Linux Shells

How to interact with a shell

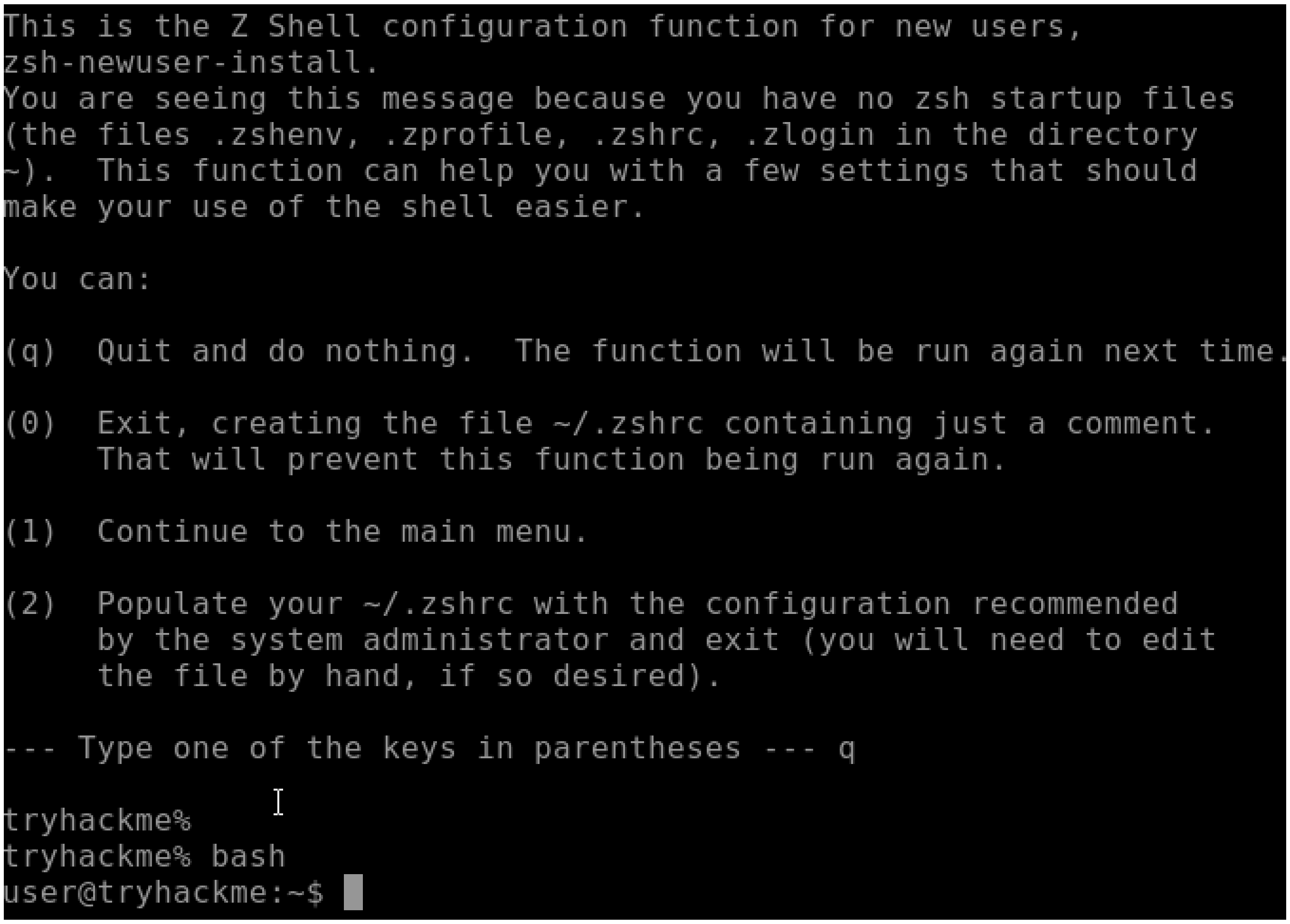
PWD - print working directory is important for linux shells #
CD- change directory
ls - list directory contents
cat - reads files
grep - searches word in file

Types of Linux Shells

To see what shell you are using you can use the command:
echo $SHELL

The file `/etc/shells` contains all the installed shells on a Linux
and can cat this to see a list of all available shells

to switch to a different shell you can just type the name of it in the command line



```
This is the Z Shell configuration function for new users,
zsh-newuser-install.
You are seeing this message because you have no zsh startup files
(the files .zshenv, .zprofile, .zshrc, .zlogin in the directory
~).  This function can help you with a few settings that should
make your use of the shell easier.

You can:

(q)  Quit and do nothing.  The function will be run again next time.

(0)  Exit, creating the file ~/.zshrc containing just a comment.
     That will prevent this function being run again.

(1)  Continue to the main menu.

(2)  Populate your ~/.zshrc with the configuration recommended
     by the system administrator and exit (you will need to edit
     the file by hand, if so desired).

--- Type one of the keys in parentheses --- q

tryhackme%
tryhackme% bash
user@tryhackme:~$
```

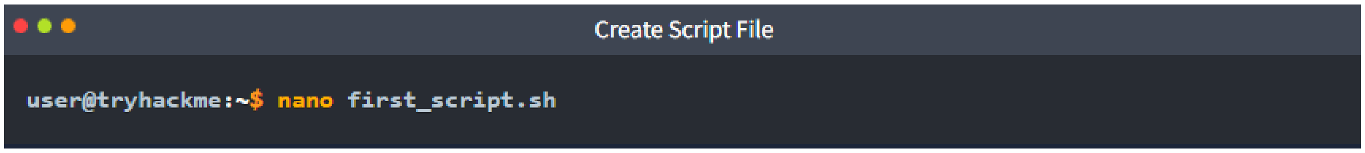
to permanently change your default shell, you can use the command: `chsh -s /usr/bin/zsh`

history - command displays all your previous commands

Feature	Bash	Fish	Zsh
Full Name	The full form of Bash is Bourne Again Shell.	The full form of Fish is Friendly Interactive Shell.	The full form of Zsh is Z Shell.
Scripting	It offers widely compatible scripting with extensive documentation available.	It has limited scripting features as compared to the other two shells.	It offers an excellent level of scripting, combining the traditional capabilities of Bash shell with some extra features.
Tab completion	It has a basic tab completion feature.	It offers advanced tab completion by giving suggestions based on your previous commands.	Its tab completion capability can be extended heavily by using plugins.
Customization	Basic level of customization.	It offers some good customization through interactive tools.	Advanced customization through oh-my-zsh framework.
User friendliness	It is less user-friendly, but being a traditional and widely used shell, its users are quite familiar and comfortable with it.	It is the most user-friendly shell.	It can be highly user-friendly with proper customization.
Syntax highlighting	The syntax highlighting feature is not available in this shell.	The syntax highlighting is built-in to this shell.	The syntax highlighting can be used with this shell by introducing some plugins.

Shell Scripting and Components

to make a script file we can create a file using any text editor (nano) and have to add the file extension **.sh**
e.g:



```
user@tryhackme:~$ nano first_script.sh
```

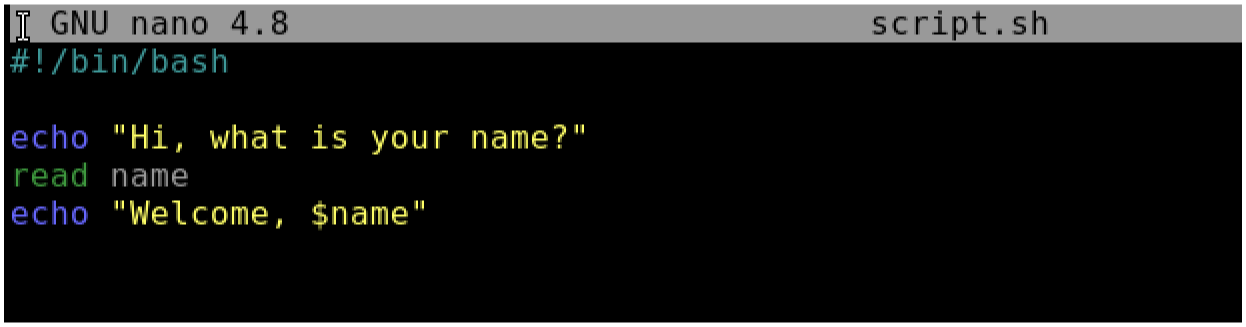
Every script should start from shebang:
#!/bin/bash

Variables:

Variables - store value inside

echo - within scripts echo displays the text followed to the user on the screen

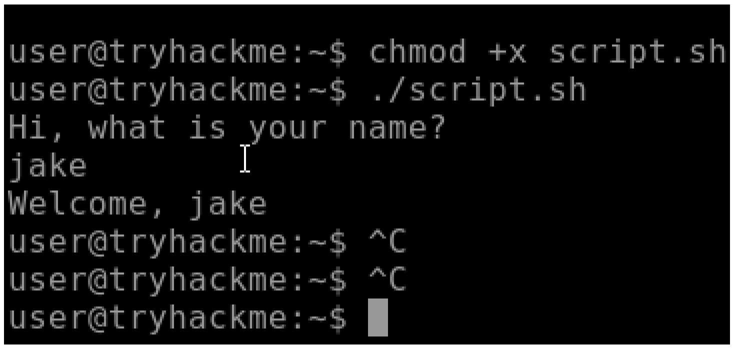
read - takes user input



```
GNU nano 4.8                                       script.sh
#!/bin/bash

echo "Hi, what is your name?"
read name
echo "Welcome, $name"
```

chmod +x [script name].sh - allows the script to become executable

./[scriptname].sh - executes the script



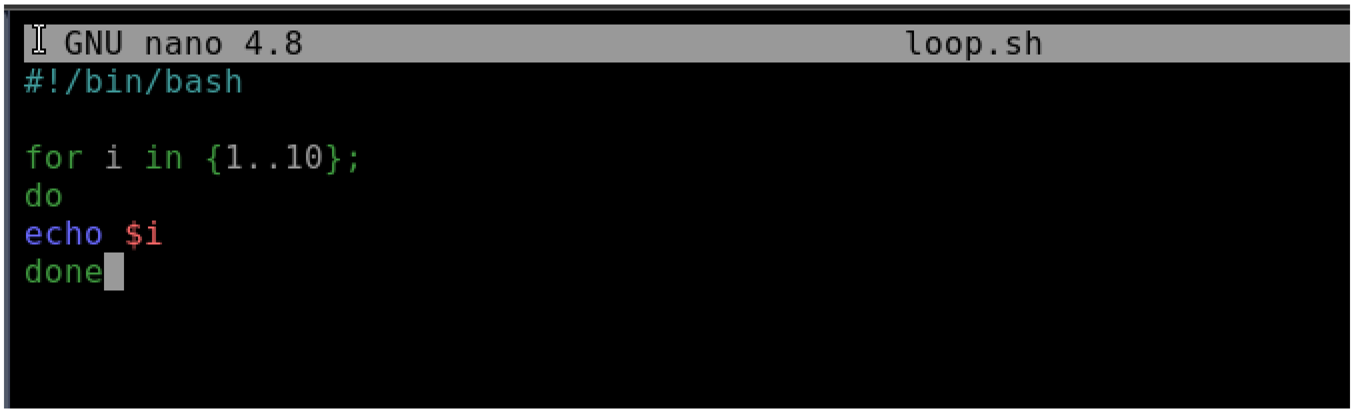
```
user@tryhackme:~$ chmod +x script.sh
user@tryhackme:~$ ./script.sh
Hi, what is your name?
jake
Welcome, jake
user@tryhackme:~$ ^C
user@tryhackme:~$ ^C
user@tryhackme:~$ 
```

Loops

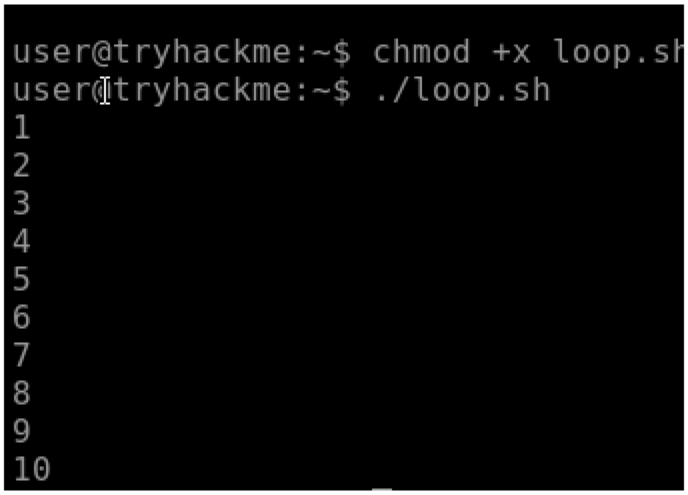
is something that is repeating

i - variable that will iterate from 1 to 10
do - indicates start of loop code
done- indicates the end
the for loop takes each number in the brackets and assigns to variable i each iteration

echo $i displays variable each iteration.



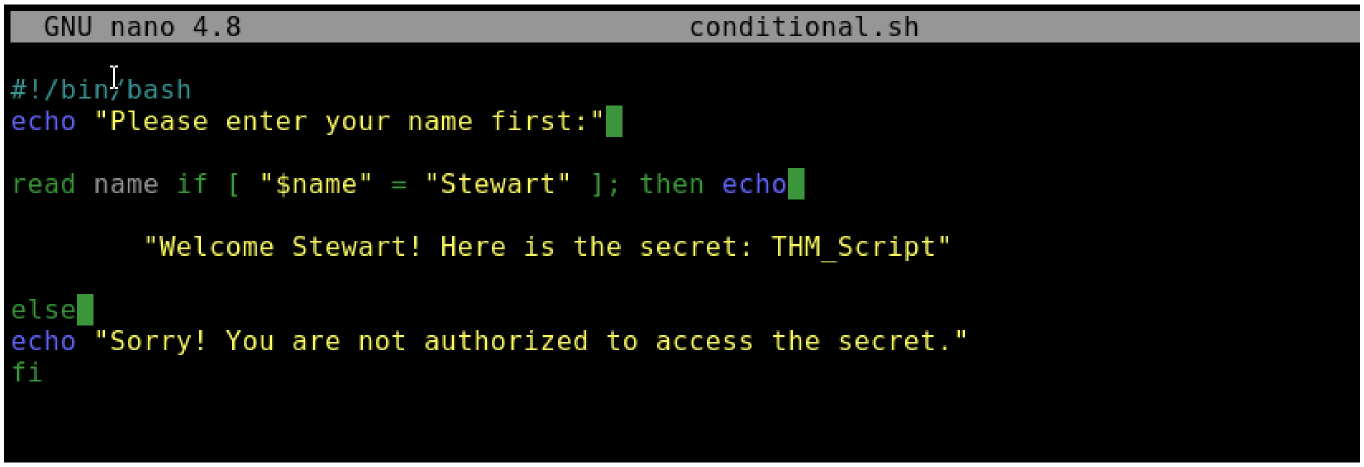
```
GNU nano 4.8                                loop.sh
#!/bin/bash

for i in {1..10};
do
echo $i
done
```



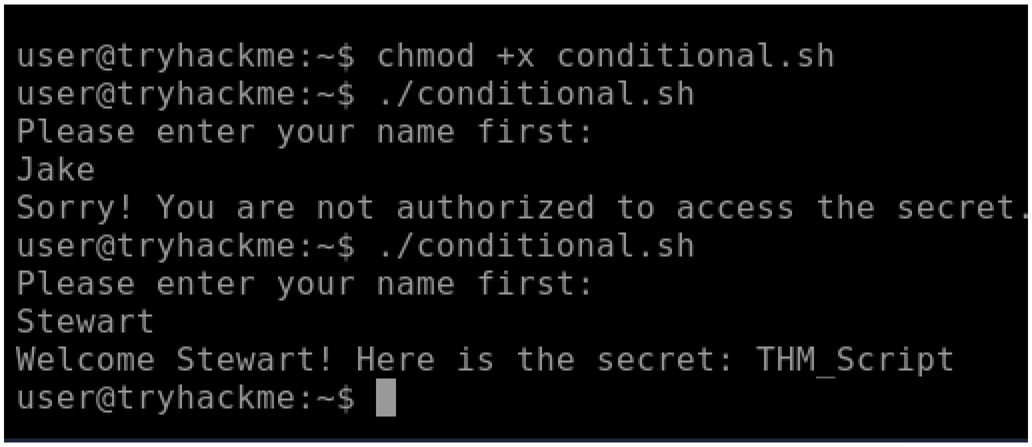
```
user@tryhackme:~$ chmod +x loop.sh
user@tryhackme:~$ ./loop.sh
1
2
3
4
5
6
7
8
9
10
```

Conditional statements

execute a specific code only when a condition is satisfied. otherwise, you can execute another code



```
 GNU nano 4.8                              conditional.sh

#!/bin/bash
echo "Please enter your name first:"

read name if [ "$name" = "Stewart" ]; then echo

        "Welcome Stewart! Here is the secret: THM_Script"

else
echo "Sorry! You are not authorized to access the secret."
fi
```

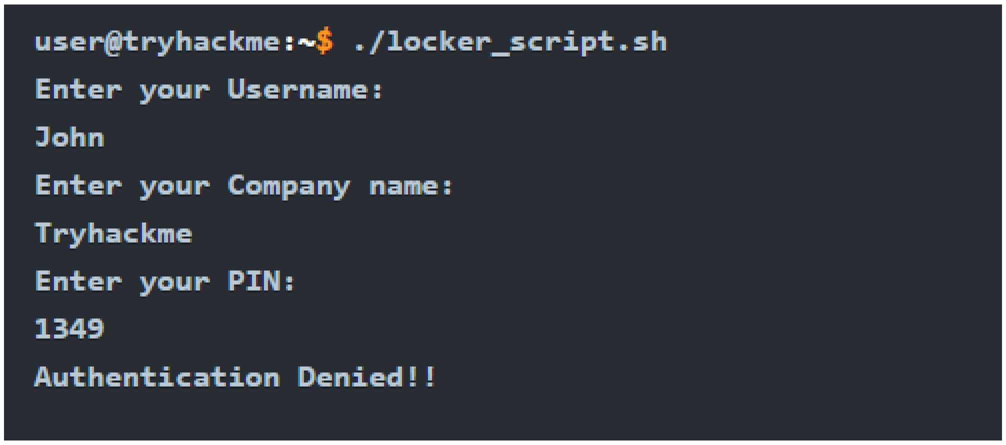

```
user@tryhackme:~$ chmod +x conditional.sh
user@tryhackme:~$ ./conditional.sh
Please enter your name first:
Jake
Sorry! You are not authorized to access the secret.
user@tryhackme:~$ ./conditional.sh
Please enter your name first:
Stewart
Welcome Stewart! Here is the secret: THM_Script
user@tryhackme:~$
```

comments

used for documentation and can be used to tell users what a bit of code does
typically comments start with #

Locker Script

```
#!/bin/bash

# Defining the variables
username=""
companyname=""
pin=""

# Defining the loop
for i in {1..3}; do
# Defining the conditional statements
        if [ "$i" -eq 1 ]; then
                echo "Enter your Username:"
                read username
        elif [ "$i" -eq 2 ]; then
                echo "Enter your Company name:"
                read companyname
        else
                echo "Enter your PIN:"
                read pin
        fi
done

# Checking if the user entered the correct details
if [ "$username" = "John" ] && [ "$companyname" = "Tryhackme" ] && [ "$pin" = "7385" ]; then
        echo "Authentication Successful. You can now access your locker, John."
else
        echo "Authentication Denied!!"
fi
```

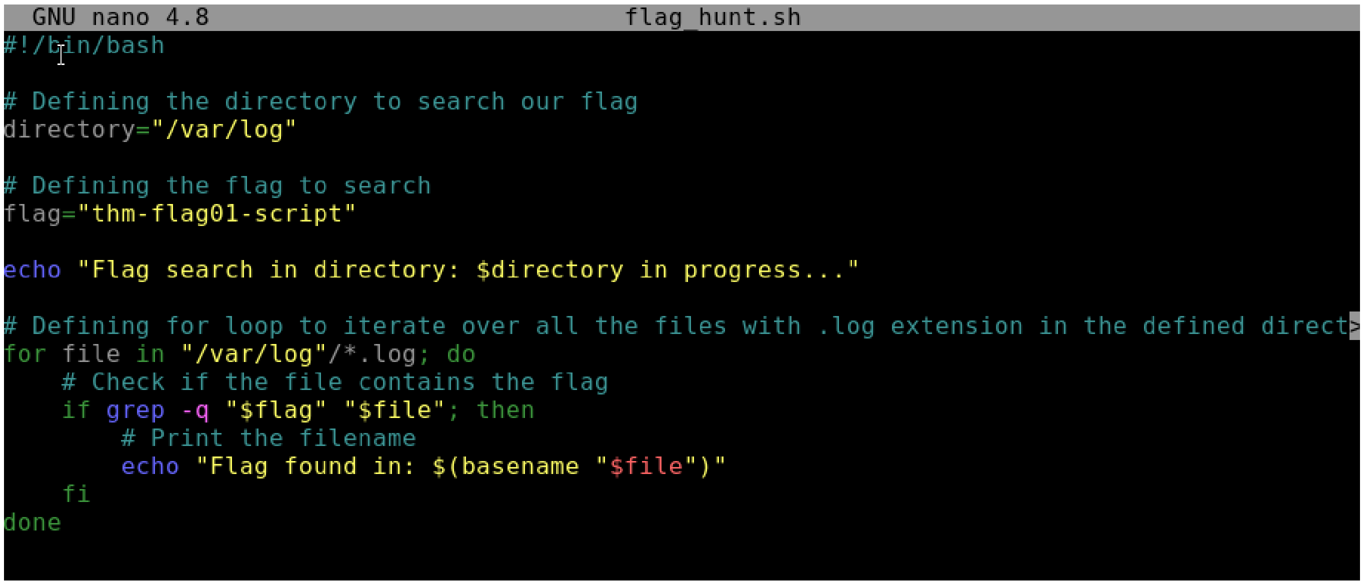


```
user@tryhackme:~$ ./locker_script.sh
Enter your Username:
John
Enter your Company name:
Tryhackme
Enter your PIN:
1349
Authentication Denied!!
```

Practical Exercise

I used command sudo su to switch user to root

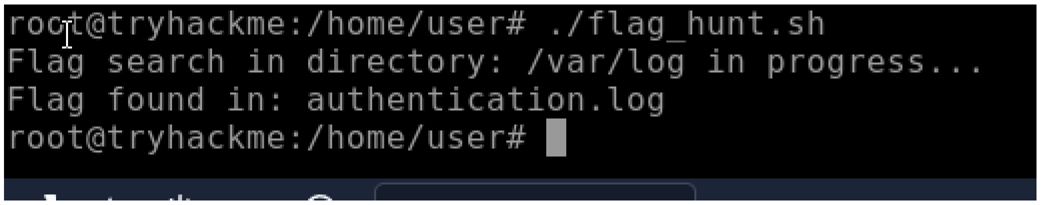
then went into the text editor to edit the existing script



```
  GNU nano 4.8                          flag_hunt.sh
#!/bin/bash

# Defining the directory to search our flag
directory="/var/log"

# Defining the flag to search
flag="thm-flag01-script"

echo "Flag search in directory: $directory in progress..."

# Defining for loop to iterate over all the files with .log extension in the defined direct>
for file in "/var/log"/*.log; do
    # Check if the file contains the flag
    if grep -q "$flag" "$file"; then
        # Print the filename
        echo "Flag found in: $(basename "$file")"
    fi
done
```

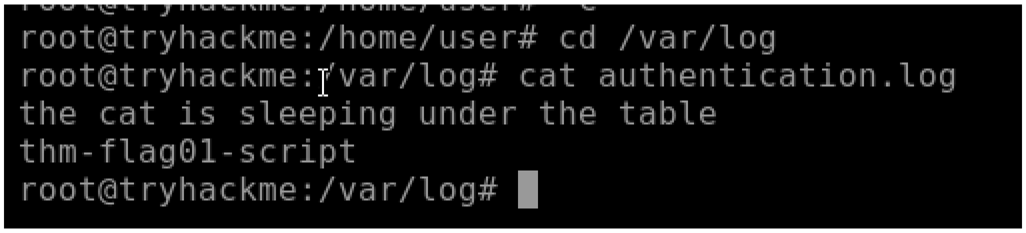
I filled in the directory and the flag and the part in the for loop

I made the script executable and ran the script



```
root@tryhackme:/home/user# ./flag_hunt.sh
Flag search in directory: /var/log in progress...
Flag found in: authentication.log
root@tryhackme:/home/user#
```

I see where the flag is and then navigate to that directory and cat into the file



```
root@tryhackme:/home/user# cd /var/log
root@tryhackme:/var/log# cat authentication.log
the cat is sleeping under the table
thm-flag01-script
root@tryhackme:/var/log# 
```