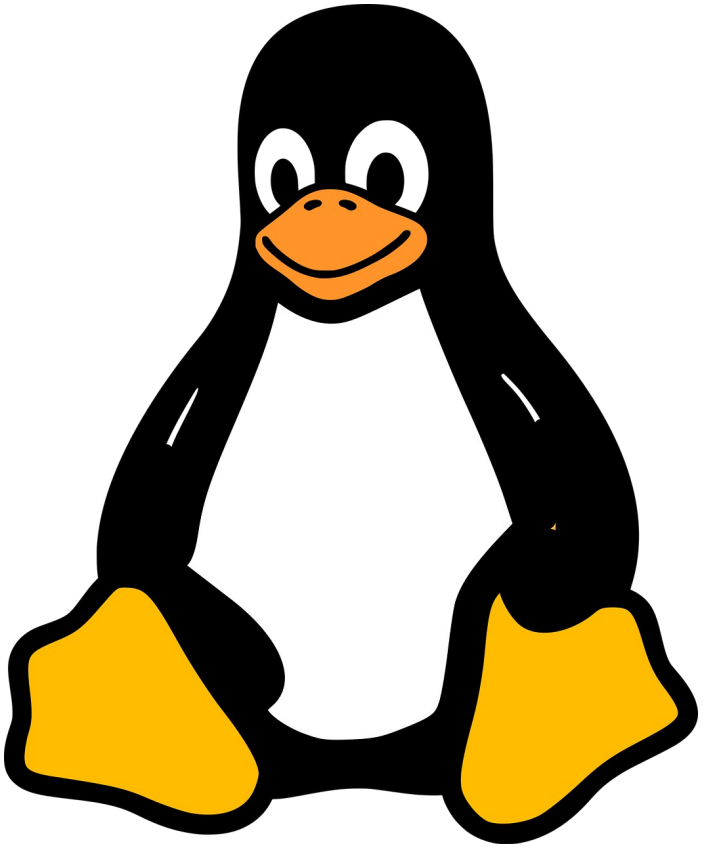
HackerFrogs Afterschool
Linux Basics /w PicoCTF: Part 4

Class:
Linux OS Operations

Workshop Number:
AS-LIN-04

Document Version:
1.75

Special Requirements:
Registered account
at tryhackme.com

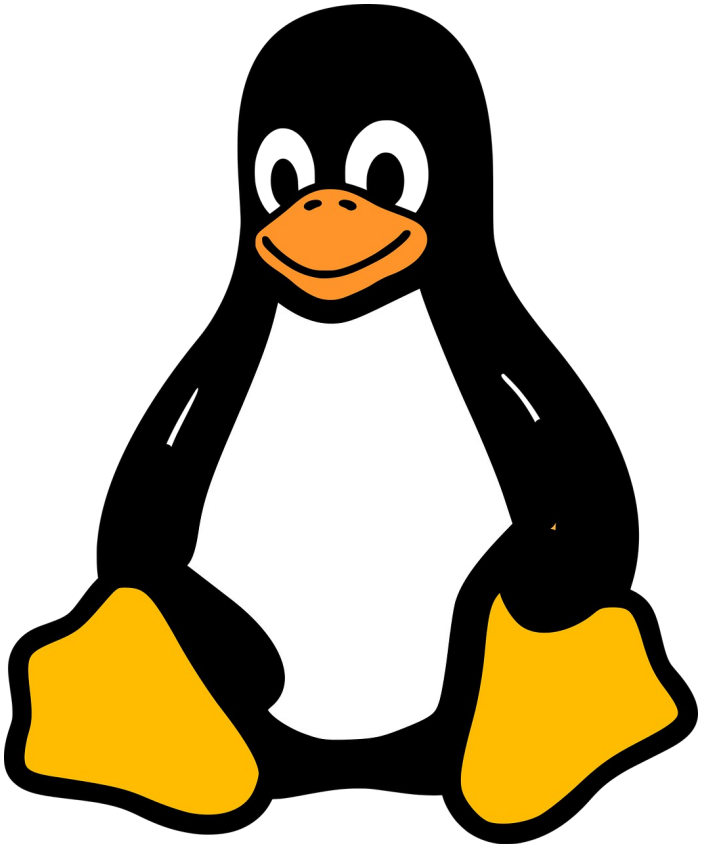


What We Learned In The Previous Workshop

Hey there HackerFrogs!

This is the fourth intro to Linux OS Operations workshop.

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



Netcat Command

Netcat is a versatile computer networking program, and one common use of netcat (with picoCTF challenges) is to connect to non-standard services on remote servers

Command Piping

Command piping is the process of passing the output of one command into the input of a second command (via use of the Linux pipe | character)

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	/

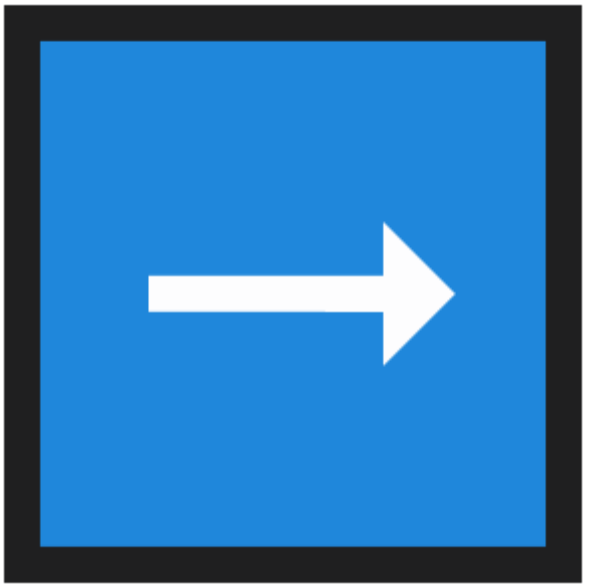
Echo Command

The **echo** command is used to create output equal to whatever is supplied to it.

It can be used to supply input to another command through command piping

Output Redirection

Output redirection is a feature in Linux which can be used to direct the output of commands and write that output to a file.



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

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.

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.

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.

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
```

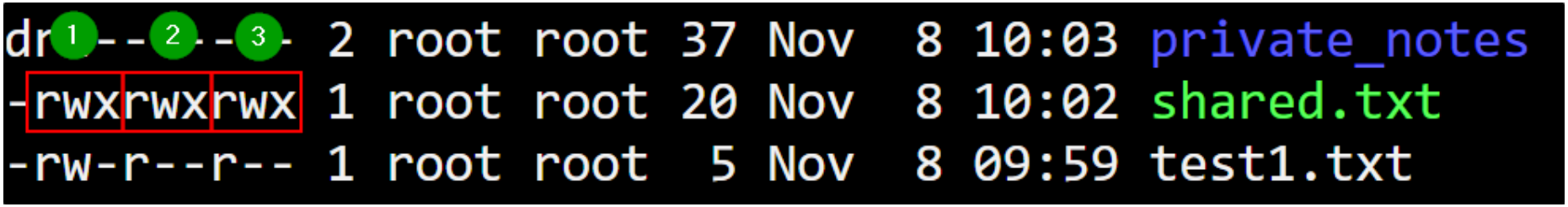
The **--help** flag can be used as part of any command to return a relatively brief explanation as to how to use the command.

Let learn with PicoCTF Wave a Flag

Let's practice some of the commands we've learned with the following challenge:

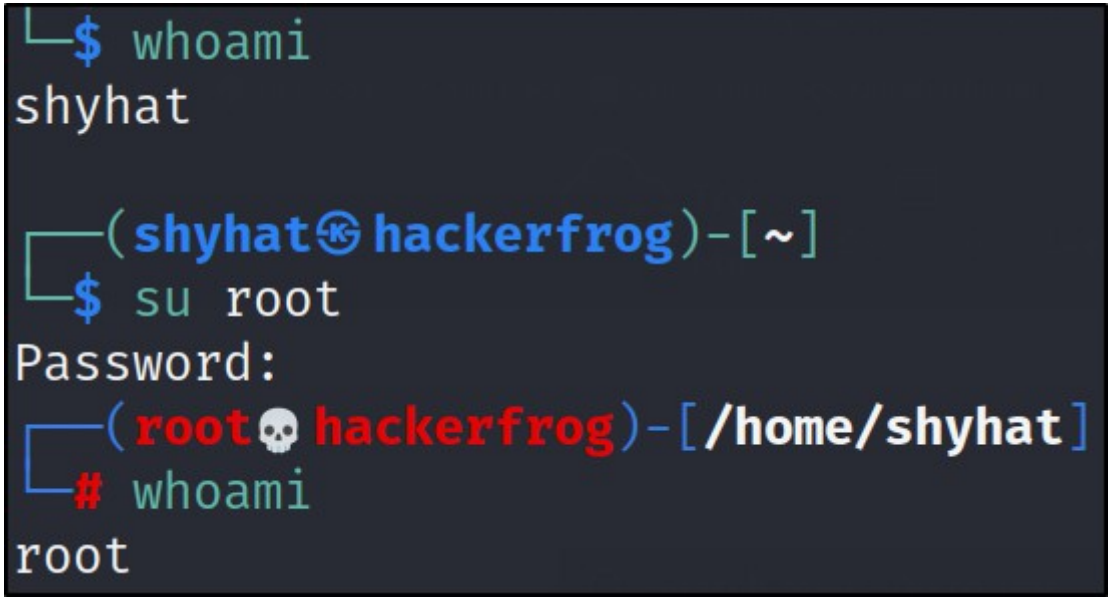
https://play.picoctf.org/practice/challenge/170

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.

Let practice with JS Linux

Let's practice working with different user accounts with JS Linux. Copy the command to create a new user account here:

https://github.com/theshyhat/hackerfrogs/blob/main/af

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

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]...
```

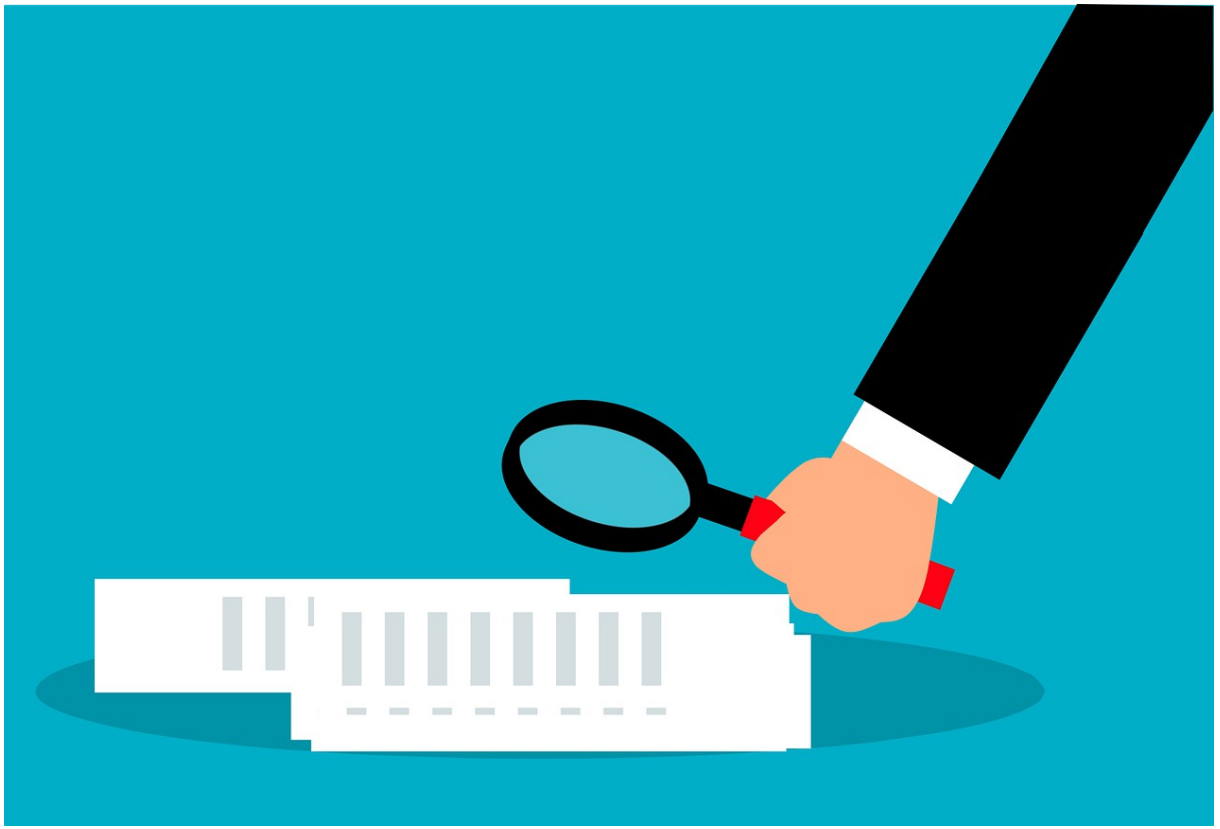
The **man** command brings up the manual document for the command specified in the argument. The documentation is more in-depth than the **–help** flag.

Let learn with PicoCTF Useless

Let's practice some of the commands we've learned with the following challenge:

https://play.picoctf.org/practice/challenge/384

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.

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.

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
```

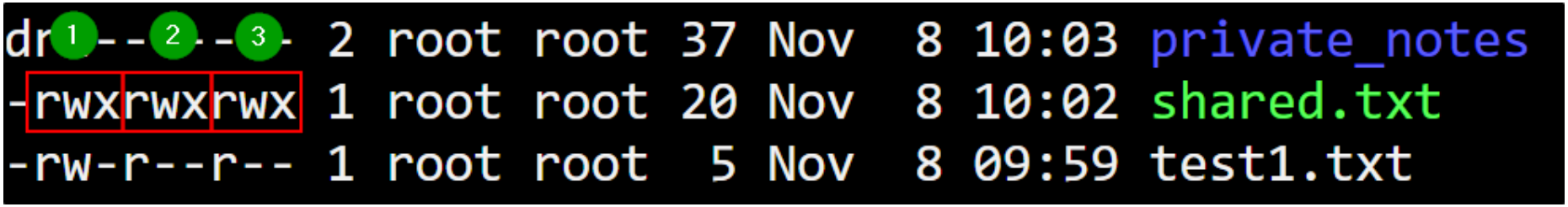
The **mv** command is used in a similar fashion to the **cp** command, except the original file will not remain in its original directory. This command is also used for modifying the names of files.

Rm Command

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

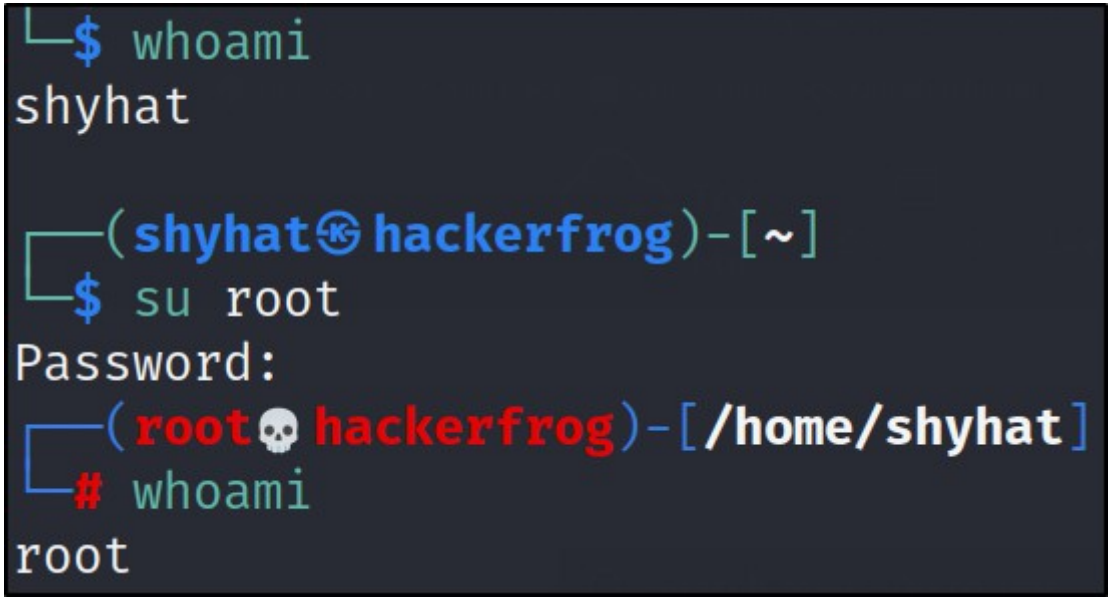
The **rm** command is used to remove files from the filesystem. Directories cannot be removed using this command unless the **-r** switch is used.

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.

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

What's Next?

In the final Linux OS operations workshop, we'll conclude our overview of Linux by working through the last TryHackMe room covering Linux.

TryHackMe

What's Next?

In the next Linux OS operations workshop, we'll keep learning about Linux commands through the PicoCTF platform.

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!