Lab 1.4 : Permissions Control

Lab Name:

Systems Administration, Permissions Control (L1-CAT-01-04)

Overview:

This lab provides comprehensive coverage of permissions control in both Windows and Linux environments and includes topics such as file ownership, access permissions in both symbolic notation and octal notation, groups, and best practices for permissions management.

Prerequisites:

- Introduction to command-line
- Introduction to virtual machines (optional)

Materials:

- Computer with access to a terminal or command prompt
- Ubuntu Linux, either in a virtual machine or as the host operating system
- Windows 10, either in a virtual machine or as the host operating system
- Administrator privileges for certain tasks
- Internet connection (for accessing additional resources)

Learning Objectives:

- Ownership of a file in both Ubuntu Linux and Windows 10.
- Permissions for a file / directory in both Ubuntu Linux and Windows 10 .
- Changing access and default permissions in Ubuntu Linux and Windows 10.
- Groups in Ubuntu Linux and Windows 10.
- Best Practices for Permissions Management.

Tasks: (Work in Progress)

1. Experiment with file ownership, answering questions for both operating systems.
2. Alter permissions in both symbolic and octal notation in both operating systems.
3. Complete the Review.

Deliverables:

- A PDF file with the given questions in the review category and your corresponding answers.
- Push the completed lab to your Git repository or submit through the designated method provided by your instructor.

Additional Resources:

- Using Powershell to Edit Permissions
- Advanced Permissions with NTFS
- Permissions Best Practices - More Info
- Microsoft Active Directory Security Groups

A message from XYZ Corporation:

Now that you have learned to navigate a command line interface (CLI), a Windows GUI and set up a new VM, let's begin permissions control.

Not all users should have the same access or the same rights to the files in our system. Best practice is to limit a user's access to what their job duties dictate they need access to.

These people should be in your system so far:

- Alec - CISO
- Joe - Assistant to CISO
- Carlos - CFO
- Victoria - IT technician

Here are some people, directories and files for you to add/create in order to give people the access they need for their jobs as XYZ corporation.

- Add users:
  - Roberta: Human Resources
  - Bob: Accountant
  - Barbara: IT Manager
- Add **Directories** & *Files*:
  - **Human Resources**
    - *employeeList.txt*
    - *hrBudget.csv*
  - **Finance**
    - *companyBudget.csv*
    - *currentExpenses.csv*
  - **Information Technology**
    - *helpdesktickets.txt*

Topic 1: Ownership of a file / directory in Linux

The owner of a file or a directory is the user (besides root) who can set access permissions for that file or directory. The default owner of a file or directory is the user that made the file or directory.

Access permissions determine what any user can do to that file or folder. There are usually default permissions as well.

These default settings are sometimes a security concern based on the needs and structure of an organization. There does not need to be complete access to all documents for all users. Here we are going to change the ownerships of a file and a directory and see how that affects the way we interact with these items.

chown: changes ownership of a file.
chown username filename

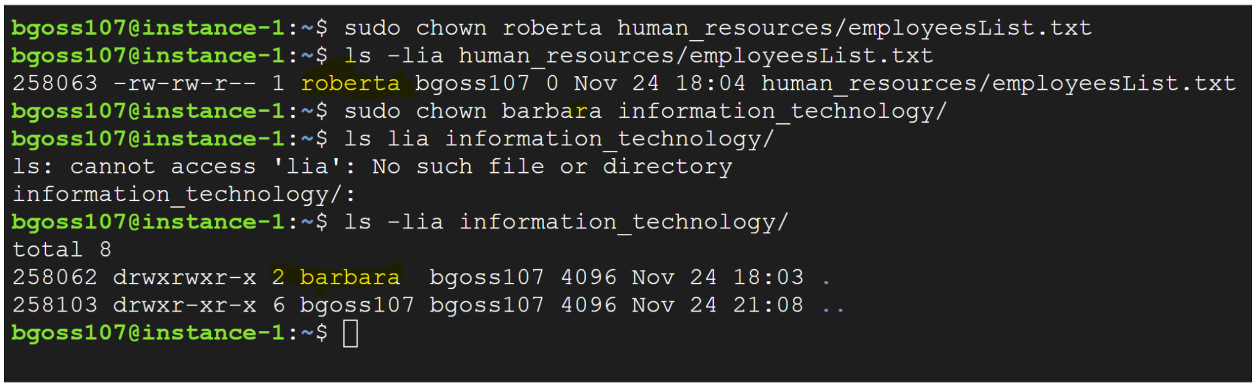
,

- What command would be able to show you who owns a file?
- Who owns the files that you have made so far?

Change ownership of the *employeeList.txt* in the Human Resources Directory.
Who do you think should own this file?

Change the ownerships of the Information Technology directory.
Who should own the IT directory?

Here is how that could look:



```
bgoss107@instance-1:~$ sudo chown roberta human_resources/employeesList.txt
bgoss107@instance-1:~$ ls -lia human_resources/employeesList.txt
258063 -rw-rw-r-- 1 roberta bgoss107 0 Nov 24 18:04 human_resources/employeesList.txt
bgoss107@instance-1:~$ sudo chown barbara information_technology/
bgoss107@instance-1:~$ ls lia information_technology/
ls: cannot access 'lia': No such file or directory
information_technology/:
bgoss107@instance-1:~$ ls -lia information_technology/
total 8
258062 drwxrwxr-x 2 barbara  bgoss107 4096 Nov 24 18:03 .
258103 drwxr-xr-x 6 bgoss107 bgoss107 4096 Nov 24 21:08 ..
bgoss107@instance-1:~$
```

The highlighted names represent the new owners listed for the file/directory.

Topic 2: Permissions for a file/directory

Now that the folders and files belong to the proper departments, the issue of access is still a concern. Can Roberta read the “companyBudget.csv” file in the Finance directory? Some of her job duties include keeping track of how much of her department’s budget is being used. Should Roberta be able to read the “companyBudget.csv” file?

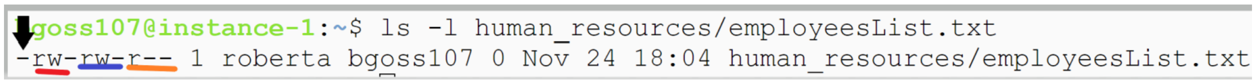
ls -l

File Permissions In Linux:
You’ll notice a few letters and dashes.

The first is either **d** or **-.** The d stands for directory and the - stands for file.
The next three refer to the **owner's** permissions.
The set of three after that refer to the **assigned group** permissions.
The last trio are **everyone else's** permissions.
For a file, the symbols are pretty straightforward:

- **r** - read
- **w** - write
- **x** - execute or run

Any dash "-" among the 2-10th placement refers to a *lack* of permissions.
Red indicates owner permissions, blue indicates group permissions, and orange indicates everyone else's permissions for this file.



```
goss107@instance-1:~$ ls -l human_resources/employeesList.txt
-rw-rw-r-- 1 roberta bgoss107 0 Nov 24 18:04 human_resources/employeesList.txt
```

Look at the following set of permissions:

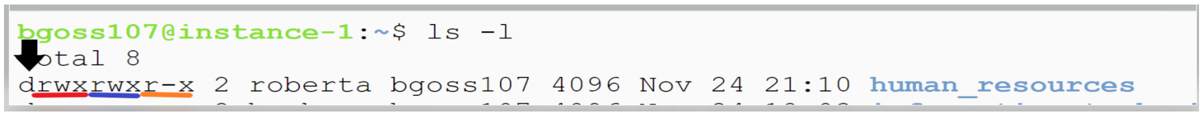
-rw-r-xr-x

This indicates that it is a file, the owner can read and write to that file but not execute, and both assigned groups and others are allowed to read and execute the file.

<u>Directory Permissions in Linux:</u>

Permissions in a directory look the same as a file, but they can mean something different.

- **r -** view the contents of a directory
- **w -** move, modify, or copy files from or to the directory
- **x -** access more detailed information within the directory, cd into that directory and its subfolders.



```
bgoss107@instance-1:~$ ls -l
otal 8
drwxrwxr-x 2 roberta bgoss107 4096 Nov 24 21:10 human_resources
```

*With this in mind, what does it mean to have the following permissions **drwxr-xr-x**? Can you see a directory with a different set of permissions?*

Topic 3: Changing Access and Default Permissions in Linux

There are two ways to change access permissions on a file or a directory.

Method 1: Symbolic notation

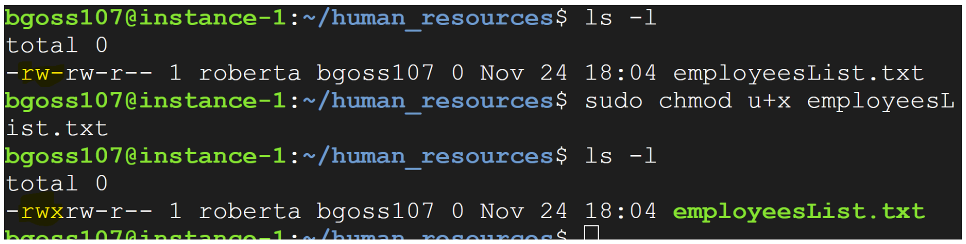
Symbolic notation requires you to state which trio (owner, group or everyone) you are changing permissions for and how you are going to change those permissions..
These are the symbols you would use for the "trio"::

- **u** - owner

g - assigned group
o - everyone else
a - all

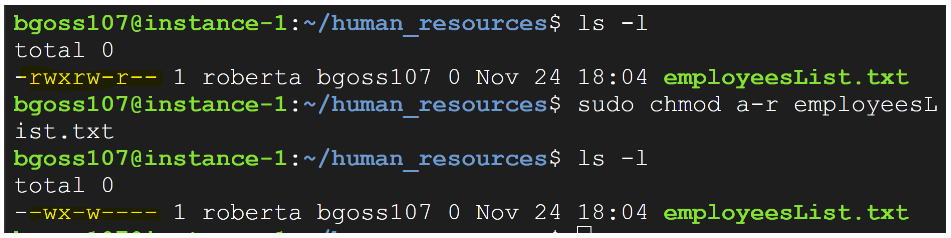
For each of these, you can add and subtract permissions. The permissions being **read - r, write - w,** and **execute - x.**

This adds executing permissions to the owner of the file.



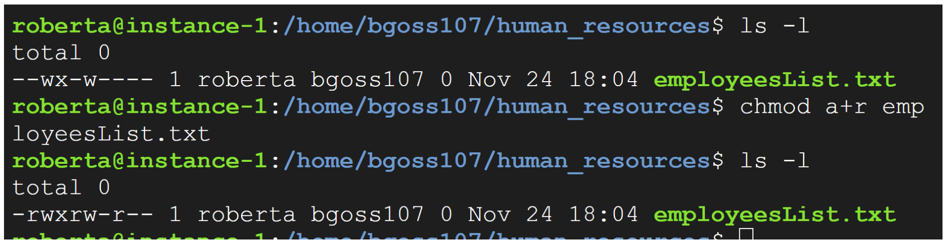
```
bgoss107@instance-1:~/human_resources$ ls -l
total 0
-rw-rw-r-- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
bgoss107@instance-1:~/human_resources$ sudo chmod u+x employeesL
ist.txt
bgoss107@instance-1:~/human_resources$ ls -l
total 0
-rwxrw-r-- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
```

This takes away read permissions for everyone.



```
bgoss107@instance-1:~/human_resources$ ls -l
total 0
-rwxrw-r-- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
bgoss107@instance-1:~/human_resources$ sudo chmod a-r employeesL
ist.txt
bgoss107@instance-1:~/human_resources$ ls -l
total 0
--wx-w---- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
```

This gives reading permissions back to all parties.



```
roberta@instance-1:/home/bgoss107/human_resources$ ls -l
total 0
--wx-w---- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
roberta@instance-1:/home/bgoss107/human_resources$ chmod a+r emp
loyeesList.txt
roberta@instance-1:/home/bgoss107/human_resources$ ls -l
total 0
-rwxrw-r-- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
```

Using symbolic method, how would you give writing privileges to all users on a file? Why did Roberta not need to use ***sudo*** *to add permissions?*

Method 2: Octal Notation

By this point, you should know what binary is, and how it relates to decimal numbers. This is some context that can be helpful when looking at changing permissions on a file or directory This is typically known as *Octal Notation* and you can use these octal numbers along with the command **chmod** to modify permissions on a file or a directory.

Permissions	Binary	Octal	Description

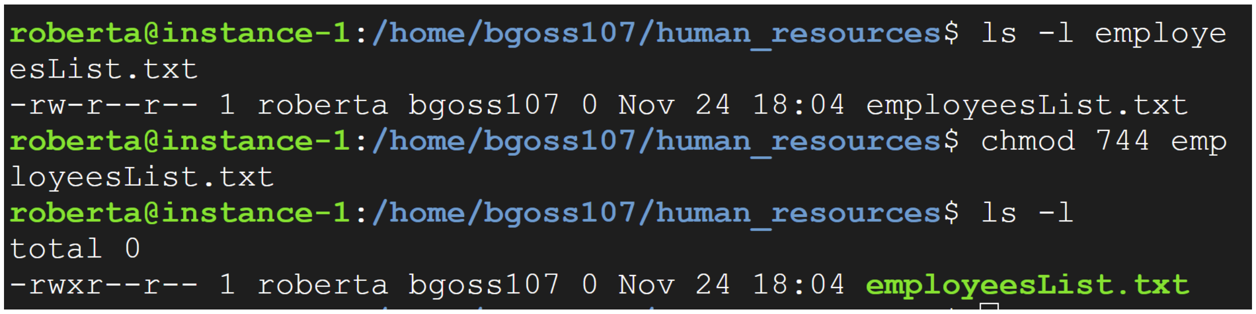
- - -	000	0	None
- - x	001	1	Execute
- w -	010	2	Write
- w x	011	3	Write & Execute
r - -	100	4	Read
r - x	101	5	Read & Execute
r w -	110	6	Read & Write
r w x	111	7	Read, Write, Execute

The way this looks with the **chmod** command is:

chmod 744 filename

This particular command says the file owner gets to read,write, and execute, while the assigned group & everyone else only has read access.

The following adds executing permissions to the owner of the file.



```
roberta@instance-1:/home/bgoss107/human_resources$ ls -l employe
esList.txt
-rw-r--r-- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
roberta@instance-1:/home/bgoss107/human_resources$ chmod 744 emp
loyeesList.txt
roberta@instance-1:/home/bgoss107/human_resources$ ls -l
total 0
-rwxr--r-- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
```

How would you use octal notation to take way **reading** permissions from everyone?

Default Permissions

Default permissions can be set using the following command

umask ###

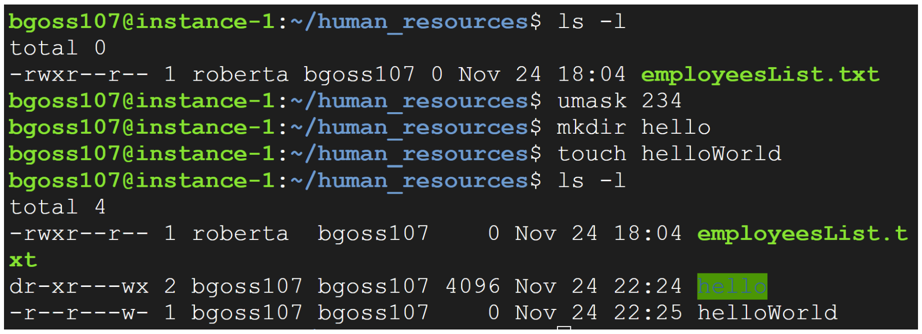
Linux has this preset to either **022** or some equivalent. What happens with the umask, is that it subtracts its ***octal*** value from the current file or folder permissions' ***octal*** value. So when a file is first created, it has **666** for its permissions' octal value, it will subtract the standard umask **022** from that number leaving you with an octal value of **644**.

What does the octal value 644 mean for your file permissions?
The standard value for a created directory is 777, what are its permissions?

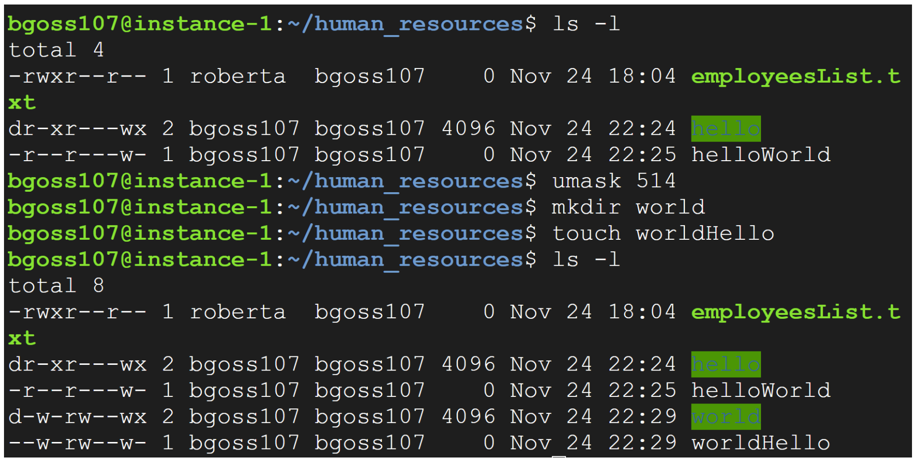
If you, as the administrator needs a different default for creating files and folders, you can change this **umask** like this:

umask 234

Try this command and then create a file and a directory, what are the resulting permissions?



```
bgoss107@instance-1:~/human_resources$ ls -l
total 0
-rwxr--r-- 1 roberta bgoss107 0 Nov 24 18:04 employeesList.txt
bgoss107@instance-1:~/human_resources$ umask 234
bgoss107@instance-1:~/human_resources$ mkdir hello
bgoss107@instance-1:~/human_resources$ touch helloWorld
bgoss107@instance-1:~/human_resources$ ls -l
total 4
-rwxr--r-- 1 roberta  bgoss107    0 Nov 24 18:04 employeesList.t
xt
dr-xr---wx 2 bgoss107 bgoss107 4096 Nov 24 22:24 hello
-r--r---w- 1 bgoss107 bgoss107    0 Nov 24 22:25 helloWorld
```



```
bgoss107@instance-1:~/human_resources$ ls -l
total 4
-rwxr--r-- 1 roberta  bgoss107    0 Nov 24 18:04 employeesList.t
xt
dr-xr---wx 2 bgoss107 bgoss107 4096 Nov 24 22:24 hello
-r--r---w- 1 bgoss107 bgoss107    0 Nov 24 22:25 helloWorld
bgoss107@instance-1:~/human_resources$ umask 514
bgoss107@instance-1:~/human_resources$ mkdir world
bgoss107@instance-1:~/human_resources$ touch worldHello
bgoss107@instance-1:~/human_resources$ ls -l
total 8
-rwxr--r-- 1 roberta  bgoss107    0 Nov 24 18:04 employeesList.t
xt
dr-xr---wx 2 bgoss107 bgoss107 4096 Nov 24 22:24 hello
-r--r---w- 1 bgoss107 bgoss107    0 Nov 24 22:25 helloWorld
d-w-rw--wx 2 bgoss107 bgoss107 4096 Nov 24 22:29 world
--w-rw--w- 1 bgoss107 bgoss107    0 Nov 24 22:29 worldHello
```

Using the command this way, the umask change lasts only as long as your current session.There is a way to change umask permanently, but it's not discussed here.

Topic 4: Groups in Linux

Something you may have noticed while looking at **ls -l** is the second name.This is the group assignment.

As discussed in Lab 1.2, groups can be really useful. When you create a user, a group with that user's name is automatically created, but a user can belong to more than one group. You've already done this when you made an administrator account in Lab 1.2.

Here you'll create groups, add users to groups and set permissions based on those groups.
The way to create a group is

sudo groupadd groupname

In order to verify the creation of the group, go to /etc/group. A good idea is to append users to new groups that you want them in

sudo usermod -aG username groupname

You can remove users from a group using

sudo gpasswd –delete username groupname

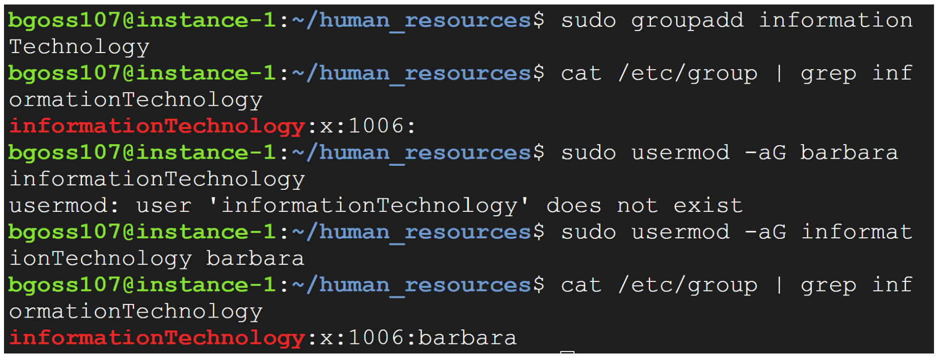
You delete a group using

Sudo groupdel groupname

This set of commands creates the group informationTechnology, searches the file /etc/group for informationTechnology and outputs a group:x:groupID: users.As you see, the first time this is done, there are no users.

Then you'll see a mistake made! The groupname and the username were switched. It's important to read the error messages that are output so you can better understand what you should do next.

The final command shows informationTechnology appended to include barbara.



```
bgoss107@instance-1:~/human_resources$ sudo groupadd information
Technology
bgoss107@instance-1:~/human_resources$ cat /etc/group | grep inf
ormationTechnology
informationTechnology:x:1006:
bgoss107@instance-1:~/human_resources$ sudo usermod -aG barbara
informationTechnology
usermod: user 'informationTechnology' does not exist
bgoss107@instance-1:~/human_resources$ sudo usermod -aG informat
ionTechnology barbara
bgoss107@instance-1:~/human_resources$ cat /etc/group | grep inf
ormationTechnology
informationTechnology:x:1006:barbara
```

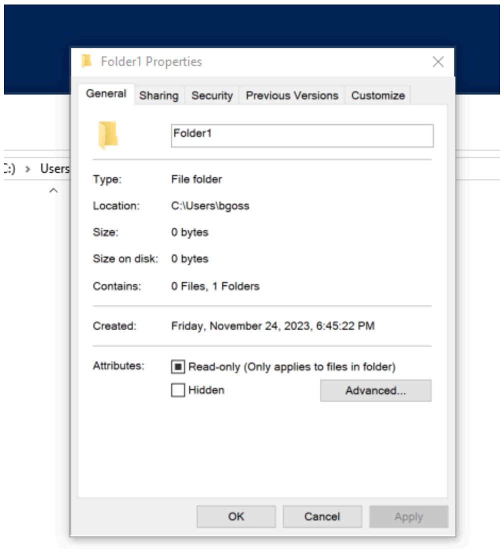
Who else would you include in the informationTechnology group? Add them now.

Topic 5: Ownership of a file/directory: PowerShell & Windows GUI

Windows folder and file permissions are different from Linux. While most Linux distributions will use the same commands and framework (octal, symbolic notation), Windows has three different file frameworks. The two are: File Allocation Table (FAT) which is usually used in external hard drives and High Performance File System (HPFS) which does not have any security implementation available. The one we will be focusing on is New Technology File System (NTFS).

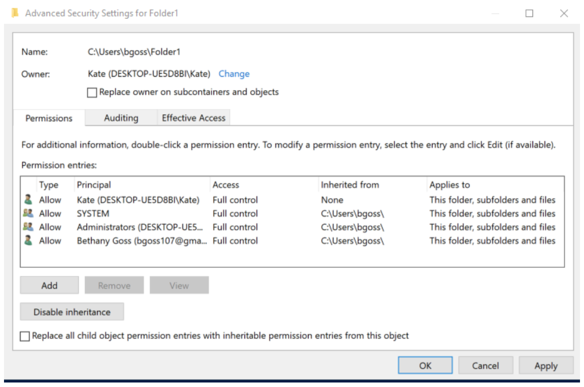
The GUI way of changing ownership of a file/directory

Right click on the file/directory then go to **Properties**
This screen should show up.



Folder1 Properties

Type:	File folder
Location:	C:\Users\bgoss
Size:	0 bytes
Size on disk:	0 bytes
Contains:	0 Files, 1 Folders
Created:	Friday, November 24, 2023, 6:45:22 PM
Attributes:	Read-only (Only applies to files in folder); Hidden



Advanced Security Settings for Folder1

Name: C:\Users\bgoss\Folder1
Owner: Kate (DESKTOP-UE5D8BI\Kate) Change
Replace owner on subcontainers and objects

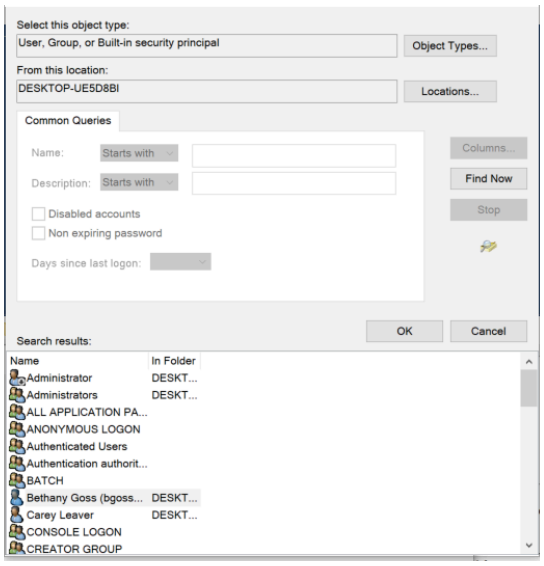
For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from	Applies to
Allow	Kate (DESKTOP-UE5D8BI\Kate)	Full control	None	This folder, subfolders and files
Allow	SYSTEM	Full control	C:\Users\bgoss\	This folder, subfolders and files
Allow	Administrators (DESKTOP-UE5...	Full control	C:\Users\bgoss\	This folder, subfolders and files
Allow	Bethany Goss (bgoss107@gma...	Full control	C:\Users\bgoss\	This folder, subfolders and files

Replace all child object permission entries with inheritable permission entries from this object

Go to **Security** then **Advanced**
Once there you'll see *owner* and a link to another screen that will allow you to change the ownership
Go to **Advanced** in this tab and click **Find Now**
This will allow you to get the exact object name needed

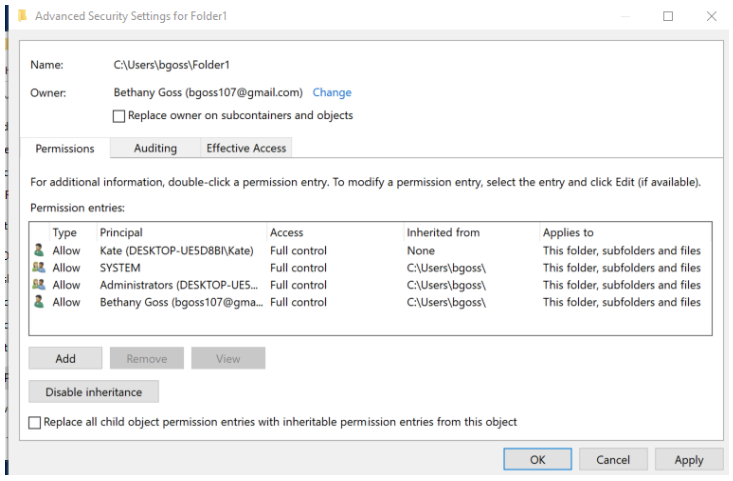


Select this object type:
User, Group, or Built-in security principal
From this location:
DESKTOP-UE5D8BI

Search results:

Name	In Folder
Administrator	DESKT...
Administrators	DESKT...
ALL APPLICATION PA...	
ANONYMOUS LOGON	
Authenticated Users	
Authentication authorit...	
BATCH	
Bethany Goss (bgoss...	DESKT...
Carey Leaver	DESKT...
CONSOLE LOGON	
CREATOR GROUP	

Then you can see the updated owner



Advanced Security Settings for Folder1

Name: C:\Users\bgoss\Folder1
Owner: Bethany Goss (bgoss107@gmail.com) Change
Replace owner on subcontainers and objects

For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from	Applies to
Allow	Kate (DESKTOP-UE5D8BI\Kate)	Full control	None	This folder, subfolders and files
Allow	SYSTEM	Full control	C:\Users\bgoss\	This folder, subfolders and files
Allow	Administrators (DESKTOP-UE5...	Full control	C:\Users\bgoss\	This folder, subfolders and files
Allow	Bethany Goss (bgoss107@gma...	Full control	C:\Users\bgoss\	This folder, subfolders and files

Replace all child object permission entries with inheritable permission entries from this object

The Powershell way:

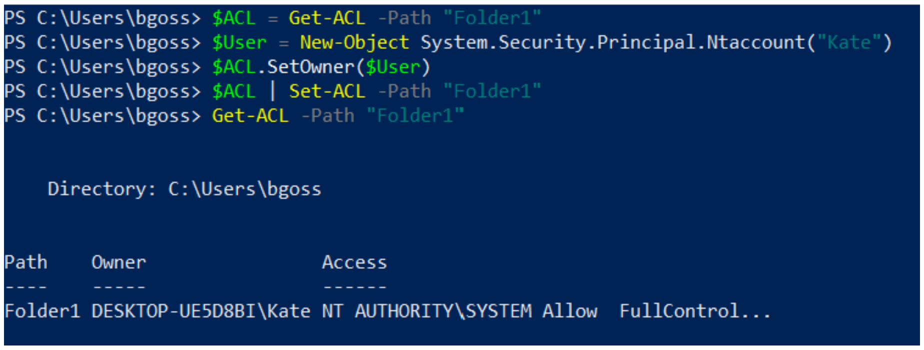
$ACL and **$User** are both variables. You will need to set these variables in each session:

$ACL = Get-Acl -Path "Folder1"
$User = New-Object System.Security.Principal.Ntaccount("Kate")

You will need to have this object as System.Security.Principal.Ntaccount in order for SetOwner to work. The user needs to have these permissions.

Like with other Powershell commands, you will Set, Get & confirm

$ACL.SetOwner($User)
$ACL | Set-Acl -Path "Folder1"
Get-ACL -Path "Folder1"



```
PS C:\Users\bgoss> $ACL = Get-ACL -Path "Folder1"
PS C:\Users\bgoss> $User = New-Object System.Security.Principal.Ntaccount("Kate")
PS C:\Users\bgoss> $ACL.SetOwner($User)
PS C:\Users\bgoss> $ACL | Set-ACL -Path "Folder1"
PS C:\Users\bgoss> Get-ACL -Path "Folder1"


    Directory: C:\Users\bgoss


Path    Owner                Access
----    -----                ------
Folder1 DESKTOP-UE5D8BI\Kate NT AUTHORITY\SYSTEM Allow  FullControl...
```

How would you change a file's ownership? What parts of the code would you change?

Topic 6: Changing access to a file/directory: PowerShell & Windows GUI

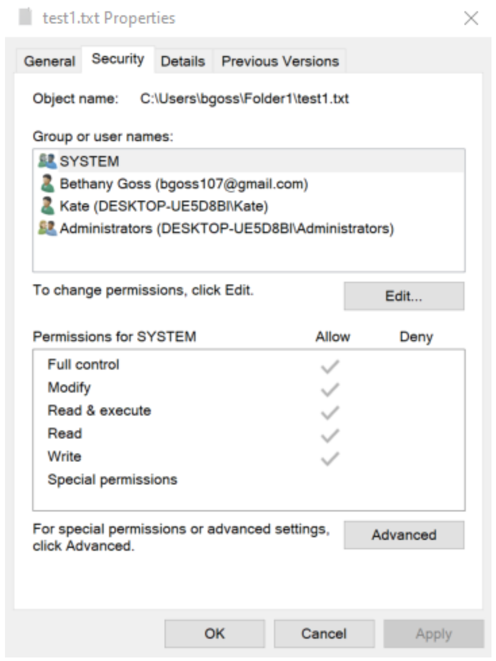
This lab will only discuss basic permissions. There are advanced permissions available, and this is a tutorial that can speak more to those advanced permissions.
Our basic permissions are:

- Full Control - equivalent to default owner privileges in Linux: users can create, delete, modify, and move folders and files
- Modify - cannot move folders or files, cannot delete the folder or file but can create, delete, and change with its files/ sub-folders (if a directory) or file properties (if a file)
- Read & Execute - users can execute the files
- Read - users can view files

- Write - users can append to a file and add files to a directories

GUI version

Again you're going to right click, go to **Properties** and go to the **Security** tab. You will see the basic set of permissions and reference to special permissions.


test1.txt Properties

General | Security | Details | Previous Versions

Object name: C:\Users\bgoss\Folder1\test1.txt

Group or user names:
- SYSTEM
- Bethany Goss (bgoss107@gmail.com)
- Kate (DESKTOP-UE5D8BI\Kate)
- Administrators (DESKTOP-UE5D8BI\Administrators)

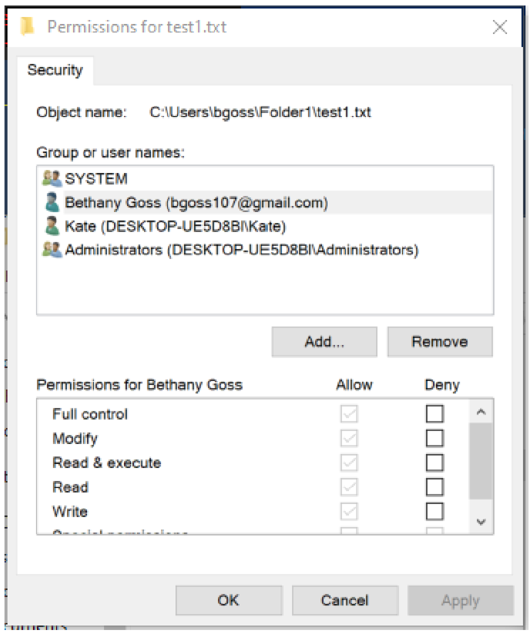
To change permissions, click Edit. [Edit...]

Permissions for SYSTEM	Allow	Deny
Full control	✓	
Modify	✓	
Read & execute	✓	
Read	✓	
Write	✓	
Special permissions		

For special permissions or advanced settings, click Advanced. [Advanced]

[OK] [Cancel] [Apply]

Now you can highlight the specific group or user and click **edit.**


Permissions for test1.txt

Security

Object name: C:\Users\bgoss\Folder1\test1.txt

Group or user names:
- SYSTEM
- Bethany Goss (bgoss107@gmail.com)
- Kate (DESKTOP-UE5D8BI\Kate)
- Administrators (DESKTOP-UE5D8BI\Administrators)

[Add...] [Remove]

Permissions for Bethany Goss	Allow	Deny
Full control	✓	☐
Modify	✓	☐
Read & execute	✓	☐
Read	✓	☐
Write	✓	☐

[OK] [Cancel] [Apply]

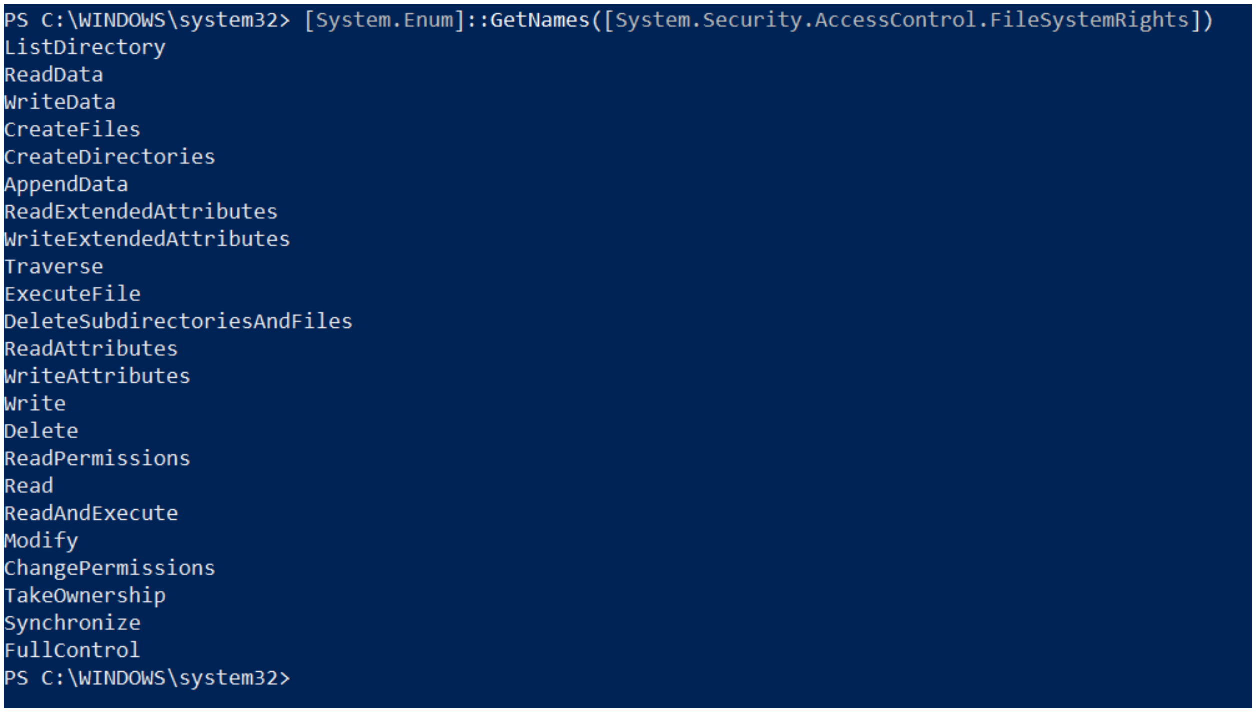
Now you can check and uncheck permissions & add users/groups

Special Permissions can be handled in PowerShell.

If you want to see all potential permissions, you can type

[System.Enum]::GetNames([System.Security.AccessControl.FileSystemRights])

Here is the result of that command:



```
PS C:\WINDOWS\system32> [System.Enum]::GetNames([System.Security.AccessControl.FileSystemRights])
ListDirectory
ReadData
WriteData
CreateFiles
CreateDirectories
AppendData
ReadExtendedAttributes
WriteExtendedAttributes
Traverse
ExecuteFile
DeleteSubdirectoriesAndFiles
ReadAttributes
WriteAttributes
Write
Delete
ReadPermissions
Read
ReadAndExecute
Modify
ChangePermissions
TakeOwnership
Synchronize
FullControl
PS C:\WINDOWS\system32>
```

In order to see a folder permissions

(Get-ACL -Path "Folder1").Access | Format-Table IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -AutoSize

You can use the same method to see a file's permissions.

In order to modify the permissions, the basic setup is this:

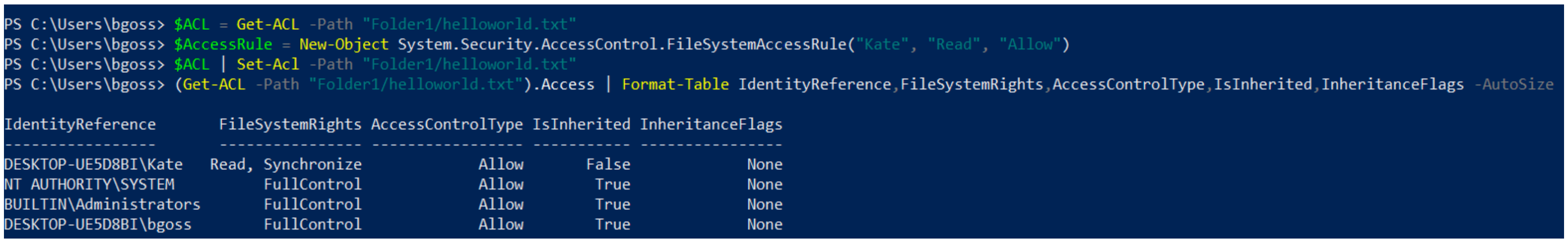
$ACL = Get-ACL -Path "Folder1/helloworld.txt"

$AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("Kate","Read","Allow")

$ACL.SetAccessRule($AccessRule)

$ACL | Set-Acl -Path "Folder1/helloworld.txt"

(Get-ACL -Path "Folder1/helloworld.txt").Access | Format-Table IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -AutoSize



```
PS C:\Users\bgoss> $ACL = Get-ACL -Path "Folder1/helloworld.txt"
PS C:\Users\bgoss> $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("Kate", "Read", "Allow")
PS C:\Users\bgoss> $ACL | Set-Acl -Path "Folder1/helloworld.txt"
PS C:\Users\bgoss> (Get-ACL -Path "Folder1/helloworld.txt").Access | Format-Table IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -AutoSize

IdentityReference       FileSystemRights AccessControlType IsInherited InheritanceFlags
-----------------       ---------------- ----------------- ----------- ----------------
DESKTOP-UE5D8BI\Kate   Read, Synchronize             Allow       False             None
NT AUTHORITY\SYSTEM          FullControl             Allow        True             None
BUILTIN\Administrators       FullControl             Allow        True             None
DESKTOP-UE5D8BI\bgoss        FullControl             Allow        True             None
```

How would you give Kate "modify" permissions?

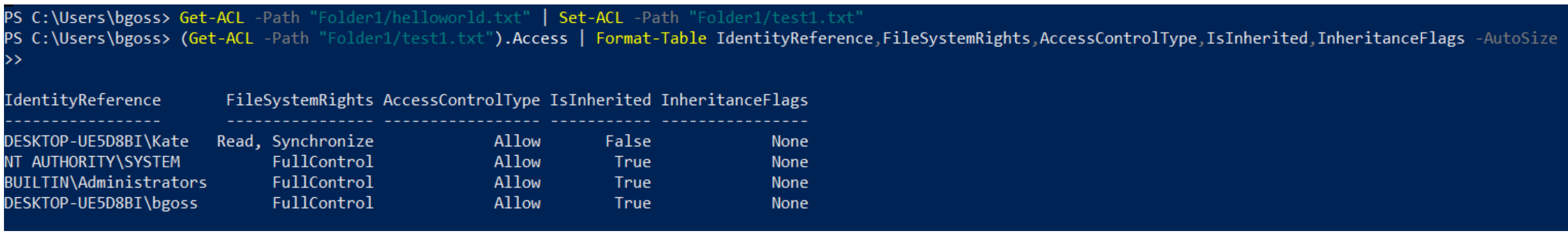
If Kate needs to have the same access on a different file, you don't necessarily need to create another $AccessRule

Instead of copying and pasting the above script you can use the following command to use one file's permissions for another file.

Get-ACL -Path "Folder1/helloworld.txt" | Set-ACL -Path "Folder1/test1.txt"

And Verify the new permissions

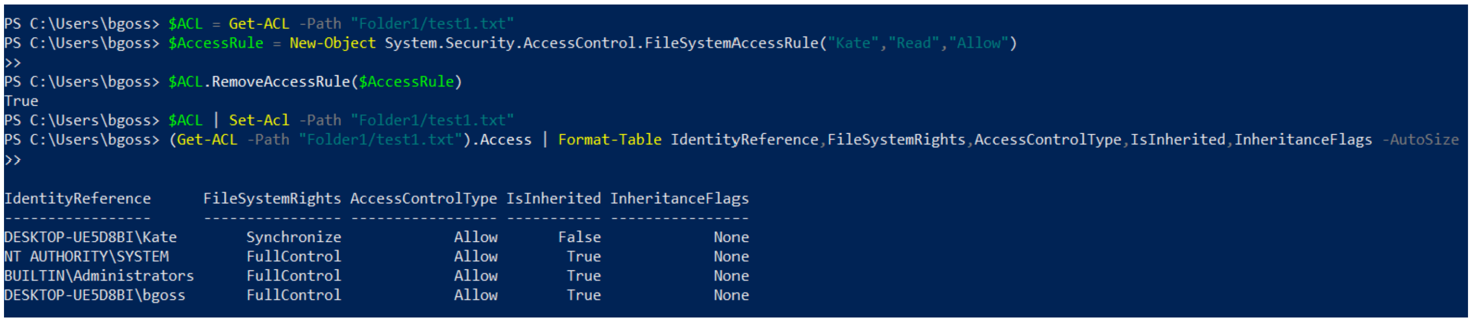
(Get-ACL -Path "Folder1/test1.txt").Access | Format-Table IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -AutoSize



```
PS C:\Users\bgoss> Get-ACL -Path "Folder1/helloworld.txt" | Set-ACL -Path "Folder1/test1.txt"
PS C:\Users\bgoss> (Get-ACL -Path "Folder1/test1.txt").Access | Format-Table IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -AutoSize
>>

IdentityReference       FileSystemRights AccessControlType IsInherited InheritanceFlags
-----------------       ---------------- ----------------- ----------- ----------------
DESKTOP-UE5D8BI\Kate   Read, Synchronize             Allow       False             None
NT AUTHORITY\SYSTEM          FullControl             Allow        True             None
BUILTIN\Administrators       FullControl             Allow        True             None
DESKTOP-UE5D8BI\bgoss        FullControl             Allow        True             None
```

In order to remove permissions, you can follow this set of instructions:

$ACL = Get-ACL -Path "Folder1/test1.txt"
$AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("Kate","Read","Allow")
$ACL.RemoveAccessRule($AccessRule)
$ACL | Set-Acl -Path "Folder1/test1.txt"
(Get-ACL -Path "Folder1/test1.txt").Access | Format-Table IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -AutoSize



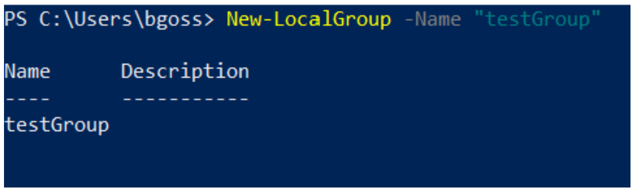
```
PS C:\Users\bgoss> $ACL = Get-ACL -Path "Folder1/test1.txt"
PS C:\Users\bgoss> $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("Kate","Read","Allow")
>>
PS C:\Users\bgoss> $ACL.RemoveAccessRule($AccessRule)
True
PS C:\Users\bgoss> $ACL | Set-Acl -Path "Folder1/test1.txt"
PS C:\Users\bgoss> (Get-ACL -Path "Folder1/test1.txt").Access | Format-Table IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -AutoSize
>>

IdentityReference      FileSystemRights AccessControlType IsInherited InheritanceFlags
-----------------      ---------------- ----------------- ----------- ----------------
DESKTOP-UE5D8BI\Kate        Synchronize             Allow       False             None
NT AUTHORITY\SYSTEM         FullControl             Allow        True             None
BUILTIN\Administrators      FullControl             Allow        True             None
DESKTOP-UE5D8BI\bgoss       FullControl             Allow        True             None
```

Topic 7: Groups in PowerShell

Here are the different kinds of groups that Windows has. We will use Local Groups here.

Creating Groups

New-LocalGroup -Name “testGroup”



```
PS C:\Users\bgoss> New-LocalGroup -Name "testGroup"

Name      Description
----      -----------
testGroup
```

Adding to the groups

Add-LocalGroupMember -Group "testGroup" -Member "bgoss"

Confirming group members

Get-LocalGroupMember -Group "testGroup"

Removing group members

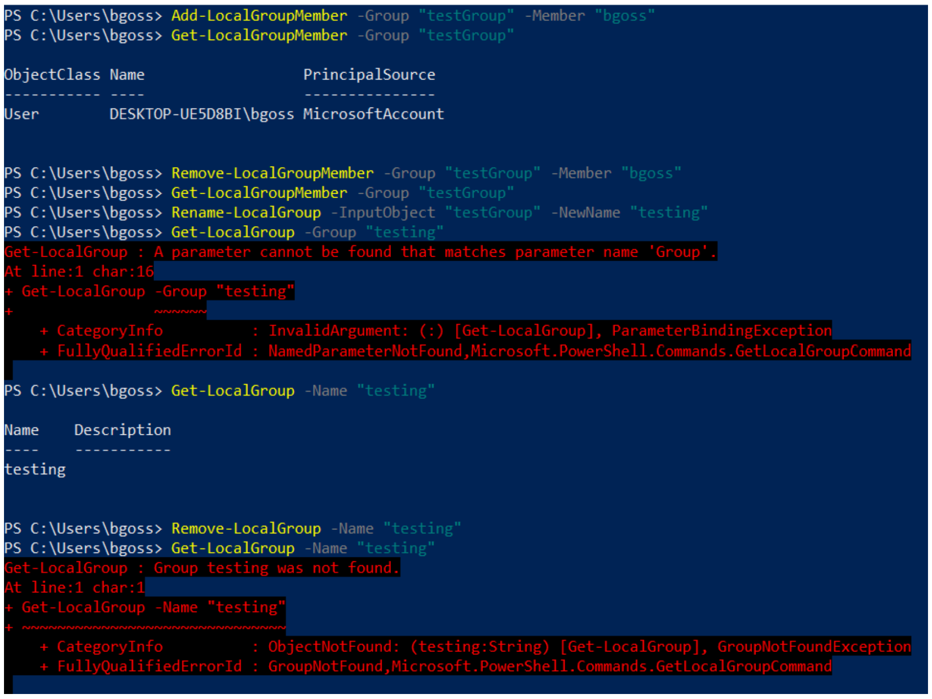
Remove-LocalGroupMember -Group "testGroup" -Member "bgoss"

Renaming the group

Rename-LocalGroup -InputObject "testGroup" -NewName "testing"

Removing the group completely

Remove-LocalGroup -Name "testing"



```
PS C:\Users\bgoss> Add-LocalGroupMember -Group "testGroup" -Member "bgoss"
PS C:\Users\bgoss> Get-LocalGroupMember -Group "testGroup"

ObjectClass Name                 PrincipalSource
----------- ----                 ---------------
User        DESKTOP-UE5D8BI\bgoss MicrosoftAccount


PS C:\Users\bgoss> Remove-LocalGroupMember -Group "testGroup" -Member "bgoss"
PS C:\Users\bgoss> Get-LocalGroupMember -Group "testGroup"
PS C:\Users\bgoss> Rename-LocalGroup -InputObject "testGroup" -NewName "testing"
PS C:\Users\bgoss> Get-LocalGroup -Group "testing"
Get-LocalGroup : A parameter cannot be found that matches parameter name 'Group'.
At line:1 char:16
+ Get-LocalGroup -Group "testing"
+                ~~~~~~
    + CategoryInfo          : InvalidArgument: (:) [Get-LocalGroup], ParameterBindingException
    + FullyQualifiedErrorId : NamedParameterNotFound,Microsoft.PowerShell.Commands.GetLocalGroupCommand

PS C:\Users\bgoss> Get-LocalGroup -Name "testing"

Name    Description
----    -----------
testing


PS C:\Users\bgoss> Remove-LocalGroup -Name "testing"
PS C:\Users\bgoss> Get-LocalGroup -Name "testing"
Get-LocalGroup : Group testing was not found.
At line:1 char:1
+ Get-LocalGroup -Name "testing"
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (testing:String) [Get-LocalGroup], GroupNotFoundException
    + FullyQualifiedErrorId : GroupNotFound,Microsoft.PowerShell.Commands.GetLocalGroupCommand
```

You can treat these groups like users when it comes to giving them permissions in the GUI method.

Windows permissions, users, and permissions all come from its Active Directory. There are ways to interact with this and it is a bit more complicated than will be discussed in this general overview, but go to the resources topic for information to continue your education further.

It may be unlikely that you are able to access the active directory because it's usually meant for on premises enterprise working environments.

However if you have used or are going to use Azure services, you will definitely have a version of active directory available to you.

Active directory offers more flexibility in terms of permissions, types of groups: there are two different kinds of administrative permissions, and bulk permissions assignment.

If you do want to see if you can work with active directory now, you can search for "Adminstrative Tools" and see if you can find "Active Directory Users and Computers." If you can, then it's available to you!

Otherwise, this is where our conversation about permissions, controls, users, and groups ends.

Topic 8: Best Practices for Permissions Management

1. Only give users the permissions they need
2. Some permissions are connected, be careful of what permissions you are giving out.
3. Before you start handing out or changing permissions, have a very clear and written down plan of who should get access to what permissions.
   If Roberta wants access to the CFO folder, ask why her job duties require her to need that access.

Now it is time to set some permissions!

1. What command would you use to change the owner of a directory/folder?
2. Set all the files in **Human Resources** to **read** only for everyone, including the assigned group and the owner. What were the original file permissions? What are they now?
3. Change the umask to give **read and write** permissions to all files in **Information Technology** to groups and all users. Then use **umask** to change all those permissions.What was the resulting umask?
    a. For Windows, look at effective access for the **Information Technology** folder, what does that information tell you?
4. How would you give the accountant, Bob, ownership of all staff files in the **Finance** folder?
5. Reset all the files in the **Human Resources** directory to their original permissions. What command or commands did you use to do this?
6. Barbara the IT manager requests **write and execute** permissions for **Human Resourcces** files. Do you give Barbara **write and execute** permissions for these files? Why or Why Not?
7. Roberta has put in a request for ownership of all the staff files located in the **Information Technology** folder. She wants to look at job duties, recent performance trends and other markers to see if the department should hire more staff. Do you give her ownership over these files? Why or why not?
8. Give **Human Resources** and **Information Technology** the same permissions. How did you go about that?

Congratulations! You have now learned how to manage permissions in both Ubuntu Linux and Windows 10. Remember that every Linux distribution is slightly different, but the process should be relatively similar with only minor changes across any debian-based distribution.

In order to receive credit for this lab, please take the time to record your screen for a short 1 to 3 minute video showing the permissions you've just changed so you can show your **lecturer.**