

Relatório: Gerenciamento de Arquivos e Permissões

1. Introdução

Este relatório descreve as operações realizadas no sistema Windows e em dispositivo Android por meio de comandos de linha (CLI), com foco em manipulação de permissões de arquivos e diretórios usando `chmod`, `icacls` e comandos ADB (`adb shell`). O objetivo é demonstrar na prática como cada comando atua, quais efeitos produz e qual sua utilidade no controle de acesso e segurança de sistemas.

2. Metodologia

- **Sistemas Operacionais:** Windows 10 pro, Ubuntu 24.04.03, Android 9.0 -R2

- **Ferramentas:** Linux: `chmod`, `chown`, `ln -s`, `df`, `du`, `file`
- Windows: `icacls`, `fsutil`
- Android: ADB shell e Comandos do Linux

Durante os testes, Nós:

- Criamos usuários e grupos em Linux e Windows.
- Aplicamos e verificamos permissões com `chmod` e `icacls`.
- Criamos e comparamos links simbólicos com cópias.
- Exploramos e alteramos permissões no Android usando `adb shell`.
Essas ações permitiram compreender a aplicação prática do controle de acesso em diferentes sistemas de arquivos.

- **Durante os experimentos realizados, foi possível compreender de forma prática como o controle de permissões e o gerenciamento de arquivos funcionam em diferentes sistemas operacionais:**

- No Linux, o controle é direto e baseado em três classes (dono, grupo e outros), configurado via `chmod` e `chown`.
- No Windows, as permissões são controladas por ACLs (Listas de Controle de Acesso), manipuladas com `icacls`, oferecendo granularidade mais fina e herança automática.
- No Android, o modelo de segurança é mais rígido, derivado do Linux, mas com isolamento adicional entre aplicativos, controlado por UIDs e políticas do sistema (SELinux).

As validações mostraram que as permissões aplicadas funcionaram conforme esperado:

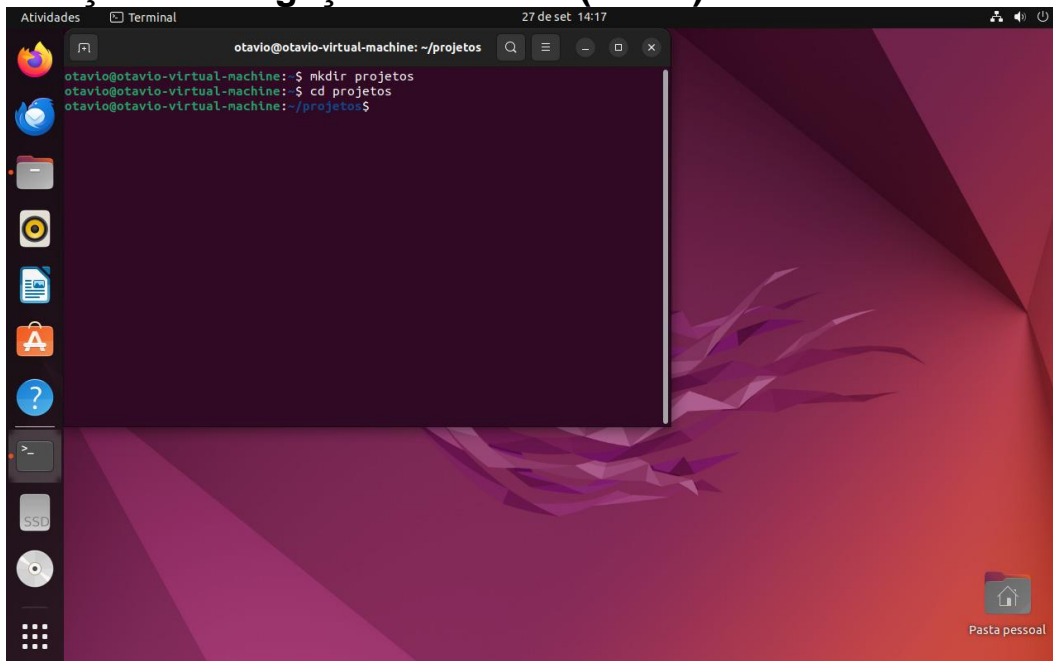
- Usuários sem permissão não conseguiram acessar ou modificar arquivos.
- Os comandos de verificação (`ls -l`, `icacls`, `df`, `du`) confirmaram a propriedade e os direitos aplicados.
- Links simbólicos foram corretamente identificados como ponteiros, e não cópias, reforçando o entendimento sobre referências de arquivos.

Com isso, o procedimento comprovou a eficácia dos comandos e consolidou o domínio sobre operações de gerenciamento de arquivos e permissões em ambientes Windows, Linux e Android.

3. Análise dos Comandos e Configurações

3.1 Linux (chmod, chown, ln -s, df, du)

Criação e Navegação de Pastas (mkdir)

A terminal window titled 'Terminal' is open on a Linux desktop. The prompt is 'otavio@otavio-virtual-machine: ~/projetos'. The user has entered three commands: 'mkdir projetos', 'cd projetos', and 'pwd'. The output of 'pwd' is '/home/otavio/projetos'. The desktop background is a red and purple geometric pattern. A dock on the left contains icons for various applications. A 'Pasta pessoal' icon is visible in the bottom right corner.

```
otavio@otavio-virtual-machine: ~/projetos
otavio@otavio-virtual-machine: $ mkdir projetos
otavio@otavio-virtual-machine: $ cd projetos
otavio@otavio-virtual-machine: ~/projetos$
otavio@otavio-virtual-machine: ~/projetos$ pwd
/home/otavio/projetos
otavio@otavio-virtual-machine: ~/projetos$
```

Comandos utilizados:

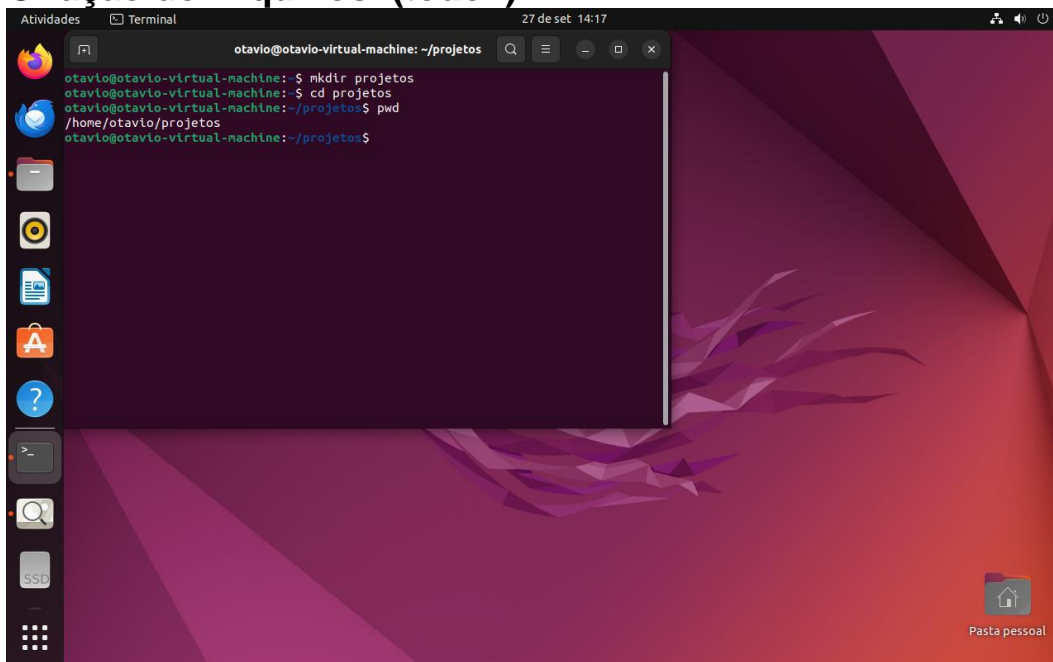
mkdir projetos

cd projetos

pwd

Explicação: Criamos a pasta 'projetos', entramos nela e confirmamos o caminho com *pwd*.

Criação de Arquivos (touch)

A terminal window titled 'Terminal' is open on a Linux desktop. The prompt is 'otavio@otavio-virtual-machine: ~/projetos'. The user has entered four commands: 'mkdir projetos', 'cd projetos', 'pwd', and 'touch grupo10.txt'. The output of 'pwd' is '/home/otavio/projetos'. The desktop background is a red and purple geometric pattern. A dock on the left contains icons for various applications. A 'Pasta pessoal' icon is visible in the bottom right corner.

```
otavio@otavio-virtual-machine: ~/projetos
otavio@otavio-virtual-machine: $ mkdir projetos
otavio@otavio-virtual-machine: $ cd projetos
otavio@otavio-virtual-machine: ~/projetos$ pwd
/home/otavio/projetos
otavio@otavio-virtual-machine: ~/projetos$ touch grupo10.txt
otavio@otavio-virtual-machine: ~/projetos$
```

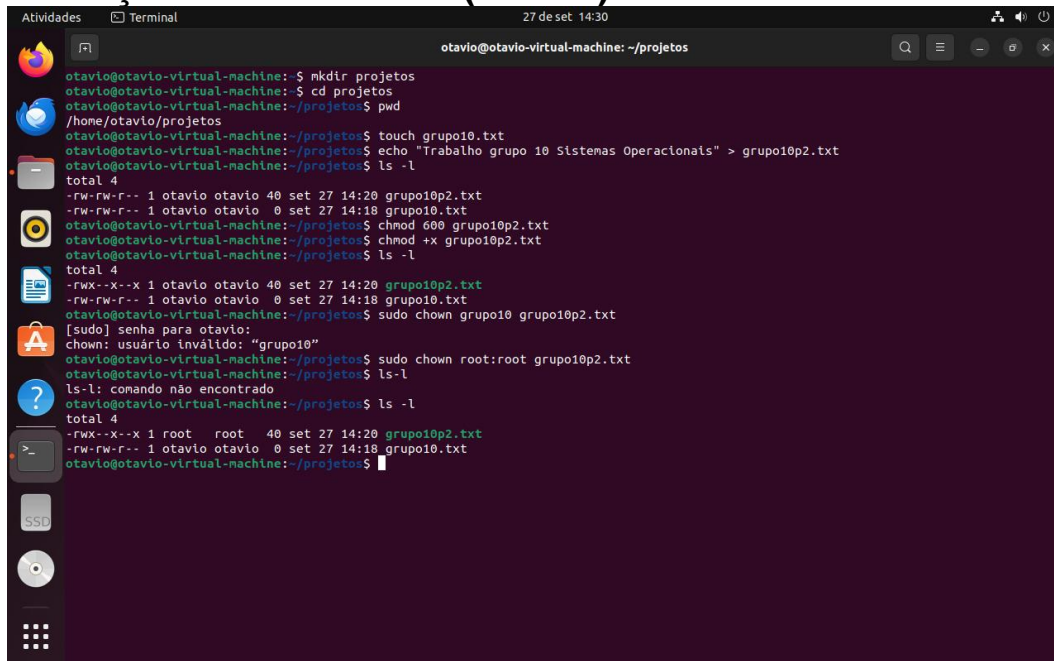
Comandos utilizados:

touch grupo10.txt

echo "Trabalho grupo 10 Sistemas Operacionais"> grupo10p2.txt

Explicação: Criamos os arquivos 'grupo10.txt' e 'grupo10p2.txt', adicionando conteúdo no segundo.

Alteração de Permissões (chmod)

A terminal window titled 'otavio@otavio-virtual-machine: ~/projetos' showing a series of commands and their outputs. The user creates a directory 'projetos', navigates to it, and creates a file 'grupo10p2.txt'. They then use 'chmod 600 grupo10p2.txt' to set permissions to read and write only for the owner. Finally, they use 'chmod +x grupo10p2.txt' to add execute permissions. The 'ls -l' command is used multiple times to show the changes in permissions from '-rw-rw-r--' to '-rwx--x--x'.

```
otavio@otavio-virtual-machine:~$ mkdir projetos
otavio@otavio-virtual-machine:~$ cd projetos
otavio@otavio-virtual-machine:~/projetos$ pwd
/home/otavio/projetos
otavio@otavio-virtual-machine:~/projetos$ touch grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ echo "Trabalho grupo 10 Sistemas Operacionais" > grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ ls -l
total 4
-rw-rw-r-- 1 otavio otavio 40 set 27 14:20 grupo10p2.txt
-rw-rw-r-- 1 otavio otavio 0 set 27 14:18 grupo10.txt
otavio@otavio-virtual-machine:~/projetos$ chmod 600 grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ chmod +x grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ ls -l
total 4
-rwx--x--x 1 otavio otavio 40 set 27 14:20 grupo10p2.txt
-rw-rw-r-- 1 otavio otavio 0 set 27 14:18 grupo10.txt
otavio@otavio-virtual-machine:~/projetos$ sudo chown grupo10 grupo10p2.txt
[sudo] senha para otavio:
chown: usuário inválido: "grupo10"
otavio@otavio-virtual-machine:~/projetos$ sudo chown root:root grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ ls -l
ls: l: comando não encontrado
otavio@otavio-virtual-machine:~/projetos$ ls -l
total 4
-rwx--x--x 1 root root 40 set 27 14:20 grupo10p2.txt
-rw-rw-r-- 1 otavio otavio 0 set 27 14:18 grupo10.txt
otavio@otavio-virtual-machine:~/projetos$
```

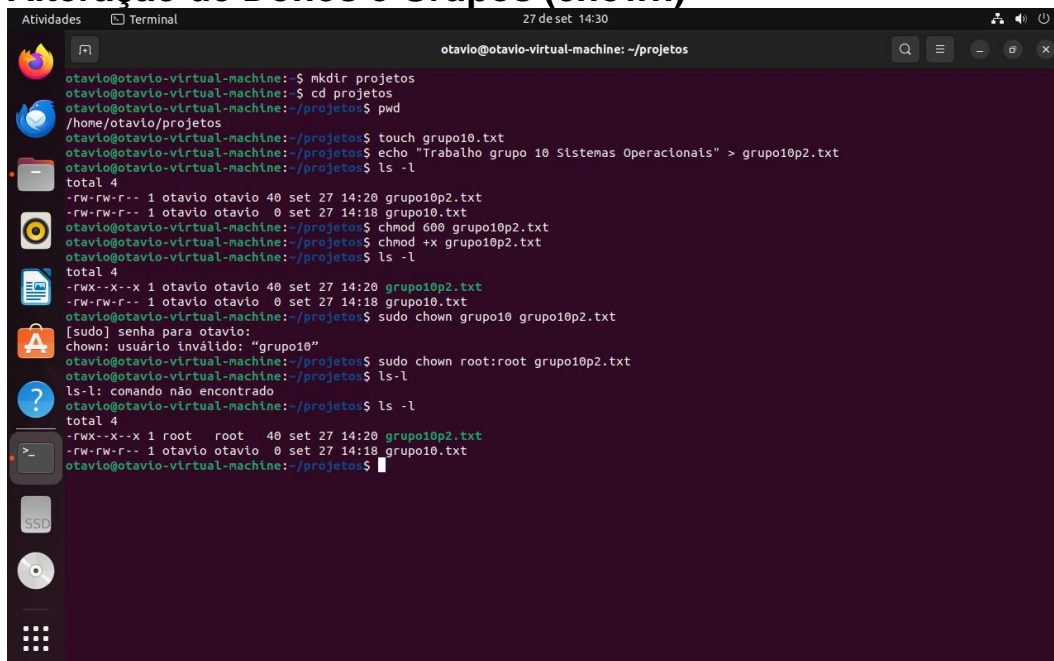
Comando utilizados:

Chmod 600 grupo10p2.txt

Chmod +x grupo10p2.txt

Explicação: Definimos permissões de leitura/escrita apenas para o dono e adicionamos permissão de execução.

Alteração de Donos e Grupos (chown)

A terminal window titled 'otavio@otavio-virtual-machine: ~/projetos' showing a series of commands and their outputs. The user creates a directory 'projetos', navigates to it, and creates a file 'grupo10p2.txt'. They then use 'chmod 600 grupo10p2.txt' and 'chmod +x grupo10p2.txt' to set permissions. Finally, they use 'sudo chown grupo10 grupo10p2.txt' to change the group to 'grupo10'. The 'ls -l' command is used multiple times to show the changes in permissions and ownership. The output shows the file is now owned by 'root' and belongs to the 'root' group.

```
otavio@otavio-virtual-machine:~$ mkdir projetos
otavio@otavio-virtual-machine:~$ cd projetos
otavio@otavio-virtual-machine:~/projetos$ pwd
/home/otavio/projetos
otavio@otavio-virtual-machine:~/projetos$ touch grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ echo "Trabalho grupo 10 Sistemas Operacionais" > grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ ls -l
total 4
-rw-rw-r-- 1 otavio otavio 40 set 27 14:20 grupo10p2.txt
-rw-rw-r-- 1 otavio otavio 0 set 27 14:18 grupo10.txt
otavio@otavio-virtual-machine:~/projetos$ chmod 600 grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ chmod +x grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ ls -l
total 4
-rwx--x--x 1 otavio otavio 40 set 27 14:20 grupo10p2.txt
-rw-rw-r-- 1 otavio otavio 0 set 27 14:18 grupo10.txt
otavio@otavio-virtual-machine:~/projetos$ sudo chown grupo10 grupo10p2.txt
[sudo] senha para otavio:
chown: usuário inválido: "grupo10"
otavio@otavio-virtual-machine:~/projetos$ sudo chown root:root grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ ls -l
ls: l: comando não encontrado
otavio@otavio-virtual-machine:~/projetos$ ls -l
total 4
-rwx--x--x 1 root root 40 set 27 14:20 grupo10p2.txt
-rw-rw-r-- 1 otavio otavio 0 set 27 14:18 grupo10.txt
otavio@otavio-virtual-machine:~/projetos$
```

Comando utilizado:

Sudo chown root:root grupo10p2.txt

Explicação: Mudamos o dono e o grupo do arquivo para root.

Teste com usuário atual

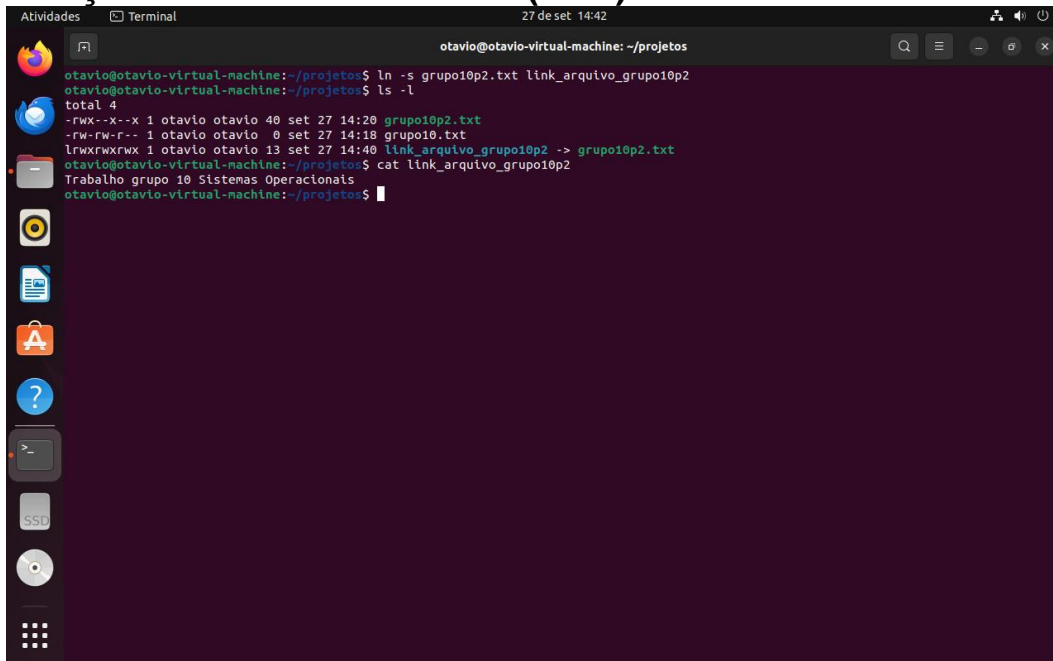
```
otavio@otavio-virtual-machine:~/projetos$ sudo chown $USER:$USER grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ ls -l
total 4
-rwx--x--x 1 otavio otavio 40 set 27 14:20 grupo10p2.txt
-rw-rw-r-- 1 otavio otavio  0 set 27 14:18 grupo10.txt
```

Comando utilizado:

Sudo chown \$USER:\$USER grupo10p2.txt

Explicação: Retornamos a posse do arquivo para o usuário atual para restaurar o acesso.

Criação de Links Simbólicos (ln -s)

A screenshot of a Linux terminal window titled 'Terminal' with the date '27 de set 14:42'. The user 'otavio' is in the directory '~/projetos'. The terminal shows the following commands and output: 1. 'ln -s grupo10p2.txt link_arquivo_grupo10p2' - creates a symbolic link. 2. 'ls -l' - shows the file listing with the new link: 'lrwxrwxrwx 1 otavio otavio 13 set 27 14:40 link_arquivo_grupo10p2 -> grupo10p2.txt'. 3. 'cat link_arquivo_grupo10p2' - displays the content of the linked file, which is 'Trabalho grupo 10 Sistemas Operacionais'. The terminal window has a sidebar with application icons on the left and search, menu, and window control buttons on the right.

```
otavio@otavio-virtual-machine:~/projetos$ ln -s grupo10p2.txt link_arquivo_grupo10p2
otavio@otavio-virtual-machine:~/projetos$ ls -l
total 4
-rwx--x--x 1 otavio otavio 40 set 27 14:20 grupo10p2.txt
-rw-rw-r-- 1 otavio otavio  0 set 27 14:18 grupo10.txt
lrwxrwxrwx 1 otavio otavio 13 set 27 14:40 link_arquivo_grupo10p2 -> grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$ cat link_arquivo_grupo10p2
Trabalho grupo 10 Sistemas Operacionais
otavio@otavio-virtual-machine:~/projetos$
```

Comandos utilizados:

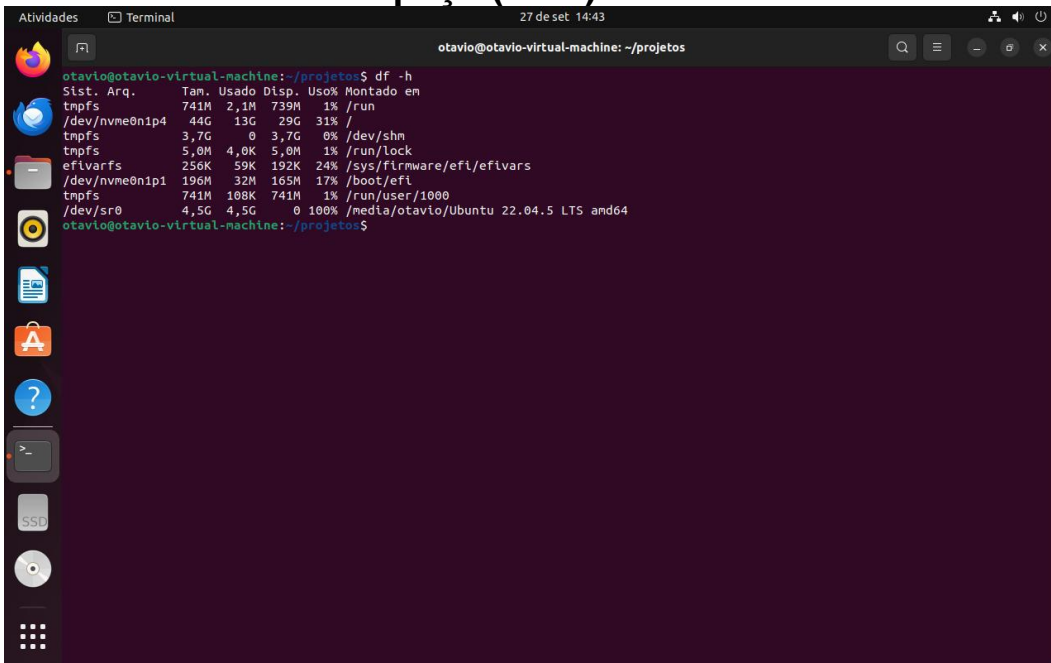
ln -s grupo10p2.txt link_arquivo_grupo10p2

ls -l

cat link_arquivo_grupo10p2

Explicação: Criamos um link simbólico chamado 'link_arquivo_grupo10p2' que aponta para o arquivo 'grupo10p2.txt'. Com o comando 'ls -l', podemos ver a seta indicando a ligação simbólica. O comando 'cat' confirma que o link acessa o mesmo conteúdo do arquivo original.

Monitoramento de Espaço (df -h)



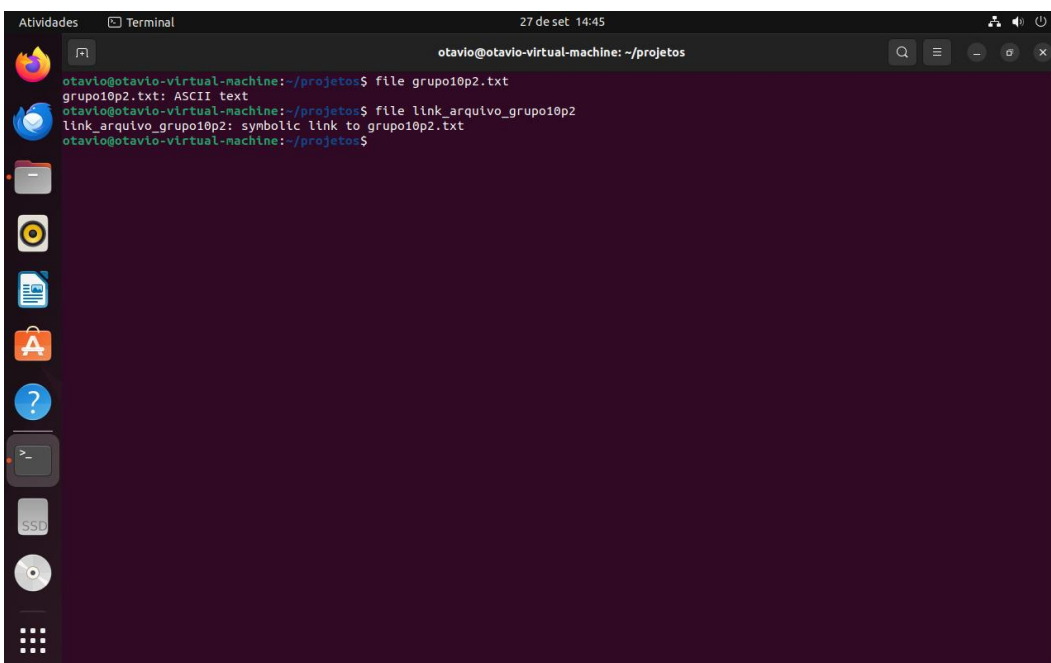
```
otavio@otavio-virtual-machine: ~/projetos
otavio@otavio-virtual-machine:~/projetos$ df -h
Sist. Arq.      Tam. Usado Disp. Uso% Montado em
tmpfs          741M  2,1M  739M   1% /run
/dev/nvme0n1p4  44G   13G   29G  31% /
tmpfs          3,7G   0  3,7G   0% /dev/shm
tmpfs          5,0M  4,0K  5,0M   1% /run/lock
efivarfs       256K   59K  192K  24% /sys/firmware/efi/efivars
/dev/nvme0n1p1 196M   32M  165M  17% /boot/efi
tmpfs          741M  108K  741M   1% /run/user/1000
/dev/sr0       4,5G   4,5G   0 100% /media/otavio/Ubuntu 22.04.5 LTS amd64
otavio@otavio-virtual-machine:~/projetos$
```

Comando utilizado:

`df -h`

Explicação: Com o comando 'df -h', verificamos o uso de espaço em disco no sistema de arquivos, mostrando o tamanho total, usado, disponível e o ponto de montagem.

Identificação de Arquivos (file)



```
otavio@otavio-virtual-machine: ~/projetos
otavio@otavio-virtual-machine:~/projetos$ file grupo10p2.txt
grupo10p2.txt: ASCII text
otavio@otavio-virtual-machine:~/projetos$ file link_arquivo_grupo10p2
link_arquivo_grupo10p2: symbolic link to grupo10p2.txt
otavio@otavio-virtual-machine:~/projetos$
```

Comandos utilizados:

`file grupo10p2.txt`

`file link_arquivo_grupo10p2`

Explicação: O comando 'file' permite identificar o tipo de arquivo. No exemplo, 'grupo10p2.txt' é um arquivo de texto ASCII, e 'link_arquivo_grupo10p2' é reconhecido como um link simbólico que aponta para 'grupo10p2.txt'.

3.2 Windows (icaccls, fsutil)

Comandos do fsutil

```
Administrator: Windows Prompt de Comando

icaccls file /grant *S-1-1-0:(D;WDAC)
- Concede ao usuário definido pela sid S-1-1-0 as permissões
  Excluir e Gravar DAC para arquivo.

C:\Windows\system32>fsutil
---- Comandos com suporte ----

8dot3name      Gerenciamento de 8dot3name
behavior       Controla o comportamento do sistema de arquivos
dax            Gerenciamento de volume Dax
dirty          Gerencia o bit sujo de volume
file           Comandos específicos de arquivo
fsinfo        Informações do sistema de arquivos
hardlink       Gerenciamento de link físico
objectID       Gerenciamento de IDs de objetos
quota          Gerenciamento de cotas
repair         Gerenciamento de autorrecuperação
reparsePoint   Gerenciamento de ponto de nova análise
storageReserve Gerenciamento de Reserva de Armazenamento
resource       Administração do Gerenciador de Recursos de Transação
sparse         Controle de arquivos esparsos
tiering        Gerenciamento de propriedades de camadas de armazenamento
transaction    Gerenciamento de transação
usn            Gerenciamento de USN
volume         Gerenciamento de volume
wim            Gerenciamento de hospedagem do Wim transparente

C:\Windows\system32>
```

Comandos Utilizados:

fsutil

Todos os comandos possíveis com o fsutil

Comandos do icaccls

```
Administrador: Prompt de Comando

C:\Windows\system32>icaccls

ICACLS name /save aclfile [/T] [/C] [/L] [/Q]
armazena as DACLS referentes aos arquivos e às pastas correspondentes
a name em arquivo_acl para uso posterior com /restore. Observe que
as SACLs, o proprietário e os rótulos de integridade não são salvos.

ICACLS directory [/substitute SidOld SidNew [...]] /restore aclfile
[/C] [/L] [/Q]
aplica as DACLS armazenadas aos arquivos de diretório.

ICACLS name /setowner user [/T] [/C] [/L] [/Q]
altera o proprietário de todos os nomes correspondentes. Essa opção
não força uma alteração de propriedade; para essa finalidade, use
o utilitário takeown.exe.

ICACLS name /findsid Sid [/T] [/C] [/L] [/Q]
localiza todos os nomes correspondentes que contêm uma ACL
com menção explícita a Sid.

ICACLS name /verify [/T] [/C] [/L] [/Q]
localiza todos os arquivos cuja ACL não está na forma canônica
ou cujo tamanho é inconsistente com as contagens de ACEs.

ICACLS name /reset [/T] [/C] [/L] [/Q]
substitui as ACLs por ACLs herdadas padrão para todos os arquivos
correspondentes.

ICACLS name [/grant[:r] Sid:perm[...]]
[/deny Sid:perm [...]]
[/remove[:g:d] Sid[...]] [/T] [/C] [/L] [/Q]
[/setintegritylevel Level:policy[...]]

/grant[:r] Sid:perm concede direitos de acesso ao usuário especificado.
Com :r, as permissões substituem todas as permissões explícitas
concedidas anteriormente. Sem :r, as permissões são adicionadas
às permissões explícitas concedidas anteriormente.

/deny Sid:perm nega explicitamente direitos de acesso ao usuário
especificado. Uma ACE de negação explícita é adicionada para as
permissões declaradas e as mesmas permissões em concessões explícitas
são removidas.
```

```
Administrador: Prompt de Comando

/remove[:[g|d]] Sid remove todas as ocorrências de SID na ACL. Com
:g, remove todas as ocorrências de direitos concedidos a essa SID.
Com :d, remove todas as ocorrências de direitos negados a essa Sid.

/setintegritylevel [(CI)(OI)]Level adiciona explicitamente uma ACE de
integridade a todos os arquivos correspondentes. O nível deverá ser
especificado como um destes:
L[ow]
M[edium]
H[igh]
Opções de herança para a ACE de integridade podem preceder o nível
e são aplicadas somente a diretórios

/inheritance:e|d|r
e - habilita a herança
d - desabilita a herança e copia as ACEs
r - remove todas as ACEs herdadas

Observação:
As SIDs podem estar no formato numérico ou de nome amigável. Se for
usado o formato numérico, afixe um * ao início da SID.

/T indica que a operação será executada em todos os arquivos/diretórios
correspondentes abaixo dos diretórios especificados em nome.

/C indica que a operação continuará em todos os erros de arquivo.
As mensagens de erro ainda serão exibidas.

/L indica que a operação será executada em um link simbólico
propriamente dito, e não no destino correspondente.

/Q indica que icacls deve suprimir as mensagens de êxito.

ICACLS preserva a ordenação canônica das entradas ACE:
Negações explícitas
Concessões explícitas
Negações herdadas
Concessões herdadas

perm é uma máscara de permissão e pode ser especificada de duas formas:
uma sequência de direitos simples:
N - sem acesso
```

```
Administrador: Prompt de Comando

M - acesso para modificar
RX - acesso para ler e executar
R - acesso somente leitura
W - acesso somente gravação
D - acesso para excluir
uma lista separada por vírgulas de direitos específicos, entre:
DE - excluir
RC - ler controle
WDAC - gravar DAC
WO - gravar proprietário
S - sincronizar
AS - acessar segurança do sistema
MA - máximo permitido
GR - leitura genérica
GW - gravação genérica
GE - execução genérica
GA - todos genéricos
RD - ler dados/listar diretório
WD - gravar dados/adicionar arquivo
AD - acrescentar dados/adicionar subdiretório
REA - ler atributos estendidos
WEA - gravar atributos estendidos
X - executar/atravessar
DC - excluir filho
RA - ler atributos
WA - gravar atributos
os direitos de herança podem preceder qualquer uma das duas formas
e aplicam-se somente a diretórios:
(OI) - objetos serão herdeiros
(CI) - contêineres serão herdeiros
(IO) - aplica-se somente aos herdeiros
(NP) - não propagar herança
(I) - permissão herdada do contêiner pai

Exemplos:

icacls c:\windows\* /save Aclfile /T
- Salva as ACLs referentes a todos os arquivos de c:\windows
e seus subdiretórios em Aclfile.

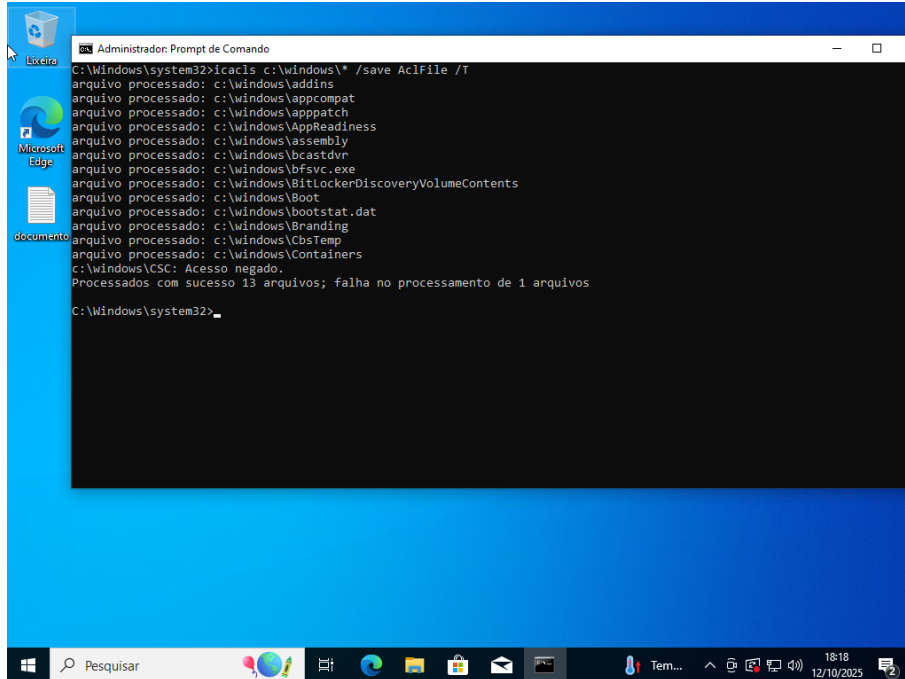
icacls c:\windows\ /restore Aclfile
- Restaura as Acls correspondentes a cada arquivo incluído em
Aclfile que exista em c:\windows e seus subdiretórios.
```

Comandos utilizados:

Icacls

Todos os comandos do icacls

Salvando ACLs



```
C:\Windows\system32>icacls c:\windows\* /save AclFile /T
arquivo processado: c:\windows\addins
arquivo processado: c:\windows\appcompat
arquivo processado: c:\windows\apppatch
arquivo processado: c:\windows\appreadiness
arquivo processado: c:\windows\assembly
arquivo processado: c:\windows\bcasdrv
arquivo processado: c:\windows\bfsvc.exe
arquivo processado: c:\windows\bitlocker\discovery\volumecontents
arquivo processado: c:\windows\boot
arquivo processado: c:\windows\bootstat.dat
arquivo processado: c:\windows\branding
arquivo processado: c:\windows\cbstemp
arquivo processado: c:\windows\containers
c:\windows\csc: Acesso negado.
Processados com sucesso 13 arquivos; falha no processamento de 1 arquivos

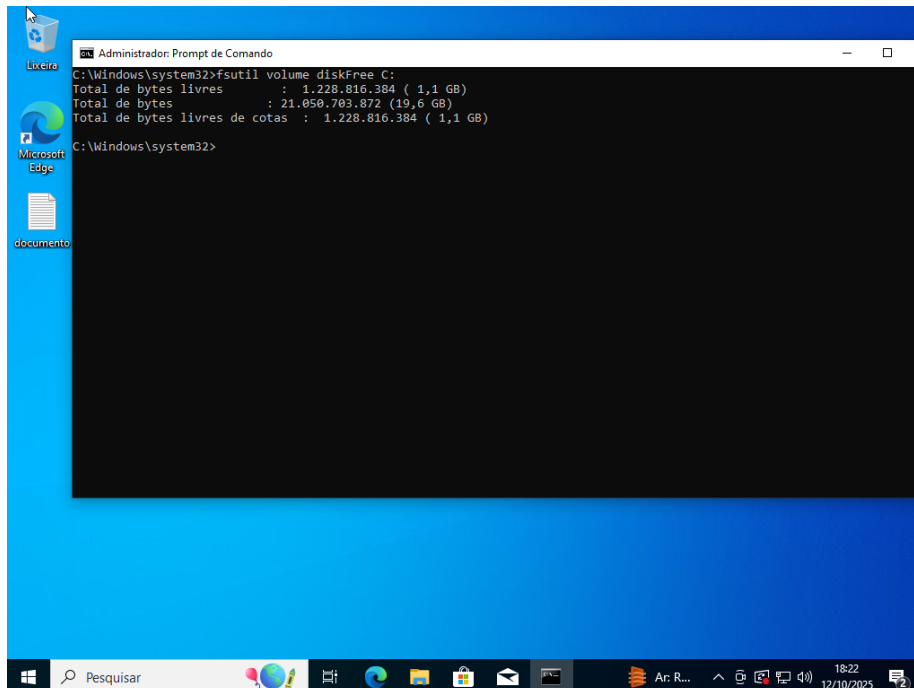
C:\Windows\system32>
```

Comandos utilizados:

Icacls c:\windows* /save AclFile /T

Explicação: Salva os ACLs referentes a todos os arquivos de C:/windows e seus subdiretórios em AclFile

Gerenciando Volume de disco



```
C:\Windows\system32>fsutil volume diskFree C:
Total de bytes livres      : 1.228.816.384 ( 1,1 GB)
Total de bytes            : 21.050.703.872 (19,6 GB)
Total de bytes livres de cotas : 1.228.816.384 ( 1,1 GB)

C:\Windows\system32>
```

Comandos utilizados:

Fsutil volume diskFree C:

Explicação: Faz o Gerenciamento de volume e mostra quanto de espaço se encontra no disco escolhido

3.3 Android (ADB shell e Comandos Linux)

Acesso ao Shell:

Usamos o **ADB** para acessar o terminal do Android com o comando:

(adb shell)

Comandos básicos utilizados:

- `ls -l /data/data/com.example.app` → lista arquivos com permissões e proprietário.
- `chmod 700 arquivo.txt` → altera permissões de arquivos.

Diferença de permissões:

No **Android**, cada aplicativo possui um **UID exclusivo**, garantindo isolamento total. Apps **não podem acessar arquivos de outros apps**, ao contrário do **Linux Desktop**, que permite mais flexibilidade de configuração de permissões.

Exemplo de saída:

```
( drwx----- 2 u0_a123 u0_a123 4096 Oct 12 10:45 files )
( -rw-r--r-- 1 u0_a123 u0_a123 1024 Oct 12 10:50 config.txt )
```

Aqui, **u0_a123** é o UID do app, mostrando como os arquivos são protegidos.

4. Comparação e Análise Crítica

- **Tabela Comparativa – Ferramentas de Gerenciamento de Permissões**

Sistema	Ferramentas	Filosofia de Permissão	Nível de Controle / Granularidade	Resumo Crítico
Linux	chmod, chown	Numérica (r, w, x) por dono, grupo e outros	Média a alta	Flexível e eficaz, mas exige conhecimento técnico.
Windows	icacls, NTFS	ACLs (listas de controle detalhadas)	Alta	Permite controle preciso, ideal para empresas, porém mais complexo.
Android	Permissões de aplicativos	UID e permissões por app	Baixa a média	Prioriza segurança e isolamento, com menor flexibilidade.

- **Análise Geral:**
O **Linux** oferece flexibilidade e simplicidade, o **Windows** fornece controle detalhado e o **Android** foca na segurança e proteção do usuário. Cada sistema reflete sua filosofia: controle técnico no Linux, equilíbrio no Windows e segurança máxima no Android.

- **Sintaxe dos Comandos.**

- **Discussão**

O **Android** prioriza a **segurança**, isolando cada aplicativo em um ambiente próprio e limitando o acesso a arquivos e recursos apenas mediante permissão do usuário.

O **Linux** oferece **flexibilidade e controle total**, baseando-se em permissões de usuário e grupo, mas depende da boa configuração do administrador.

Já o **Windows** busca um **equilíbrio entre segurança e usabilidade**, usando listas de controle de acesso (ACLs) e o UAC para garantir proteção com granularidade e praticidade.

5. Conclusão

Resumo dos aprendizados:

Aprendemos que cada sistema operacional possui um modo diferente de controlar o acesso. O **Linux** utiliza permissões por usuários e grupos, garantindo flexibilidade; o **Windows** usa **ACLs**, oferecendo controle detalhado; e o **Android** isola os aplicativos para priorizar a **segurança**. As permissões são fundamentais para proteger dados e evitar acessos indevidos.

Objetivo:

Conseguimos compreender e aplicar as **operações e os conceitos de controle de acesso e sistemas de arquivos** nos três ambientes — **Windows, Linux e Android** — entendendo como cada um equilibra segurança, flexibilidade e proteção dos dados.

6. Autoavaliação e Referências

Compreendemos o uso de `icacls` e `chmod` para manipular permissões em diferentes sistemas.

Ponto forte: execução prática e comparação entre plataformas.

Ponto de melhoria: aprofundar uso de ACLs no Android e SELinux.

Referências

Manual do chmod: <https://man7.org/linux/man-pages/man1/chmod.1.html>

Manual do chown: <https://man7.org/linux/man-pages/man1/chown.1.html>

Manual do ln: <https://man7.org/linux/man-pages/man1/ln.1.html>

Documentação GNU Coreutils:

<https://www.gnu.org/software/coreutils/manual/>

Microsoft Docs – ICACLS Command:

<https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/icacs>

Petri.com – How to Use the ICACLS Command to Manage File Permissions

LazyJobSeeker Blog – ADB chmod & chown Explained

Android StackExchange – chmod falhando em partições protegidas

SuperUser – Equivalent of chmod in Windows

Tabela Comparativa – Ferramentas de Gerenciamento de Permissões

Sistema	Ferramentas	Filosofia de Permissão	Nível de Controle / Granularidade	Resumo Crítico
Linux	chmod, chown	Numérica (r, w, x) por dono, grupo e outros	Média a alta	Flexível e eficaz, mas exige conhecimento técnico.
Windows	icacls, NTFS	ACLs (listas de controle detalhadas)	Alta	Permite controle preciso, ideal para empresas, porém mais complexo.
Android	Permissões de aplicativos	UID e permissões por app	Baixa a média	Prioriza segurança e isolamento, com menor flexibilidade.

Análise Geral:

O **Linux** oferece flexibilidade e simplicidade, o **Windows** fornece controle detalhado e o **Android** foca na segurança e proteção do usuário. Cada sistema reflete sua filosofia: controle técnico no Linux, equilíbrio no Windows e segurança máxima no Android.