

## TRABALHO

### Criação e Desenvolvimento de Scripts

**Artur José Gomes Pereira**

Nº 2040415

**João José da Costa Cabral**

Nº 2020919

### Tecnologias e Programação de Sistemas de Informação

**UNIDADE CURRICULAR:**  
Sistemas Operativos e Redes

**DOCENTE:**  
Milton Aguiar

**DATA:**

28 de Janeiro de 2020

# ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Cofinanciado por:



# Índice

---

Introdução.....	2
Shell Scripts .....	3
Comando CHOWN .....	8
Conclusão .....	10
WEBGRAFIA .....	11

# Introdução

---

Este relatório é referente ao trabalho solicitado pelo professor Milton Aguiar. Este trabalho consistia da realização de 5 Shell Scripts e da exploração de um comando Unix escolhido pelo grupo, que, neste caso, foi o "chown".

O primeiro Shell Script pergunta ao utilizador a sua nota numa disciplina e, dependendo da nota o programa devolveria se o aluno estaria aprovado ou reprovado na disciplina. O segundo Script procuraria os ficheiros ou diretórios referentes ao utilizador. O terceiro procuraria a existência de um utilizador através do ficheiro "passwd". O quarto pediria 2 números ao utilizador e calcularia, dependendo do que o mesmo pedisse, a soma dos números, subtração, multiplicação, divisão, e qual dos 2 números é maior. O ultimo Shell Script faz o backup dos dados do nosso user e os guarda dentro de uma determinada pasta, todas as terças feiras pelas 21 horas.

Relativo ao comando escolhido, temos de fazer uma apresentação sobre o mesmo e dar uns exemplos do seu funcionamento.

Este trabalho como um todo, tinha como objetivo utilizar conhecimentos adquiridos em sala de aula e aplicar na criação de Shell scripts.

# Shell Scripts

## Shell Script 1

A forma como desenvolvemos este foi utilizar um ciclo Until Do.

```
GNU nano 2.9.3 exercicioUM.sh
#!/bin/bash

i="1"
until [ "$i" = "0" ]
do
    echo "Qual é o teu nome ?"
    read nome
    echo "Ola, $nome"

    echo "Nome da disciplina?"
    read disciplina
    echo "Qual foi a nota?"
    read nota

    if [ $nota -ge "10" ]
    then
        echo "Aprovado"
    else
        echo "Reprovado"
    fi

    echo "Se queres sair escreve 0, senao carrega enter."
    read i
    if [ "$i" == "0" ];then
        echo "Obrigado"
    fi
done

[ 29 linhas lidas ]
^G Ajuda      ^O Gravar     ^W Procurar   ^K Cortar txt ^J Justificar  ^C Pos cursor M-U Desfazer
^X Sair        ^R Carregar   ^_ Substituir ^U Repor txt  ^T Limpar     ^I Ir p/ linha M-E Refazer
```

A forma como este programa funciona é basicamente ele pede a nota ao utilizador e guarda esse valor dentro uma variável chamada "nota". Essa variável entra num if onde é comparada com o valor 10. Se o valor maior ou igual a 10 é exibido a mensagem "Aprovado" e se for inferior a 10 a mensagem "Reprovado" é exibida.

## Shell Script 2

```
#!/bin/bash

echo "Mostrar TODOS os diretorios de $USER ? - Selecione 1"

echo "Mostrar TODOS os ficheiros de $USER? - Selecione 2"
read encontrar

if [ $encontrar == "1" ];then
echo "${find /home/$USER -type d}"

elif [ $encontrar == "2" ];then
echo "${find /home/$USER -type f}"
fi
```

Este script pergunta ao utilizador se quer ver os ficheiros ou as pastas dentro do seu diretório. Dependendo da opção escolhida pelo utilizador o comando "find" vai ser executado para procurar tudo do tipo D (diretório) ou F(ficheiro)

## Shell Script 3

```
#!/bin/bash

echo "Qual é o user ?"
read i
confirma="$(grep -w $i /etc/passwd)"

if [ "$confirma" ]; then
    echo "$i existe"
else
    echo "$i não existe"
fi
```

Neste script o utilizador insere a variável do user a ser procurado. Essa variável é procurada na pasta "/etc/passwd" se existir indica o nome do user e que o mesmo existe, se não existir indica que não existe.

## Shell Script

```
#!/bin/bash
x="0"
y="0"

um="Introduza o primeiro numero"
dois="Introduza o segundo numero"

echo "1 - SOMA de dois numeros"
echo "2 - SUBTRACAO de dois numeros"
echo "3 - MULTIPLICACAO de dois numeros"
echo "4 - DIVISÃO de dois numeros"
echo "5 - MAIOR DE DOIS numeros"
read resposta
if [ "$resposta" == "1" ]; then
    echo "$um"
    read x
    echo "$dois"
    read y
    echo "Soma: $((x+y)) "

elif [ "$resposta" == "2" ]; then
    echo "$um"
    read x
    echo "$dois"
    read y
    echo "Subtração: $((x-y))"

elif [ "$resposta" == "3" ]; then
    echo "$um"
    read x

    echo "$dois"
    read y
    echo "Multiplicação: $((x*y))"

elif [ "$resposta" == "4" ]; then
    echo "$um"
    read x
    echo "$dois"
    read y
    echo "Divisão: $((x/y))"

elif [ "$resposta" == "5" ]; then
    echo "$um"
    read x
    echo "$dois"
    read y
    if [ $x -gt $y ]; then
        echo "0 numero maior é: $x"
    elif [ $y -gt $x ]; then
        echo "0 numero maior é: $y"
    else
        echo "0s numeros são iguais: $x e $y"
    fi
fi
fi
```

O quarto script basicamente pede ao utilizador 2 números que são guardados dentro de 2 variáveis. Depois um menu com as opções é exibido. A opção escolhida passa por um "if" onde os números são, relativos à opção, somados, subtraídos, multiplicados, divididos, ou comparados para saber qual o maior.

## Shell Script 5

```
#!/bin/bash

dir_orig="/home/sor"
dir_dest="/home/sor/backup"

data_dia=`date +%d`
data_mes=`date +%m`
data_ano=`date +%y`
data="$data_dia.$data_mes.$data_ano"

if [ -e $dir_orig ]; then
    echo "Diretório Origem já existe!"
else
    echo "Diretório não existe"
    exit 1
fi

dir_final="$dir_dest/bkp_$data/"

if [ -e $dir_final ]; then
    echo "Diretório Destino já existente!"
else
    echo "Diretório não existe. Será criado!"
    mkdir -p $dir_final
fi

for file in $(ls *.sh)
do
```



```

for file in $(ls *.sh)
do
    cp $file $dir_final
done

echo "Backup foi realizado"
echo "Diretório: ${dir_orig}"
echo "Diretório: ${dir_final}";
exit 0

```

```

GNU nano 2.9.3 /tmp/crontab.ctr7i0/crontab
Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
# m h dom mon dow   command
00 21 * * 2 /bin/sh /home/sor/ShellScripts/backup.sh

```

[ 24 linhas lidas ]

^C Ajuda    ^O Gravar    ^W Procurar    ^K Cortar txt    ^J Justificar    ^C Pos cursor    M-U Desfazer  
 ^X Sair    ^R Carregar    ^\_ Substituir    ^U Repor txt    ^T Ortografia    ^ Ir p/ linha    M-E Refazer

Os 2 primeiros prints são relativos ao script que faz uma copia de backup para dentro da pasta "backup". Basicamente, uma copia de todos os dados é guardada dentro da pasta "backup" com a data da copia.

A terceira print é o timer responsável por repetir o script todos as terças as 21 horas.



# Comando CHOWN

---

O comando `chown` existe em todas as variações de Unix. `Chown` é a abreviação de `change owner`, e serve para mudar o dono dos ficheiros, diretórios e links.

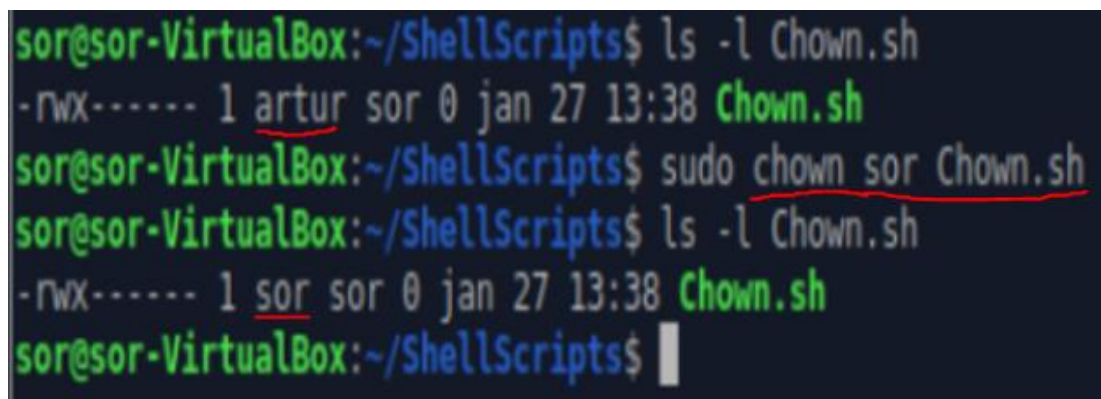
Linux é um sistema operativo desenhado para suportar múltiplos utilizadores em simultâneo, e por causa disso é necessário controlar o acesso aos ficheiros, e por isso um grupo de regras foram criadas chamadas de permissões.

As permissões são divididas em 3 principais grupos: User, Group, Others. Quando um ficheiro é criado, o criador é o User e o Group é o grupo atual do criador, os Others são os restantes utilizadores não pertencentes ao grupo.

A forma de usar este comando é

`"chown <<novo dono>> <<nome do ficheiro / diretório / link>>"`

Ex:



```
sor@sor-VirtualBox:~/ShellScripts$ ls -l Chown.sh
-rwx----- 1 artur sor 0 jan 27 13:38 Chown.sh
sor@sor-VirtualBox:~/ShellScripts$ sudo chown sor Chown.sh
sor@sor-VirtualBox:~/ShellScripts$ ls -l Chown.sh
-rwx----- 1 sor sor 0 jan 27 13:38 Chown.sh
sor@sor-VirtualBox:~/ShellScripts$
```

Neste exemplo vemos que o ficheiro "`Chown.sh`" é um ficheiro do utilizador "`artur`" e usando o comando `CHOWN`, da forma anteriormente dita, o dono do ficheiro mudou para o "`sor`"

EX:

```
sor@sor-VirtualBox:~/ShellScripts$ ls -l Chown.sh
-rwx----- 1 sor testingCHOWN 0 jan 27 13:38 Chown.sh
sor@sor-VirtualBox:~/ShellScripts$ sudo chown vanessa:contabilidade Chown.sh
sor@sor-VirtualBox:~/ShellScripts$ ls -l Chown.sh
-rwx----- 1 vanessa contabilidade 0 jan 27 13:38 Chown.sh
```

Neste exemplo vemos como utilizar tanto o dono como o grupo ao qual está associado o ficheiro.

# Conclusão

---

Após concluirmos os Shell scripts, chegamos a conclusão que a forma de ser eficiente é organização. Para cumprir os objetivos é necessária comunicação entre os membros, assim como a partilha de informação, apesar de algumas dificuldades iniciais em definir como íamos realizar os programas conseguimos realizar os scripts solicitados.

Finalizamos este relatório com a certeza de ter aumentado os nossos conhecimentos sobre Linux e sobre Shell Scripts, que com certeza será útil durante toda a nossa jornada futura. É também muito importante agradecer vividamente ao professor Milton Aguiar por todos os conhecimentos fornecidos durante o passado semestre.

# Webgrafia

---

<https://linuxize.com/post/how-to-list-users-in-linux/>

<https://linuxconfig.org/how-to-find-all-files-with-a-specific-text-using-linux-shell>

<https://www.cyberciti.biz/tips/find-out-if-file-exists-with-conditional-expressions.html>

<https://www.tecmint.com/linux-find-command-to-search-multiple-filenames-extensions>

<https://unix.stackexchange.com/questions/48492/list-only-regular-files-but-not-directories-in-current-directory>

<https://superuser.com/questions/336275/find-out-if-user-name-exists>

<https://www.thegeekstuff.com/2009/06/15-practical-crontab-examples/>