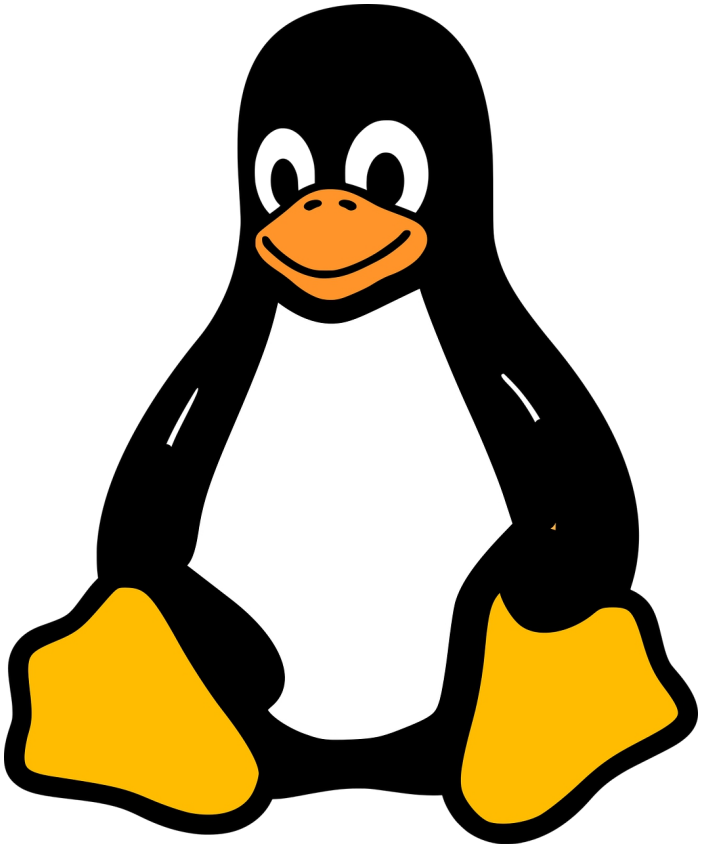
HackerFrogs Afterschool Linux Basics /w TryHackMe

Class:
Linux OS Operations

Workshop Number:
AS-LIN-03

Document Version:
1.0

Special Requirements:
Registered account
at tryhackme.com

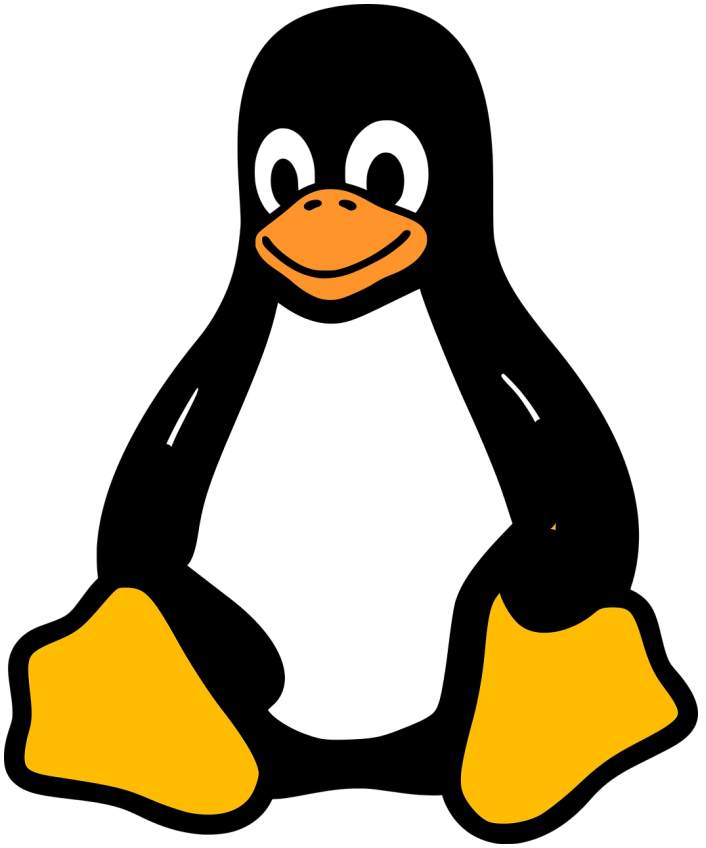


What We Learned In The Previous Workshop

Hey there HackerFrogs!

This is the third intro to Linux OS Operations workshop.

In the previous workshop we learned about the following Linux commands:



Find Command

The Find command is used to search for files on the system. It can used with many different arguments and flags to refine the search parameters.

Grep Command

The Grep command searches within the contents of files for specified strings. It is very commonly used to pick out specific words or phrases.

Sort Command

The Sort command takes all of the lines contained within a given file and returns them in alphabetical / numerical order.

Uniq Command

The Uniq command takes all of the lines in a file and removes any lines with identical contents to the one above it. This command is very useful for removing consecutive blank lines in a given file

Command Piping

Command piping is the process of passing the output of one command to the input of another command (via use of the Linux pipe | character)

Strings Command

The Strings command is used to return human-readable text from files. It is often used to find text inside of files that also contain both text and binary data.

Base64 Command

The Base64 command encodes / decodes data according to the Base64 codec format. It is often used to convert data for transmission across computer networks.

0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	I	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

TryHackMe

TryHackMe is a popular cybersecurity education platform with educational modules (called rooms) that span a great number of cybersecurity disciplines.

TryHackMe

In this workshop, we'll be making use of the one of TryHackMe's interactive rooms to learn more about using the Linux operating system.

TryHackMe

But if we want use the TryHackMe platform, we must first have a valid user account for the website. If needed, signup for an account at the following URL:

https://tryhackme.com/signup

Let's Do Some Pre-Studying

Since the TryHackMe virtual lab machines have a relatively short time limit for free user accounts, we'll do some pre-studying on another web-accessible Linux console first, so we can maximize our time at TryHackMe later.

Let's Access a Linux Console

We can access a web-based Linux console for our educational needs at the following URL:

https://bellard.org/jslinux/index.html

On to the TryHackMe Room!

Now that we've studied the commands we will encounter in the TryHackMe room, let's access the room at the following URL:

https://tryhackme.com/room/linuxfundamentalspart2

Note: We'll need to be logged in to complete the room tasks.

The Help Flag

```
[root@localhost ~]# cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonempty output lines, overrides -n
```

The **--help** flag can be used as part of any command to return a relatively brief explanation as to how to use the command.

The Man Command

```
[root@localhost ~]# man cat
CAT(1)                          User Commands

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...
```

The **man** command brings up the manual document for the command specified in the argument. The documentation is more in-depth than the **–help** flag.

The & Operator

```
[root@localhost ~]# find / -name shyhat &
[1] 140
[root@localhost ~]#
```

The **&** operator is used to execute a command as a background process, which means that the CLI terminal will be free to execute other commands while the process is running.

The && Operator

```
[root@localhost ~]# whoami && pwd
root
/root
```

The **&&** operator is used to execute the two commands to the left and right of the operator, one after another, but will only execute the second command if the first command executed successfully.

The > Operator

```
[root@localhost ~]# echo test > test.txt
[root@localhost ~]# cat test.txt
test
```

The **>** operator takes the output of the command and redirects it into a file. If there is already a file with the same name in the directory, it will be overwritten.

The >> Operator

```
[root@localhost ~]# echo test1 >> test.txt
[root@localhost ~]# echo test2 >> test.txt
[root@localhost ~]# cat test.txt
test1
test2
```

The **>>** operator works in a very similar fashion to the **>** operator, but if a file with the same name exists in the directory, the output is added to the file instead of overwriting the file.

Whoami Command

```
[root@localhost ~]# whoami
root
```

The **whoami** command is used to return the name of the current user. This is important because most user accounts only have access to certain portions of the file system.

Touch Command

```
[root@localhost ~]# touch test.txt
[root@localhost ~]# ls
test.txt
```

The **touch** command is used to create empty files with the file name specified in the argument.

Mkdir Command

```
[root@localhost ~]# mkdir test
[root@localhost ~]# cd test
[root@localhost test]# pwd
/root/test
```

The **mkdir** command command is used to create a directory with the directory name specified in the argument.

Cp Command

```
[root@localhost ~]# echo 'we copied test.txt to another directory!'> test.txt
[root@localhost ~]# cp test.txt test
[root@localhost ~]# cd test && cat test.txt
we copied test.txt to another directory!
```

The **cp** command copies files from one directory to another. The first argument is the file to be copied, and the second argument is the directory the file is to be copied to.

Mv Command

```
[root@localhost ~]# ls
test1.txt
[root@localhost ~]# cat test2.txt
cat: test2.txt: No such file or directory
[root@localhost ~]# mv test1.txt test2.txt
[root@localhost ~]# cat test2.txt
test text
```

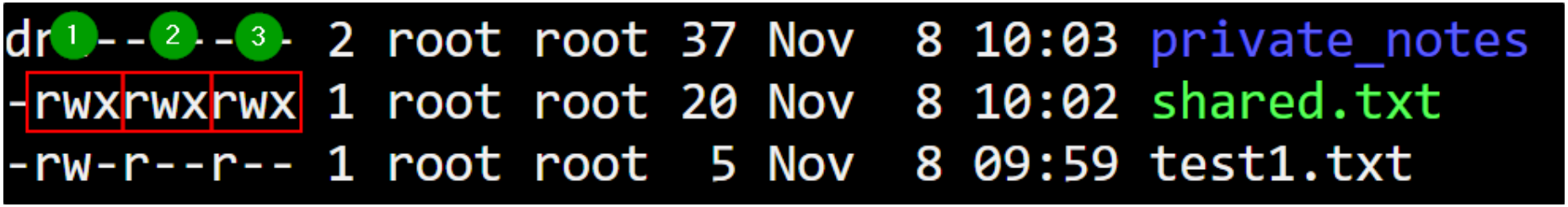
The **mv** command is used in a similar fashion to the **cp** command, except the original file will not remain in its original directory. This command is also used for modifying the names of files.

Rm Command

```
[root@localhost ~]# ls
test.txt
[root@localhost ~]# rm test.txt
[root@localhost ~]# ls
[root@localhost ~]#
```

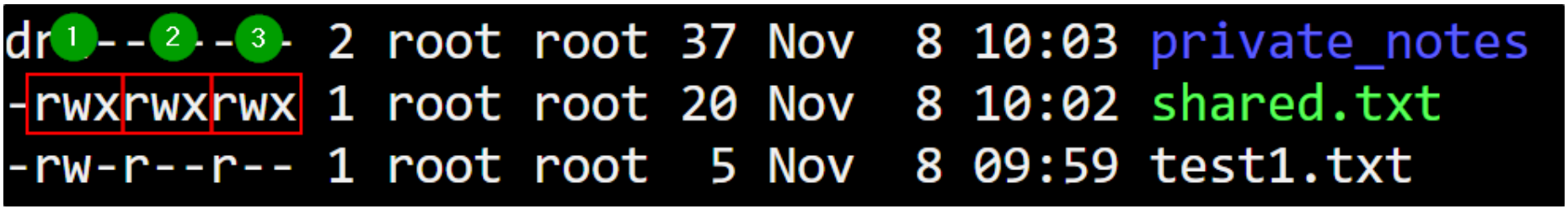
The **rm** command is used to remove files from the filesystem. Directories cannot be removed using this command unless the **-r** switch is used.

Linux File Permissions



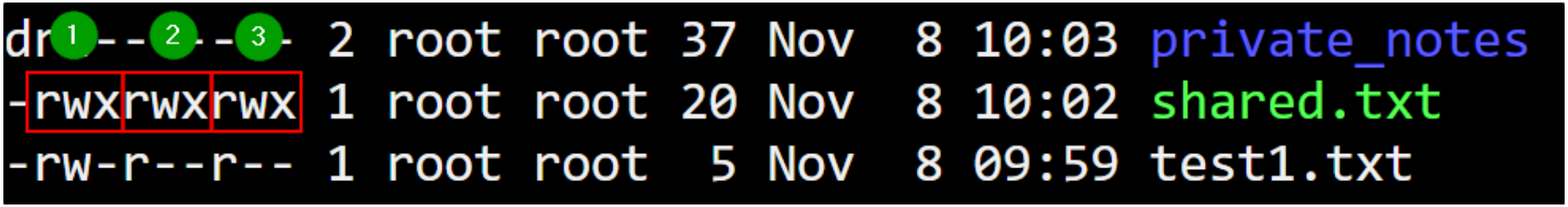
Each file in the Linux filesystem has different (r)ead, (w)rite, and e(x)ecute permissions, depending on whether a user is (1), the file owner, (2), part of a certain group, or (3), any other user.

Linux File Permissions



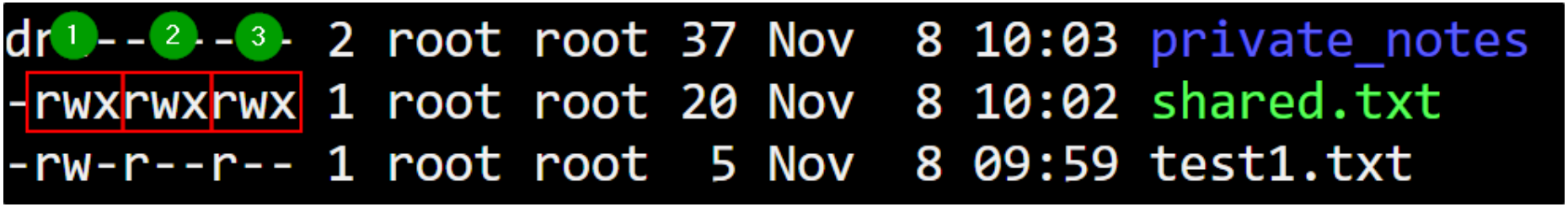
Each file in the Linux filesystem has different (r)ead, (w)rite, and e(x)ecute permissions, depending on whether a user is (1), the file owner, (2), part of a certain group, or (3), any other user.

Linux File Permissions



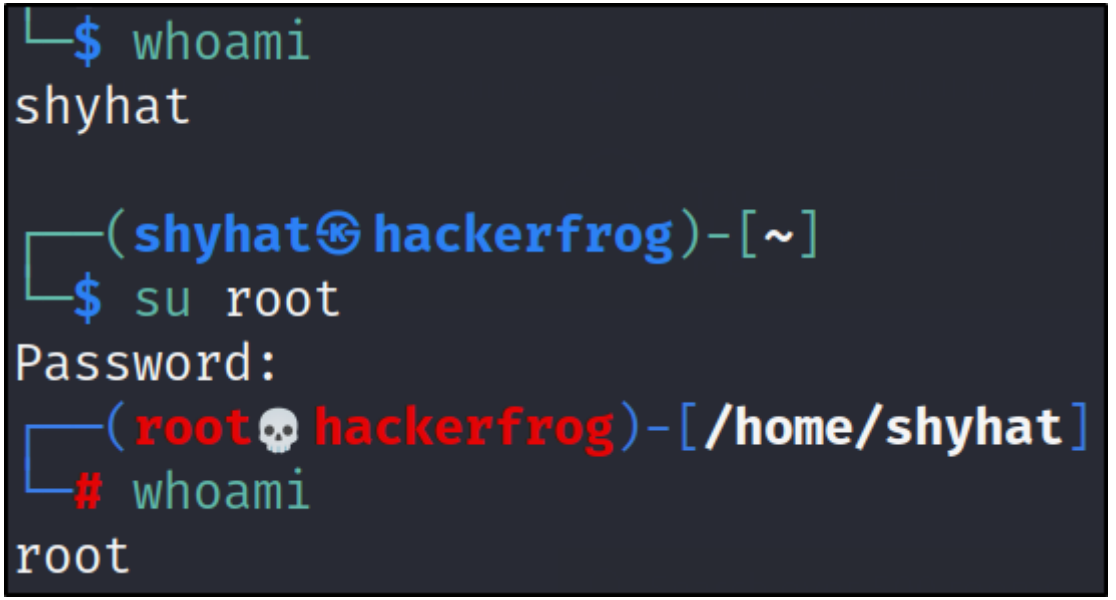
Each file in the Linux filesystem has different (r)ead, (w)rite, and e(x)ecute permissions, depending on whether a user is (1), the file owner, (2), part of a certain group, or (3), any other user.

Linux File Permissions



Each file in the Linux filesystem has different (r)ead, (w)rite, and e(x)ecute permissions, depending on whether a user is (1), the file owner, (2), part of a certain group, or (3), any other user.

Su Command



The **su** (switch user) command is used to switch between active user accounts during a CLI session. When using the **su** command, the password of the user account being accessed must be entered.

Noteworthy Linux File Directories

/etc where system files (user passwords) are stored

/var where log files and database files are stored

/root typically the folder with the most secure access

/tmp all files here are deleted on a regular basis

/home where individual user account files are stored

Noteworthy Linux File Directories

/etc	where system files (user passwords) are stored
/var	where log files and database files are stored
/root	typically the folder with the most secure access
/tmp	all files here are deleted on a regular basis
/home	where individual user account files are stored

Noteworthy Linux File Directories

/etc where system files (user passwords) are stored

/var where log files and database files are stored

/root typically the folder with the most secure access

/tmp all files here are deleted on a regular basis

/home where individual user account files are stored

Noteworthy Linux File Directories

/etc	where system files (user passwords) are stored
/var	where log files and database files are stored
/root	typically the folder with the most secure access
/tmp	all files here are deleted on a regular basis
/home	where individual user account files are stored

Noteworthy Linux File Directories

/etc	where system files (user passwords) are stored
/var	where log files and database files are stored
/root	typically the folder with the most secure access
/tmp	all files here are deleted on a regular basis
/home	where individual user account files are stored

Noteworthy Linux File Directories

/etc where system files (user passwords) are stored

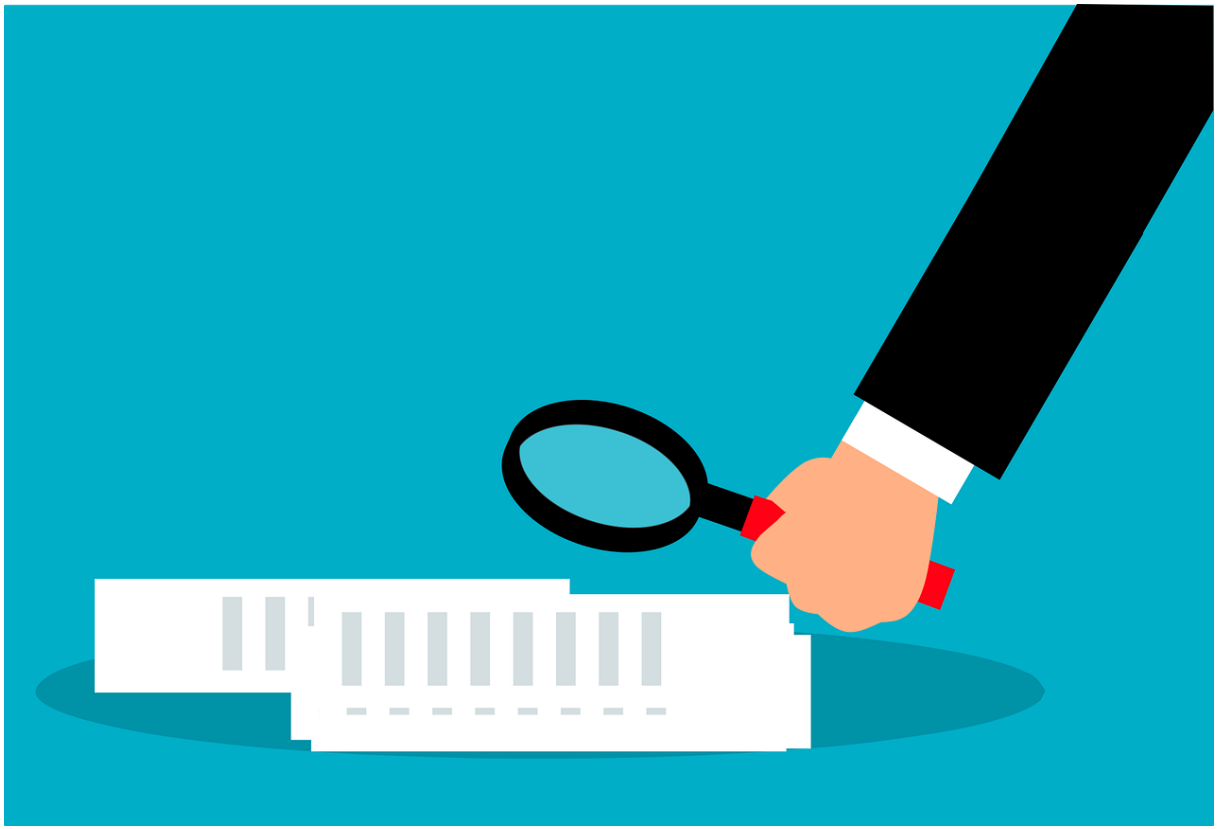
/var where log files and database files are stored

/root typically the folder with the most secure access

/tmp all files here are deleted on a regular basis

/home where individual user account files are stored

Summary



Let's review the Linux commands we learned in this workshop:

The Help Flag

```
[root@localhost ~]# cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonempty output lines, overrides -n
```

The **--help** flag can be used as part of any command to return a relatively brief explanation as to how to use the command.

The Man Command

```
[root@localhost ~]# man cat
CAT(1)                          User Commands

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...
```

The **man** command brings up the manual document for the command specified in the argument. The documentation is more in-depth than the **–help** flag.

The & Operator

```
[root@localhost ~]# find / -name shyhat &
[1] 140
[root@localhost ~]#
```

The **&** operator is used to execute a command as a background process, which means that the CLI terminal will be free to execute other commands while the process is running.

The && Operator

```
[root@localhost ~]# whoami && pwd
root
/root
```

The **&&** operator is used to execute the two commands to the left and right of the operator, one after another, but will only execute the second command if the first command executed successfully.

The > Operator

```
[root@localhost ~]# echo test > test.txt
[root@localhost ~]# cat test.txt
test
```

The **>** operator takes the output of the command and redirects it into a file. If there is already a file with the same name in the directory, it will be overwritten.

The >> Operator

```
[root@localhost ~]# echo test1 >> test.txt
[root@localhost ~]# echo test2 >> test.txt
[root@localhost ~]# cat test.txt
test1
test2
```

The **>>** operator works in a very similar fashion to the **>** operator, but if a file with the same name exists in the directory, the output is added to the file instead of overwriting the file.

Whoami Command

```
[root@localhost ~]# whoami
root
```

The **whoami** command is used to return the name of the current user. This is important because most user accounts only have access to certain portions of the file system.

Touch Command

```
[root@localhost ~]# touch test.txt
[root@localhost ~]# ls
test.txt
```

The **touch** command is used to create empty files with the file name specified in the argument.

Mkdir Command

```
[root@localhost ~]# mkdir test
[root@localhost ~]# cd test
[root@localhost test]# pwd
/root/test
```

The **mkdir** command command is used to create a directory with the directory name specified in the argument.

Cp Command

```
[root@localhost ~]# echo 'we copied test.txt to another directory!'> test.txt
[root@localhost ~]# cp test.txt test
[root@localhost ~]# cd test && cat test.txt
we copied test.txt to another directory!
```

The **cp** command copies files from one directory to another. The first argument is the file to be copied, and the second argument is the directory the file is to be copied to.

Mv Command

```
[root@localhost ~]# ls
test1.txt
[root@localhost ~]# cat test2.txt
cat: test2.txt: No such file or directory
[root@localhost ~]# mv test1.txt test2.txt
[root@localhost ~]# cat test2.txt
test text
```

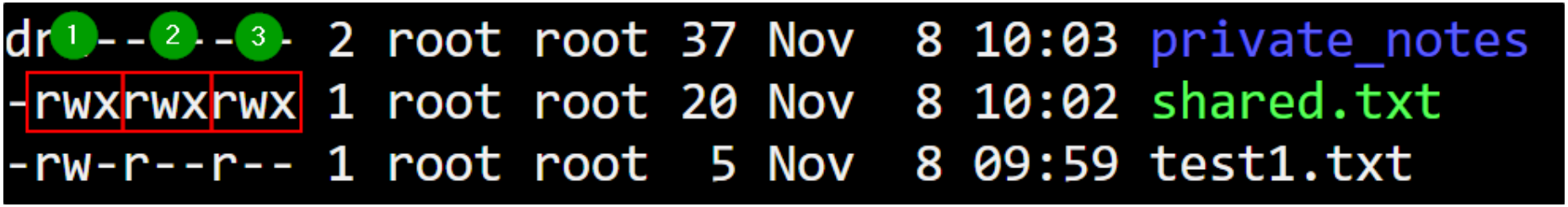
The **mv** command is used in a similar fashion to the **cp** command, except the original file will not remain in its original directory. This command is also used for modifying the names of files.

Rm Command

```
[root@localhost ~]# ls
test.txt
[root@localhost ~]# rm test.txt
[root@localhost ~]# ls
[root@localhost ~]#
```

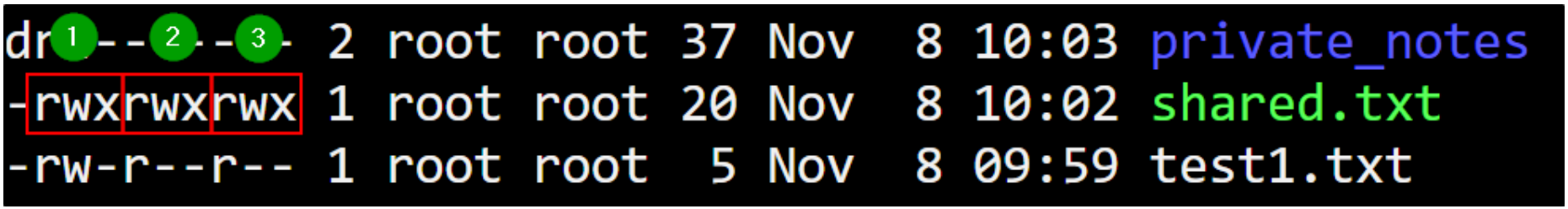
The **rm** command is used to remove files from the filesystem. Directories cannot be removed using this command unless the **-r** switch is used.

Linux File Permissions



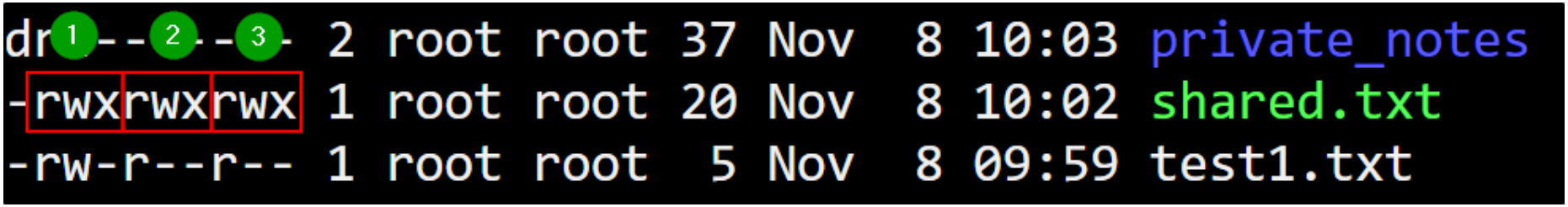
Each file in the Linux filesystem has different (r)ead, (w)rite, and e(x)ecute permissions, depending on whether a user is (1), the file owner, (2), part of a certain group, or (3), any other user.

Linux File Permissions



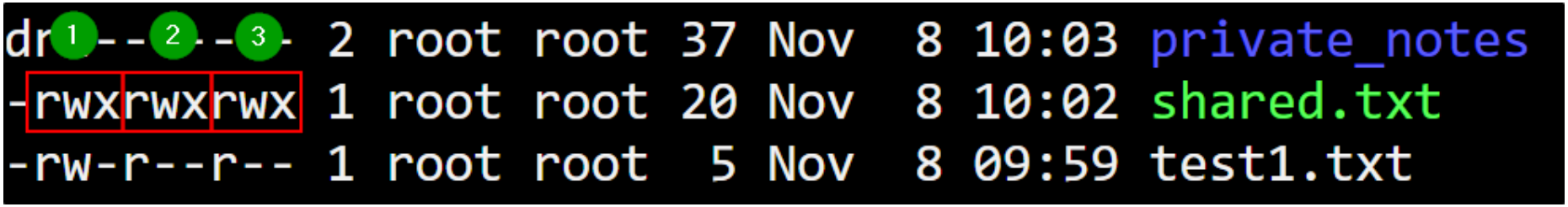
Each file in the Linux filesystem has different (r)ead, (w)rite, and e(x)ecute permissions, depending on whether a user is (1), the file owner, (2), part of a certain group, or (3), any other user.

Linux File Permissions



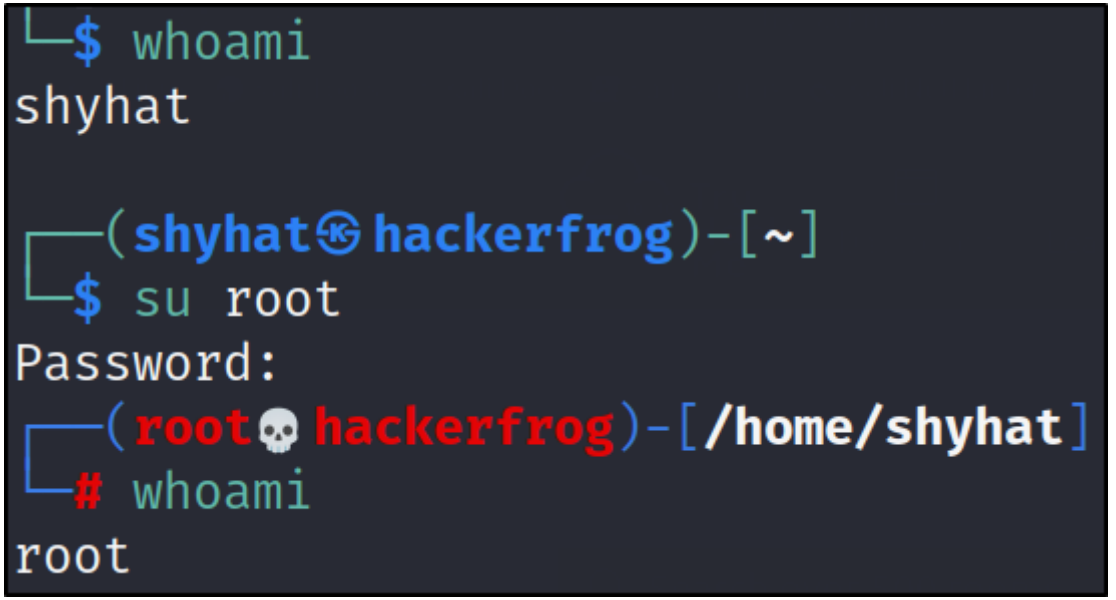
Each file in the Linux filesystem has different (r)ead, (w)rite, and e(x)ecute permissions, depending on whether a user is (1), the file owner, (2), part of a certain group, or (3), any other user.

Linux File Permissions



Each file in the Linux filesystem has different (r)ead, (w)rite, and e(x)ecute permissions, depending on whether a user is (1), the file owner, (2), part of a certain group, or (3), any other user.

Su Command



The **su** (switch user) command is used to switch between active user accounts during a CLI session. When using the **su** command, the password of the user account being accessed must be supplied.

Noteworthy Linux File Directories

/etc	where system files are stored
/var	where log files and database files are stored
/root	typically the folder with the most secure access
/tmp	all files here are deleted on a regular basis
/home	where individual user account files are stored

Noteworthy Linux File Directories

/etc	where system files are stored
/var	where log files and database files are stored
/root	typically the folder with the most secure access
/tmp	all files here are deleted on a regular basis
/home	where individual user account files are stored

Noteworthy Linux File Directories

/etc where system files are stored

/var where log files and database files are stored

/root typically the folder with the most secure access

/tmp all files here are deleted on a regular basis

/home where individual user account files are stored

Noteworthy Linux File Directories

/etc where system files are stored

/var where log files and database files are stored

/root typically the folder with the most secure access

/tmp all files here are deleted on a regular basis

/home where individual user account files are stored

Noteworthy Linux File Directories

/etc where system files are stored

/var where log files and database files are stored

/root typically the folder with the most secure access

/tmp all files here are deleted on a regular basis

/home where individual user account files are stored

Noteworthy Linux File Directories

/etc where system files are stored

/var where log files and database files are stored

/root typically the folder with the most secure access

/tmp all files here are deleted on a regular basis

/home where individual user account files are stored

What's Next?

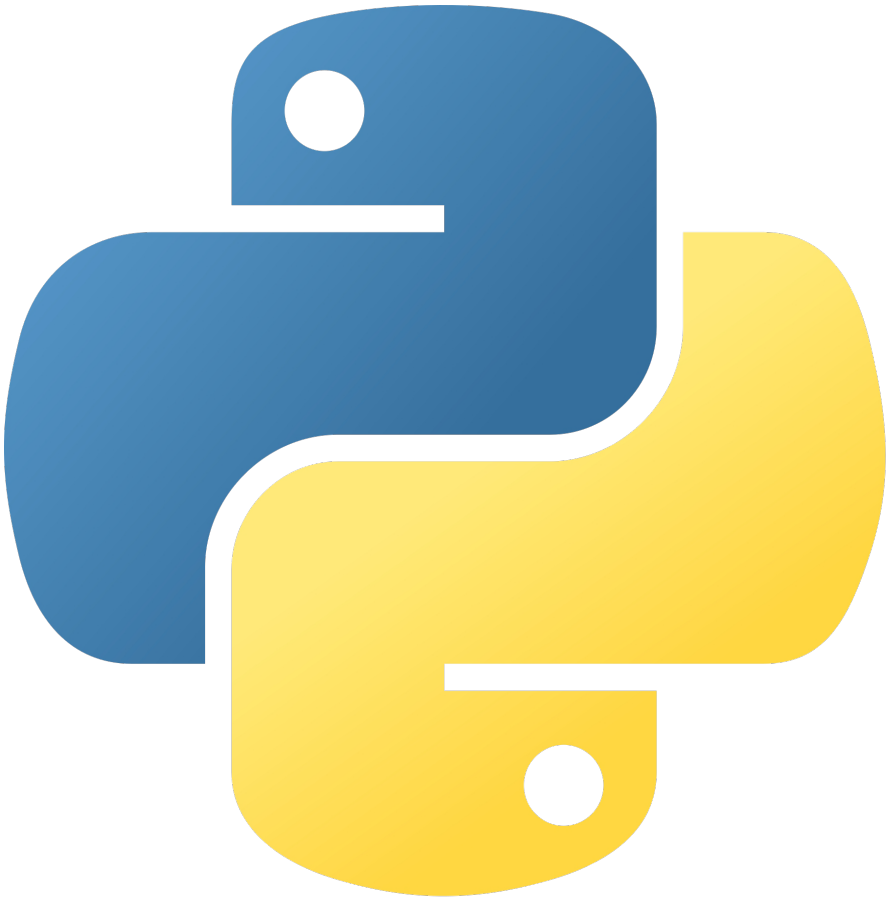
This is the last class for Linux OS operations, but if we want to learn more about working with Linux, we could continue with the Linux Fundamentals series of rooms at TryHackMe.

Try Hack Me

What's Next?

In the next workshop, we'll be starting a new subject:

Programming



Extra Credit

Looking for more study material on this workshop's topics?

See this video's description for links to supplemental documents and exercises!

Until Next Time, HackerFrogs!