BES - Segurança da Informação

# Roteiro de Atividade Prática

*Segurança no Ambiente Linux*

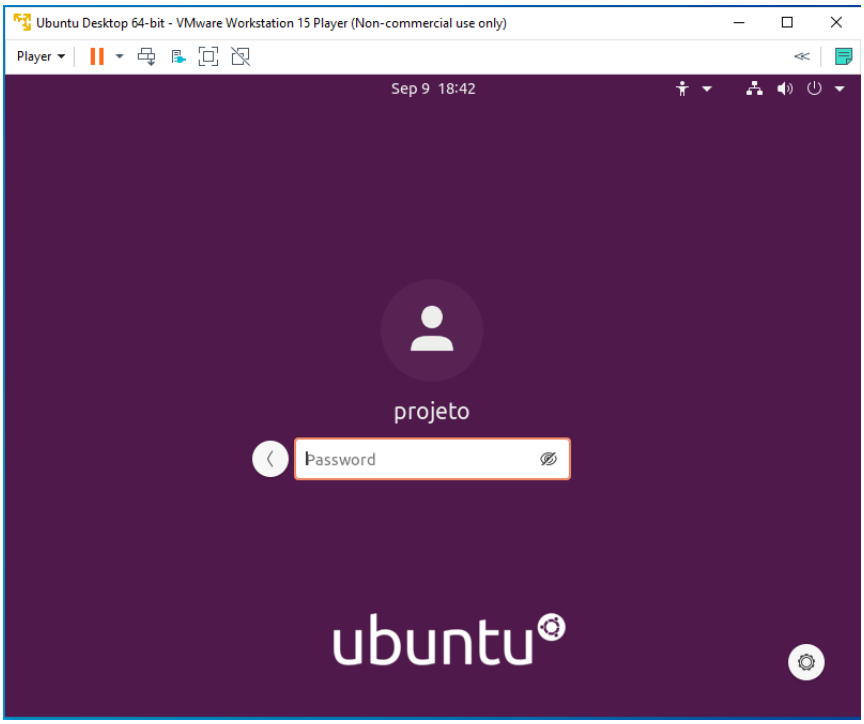
## Atividade Prática – Controle de Acesso a Arquivos

Este roteiro vai auxiliar o estudante na realização da primeira etapa da atividade prática. Nesta atividade iremos explorar o mecanismo de controle de acesso a arquivos no Linux, vamos apresentar primeiramente o modelo tradicional de controle de acesso, na sequência as listas de controle de acesso estentidas. Como pré-requisito para esta atividade, o estudante deve ter acesso a uma máquina com o sistema operacional Linux, sugerimos fortemente que você utilize uma máquina virtual com o Linux. Para auxiliar você neste processo, criamos um material passo a passo, caso tenha dúvidas como criar uma máquina virtual Linux consulte o material da pré-atividade.

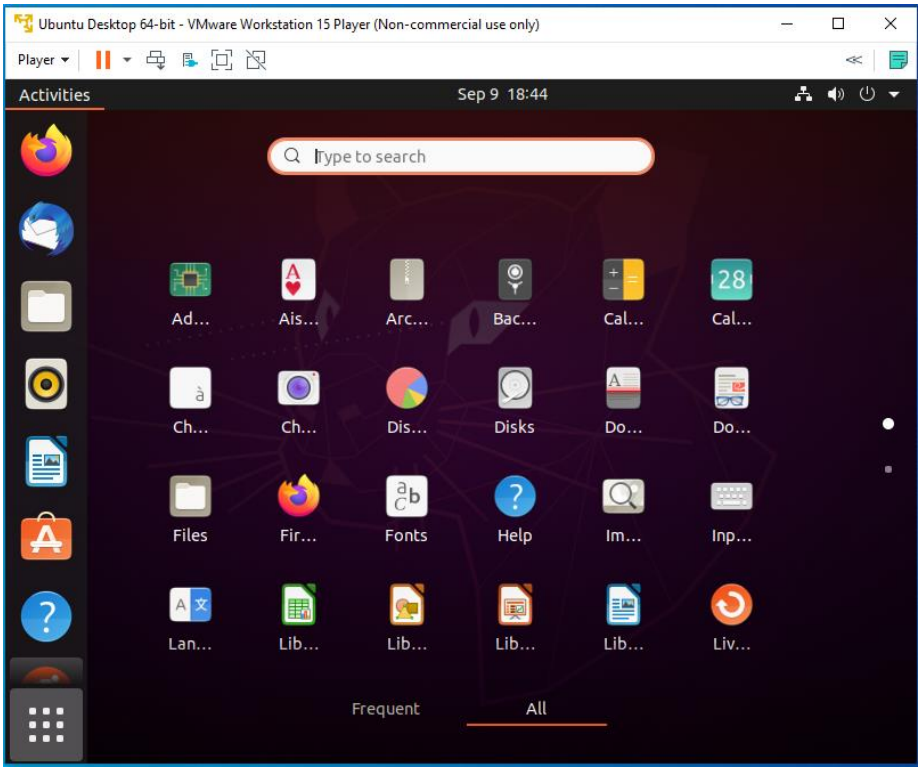
## Roteiro Atividade:

Neste roteiro estaremos utilizando a distribuição do Linux do Ubuntu Desktop, esta atividade poderá ser realizada em qualquer distribuição Linux, porém se você não tiver familiaridade com o ambiente Linux, utilize a mesma distribuição adotada neste documento.

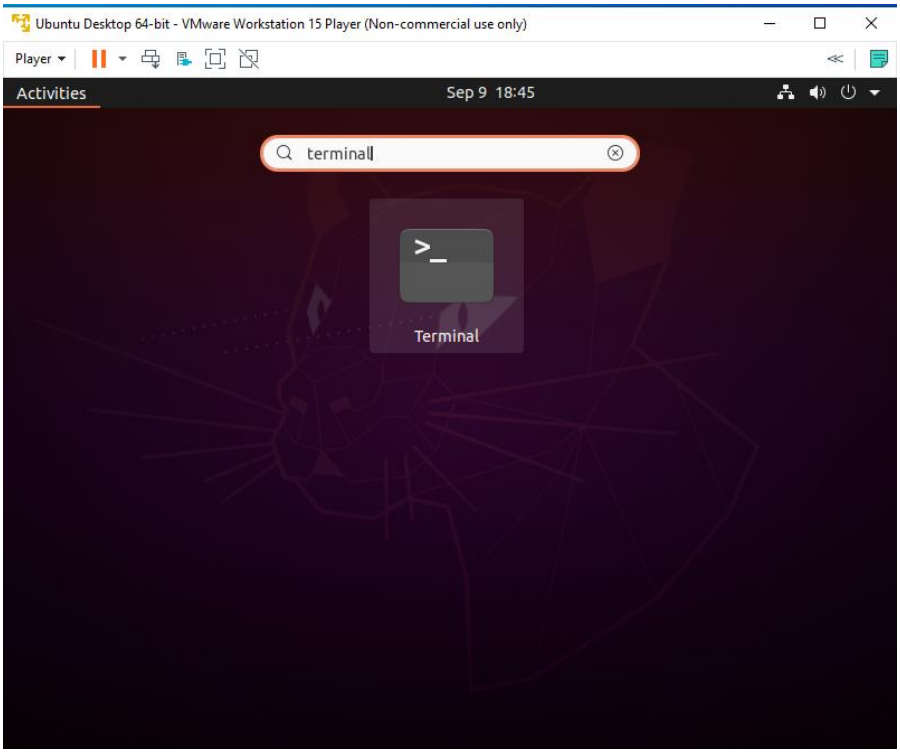
A proposta deste roteiro é aplicar na prática o conhecimento de controle de acesso a arquivos introduzidos na unidade 4. Esta atividade está estruturada em duas etapas. Na primeira etapa do modelo tradicional, estaremos demostrando como manipular corretamente as permissões básicas e especiais no Linux, consequentemente os principais comandos. Na segunda etapa, iremos mostrar como expandir o controle de acesso a arquivos utilizando as listas de controle de acesso estendida, bem como os comandos correspondentes no Linux. Então vamos lá, é hora de colocar a mão na massa, primeiramente acesse a máquina com Linux, se tiver com dúvidas consulte o roteiro da pré-atividade. Neste momento, considerarei que você já tem acesso ao ambiente Linux.



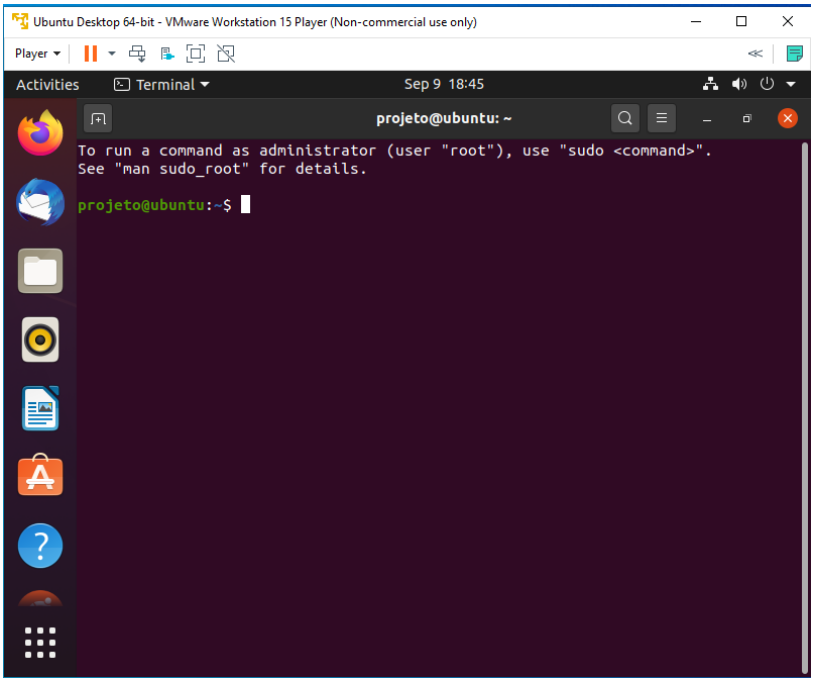
Na opção “Search” nos aplicativos localize o terminal:




Digite “Terminal” e abra uma sessão no aplicativo:



O Ubuntu Desktop é semelhante ao Windows, fornece um ambiente gráfico repleto de ferramentas que são bastante úteis no cotidiano, porém uma das grandes vantagens do Linux é a utilização do terminal dado o grande potencial da linha de comando. Prontinho, nossa aventura inicia aqui.




### Manipulação de Arquivos

No Linux, normalmente cada usuário tem acesso ao seu próprio diretório de trabalho, neste diretório o usuário tem permissão total, denominamos este diretório como home do usuário, ao abrir o terminal você será automaticamente direcionado para este diretório. Para que você explore este ambiente segue alguns comandos que serão bastante uteis:

| Comando | Descrição | Sintaxe |
|---|---|---|
| cd | Acessar um diretório | cd pasta1 |
| cd .. | Voltar um nível acima do diretório corrente | cd .. |
| cd ~ | Acessar o diretório home do usuário | cd ~ |
| cd - | Acessar o diretório anterior | cd - |
| pwd | Verificar o diretório corrente | pwd |

| | | |
|---|---|---|
| mkdir | Criar um diretório | mkdir pasta1 |
| ls | Listar arquivos e diretórios | ls |
| ls -l | Listar arquivos e diretórios de forma detalhada | ls -l |
| ls -la | Listar arquivos e diretórios de forma detalhada incluindo arquivos ocultos | ls -la |
| rm | Remover arquivo ou diretório | rm arquivo1.txt<br>rm -rf pasta1 |
| touch | Criar um arquivo vazio | touch arquivo1.txt |

**Atividade Manipulação de Arquivos:**

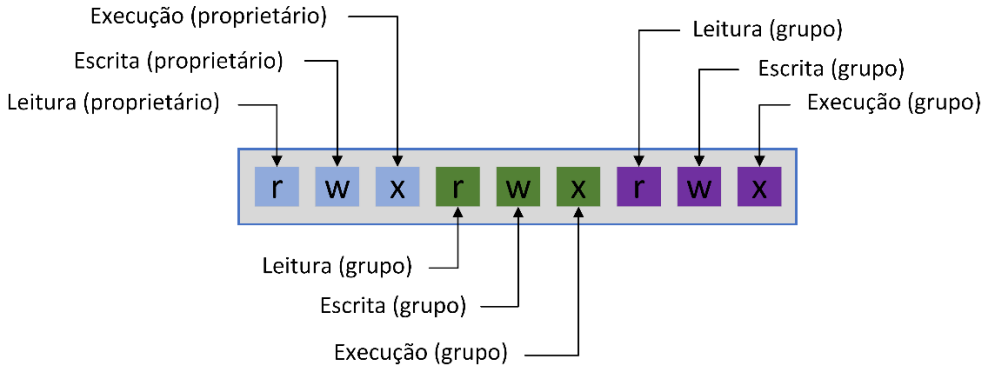
1. Verifique qual o seu diretório corrente
2. Crie dois diretórios “tsi_pasta1” e outro “tsi_pasta2”.
3. Acesse o diretório “tsi_pasta1”
4. Crie dois arquivos “arquivo1.txt” e “arquivo2.txt”.
5. Retorne um nível na hierarquia de diretório. (cd ..)
6. Verifique qual o seu diretório corrente.
7. Retorne ao diretório anterior. (cd -)
8. Verifique o seu diretório corrente.
9. Liste os arquivos em detalhes.
10. Remova o arquivo “arquivo1.txt”.

Gabarito:

```
jhonatan@ubuntu: ~/tsi_pasta1
jhonatan@ubuntu:~$ pwd
/home/jhonatan
jhonatan@ubuntu:~$ mkdir tsi_pasta1
jhonatan@ubuntu:~$ mkdir tsi_pasta2
jhonatan@ubuntu:~$ cd tsi_pasta1
jhonatan@ubuntu:~/tsi_pasta1$ touch arquivo1.txt arquivo2.txt
jhonatan@ubuntu:~/tsi_pasta1$ cd ..
jhonatan@ubuntu:~$ pwd
/home/jhonatan
jhonatan@ubuntu:~$ cd -
/home/jhonatan/tsi_pasta1
jhonatan@ubuntu:~/tsi_pasta1$ pwd
/home/jhonatan/tsi_pasta1
jhonatan@ubuntu:~/tsi_pasta1$ ls -l
total 0
-rw-rw-r-- 1 jhonatan jhonatan 0 Jul 12 05:40 arquivo1.txt
-rw-rw-r-- 1 jhonatan jhonatan 0 Jul 12 05:40 arquivo2.txt
jhonatan@ubuntu:~/tsi_pasta1$ rm arquivo1.txt
jhonatan@ubuntu:~/tsi_pasta1$ ls -l
total 0
-rw-rw-r-- 1 jhonatan jhonatan 0 Jul 12 05:40 arquivo2.txt
jhonatan@ubuntu:~/tsi_pasta1$
```

## Modelo Tradicional de Controle de Acesso a Arquivos

Conforme estudamos na unidade 4 as permissões básicas são: leitura, escrita e execução. Tais permissões são concedidas aos usuários considerando três blocos, proprietário, grupo e outros. Retomando este conceito temos a representação disposta na figura abaixo:




Para alterar as permissões básicas no Linux utilizamos o comando chmod, este comando pode ser utilizado de forma nominal ou por representação binária. Primeiramente, vamos aprender a definir as

permissões de forma nominal. Na forma nominal, modificamos a permissão atual utilizando uma máscara que permite alterar a permissão do objeto. Segue a sintaxe do comando chmod:

chmod <máscara> <objeto>

Para alterar as permissões de um objeto (arquivo ou diretório) é necessário ter privilégios sobre ele. Assim, para obter os privilégios de superusuário podemos utilizar o comando "sudo" antecedendo o comando que queremos executar, este comando vai fornecer os privilégios necessários, será solicitado que você forneça a senha do usuário para elevar os privilégios.

sudo chmod <máscara> <objeto>

A máscara na representação nominal é sempre estruturada em três partes, respeitando sua ordem:

1. **Bloco de destino:** usuário (u), grupo (g), outros (o) e todos (a);
2. **Operação:** incluir (+) ou remover permissão (-);
3. **Permissão:** leitura (r), escrita (w) e execução (x).

Por exemplo para adicionar a permissão de execução para o usuário você deve combinar o bloco de destino, neste caso o usuário (u), operação (+) e permissão é execução (x), assim teremos a seguinte máscara u+x. Esta máscara deve ser utilizada com o comando chmod na sequência o objeto a ser alterado. O objeto corresponde ao arquivo ou diretório. Por exemplo, podemos alterar a permissão do arquivo arquivo1.txt, a permissão seria:

chmod u+x arquivo1.txt

Em contra partida para remover a permissão de execução do usuário você realiza o mesmo procedimento, a única diferença é que será necessário utilizar o sinal de subtração (-) para remover a permissão, assim devemos executar o seguinte comando:

chmod u-x arquivo1.txt

Para adicionar a permissão de execução para o grupo o processo é mesmo, devemos alterar apenas o bloco de destino, neste caso o grupo corresponde a letra "g".

chmod g+x arquivo1.txt

Para adicionar a permissão de execução para o outros, devemos alterar apenas o bloco de destino, neste caso outros corresponde a letra "o".

chmod o+x arquivo1.txt

Para definir a permissão de leitura (r) e escrita (w) devemos seguir a mesma lógica. Por exemplo definir a permissão de leitura para o usuário:

```
chmod u+r arquivo1.txt
```

Ainda é possível atribuir a permissão para todos os blocos utilizando a letra “a” (all), neste sentido a permissão será aplicada para o usuário, grupo e outros. Por exemplo adicionar a permissão de execução para todos.

```
chmod a+x arquivo1.txt
```

Adicionalmente podemos combinar mais de um bloco de destino, ou mais de um tipo de permissão. Alternativamente ao invés de utilizar a letra “a” poderíamos combinar as três letras. Por exemplo adicionar a permissão de execução para o usuário, grupo e outros.

```
chmod ugo+x arquivo1.txt
```

Poderíamos definir a permissão apenas sobre o usuário (u) e o grupo (g).

```
chmod ug+x arquivo1.txt
```

Ainda definir mais de um tipo de permissão simultaneamente, por exemplo o usuário com a permissão de leitura (r) e escrita (w).

```
chmod u+rw arquivo1.txt
```

As permissões nominais podem ser atribuídas de única vez, devem ser separadas por virgula. Por exemplo adicionar a permissão de execução (x) para o usuário (u) e remover (-) a permissão de escrita (w) do grupo (g).

```
chmod u+x,g-w arquivo1.txt
```

As permissões podem ser aplicadas sobre os diretórios, você precisa adicionar a opção de recursividade (-R). Por exemplo, adicionar a permissão de escrita para o usuário no diretório “tsi_pasta1”.

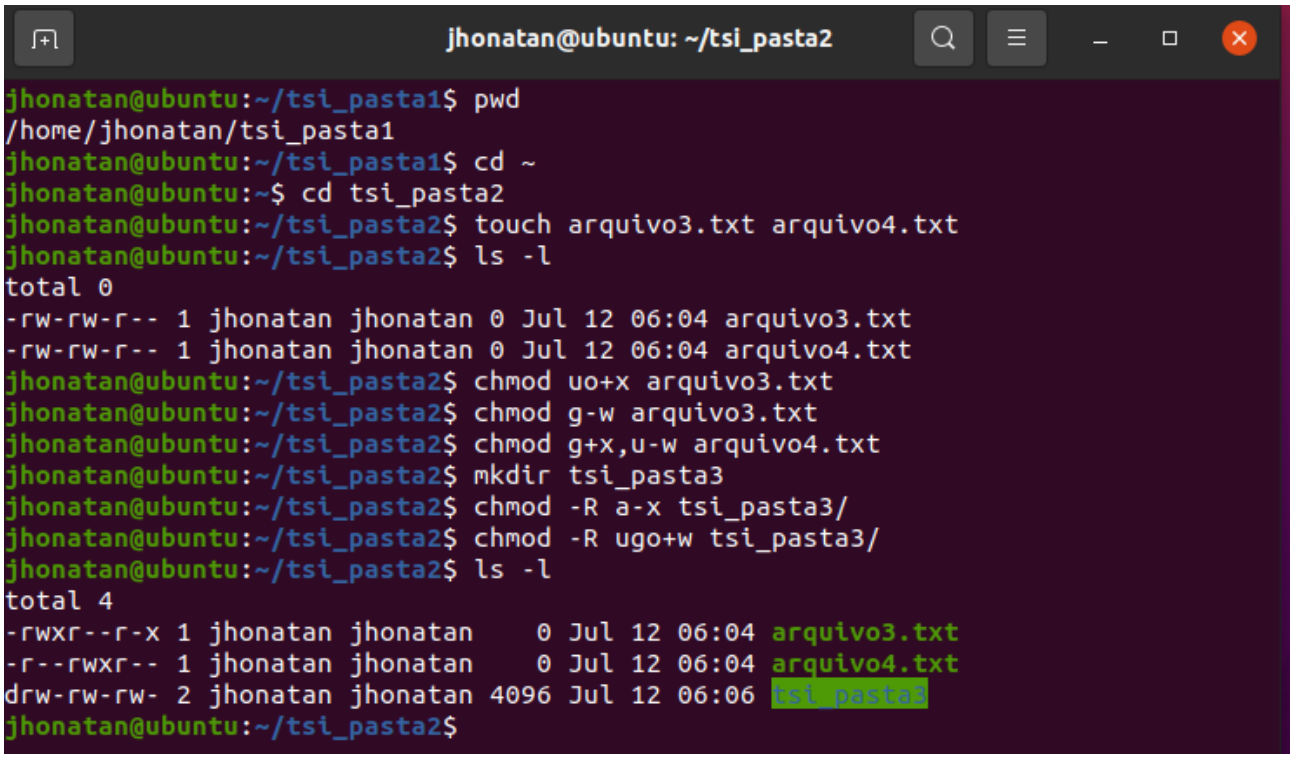
```
chmod -R u+w tsi_pasta1
```

**Atividade Permissões Básicas:**

1. Verifique qual o seu diretório corrente.
2. Acesse o diretório home do usuário. (cd ~)
3. Acesse o diretório “tsi_pasta2”
4. Crie dois arquivos “arquivo3.txt” e “arquivo4.txt”.
5. Liste os arquivos no diretório e verifique as permissões padrões.
6. Adicione a permissão de execução para o usuário e outros no arquivo arquivo3.txt.

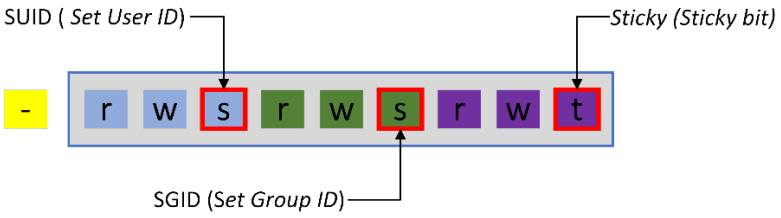
7. Remova a permissão de escrita do grupo no arquivo arquivo3.txt.
8. Adicione a permissão de execução para o grupo no arquivo arquivo4.txt e remova a permissão de escrita do usuário utilizando uma única linha de comando (separar por virgula).
9. Crie o diretório tsi_pasta3.
10. Remova a permissão de execução de todos no diretório tsi_pasta3. (utilizar o "a" - todos).
11. Adicione a permissão de escrita do usuário, grupo e outros no diretório tsi_pasta3. (utilizar a permissão "ugo" – usuário, grupo e outros).
12. Liste os arquivos e diretórios e verifique as permissões concedidas.

Gabarito:




## Permissões Especiais

Conforme estudamos na unidade 4, podemos definir permissões adicionais para os arquivos e diretórios. Os três modelos especiais para controle de acesso são denominados *Set User ID* (SUID), *Set Group ID* (SGID) e *Sticky Bit* (Sticky).



As permissões especiais alteram o comportamento padrão do sistema operacional na manipulação dos arquivos e diretórios que possuem tais permissões.

A propriedade SUID permite ajustar o ID do usuário (SetUID), é aplicada apenas para arquivos executáveis não tendo qualquer efeito sob diretórios. Esta permissão de acesso só pode ser definida no campo de execução do proprietário do arquivo, atribuição realizada com a letra “s”. Tal funcionalidade proporciona a criação e utilização de programas privilegiados que podem usar arquivos que são normalmente inacessíveis a outros usuários.

```
chmod u+s arquivo4.txt
```

Alternativamente, a propriedade SGID é utilizada para ajustar o ID do grupo. Esta propriedade tem uma função bastante semelhante a propriedade SUID para arquivos executáveis, contudo esta propriedade tem um efeito especial quando aplicado sob diretórios. Esta permissão de acesso especial só pode ser definida no campo que habilita a execução para o grupo, atribuição realizada com a letra “s”.

```
chmod g+s tsi_pasta3
```

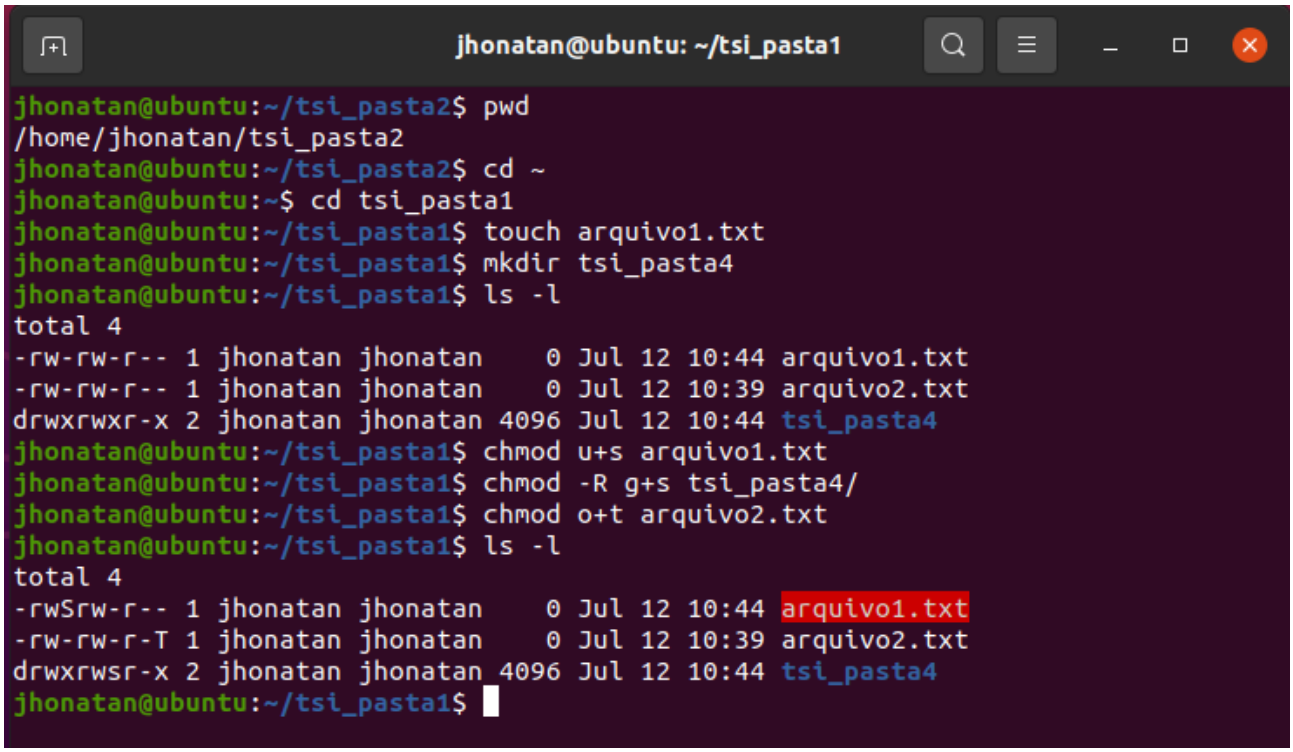
Adicionalmente, a propriedade *Sticky,* quando habilitada em arquivos executáveis faz com que o sistema mantenha uma imagem do programa em memória depois que o programa finalizar. Esta permissão de acesso especial é definida no campo que habilita a execução para outros usuários, atribuição realizada com a letra “t”.

```
chmod o+t arquivo4.txt
```

**Atividade Permissões Especiais:**

1. Verifique qual o seu diretório corrente.
2. Acesse o diretório home do usuário. (cd ~)
3. Acesse o diretório “tsi_pasta1”
4. Crie o novamente o arquivo “arquivo1.txt”
5. Crie o diretório “tsi_pasta4”
6. Liste os arquivos no diretório e verifique as permissões padrões.
7. Adicione a permissão *Set User ID* sobre o arquivo “arquivo1.txt”.
8. Adicione a permissão *Set Group ID* sobre o diretório “tsi_pasta4”.
9. Adicione a permissão *Sticky Bit* sobre o arquivo “arquivo2.txt”.
10. Liste os arquivos e diretórios e verifique as permissões concedidas.

Gabarito:




## Permissão Básica Utilizando Representação Binária

Para modificar as permissões utilizando a representação binária você deve atribuir a nova permissão utilizando uma máscara numérica. Basicamente, a máscara é composta por três dígitos consecutivos (ex.: 000). Nesta máscara cada dígito deve ser considerado individualmente. O primeiro dígito refere-se a permissão do usuário ([**0**]00), na sequência o próximo dígito é utilizado para definir a permissão do grupo (0[**0**]0) e o último dígito é utilizado para definir a permissão de outros (00[**0**]).

A representação da máscara binária é apresentada na tabela abaixo, deve-se utilizar o valor decimal na permissão desejada:

| Permissão | Binário | Decimal |
|---|---|---|
| --- | 000 | 0 |
| --x | 001 | 1 |
| -w- | 010 | 2 |
| -wx | 011 | 3 |

| | | |
|---|---|---|
| r-- | 100 | 4 |
| r-x | 101 | 5 |
| rw- | 110 | 6 |
| rwx | 111 | 7 |

Deste modo, você define as permissões de maneira individual para cada um dos dígitos, atribuindo o valor decimal. Por exemplo, se você quer definir o usuário com permissão de leitura, escrita e execução o primeiro dígito deve receber o valor 7; definir o grupo com permissão de leitura e execução o segundo dígito deve receber o valor 5; definir outros com a permissão de leitura o terceiro dígito deve receber o valor 4. Assim, como resultado teremos uma máscara formada pela sequência desses números, ou seja 754, esta máscara deve ser utilizada com o comando chmod.

chmod 754 arquivo4.txt

A maneira mais simples de você recordar como aplicar essas permissões é lembrar o valor das três permissões básicas de leitura, escrita e execução. Para o valor da permissão de execução é 1, a escrita é o seu dobro, tem o valor é 2, e a leitura é o seu dobro temos o valor é 4. Diante disto, é só somar o valor que você precisa. Por exemplo a permissão anterior: usuário com permissão de leitura=4, escrita=2 e execução=2, é só somar os valores das permissões 4+2+1=7, aqui temos a permissão do usuário; grupo com permissão de leitura=4 e execução=1, é só somar os valores 4 + 1 = 5; por fim a permissão para outros como a permissão é apenas permissão de leitura adicionamos apenas o valor 4. Assim, como resultado teremos a mesma máscara formada pela sequência desses números, ou seja 754.

Então, para reforçar vamos utilizar um segundo exemplo: usuário com permissão de escrita e execução; grupo com permissão de leitura e execução, outros com permissão de leitura e escrita.

- Usuário: escrita (2) e execução (1), 2 + 1 = **3**;
- Grupo: leitura (4) e execução (1), 4 + 1 = **5**;
- Outros: leitura (4) e escrita (2), 4 + 2 = **6**;

Assim, como resultado teremos uma máscara formada pela sequência desses números, ou seja 356, esta máscara deve ser utilizada com o comando chmod.

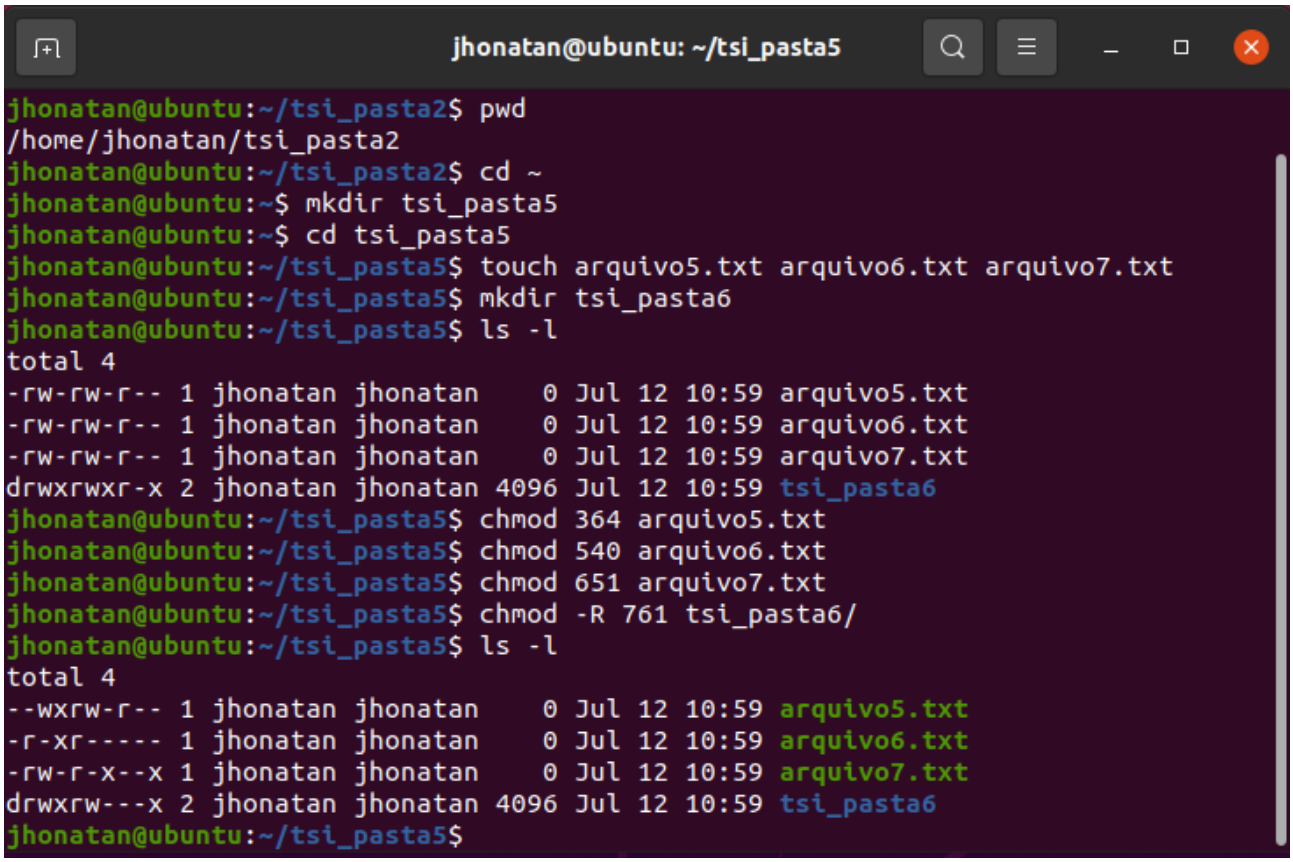
chmod 356 arquivo4.txt

**Atividade Permissões Representação Binária:**

1. Verifique qual o seu diretório corrente.

2. Acesse o diretório home do usuário. (cd ~)
3. Crie o diretório “tsi_pasta5”.
4. Acesse o diretório “tsi_pasta5”.
5. Crie os arquivos “arquivo5.txt”, “arquivo6.txt” e “arquivo7.txt”.
6. Crie o diretório “tsi_pasta6”.
7. Liste os arquivos e diretórios e verifique as permissões concedidas.
8. Adicione a permissões no arquivo “arquivo5.txt”, usuário: escrita e execução; grupo: leitura e escrita, outros: leitura. (Utilizar a representação binária).
9. Adicione a permissões no arquivo “arquivo6.txt”, usuário: leitura e execução; grupo: leitura; outros: sem nenhuma permissão. (Utilizar a representação binária).
10. Adicione a permissões no arquivo “arquivo7.txt”, usuário: leitura e escrita; grupo: leitura e execução, outros: execução. (Utilizar a representação binária).
11. Adicione a permissões no diretório “tsi_pasta5”, usuário: leitura, escrita e execução; grupo: leitura e escrita, outros: execução. (Utilizar a representação binária).
12. Liste os arquivos e diretórios e verifique as permissões concedidas.

Gabarito:



## Alterar o Proprietário e Grupo do Objeto

Para alterar o proprietário e grupo de um objeto (diretório e arquivo) é utilizado o comando chown. Segue a sintaxe do comando:

chown <usuário>:<grupo> <objeto>

Antes de efetivamente utilizar o comando chown, vou mostrar outros dois outros comandos um para criar um usuário e outro comando para criar um outro grupo.

useradd <usuário>

Como exemplo, vamos criar um usuário “user_tsi”:

useradd user_tsi

Por sua vez, o comando para criar um grupo é groudadd. Segue a sintaxe do comando:

groupadd <grupo>

Como exemplo, vamos criar um grupo “grupo_tsi”:

groupadd grupo_tsi

Agora sim, vamos utilizar o comando chown para modificar o proprietário de um arquivo:

chown user_tsi:grupo_tsi arquivo1.txt

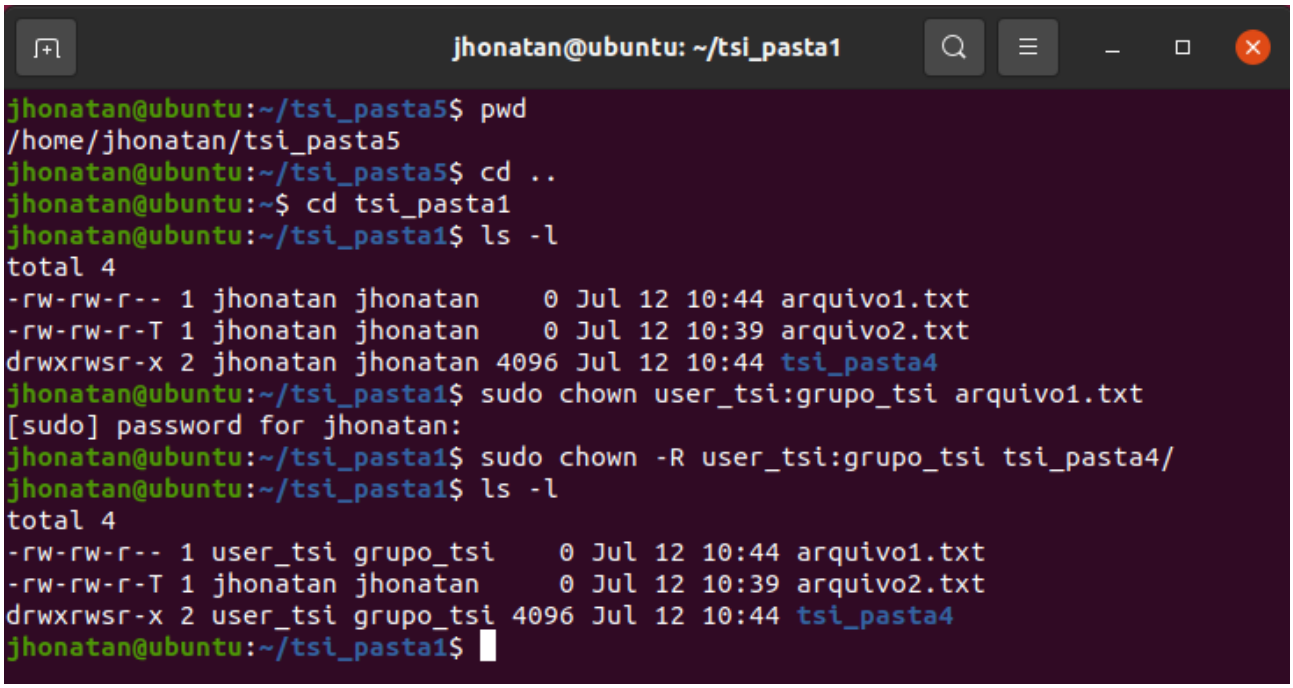
Para modificar o proprietário de um diretório a sintaxe é a mesma, acrescenta-se apenas a opção de recursividade:

chown -R user_tsi:grupo_tsi tsi_pasta1

**Atividade Modificar Proprietário:**

1. Verifique qual o seu diretório corrente.
2. Acesse o diretório home do usuário. (cd ~)
3. Acesse o diretório “tsi_pasta1”.
4. Liste os arquivos e diretórios e verifique o proprietário do arquivo.
5. Altere o proprietário do arquivo “arquivo1.txt” para “user_tsi” e grupo para “grupo_tsi”.
6. Altere o proprietário do diretório “tsi_pasta4” para “user_tsi” e grupo para “grupo_tsi”.
7. Liste os arquivos e diretórios e verifique o proprietário do arquivo.

Gabarito:




## Lista de Controle de Acesso Estendida

A permissão da lista de controle de acesso (ACL) estendida é aplicada utilizando o comando setfact. Na sintaxe do comando setfact devemos passar o parâmetro -m para adicionar a permissão, na sequência fornecer o usuário “u:<usuário>”, um sinal de dois pontos “:” e a permissão a ser atribuída (leitura (r), escrita (w) e execução (x)), conforme disposto abaixo:

setfact -m u:<usuário>:<permissão> <objeto>

Por exemplo, vamos definir uma permissão ACL estendida direcionada a um usuário específico, o usuário “user_tsi” recebendo permissão de leitura, escrita e execução.

setfact -m u:user_tsi:rwx arquivo1.txt

Em contrapartida para remover a permissão aplicada sobre o objeto utilizamos o parâmetro “-b”, conforme a sintaxe abaixo:

setfact -b arquivo1.txt

Para consultar a permissão da ACL estendida definida sobre o objeto utilizamos o comando getfact. Segue abaixo a sintaxe do comando:

getfact arquivo1.txt

**Atividade Lista de Controle de Acesso Estendida:**

1. Verifique qual o seu diretório corrente.
2. Acesse o diretório home do usuário. (cd ~)
3. Crie um diretório “acl_estendida1”.
4. Acesse o diretório “acl_estendida1”.
5. Crie os arquivos “arquivos1.txt” e “arquivos2.txt”.
6. Crie o diretório “acl_estendida2”.
7. Liste os arquivos e diretório para verificar as propriedades da pasta
8. Adicione permissão de leitura e escrita para o usuário “user_tsi” utilizando a ACL estendida sobre o arquivo “arquivos1.txt”.
9. Adicione permissão de leitura e execução para o grupo, “grupo_tsi” utilizando a ACL estendida sobre o arquivo “arquivos2.txt”.
10. Adicione permissão de leitura, escrita e execução para o usuário “user_tsi” utilizando a ACL estendida sobre o diretório “acl_estendida2”.
11. Liste os arquivos e diretório, verifique se foi adicionado um sinal de adição (+) ao lado dos objetos que possuem permissão de ACL estendida.
12. Verifique a permissão do arquivo “arquivos1.txt” utilizando o comando getfacl.
13. Verifique a permissão do arquivo “arquivos2.txt” utilizando o comando getfacl.
14. Verifique a permissão do diretório “acl_estendida2” utilizando o comando getfacl.
15. Remova a ACL estendida definida sobre o arquivo “arquivos2.txt”.
16. Liste os arquivos e diretório, verifique se foi adicionado um sinal de adição (+) ao lado dos objetos que possuem permissão de ACL estendida.

Gabarito:

```
root@ubuntu:/home/jhonatan/tsi_pasta1# pwd
/home/jhonatan/tsi_pasta1
root@ubuntu:/home/jhonatan/tsi_pasta1# cd ~
root@ubuntu:~# mkdir acl_estendida1
root@ubuntu:~# cd acl_estendida1/
root@ubuntu:~/acl_estendida1# touch arquivo1.txt arquivo2.txt
root@ubuntu:~/acl_estendida1# mkdir acl_estendida2
root@ubuntu:~/acl_estendida1# ls -l
total 4
drwxr-xr-x 2 root root 4096 Jul 12 13:21 acl_estendida2
-rw-r--r-- 1 root root    0 Jul 12 13:21 arquivo1.txt
-rw-r--r-- 1 root root    0 Jul 12 13:21 arquivo2.txt
root@ubuntu:~/acl_estendida1# setfacl -m u:user_tsi:rw arquivo1.txt
root@ubuntu:~/acl_estendida1# setfacl -m g:grupo_tsi:rx arquivo2.txt
root@ubuntu:~/acl_estendida1# setfacl -m u:user_tsi:rwx acl_estendida2/
root@ubuntu:~/acl_estendida1# ls -l
total 4
drwxrwxr-x+ 2 root root 4096 Jul 12 13:21 acl_estendida2
-rw-rw-r--+ 1 root root    0 Jul 12 13:21 arquivo1.txt
-rw-r-xr--+ 1 root root    0 Jul 12 13:21 arquivo2.txt
root@ubuntu:~/acl_estendida1# getfacl arquivo1.txt
# file: arquivo1.txt
# owner: root
# group: root
user::rw-
user:user_tsi:rw-
group::r--
mask::rw-
other::r--

root@ubuntu:~/acl_estendida1#
```

```
root@ubuntu:~/acl_estendida1# getfacl arquivo2.txt
# file: arquivo2.txt
# owner: root
# group: root
user::rw-
group::r--
group:grupo_tsi:r-x
mask::r-x
other::r--

root@ubuntu:~/acl_estendida1# getfacl acl_estendida2/
# file: acl_estendida2/
# owner: root
# group: root
user::rwx
user:user_tsi:rwx
group::r-x
mask::rwx
other::r-x

root@ubuntu:~/acl_estendida1# setfacl -b arquivo2.txt
root@ubuntu:~/acl_estendida1# ls -l
total 4
drwxrwxr-x+ 2 root root 4096 Jul 12 13:21 acl_estendida2
-rw-rw-r--+ 1 root root    0 Jul 12 13:21 arquivo1.txt
-rw-r--r--  1 root root    0 Jul 12 13:21 arquivo2.txt
root@ubuntu:~/acl_estendida1#
```

## Definindo Permissão em Arquivos Utilizando Python

Para definir permissões em arquivos utilizando o Python vamos utililzar o comando chmod da biblioteca “os”. Este comando funciona de maneira semelhante as permissões definidas no Linux, porém a máscara é definida em formato octal. Existe uma segunda que fornece as permissões no formato corresponte, a biblioteca “stat”. As permissões básicas definidas na biblioteca “stat” são listadas abaixo:

- stat.S_IRWXU – Leitura, escrita e execução pelo proprietário.
- stat.S_IRUSR – Leitura pelo proprietário.
- stat.S_IWUSR – Escrita pelo proprietário.
- stat.S_IXUSR – Execução pelo proprietário.
- stat.S_IRWXG – Leitura, escrita e execução pelo grupo.
- stat.S_IRGRP – Leitura pelo grupo.
- stat.S_IWGRP – Escrita pelo grupo.
- stat.S_IXGRP – Execução pelo grupo.
- stat.S_IRWXO – Leitura, escrita e execução por outros
- stat.S_IROTH – Leitura por outros
- stat.S_IWOTH – Escrita por outros
- stat.S_IXOTH – Execução por outros

Além das permissões básicas, a biblioteca “stat” dispõe de algumas permissões especiais, foram elencadas na lista abaixo:

- stat.S_ISUID – Execução como *Set user ID*.
- stat.S_ISGID – Execução como *Set group ID*.
- stat.S_ENFMT – Bloqueio de registro aplicado.
- stat.S_ISVTX – Salva a imagem do texto após a execução.
- stat.S_IREAD – Leitura pelo proprietário.
- stat.S_IWRITE – Escrita pelo proprietário.
- stat.S_IEXEC – Execução pelo proprietário.

Para alterar as permissões em um arquivo utilizando o Python é bastante simples, crie um arquivo texto no mesmo diretório do seu projeto, vou nomear este arquivo como “exemplo.txt”. Como você é proprietário deste arquivo, terá permissão total sobre ele, tente modificar o arquivo. Para deixar o arquivo apenas com permissão de leitura basta implementar as linhas abaixo:

```
1  import os
2  import stat
3
4  os.chmod("exemplo.txt", stat.S_IRUSR)
```

Observe que apenas importamos as bibliotecas “os” e “stat”, na sequência chamamos a função chmod da biblioteca “os”. Esta função recebe dois parâmetros, o arquivo a ser manipulado e a permissão a ser aplicada. Para definir o proprietário com apenas permissão de leitura sobre o arquivo “exemplo.txt” utilizamos a macro “stat.S_IRUSR”. Perceba que agora o arquivo “exemplo.txt” só possui permissão de leitura, ao tentar modificar este arquivo você receberá a mensagem que não foi possível modificar o arquivo. No Pycharm também é possível notar que foi adicionado um icone de um cadeado sobre o arquivo “exemplo.txt”. Para retornar as permissões anteriores, podemos utilizar o código abaixo:

```
1  import os
2  import stat
3
4  os.chmod("exemplo.txt", stat.S_IRWXU)
```

A macro stat.S_IRWXU permite fornecer as permissões de leitura, escrita e execução sobre o arquivo “exemplo.txt”. Veja que alterar as permissões é bem tranquilo, você pode utilizar qualquer uma das permissões da biblioteca stat fornecida na lista acima. Agora vamos incrementar um pouco nosso código, sempre que este código for executado vamos modificar as permissões (permissão total) do arquivo “exemplo.txt”, abrir o arquivo para escrita, escrever no arquivo e modificar novamente a permissão (apenas leitura).

```
1  import os
2  import stat
3
4  # Verifica se o arquivo existe
5  if os.path.isfile("exemplo.txt"):
6      # Modifica a permissão do arquivo para leitura, escrita e execução
7      os.chmod("exemplo.txt", stat.S_IRWXU)
8
9  # Abre o arquivo para escrita
10 arquivo = open("exemplo.txt", 'w')
11 # Escreve no arquivo
12 arquivo.write("Escrevendo Arquivo!")
13 # Fecha o arquivo
14 arquivo.close()
15
16 # Modifica o arquivo apenas para leitura
17 os.chmod("exemplo.txt", stat.S_IRUSR)
```

**Atividade Permissão Arquivos Pelo Python:**

Para auxiliar você nesta atividade baseie-se no código da figura acima ou no arquivo “Permissao.py”.

1. Crie um programa em Python para armazenar a data e horário que o arquivo foi executado.
2. A informação da data e horário deve ser armazenada em um arquivo denominado “permissao.txt”
3. Sempre que o arquivo for executado você deve verificar se este arquivo existe.
4. Caso o arquivo exista você deve modificar a permissão deste arquivo para leitura, escrita e execução do proprietário.
5. Obtenha as informações de data e hora do sistema e armazene em uma variável
6. Abra o arquivo para escrita.
7. Grave as informações das variáveis no arquivo “permissão.txt”.
8. Modifique as permissões do arquivo “permissão.txt” apenas para escrita.

Prepare um arquivo TEXTO ou PDF com as respostas para as seguintes questões:

1) Você obteve sucesso em todas as etapas desta atividade? Se não, quais foram os problemas encontrados?
2) Você conseguiu realizar a atividade sem consultar outras fontes (colegas, internet, bibliografia...)?
3) Quanto tempo (em minutos) você dedicou à esta atividade?
4) Qual é a sua opinião sobre esta atividade?
5) Você tem sugestões para melhorar atividades práticas desta natureza?

**PROFESSOR-AUTOR**

Jhonatan Geremias

**REVISÃO E CORREÇÃO**

Luis Gonzaga

PUCPR

GRUPO MARISTA