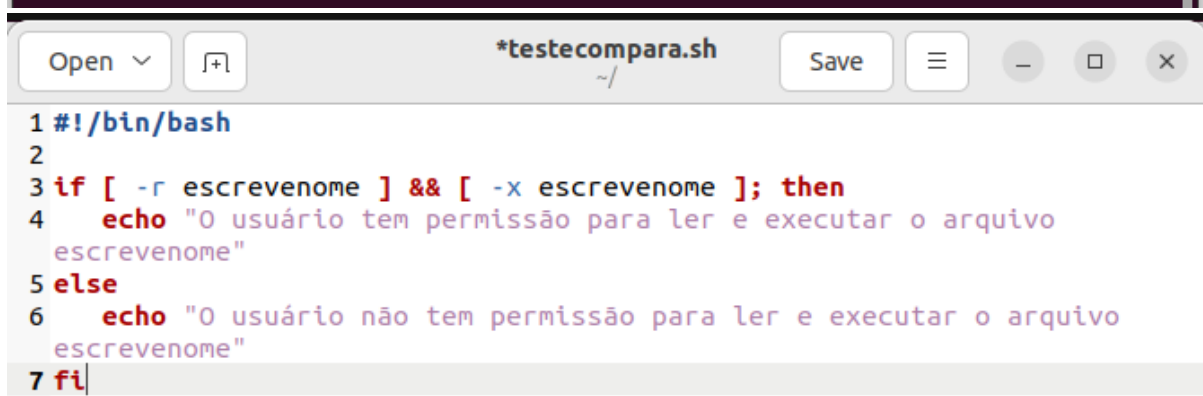


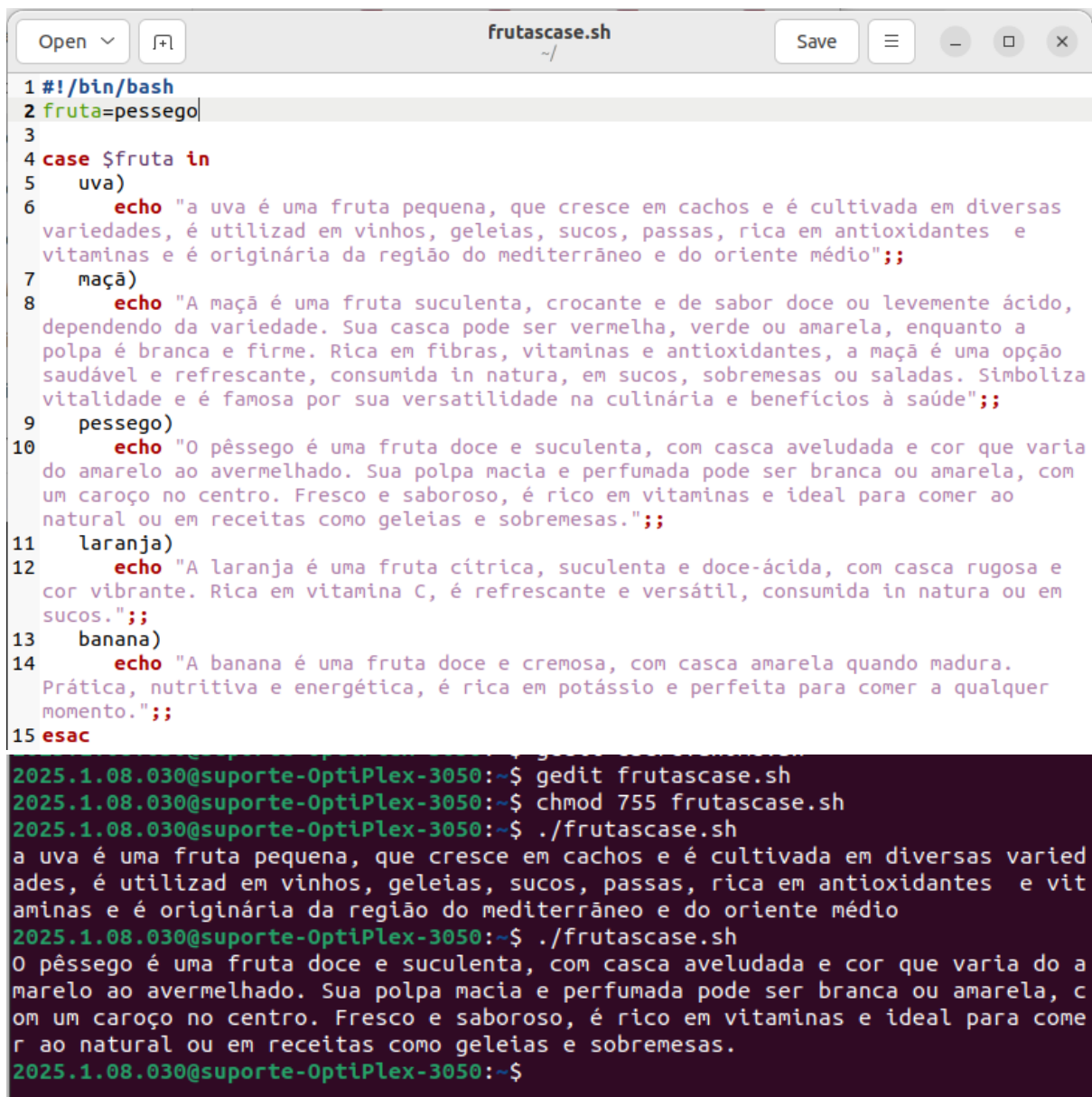
1) Crie um script chamado escrevenome, faça com que a saída desse script seja seu nome completo. Não utilize o comando chmod. Depois crie um script chamado testecompara, utilize o operador AND e verifique se o usuário logado tem permissão r e x sobre o script escrevenome. Mostre o resultado da saída.

```
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit escrevenome.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit testecompara.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ chmod 755 testecompara.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ ./testecompara.sh
O usuário não tem permissão para ler e executar o arquivo escrevenome
2025.1.08.030@suporte-OptiPlex-3050:~$
```



```
1 #!/bin/bash
2
3 if [ -r escrevenome ] && [ -x escrevenome ]; then
4     echo "O usuário tem permissão para ler e executar o arquivo
    escrevenome"
5 else
6     echo "O usuário não tem permissão para ler e executar o arquivo
    escrevenome"
7 fi
```

2) Crie um script chamado frutascase. Com base no valor da variável fruta mostre uma breve descrição da fruta. Faça com 5 frutas. Exemplo: fruta=uva, echo “A uva é o fruto da videira ou parreira, uma planta da família Vitaceae. É originária da Ásia e uma das frutas mais antigas utilizadas na alimentação humana. Existem mais de 60 mil variedades da fruta. A cor, o sabor e o tamanho variam de acordo com cada espécie. A uva também é classificada quanto ao destino de produção, de mesa ou para vinicultura. Pode ser consumida in natura ou usada na preparação de doce, vinho, passas, mussels, geléias, tortas, gelatinas, sucos.

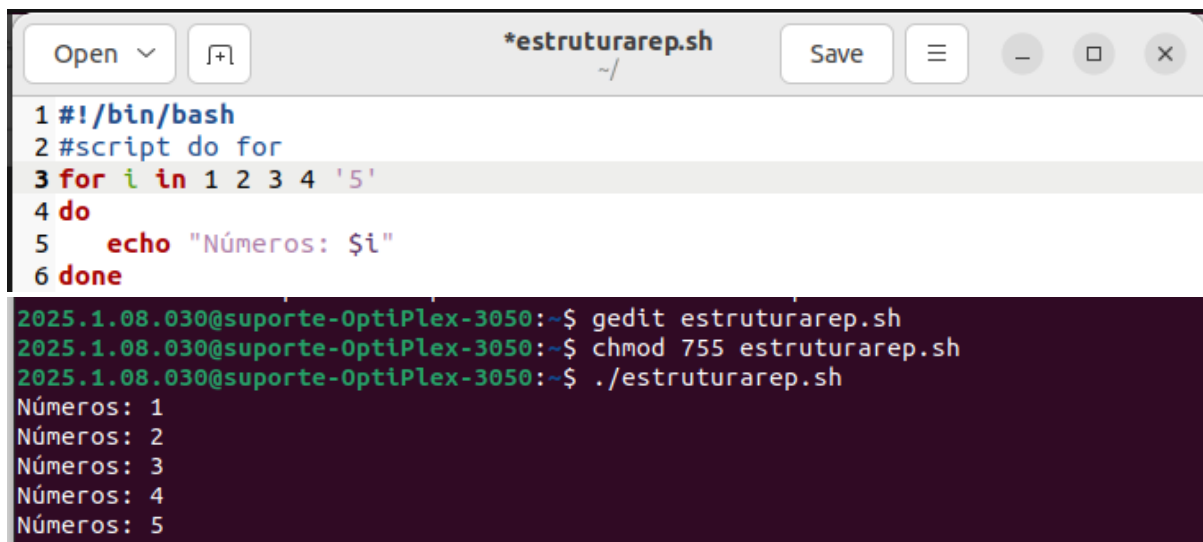


```
1 #!/bin/bash
2 fruta=pessego
3
4 case $fruta in
5     uva)
6         echo "a uva é uma fruta pequena, que cresce em cachos e é cultivada em diversas
        variedades, é utilizad em vinhos, geleias, sucos, passas, rica em antioxidantes e vit
        aminas e é originária da região do mediterrâneo e do oriente médio";;
7     maçã)
8         echo "A maçã é uma fruta succulenta, crocante e de sabor doce ou levemente ácido,
        dependendo da variedade. Sua casca pode ser vermelha, verde ou amarela, enquanto a
        polpa é branca e firme. Rica em fibras, vitaminas e antioxidantes, a maçã é uma opção
        saudável e refrescante, consumida in natura, em sucos, sobremesas ou saladas. Simboliza
        vitalidade e é famosa por sua versatilidade na culinária e benefícios à saúde";;
9     pessego)
10        echo "O pêssego é uma fruta doce e succulenta, com casca aveludada e cor que varia
        do amarelo ao avermelhado. Sua polpa macia e perfumada pode ser branca ou amarela, com
        um caroço no centro. Fresco e saboroso, é rico em vitaminas e ideal para comer ao
        natural ou em receitas como geleias e sobremesas.";;
11    laranja)
12        echo "A laranja é uma fruta cítrica, succulenta e doce-ácida, com casca rugosa e
        cor vibrante. Rica em vitamina C, é refrescante e versátil, consumida in natura ou em
        sucos.";;
13    banana)
14        echo "A banana é uma fruta doce e cremosa, com casca amarela quando madura.
        Prática, nutritiva e energética, é rica em potássio e perfeita para comer a qualquer
        momento.";;
15 esac
```

```
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit frutascase.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ chmod 755 frutascase.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ ./frutascase.sh
a uva é uma fruta pequena, que cresce em cachos e é cultivada em diversas varied
ades, é utilizad em vinhos, geleias, sucos, passas, rica em antioxidantes e vit
aminas e é originária da região do mediterrâneo e do oriente médio
2025.1.08.030@suporte-OptiPlex-3050:~$ ./frutascase.sh
O pêssego é uma fruta doce e succulenta, com casca aveludada e cor que varia do a
marelo ao avermelhado. Sua polpa macia e perfumada pode ser branca ou amarela, c
om um caroço no centro. Fresco e saboroso, é rico em vitaminas e ideal para come
r ao natural ou em receitas como geleias e sobremesas.
2025.1.08.030@suporte-OptiPlex-3050:~$
```

3) Cite, explique e faça um script simples para cada estrutura de repetição do shell bash. Use sua criatividade para os scripts.

No shell temos três estruturas de repetição o for, while e until. O for é utilizado pois ele percorre por cada item listado, como exemplo:



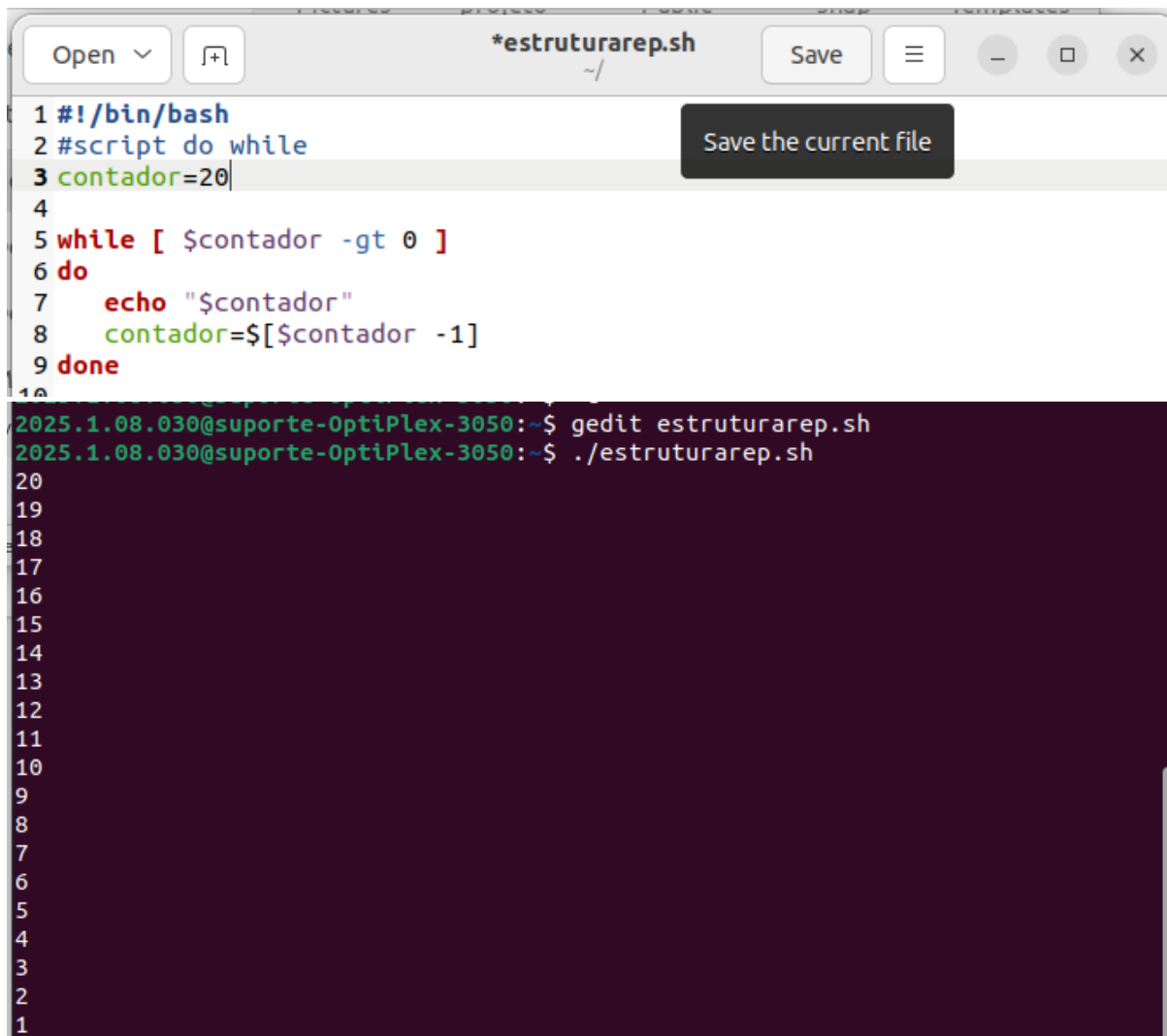
The screenshot shows a gedit editor window titled `*estruturarep.sh` with a file icon and a 'Save' button. The script content is as follows:

```
1 #!/bin/bash
2 #script do for
3 for i in 1 2 3 4 '5'
4 do
5     echo "Números: $i"
6 done
```

Below the editor, a terminal window shows the execution of the script:

```
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit estruturarep.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ chmod 755 estruturarep.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ ./estruturarep.sh
Números: 1
Números: 2
Números: 3
Números: 4
Números: 5
```

O while é utilizado pois ele percorre a condição até que ela deixe de ser verdadeira, como por exemplo:



The screenshot shows a gedit editor window titled `*estruturarep.sh` with a file icon, a 'Save' button, and a 'Save the current file' tooltip. The script content is as follows:

```
1 #!/bin/bash
2 #script do while
3 contador=20
4
5 while [ $contador -gt 0 ]
6 do
7     echo "$contador"
8     contador=$((contador - 1))
9 done
```

Below the editor, a terminal window shows the execution of the script:

```
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit estruturarep.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ ./estruturarep.sh
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
```

O until é usado pois ele percorre a condição até que a condição seja verdadeira, como por exemplo:

```
2025.1.08.030@suporte-OptiPlex-3050:~$ ./estruturarep.sh
./estruturarep.sh: line 3: [: -gt: unary operator expected
Contagem:
Contagem: 1
Contagem: 2
Contagem: 3
```

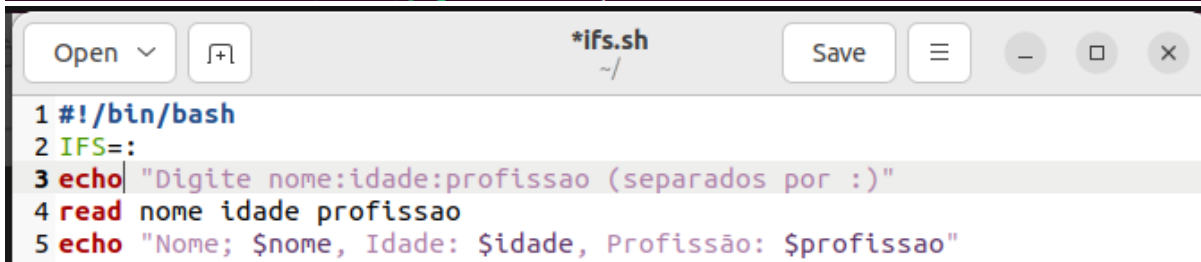
A screenshot of a code editor window titled '*estruturarep.sh' with a file icon and a '~/' symbol. The editor contains a shell script with 7 lines: 1. '#!/bin/bash', 2. '#script do until', 3. 'until [\$contador -gt 3];', 4. 'do', 5. ' echo "Contagem: \$contador"', 6. ' contador=\$((contador + 1))', 7. 'done'. The script is displayed with syntax highlighting: blue for comments, red for keywords, green for variables, and black for other text.

```
1 #!/bin/bash
2 #script do until
3 until [ $contador -gt 3 ];
4 do
5     echo "Contagem: $contador"
6     contador=$((contador + 1))
7 done
```

4) Explique o que é IFS e faça um script diferente do que foi visto em aula. Use sua criatividade.

O IFS é uma variável que define os limitadores de campo para o shell.

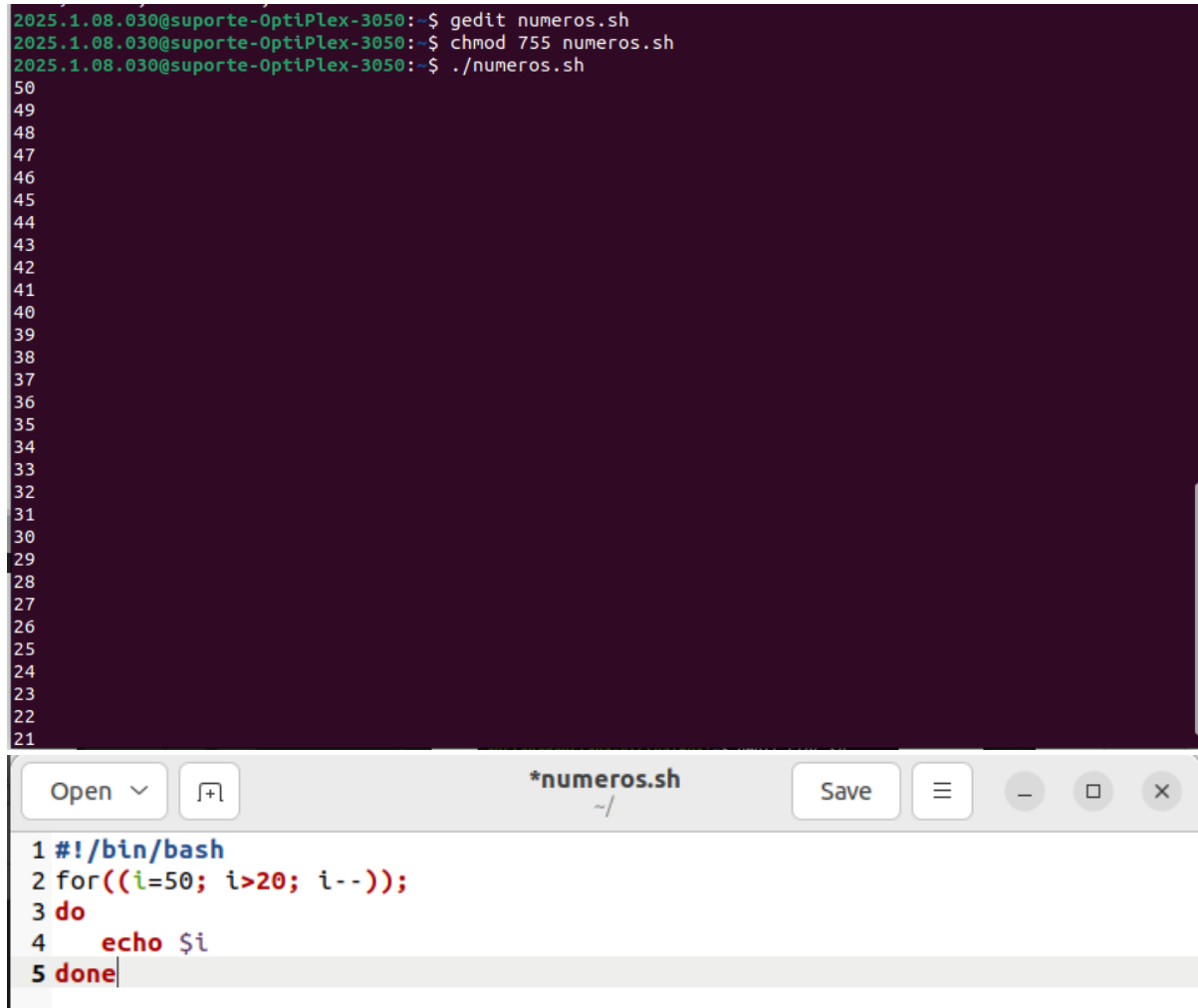
```
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit estruturarep.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit ifs.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ chmod 755 ifs.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ ./ifs.sh
Digite nome:idade:profissao (separados por :)
Pedro:25:barbeiro
Nome; Pedro, Idade: 25, Profissão: barbeiro
```

A screenshot of a code editor window titled '*ifs.sh' with a file icon and a '~/' symbol. The editor contains a shell script with 5 lines: 1. '#!/bin/bash', 2. 'IFS=: ', 3. 'echo "Digite nome:idade:profissao (separados por :)"', 4. 'read nome idade profissao', 5. 'echo "Nome; \$nome, Idade: \$idade, Profissão: \$profissao"'. The script is displayed with syntax highlighting: blue for comments, green for variables, red for keywords, and black for other text.

```
1 #!/bin/bash
2 IFS=: 
3 echo "Digite nome:idade:profissao (separados por :)"
4 read nome idade profissao
5 echo "Nome; $nome, Idade: $idade, Profissão: $profissao"
```

5) Crie um script for no estilo C que mostre na tela os números de 50 a 20.

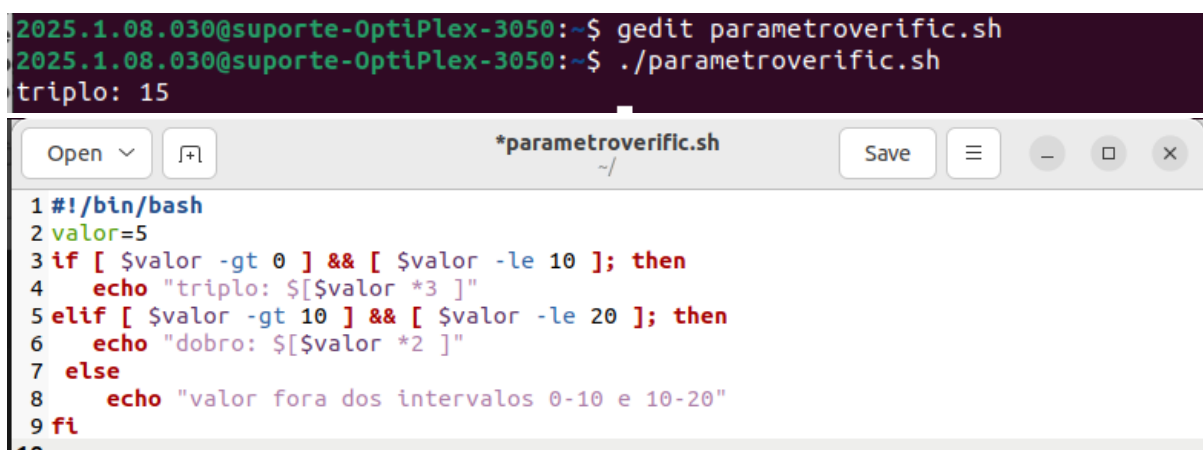
```
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit numeros.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ chmod 755 numeros.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ ./numeros.sh
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
```



```
1 #!/bin/bash
2 for((i=50; i>20; i--));
3 do
4     echo $i
5 done
```

6) Desenvolva um script que receba um parâmetro e verifique se o valor está entre 0 e 10. Caso sim mostre o triplo do valor. Caso ele esteja entre 10 e 20 mostre o dobro. Caso não esteja nos anteriores apresente uma mensagem.

```
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit parametroverific.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ ./parametroverific.sh
triplo: 15
```

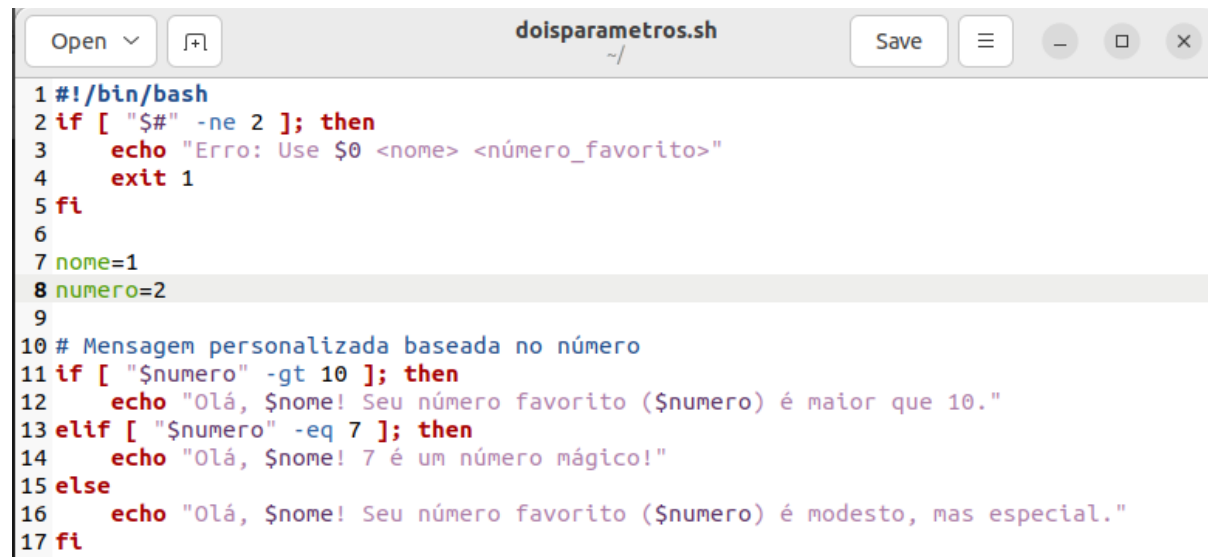


```
1 #!/bin/bash
2 valor=5
3 if [ $valor -gt 0 ] && [ $valor -le 10 ]; then
4     echo "triplo: ${valor * 3}"
5 elif [ $valor -gt 10 ] && [ $valor -le 20 ]; then
6     echo "dobro: ${valor * 2}"
7 else
8     echo "valor fora dos intervalos 0-10 e 10-20"
9 fi
10
```

7) Explique o que é \$# e faça um script diferente do que foi visto em aula. Faça com dois parâmetros. Use sua criatividade.

O \$# é uma variável especial que está disponível no bash, ela verifica o número de parâmetros passados para o script .

```
2025.1.08.030@suporte-OptiPlex-3050:~$ gedit doisparametros.sh
2025.1.08.030@suporte-OptiPlex-3050:~$ ./doisparametros.sh Victoria 13
Olá, 1! Seu número favorito (2) é modesto, mas especial.
```



```
1 #!/bin/bash
2 if [ "$#" -ne 2 ]; then
3     echo "Erro: Use $0 <nome> <número_favorito>"
4     exit 1
5 fi
6
7 nome=1
8 numero=2
9
10 # Mensagem personalizada baseada no número
11 if [ "$numero" -gt 10 ]; then
12     echo "Olá, $nome! Seu número favorito ($numero) é maior que 10."
13 elif [ "$numero" -eq 7 ]; then
14     echo "Olá, $nome! 7 é um número mágico!"
15 else
16     echo "Olá, $nome! Seu número favorito ($numero) é modesto, mas especial."
17 fi
```