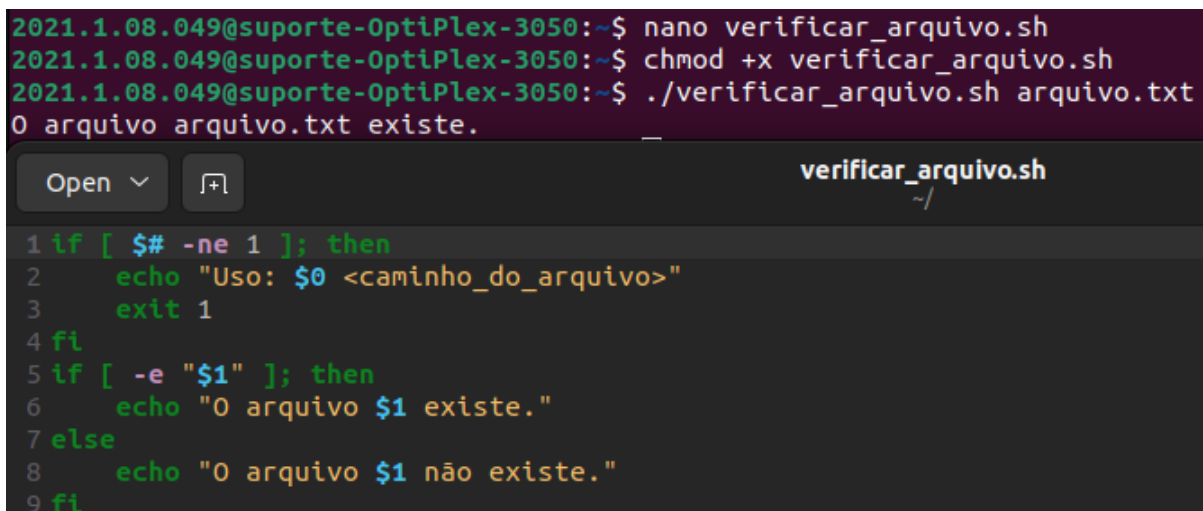


```
1 string1=""
2 string2="conteudo"
3 if [ -z "$string1" ]; then
4     echo "A primeira string está vazia."
5 else
6     echo "A primeira string tem conteúdo."
7 fi
8
9 if [ -z "$string2" ]; then
10    echo "A segunda string está vazia."
11 else
12    echo "A segunda string tem conteúdo."
13 fi
```

12) Cite 5 opções de comparações envolvendo arquivos. Escolha uma das opções e crie um script envolvendo essa opção.

As 5 opções de comparações envolvendo arquivos são: -e arquivo, -f arquivo, -d arquivo, -r arquivo, -s arquivo

```
2021.1.08.049@suporte-OptiPlex-3050:~$ nano verificar_arquivo.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ chmod +x verificar_arquivo.sh
2021.1.08.049@suporte-OptiPlex-3050:~$ ./verificar_arquivo.sh arquivo.txt
0 arquivo arquivo.txt existe.
```



```
1 if [ $# -ne 1 ]; then
2     echo "Uso: $0 <caminho_do_arquivo>"
3     exit 1
4 fi
5 if [ -e "$1" ]; then
6     echo "O arquivo $1 existe."
7 else
8     echo "O arquivo $1 não existe."
9 fi
```