



# Bash Scripting

## Introduction

# Bash Tutorial on GitHub

<https://github.com/Hacking-Lab/bash-scripting-tutorial.git>

# First Line: hash bang

```
#!/bin/bash
```

Im Unix-Jargon wird das Ausrufezeichen als bang und das Doppelkreuz als hash oder auch sharp bezeichnet.

<https://de.wikipedia.org/wiki/Shebang>

# Comments

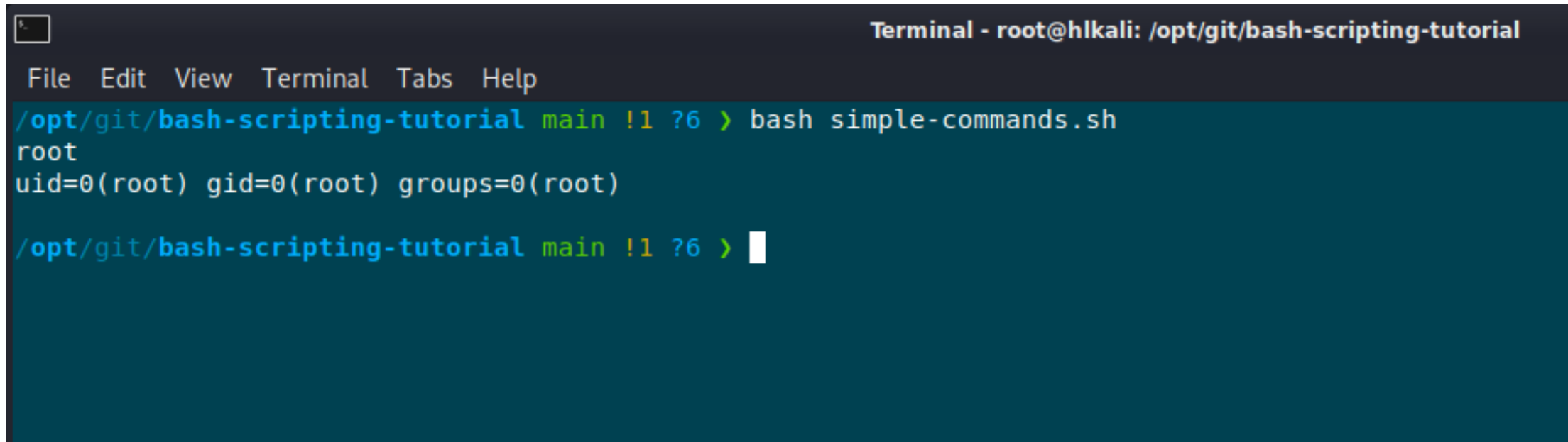
# this is a comment

# Commands

`#!/bin/bash`

`whoami`

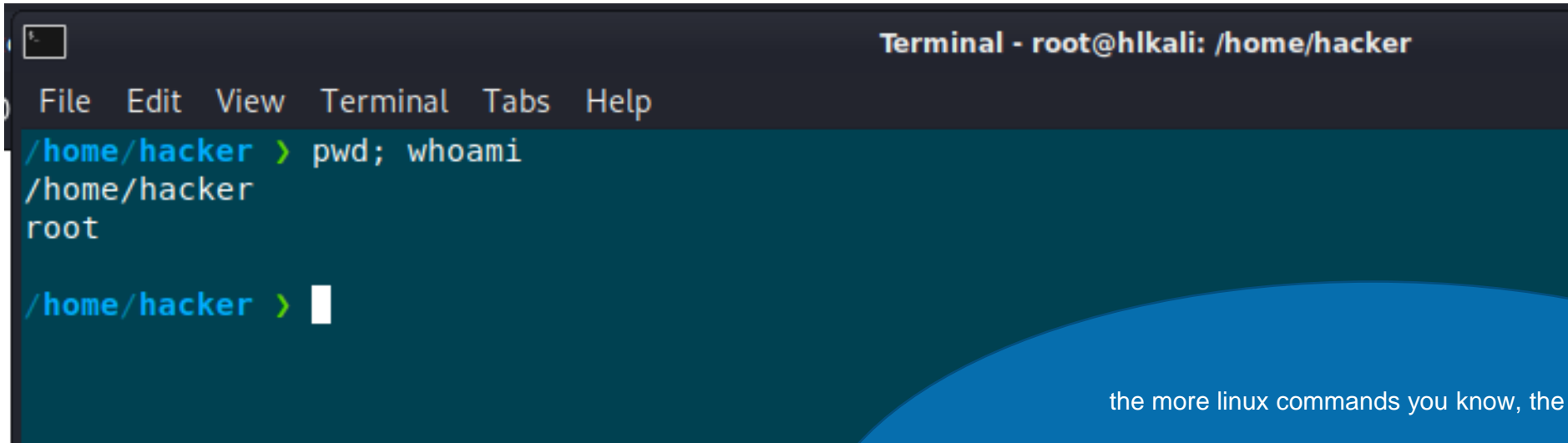
`id`

A terminal window titled "Terminal - root@hikali: /opt/git/bash-scripting-tutorial". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows a prompt `/opt/git/bash-scripting-tutorial main !1 ?6 >` followed by the command `bash simple-commands.sh`. The output of the script is `root` followed by `uid=0(root) gid=0(root) groups=0(root)`. The prompt `/opt/git/bash-scripting-tutorial main !1 ?6 >` is shown again with a cursor at the end.

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?6 > bash simple-commands.sh
root
uid=0(root) gid=0(root) groups=0(root)
/opt/git/bash-scripting-tutorial main !1 ?6 > 
```

# Multiple Commands

```
pwd ; whoami
```



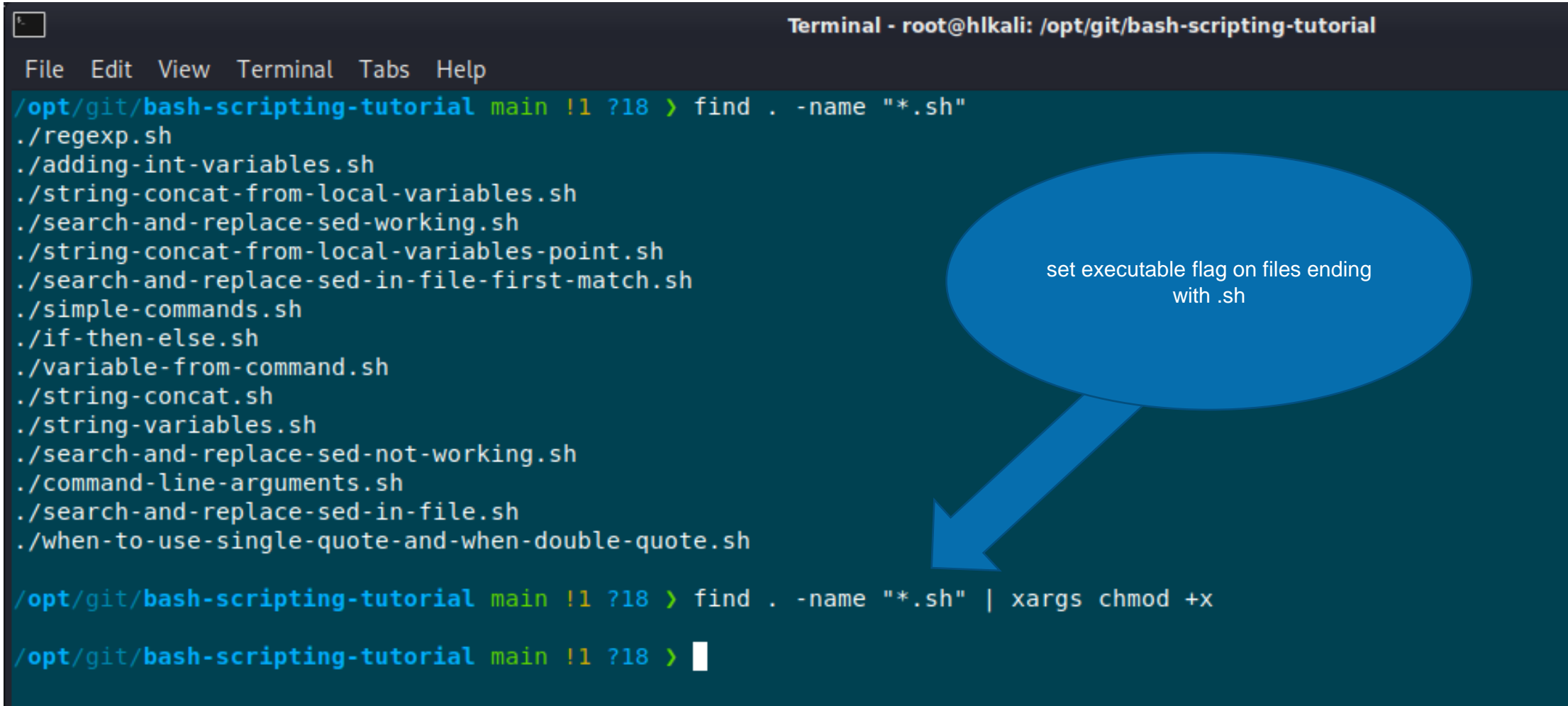
A terminal window titled "Terminal - root@hlkali: /home/hacker" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is `/home/hacker >`. The command `pwd ; whoami` has been entered and executed, resulting in two lines of output: `/home/hacker` and `root`. The prompt is now `/home/hacker >` with a cursor.

the more linux commands you know, the better

awk  
sed  
grep  
wc  
uniq  
sort  
tee  
jq

# Piping Commands – xargs

```
find . -name "*.sh" | xargs chmod +x
```



The image shows a terminal window titled "Terminal - root@hklali: /opt/git/bash-scripting-tutorial". The terminal has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The prompt is `/opt/git/bash-scripting-tutorial main !1 ?18 >`. The first command entered is `find . -name "*.sh"`, which lists 17 shell script files in the current directory. The second command is `find . -name "*.sh" | xargs chmod +x`. A blue oval callout with an arrow points to the `xargs` command, containing the text "set executable flag on files ending with .sh".

```
Terminal - root@hklali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?18 > find . -name "*.sh"
./regexp.sh
./adding-int-variables.sh
./string-concat-from-local-variables.sh
./search-and-replace-sed-working.sh
./string-concat-from-local-variables-point.sh
./search-and-replace-sed-in-file-first-match.sh
./simple-commands.sh
./if-then-else.sh
./variable-from-command.sh
./string-concat.sh
./string-variables.sh
./search-and-replace-sed-not-working.sh
./command-line-arguments.sh
./search-and-replace-sed-in-file.sh
./when-to-use-single-quote-and-when-double-quote.sh

/opt/git/bash-scripting-tutorial main !1 ?18 > find . -name "*.sh" | xargs chmod +x

/opt/git/bash-scripting-tutorial main !1 ?18 > 
```

set executable flag on files ending with .sh

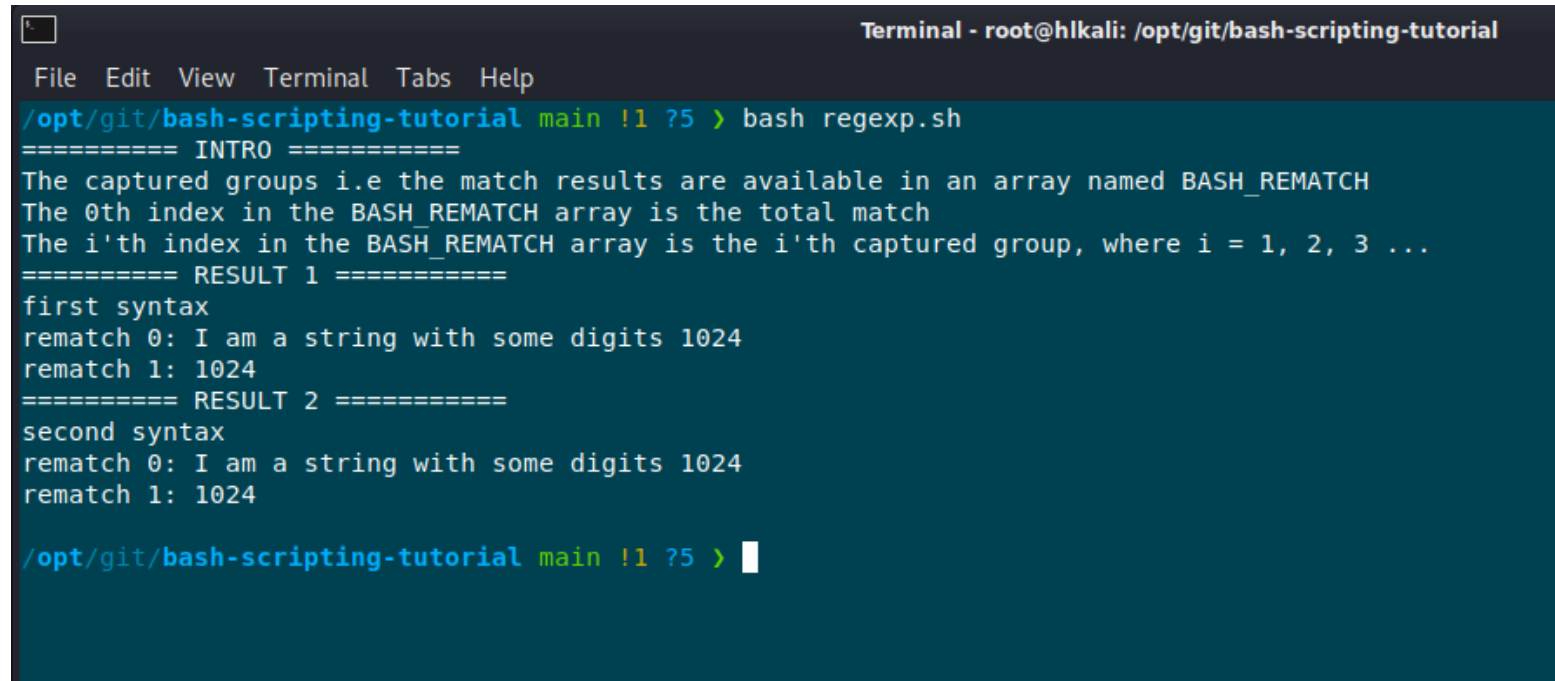
# Regexp

```
#!/bin/bash
pat='^[0-9]+([0-9]+) '
s='I am a string with some digits 1024'
```

```
echo "===== INTRO ====="
echo "The captured groups i.e the match results are available in an array named BASH_REMATCH
The 0th index in the BASH_REMATCH array is the total match
The i'th index in the BASH_REMATCH array is the i'th captured group, where i = 1, 2, 3 ..."
```

```
echo "===== RESULT 1 ====="
echo "first syntax"
[[ $s =~ $pat ]] # $pat must be unquoted
echo "rematch 0: ${BASH_REMATCH[0]}"
echo "rematch 1: ${BASH_REMATCH[1]}"

echo "===== RESULT 2 ====="
echo "second syntax"
[[ $s =~ [^0-9]+([0-9]+) ]]
echo "rematch 0: ${BASH_REMATCH[0]}"
echo "rematch 1: ${BASH_REMATCH[1]}"
```

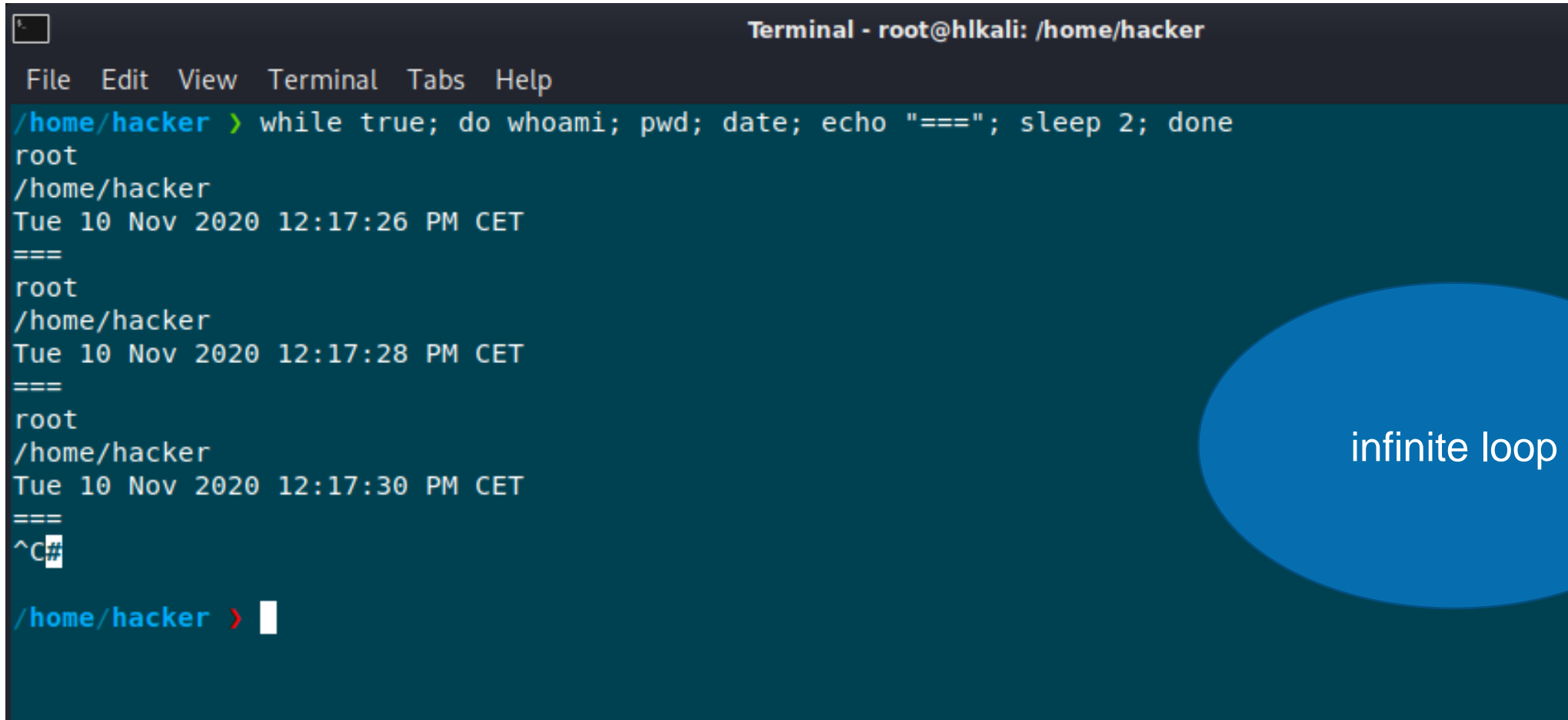
A terminal window titled "Terminal - root@hklali: /opt/git/bash-scripting-tutorial" showing the execution of a script named "regexp.sh". The terminal output matches the code shown in the previous blocks, displaying introductory text and two test results. The first result, labeled "RESULT 1", shows the first syntax working with the unquoted pattern. The second result, labeled "RESULT 2", shows the second syntax working with the quoted pattern. Both tests show the full match at index 0 and the first captured group at index 1.

```
Terminal - root@hklali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?5 > bash regexp.sh
===== INTRO =====
The captured groups i.e the match results are available in an array named BASH_REMATCH
The 0th index in the BASH_REMATCH array is the total match
The i'th index in the BASH_REMATCH array is the i'th captured group, where i = 1, 2, 3 ...
===== RESULT 1 =====
first syntax
rematch 0: I am a string with some digits 1024
rematch 1: 1024
===== RESULT 2 =====
second syntax
rematch 0: I am a string with some digits 1024
rematch 1: 1024
/opt/git/bash-scripting-tutorial main !1 ?5 > 
```



# One Liner

```
while true; do whoami; pwd; date; echo "==="; sleep 2; done
```



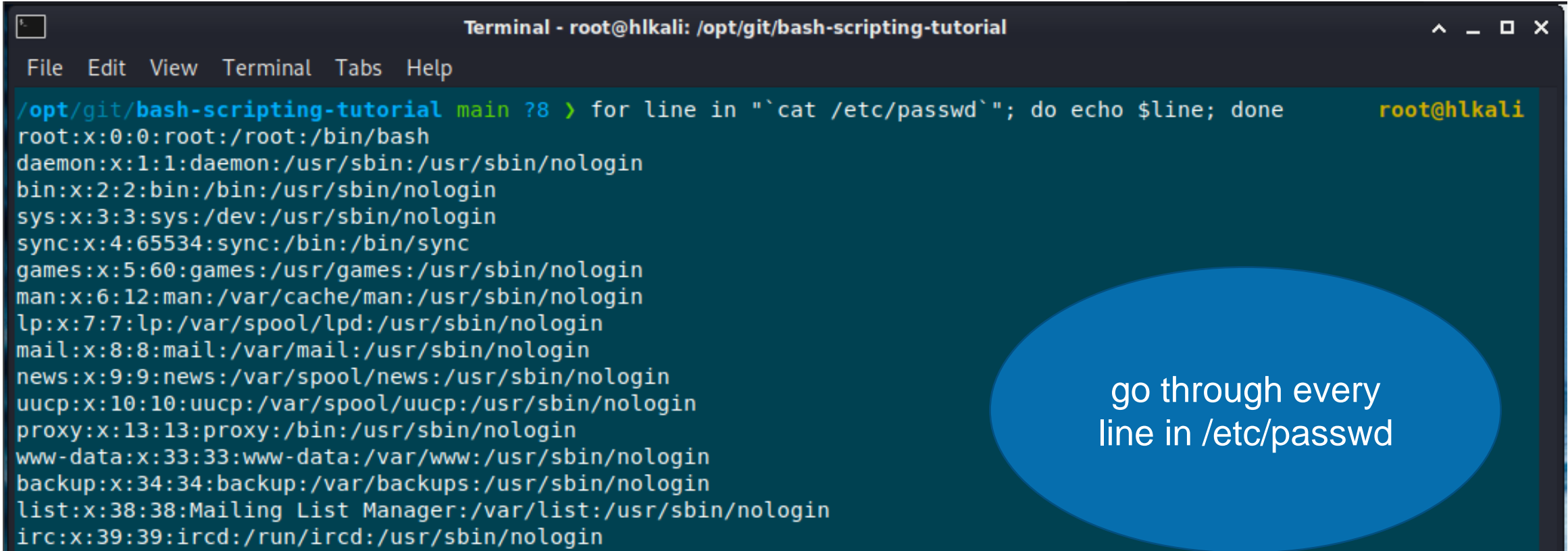
A terminal window titled "Terminal - root@h1kali: /home/hacker" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is `/home/hacker >`. The command `while true; do whoami; pwd; date; echo "==="; sleep 2; done` is entered. The output shows the command executing repeatedly, displaying `root`, `/home/hacker`, `Tue 10 Nov 2020 12:17:26 PM CET`, and `===` on each iteration. The loop is interrupted by a `^C` signal, resulting in a `##` prompt. The prompt returns to `/home/hacker >`.

```
Terminal - root@h1kali: /home/hacker
File Edit View Terminal Tabs Help
/home/hacker > while true; do whoami; pwd; date; echo "==="; sleep 2; done
root
/home/hacker
Tue 10 Nov 2020 12:17:26 PM CET
===
root
/home/hacker
Tue 10 Nov 2020 12:17:28 PM CET
===
root
/home/hacker
Tue 10 Nov 2020 12:17:30 PM CET
===
^C##
/home/hacker >
```

infinite loop

# One Liner

```
for line in "`cat /etc/passwd`"; do echo $line; done
```



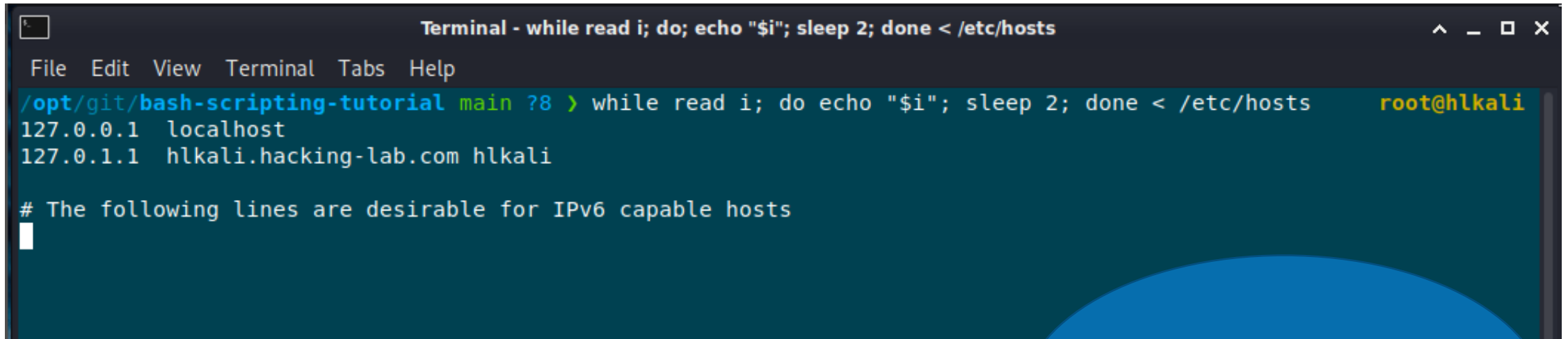
A terminal window titled "Terminal - root@hlkali: /opt/git/bash-scripting-tutorial" displays the execution of the command `for line in "`cat /etc/passwd`"; do echo $line; done`. The output lists system users and their details, such as `root:x:0:0:root:/root:/bin/bash` and `daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin`. A blue oval on the right side of the terminal contains the text "go through every line in /etc/passwd".

```
Terminal - root@hlkali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help

/opt/git/bash-scripting-tutorial main ?8 > for line in "`cat /etc/passwd`"; do echo $line; done root@hlkali
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
```

# One Liner

```
while read i; do echo "$i"; sleep 2; done < /etc/hosts
```



A terminal window titled "Terminal - while read i; do; echo \"\$i\"; sleep 2; done < /etc/hosts". The window shows the command being executed and its output. The output lists the first two lines of /etc/hosts: "127.0.0.1 localhost" and "127.0.1.1 hlkali.hacking-lab.com hlkali". Below this, a comment line is visible: "# The following lines are desirable for IPv6 capable hosts". The prompt shows the user is root@hlkali.

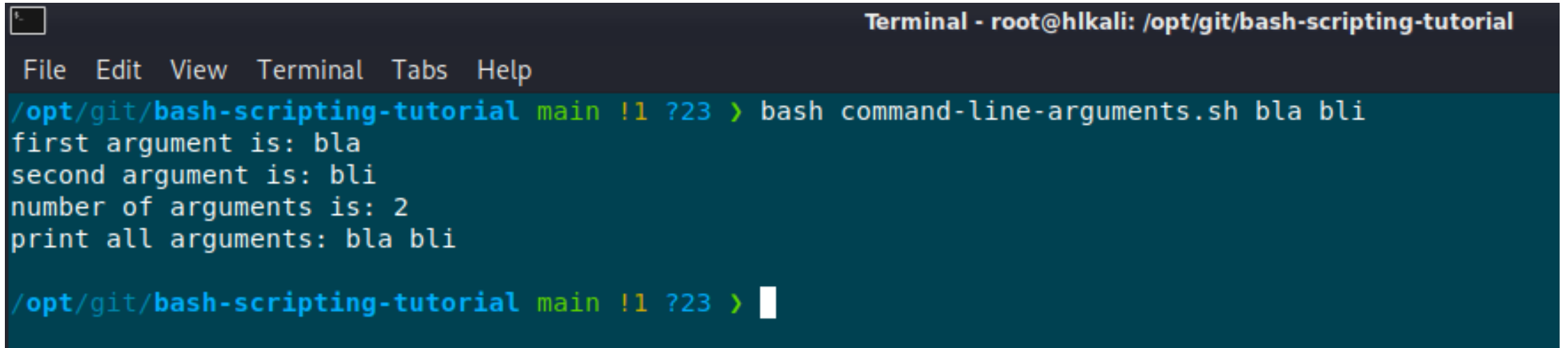
```
Terminal - while read i; do; echo "$i"; sleep 2; done < /etc/hosts
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main ?8 > while read i; do echo "$i"; sleep 2; done < /etc/hosts root@hlkali
127.0.0.1 localhost
127.0.1.1 hlkali.hacking-lab.com hlkali
# The following lines are desirable for IPv6 capable hosts
█
```

go through every  
line in /etc/hosts

# Command Line Arguments

```
#!/bin/bash
if [ "$#" -ne 2 ]; then
    echo "You must enter exactly 2 command line arguments"
fi

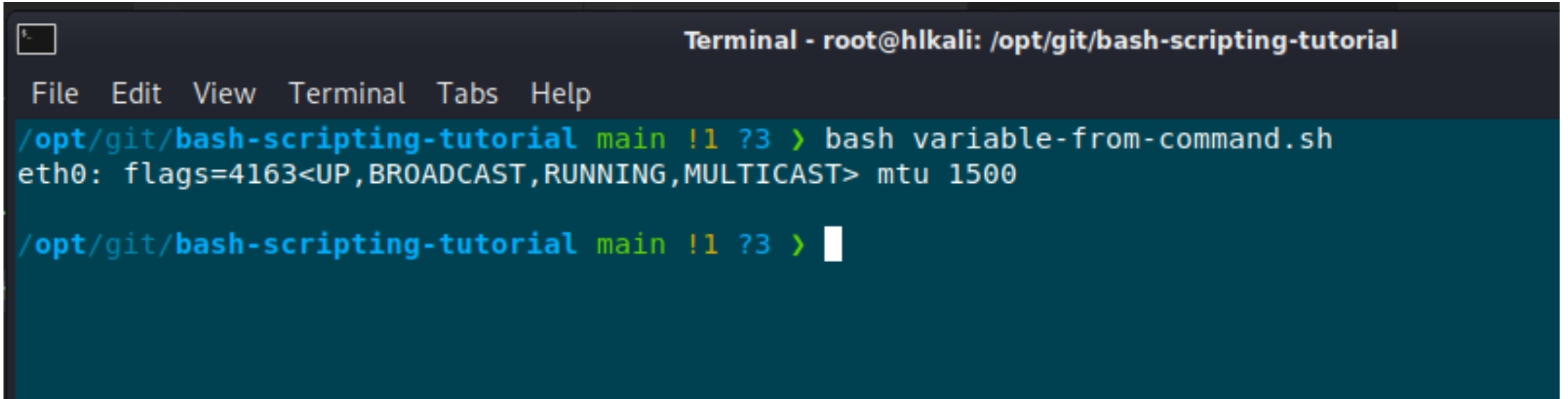
echo "first argument is: $1"
echo "second argument is: $2"
echo "number of arguments is: $#\"
echo "print all arguments: $*\"
```

A terminal window titled "Terminal - root@hikali: /opt/git/bash-scripting-tutorial". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the command `bash command-line-arguments.sh bla bli` being executed. The output is:  
first argument is: bla  
second argument is: bli  
number of arguments is: 2  
print all arguments: bla bli  
The prompt `/opt/git/bash-scripting-tutorial main !1 ?23 >` is visible at the bottom.

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?23 > bash command-line-arguments.sh bla bli
first argument is: bla
second argument is: bli
number of arguments is: 2
print all arguments: bla bli
/opt/git/bash-scripting-tutorial main !1 ?23 > █
```

# Output from Command into Variable

```
#!/bin/bash
gugus=`ifconfig -a | grep eth0`
echo $gugus
```

A terminal window titled "Terminal - root@hikali: /opt/git/bash-scripting-tutorial". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows a prompt "/opt/git/bash-scripting-tutorial main !1 ?3 >" followed by the command "bash variable-from-command.sh". The output is "eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500". The prompt is repeated on the next line with a cursor.

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?3 > bash variable-from-command.sh
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
/opt/git/bash-scripting-tutorial main !1 ?3 > █
```

# Variables – default values

```
#!/bin/bash
```

```
var=${USER:=value}
```

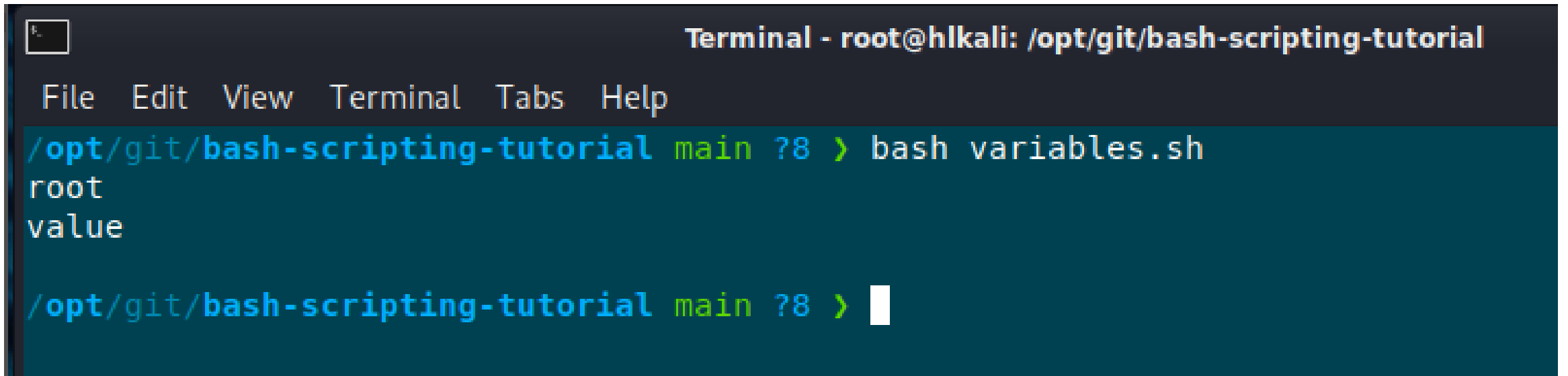
```
echo $USER
```

```
var=${GUGUS:=value}
```

```
echo $GUGUS
```

\$USER exists = root

\$GUGUS not available



The screenshot shows a terminal window titled "Terminal - root@h1kali: /opt/git/bash-scripting-tutorial". The terminal has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The prompt is `/opt/git/bash-scripting-tutorial main ?8 >`. The user enters `bash variables.sh`. The script outputs `root` and `value` on two separate lines. The prompt returns to `/opt/git/bash-scripting-tutorial main ?8 >` with a white cursor.

```
Terminal - root@h1kali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main ?8 > bash variables.sh
root
value
/opt/git/bash-scripting-tutorial main ?8 > 
```

# Executing Bash Script

## Approach 1

- `chmod +x myscript.sh`
- `./myscript`

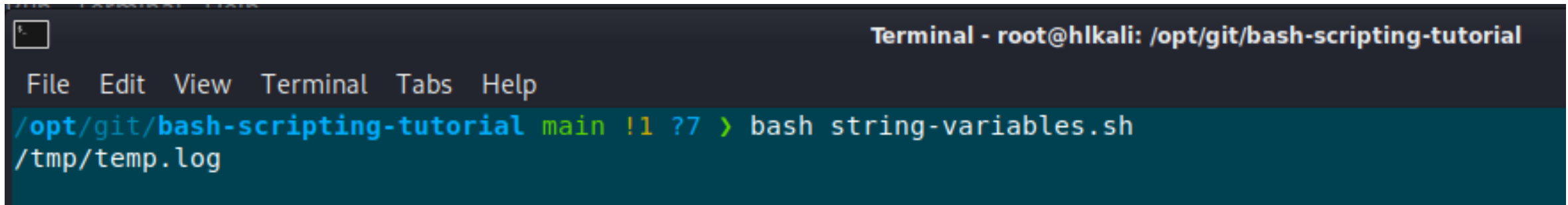
## Approach 2

- `bash myscript.sh`

# string Variables

```
mylog="/tmp/temp.log"
```

```
echo $mylog
```

A terminal window with a dark background. The title bar reads "Terminal - root@hikali: /opt/git/bash-scripting-tutorial". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows a prompt at "/opt/git/bash-scripting-tutorial main !1 ?7 >" followed by the command "bash string-variables.sh". The output of the command is "/tmp/temp.log".

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?7 > bash string-variables.sh  
/tmp/temp.log
```



# string Concat

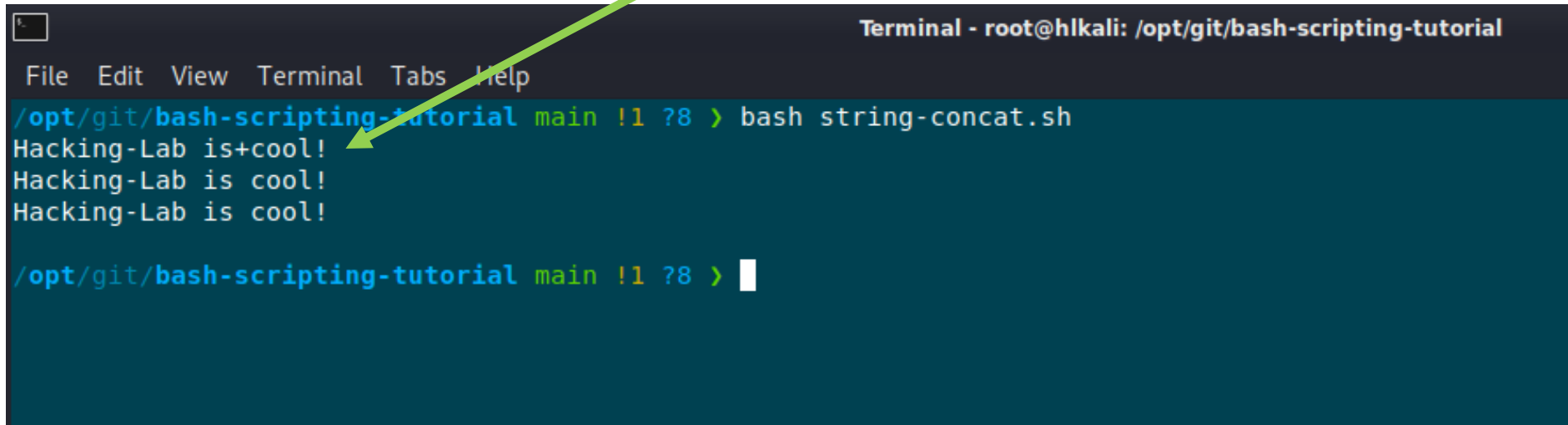
```
#!/bin/bash
firststring="Hacking-Lab is"
secondstring="cool!"
```

```
concat=$firststring+$secondstring
echo $concat
```

```
concat1=$firststring" "$secondstring
echo $concat1
```

```
concat2="$firststring $secondstring"
echo $concat2
```

does not work



```
Terminal - root@hlkali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?8 > bash string-concat.sh
Hacking-Lab is+cool!
Hacking-Lab is cool!
Hacking-Lab is cool!

/opt/git/bash-scripting-tutorial main !1 ?8 > 
```

# Loading Variables from File using the keyword «**source**»

```
#!/bin/bash  
source myvariables
```

```
concat2="$firststring $secondstring"  
echo $concat2
```

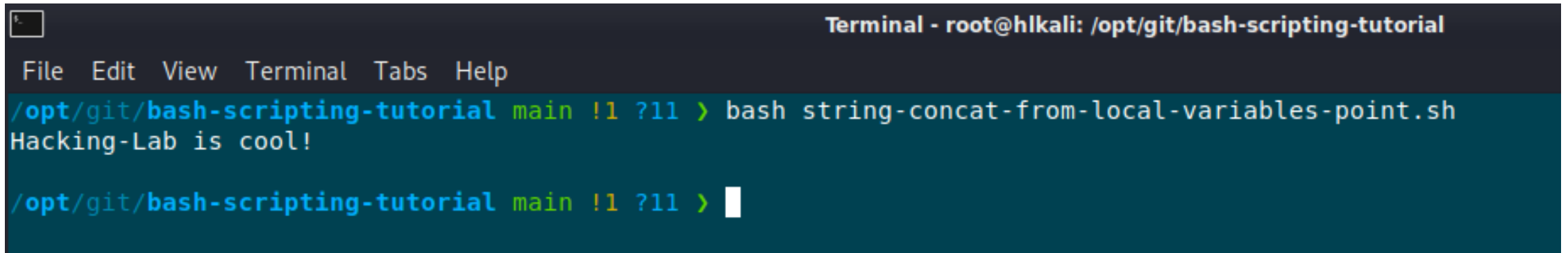
A terminal window titled "Terminal - root@h1kali: /opt/git/bash-scripting-tutorial". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows a prompt "/opt/git/bash-scripting-tutorial main !1 ?10 >" followed by the command "bash string-concat-from-local-variables.sh". The output of the script is "Hacking-Lab is cool!". The prompt is repeated on the next line, followed by a white cursor.

```
Terminal - root@h1kali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?10 > bash string-concat-from-local-variables.sh  
Hacking-Lab is cool!  
/opt/git/bash-scripting-tutorial main !1 ?10 > |
```

# Loading Variables from File using the keyword «.»»

```
#!/bin/bash  
.  
myvariables
```

```
concat2="$firststring $secondstring"  
echo $concat2
```

A terminal window titled "Terminal - root@h1kali: /opt/git/bash-scripting-tutorial". The terminal shows a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The prompt is "/opt/git/bash-scripting-tutorial main !1 ?11 >". The user enters "bash string-concat-from-local-variables-point.sh", and the output is "Hacking-Lab is cool!". The prompt is now "/opt/git/bash-scripting-tutorial main !1 ?11 >".

```
Terminal - root@h1kali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?11 > bash string-concat-from-local-variables-point.sh  
Hacking-Lab is cool!  
/opt/git/bash-scripting-tutorial main !1 ?11 > 
```

# ~~int Variables~~

~~id=5~~

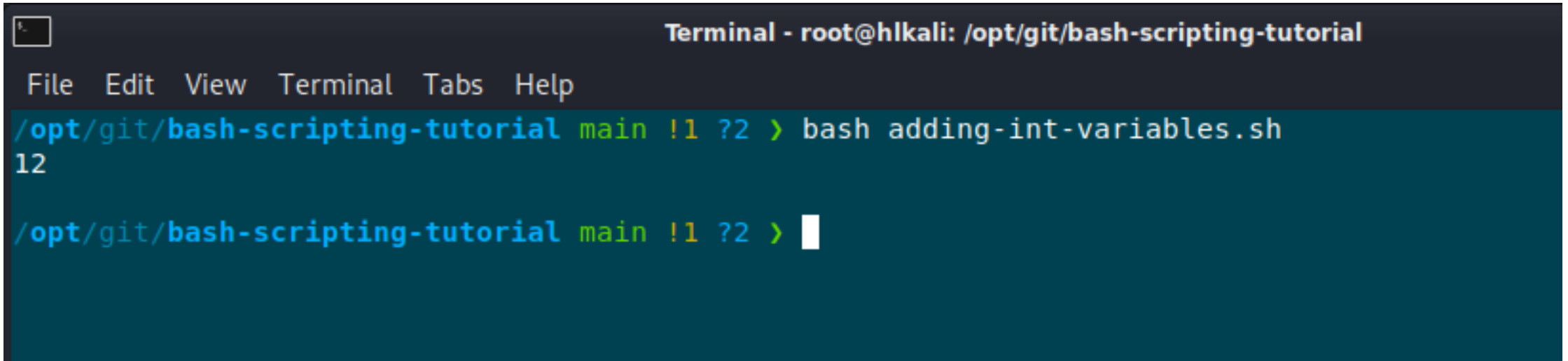
~~echo \$id~~

everything considered as «string»

int variables not available

# int Addition

```
#!/bin/bash
bla=5
bli=7
echo $((bla+bli))
```

A terminal window titled "Terminal - root@hikali: /opt/git/bash-scripting-tutorial". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the command `bash adding-int-variables.sh` being executed, which outputs `12`. The prompt `/opt/git/bash-scripting-tutorial main !1 ?2 >` is visible on the next line.

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?2 > bash adding-int-variables.sh
12
/opt/git/bash-scripting-tutorial main !1 ?2 > 
```

# int Arithmetic

```
#!/bin/bash
```

```
expr 2 + 2
```

```
expr 6 \* 6
```

# (escape \*)

\* must be escaped

```
let a=2+2
```

```
let b=6*6
```

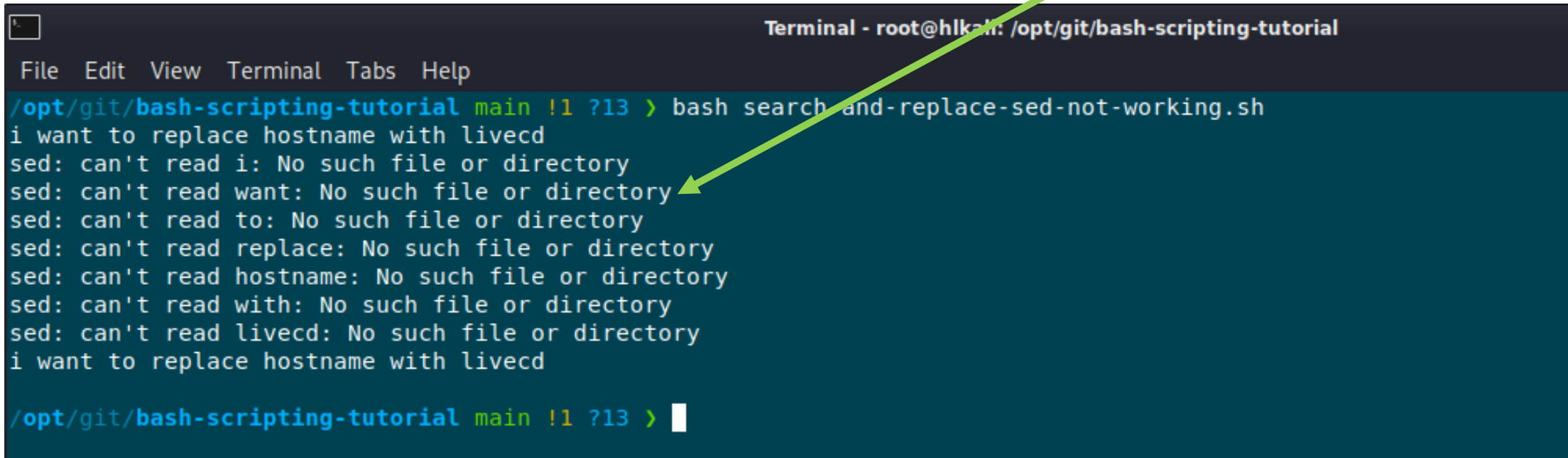
```
Terminal - root@hlkali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?27 > bash int-arithmetic.sh
4
36
```

# Search and Replace in Variable

```
#!/bin/bash
```

```
mystring="i want to replace hostname with livedcd"  
echo $mystring  
sed -e "s/hostname/livedcd/g" $mystring  
echo $mystring
```

does not work



The terminal window shows the execution of a script. The prompt is `/opt/git/bash-scripting-tutorial main !1 ?13 >`. The user runs `bash search-and-replace-sed-not-working.sh`. The script outputs `i want to replace hostname with livedcd`. Then, it runs `sed -e "s/hostname/livedcd/g" $mystring`, which results in seven error messages: `sed: can't read i: No such file or directory`, `sed: can't read want: No such file or directory`, `sed: can't read to: No such file or directory`, `sed: can't read replace: No such file or directory`, `sed: can't read hostname: No such file or directory`, `sed: can't read with: No such file or directory`, and `sed: can't read livedcd: No such file or directory`. Finally, it outputs `i want to replace hostname with livedcd`. The prompt is `/opt/git/bash-scripting-tutorial main !1 ?13 >`.

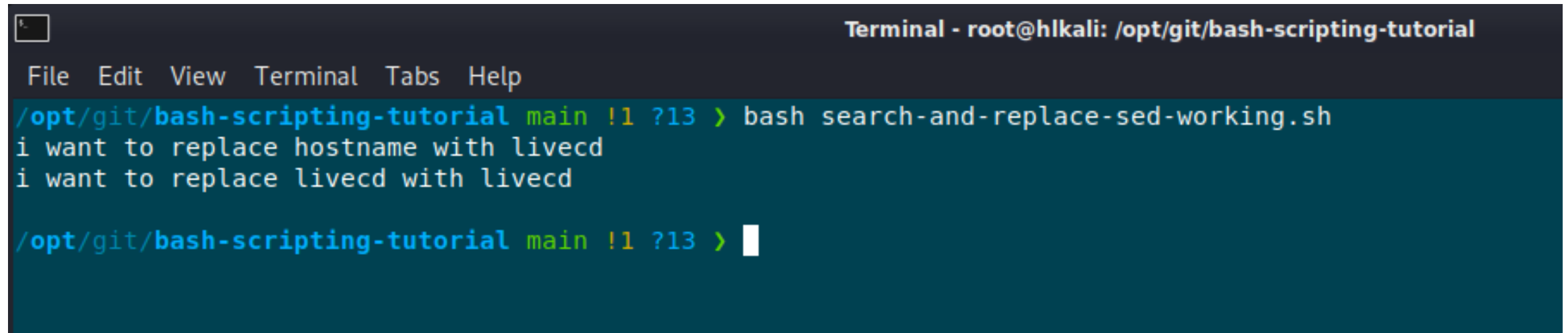
# Search and Replace in Variable

```
#!/bin/bash
```

```
mystring="i want to replace hostname with livedcd"
```

```
echo $mystring
```

```
echo $mystring | sed "s/hostname/livedcd/g"
```



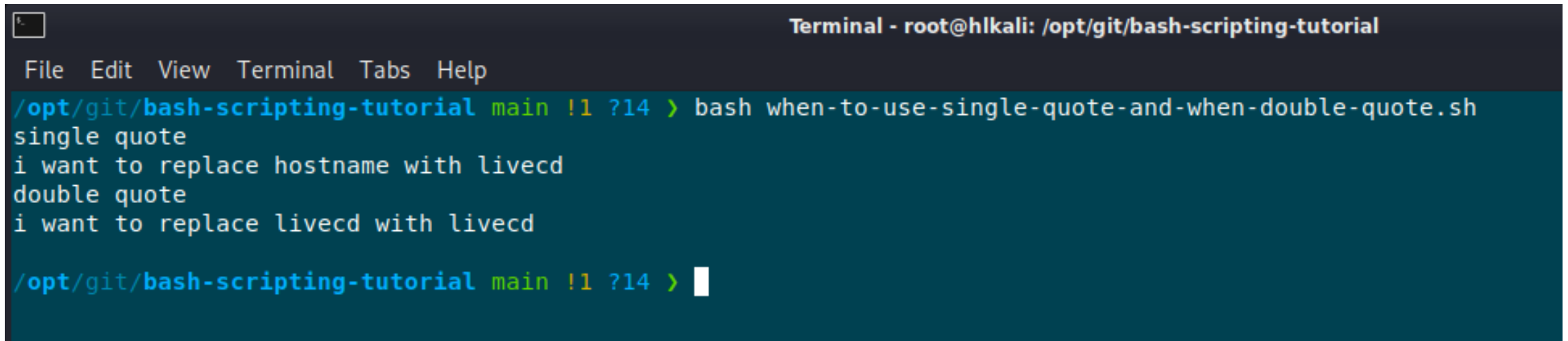
A terminal window titled "Terminal - root@h1kali: /opt/git/bash-scripting-tutorial" showing the execution of a script. The terminal has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The prompt is `/opt/git/bash-scripting-tutorial main !1 ?13 >`. The user runs `bash search-and-replace-sed-working.sh`. The script outputs two lines: `i want to replace hostname with livedcd` and `i want to replace livedcd with livedcd`. The prompt is now `/opt/git/bash-scripting-tutorial main !1 ?13 >` with a cursor.



# when to use ' or use "

```
#!/bin/bash
```

```
mystring="i want to replace hostname with livedcd"  
myreplace="hostname"  
echo "single quote"  
echo $mystring | sed 's/$myreplace/livedcd/g'  
echo "double quote"  
echo $mystring | sed "s/$myreplace/livedcd/g"
```

A terminal window titled "Terminal - root@hikali: /opt/git/bash-scripting-tutorial" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the execution of a script. The prompt is "/opt/git/bash-scripting-tutorial main !1 ?14 >". The script runs "bash when-to-use-single-quote-and-when-double-quote.sh". The output is: "single quote", "i want to replace hostname with livedcd", "double quote", and "i want to replace livedcd with livedcd". The prompt is now "/opt/git/bash-scripting-tutorial main !1 ?14 >".

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?14 > bash when-to-use-single-quote-and-when-double-quote.sh  
single quote  
i want to replace hostname with livedcd  
double quote  
i want to replace livedcd with livedcd  
/opt/git/bash-scripting-tutorial main !1 ?14 > █
```

# Search and Replace in File – sed – global match

```
#!/bin/bash
```

```
echo "=====  
echo "original content of myfile"  
cat myfile  
sed -i -e 's/hostname/livedcd/g' ./myfile
```

```
echo "=====  
echo "replace content of myfile"  
cat myfile
```

/g == all matches

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?17 > bash search-and-replace-sed-in-file.sh  
=====  
original content of myfile  
i want to replace hostname with livedcd  
i want to replace hostname with livedcd i want to replace hostname with livedcd  
  
=====  
replace content of myfile  
i want to replace livedcd with livedcd  
i want to replace livedcd with livedcd i want to replace livedcd with livedcd  
  
/opt/git/bash-scripting-tutorial main !1 ?17 > |
```

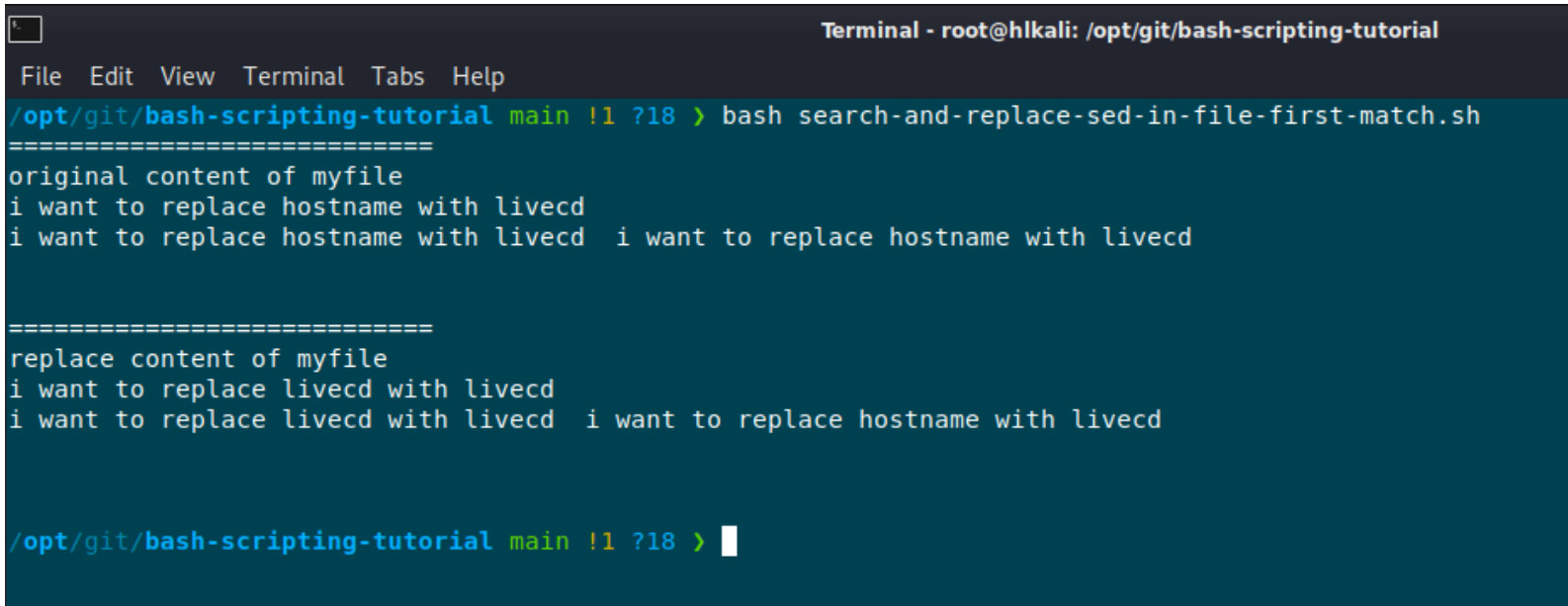
# Search and Replace in File – sed – first match

```
#!/bin/bash

echo "=====
echo "original content of myfile"
cat myfile
sed -i -e 's/hostname/livecd/' ./myfile

echo "=====
echo "replace content of myfile"
cat myfile
```

without /g ==  
first match

A terminal window titled "Terminal - root@hlkali: /opt/git/bash-scripting-tutorial" shows the execution of a script. The script's output is as follows:

```
/opt/git/bash-scripting-tutorial main !1 ?18 > bash search-and-replace-sed-in-file-first-match.sh
=====
original content of myfile
i want to replace hostname with livecd
i want to replace hostname with livecd i want to replace hostname with livecd

=====
replace content of myfile
i want to replace livecd with livecd
i want to replace livecd with livecd i want to replace hostname with livecd

/opt/git/bash-scripting-tutorial main !1 ?18 > |
```

# Search and Replace recursively

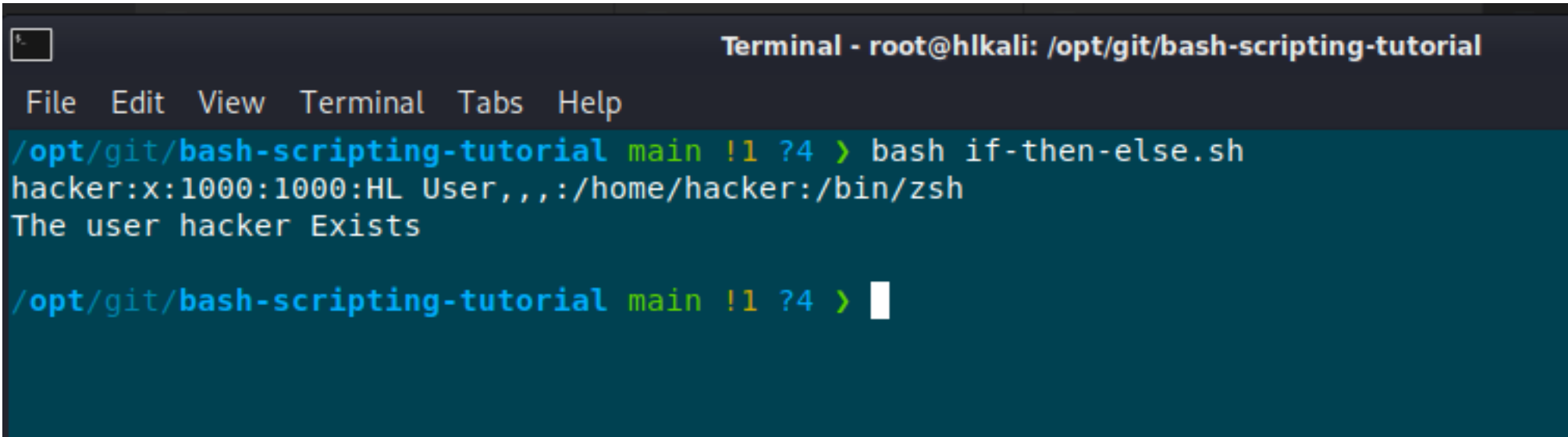
```
find . -type f -name "*.md" -print0 | xargs -0 sed -i 's/foo/bar/g'
```

or

```
grep -r1Z 'foo' . | xargs -0 sed -i.bak 's/foo/bar/g'
```

## if – then - else

```
#!/bin/bash
user="hacker"
if grep $user /etc/passwd
then
    echo "The user $user Exists"
else
    echo "The user $user doesn't exist"
fi
```

A terminal window titled "Terminal - root@h1kali: /opt/git/bash-scripting-tutorial". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows the command `bash if-then-else.sh` being executed. The output is `hacker:x:1000:1000:HL User,,,:/home/hacker:/bin/zsh` followed by `The user hacker Exists`. The prompt `/opt/git/bash-scripting-tutorial main !1 ?4 >` is visible at the bottom with a cursor.

```
Terminal - root@h1kali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?4 > bash if-then-else.sh
hacker:x:1000:1000:HL User,,,:/home/hacker:/bin/zsh
The user hacker Exists
/opt/git/bash-scripting-tutorial main !1 ?4 > |
```

# Comparison

Description	Numeric Comparison	String Comparison
less than	-lt	<
greater than	-gt	>
equal	-eq	=
not equal	-ne	!=
less or equal	-le	N/A
greater or equal	-ge	N/A
Shell comparison example:	[ 100 -eq 50 ]; echo \$?	[ "GNU" = "UNIX" ]; echo \$?

# Command Return Status - \$?

```
#!/bin/bash
```

```
string_a="UNIX"  
string_b="GNU"
```

```
echo "Are $string_a and $string_b strings equal?"  
[ $string_a = $string_b ]  
echo $?
```

```
num_a=100  
num_b=100
```

```
echo "Is $num_a equal to $num_b ?"  
[ $num_a -eq $num_b ]  
echo $?
```

\$?

0 signals true

1 indicates false



```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?23 > bash command-return-status.sh  
Are UNIX and GNU strings equal?  
1  
Is 100 equal to 100 ?  
0  
/opt/git/bash-scripting-tutorial main !1 ?23 > |
```

# if – then – else // comparing int

```
#!/bin/bash  
val1=6
```

```
if [ $val1 -gt 5 ]  
then  
    echo "The test value $val1 is greater than 5"  
else  
    echo "The test value $val1 is not greater than 5"  
fi
```

`n1 -eq n2` Returns true if `n1` equal `n2` .

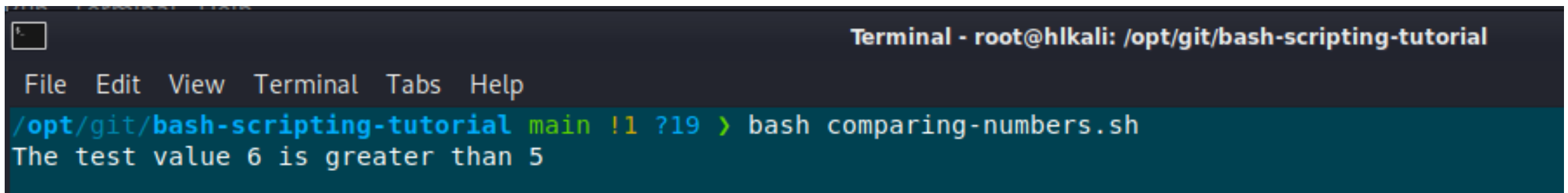
`n1 -ge n2` Returns true if `n1` greater or equal `n2` .

`n1 -gt n2` Returns true if `n1` greater `n2` .

`n1 -le n2` Returns true if `n1` less than or equal `n2` .

`n1 -lt n2` Returns true if `n1` is less `n2` .

`n1 -ne n2` Returns true if `n1` not equal `n2` .

A terminal window titled "Terminal - root@h1kali: /opt/git/bash-scripting-tutorial". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the command `/opt/git/bash-scripting-tutorial main !1 ?19 > bash comparing-numbers.sh` being executed, followed by the output `The test value 6 is greater than 5`.

```
Terminal - root@h1kali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?19 > bash comparing-numbers.sh  
The test value 6 is greater than 5
```



# if – then – else // comparing strings

```
#!/bin/bash
id
user="root"
if [ $user = $USER ]
then
    echo "The user $user is the current logged in user"
fi
```

`str1 = str2` Tests strings for equality, returns true if strings are identical.

`str1 != str2` Returns true if the strings are not identical.

`str1 < str2` Returns true if `str1` less than `str2`.

`str1 > str2` Returns true if `str1` greater than `str2`.

`-n str1` Returns true if the length is `str1` greater than zero.

`-z str1` Returns true if the length `str1` is zero.

Terminal - root@hikali: /opt/git/bash-scripting-tutorial

File Edit View Terminal Tabs Help

/opt/git/bash-scripting-tutorial main !1 ?20 > bash comparing-strings.sh

uid=0(root) gid=0(root) groups=0(root)

The user root is the current logged in user

# File checks

```
#!/bin/bash
mydir="/home/hacker"
if [ -d $mydir ]
then
    echo "The $mydir directory exists"
    cd $mydir
    ls -ll | wc -l
else
    echo "The $mydir directory does not exist"
fi
```

`-d file` Checks if a file exists and is a directory.

`-e file` Checks if the file exists.

`-f file` Checks if a file exists and is a file.

`-r file` Checks if the file exists and is readable.

`-s file` Checks if the file exists and if it is empty.

`-w file` Checks if the file exists and is writable.

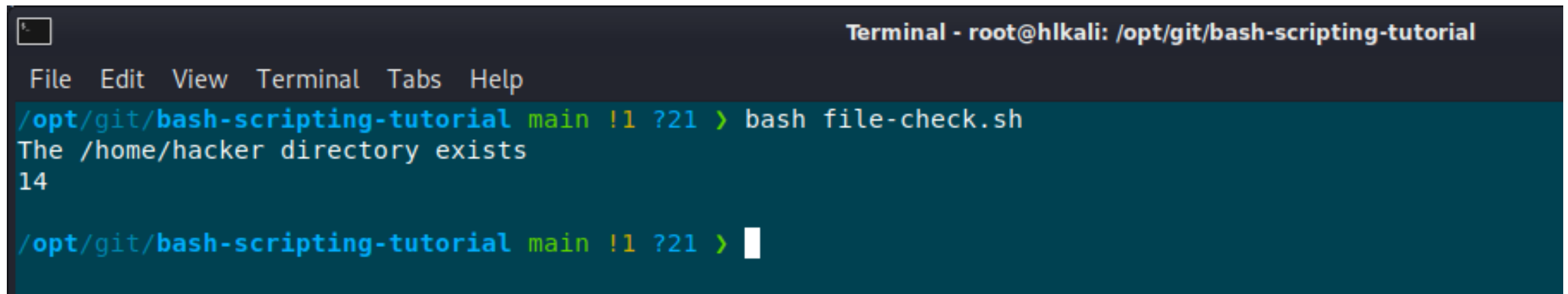
`-x file` Checks if a file exists and is executable.

`file1 -nt file2` Checks if newer `file1` than `file2`.

`file1 -ot file2` Checks if it's older `file1` than `file2`.

`-o file` Checks if the file exists and is owned by the current user.

`-G file` Checks if the file exists and if its group ID matches the group ID of the current user.



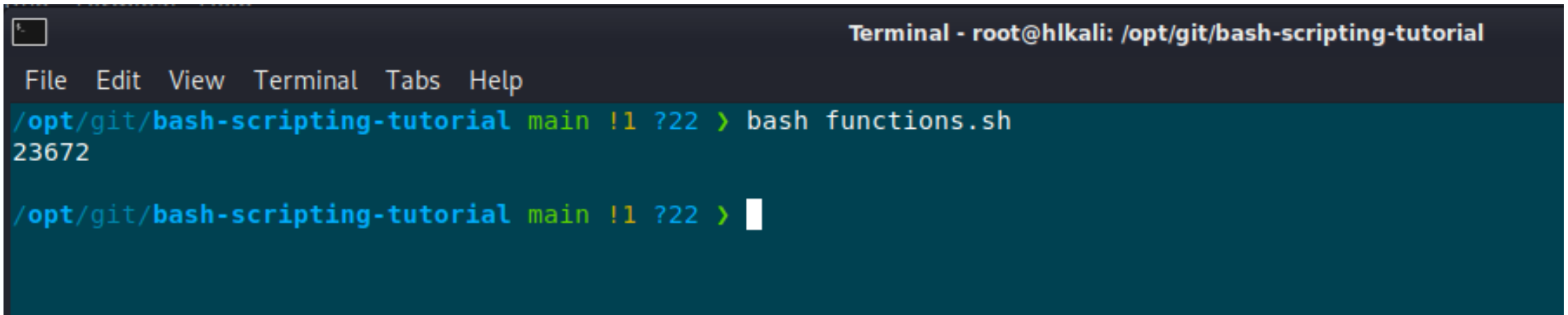
```
Terminal - root@hlkali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?21 > bash file-check.sh
The /home/hacker directory exists
14
/opt/git/bash-scripting-tutorial main !1 ?21 > 
```

# Functions

```
#!/bin/bash
```

```
function total_files {  
    find $1 -type f | wc -l  
}
```

```
mydir="/home/hacker"  
total_files $mydir
```

A terminal window titled "Terminal - root@hlkali: /opt/git/bash-scripting-tutorial". The terminal has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The prompt is "/opt/git/bash-scripting-tutorial main !1 ?22 >". The user enters "bash functions.sh", and the terminal outputs "23672". The prompt is then "/opt/git/bash-scripting-tutorial main !1 ?22 >".

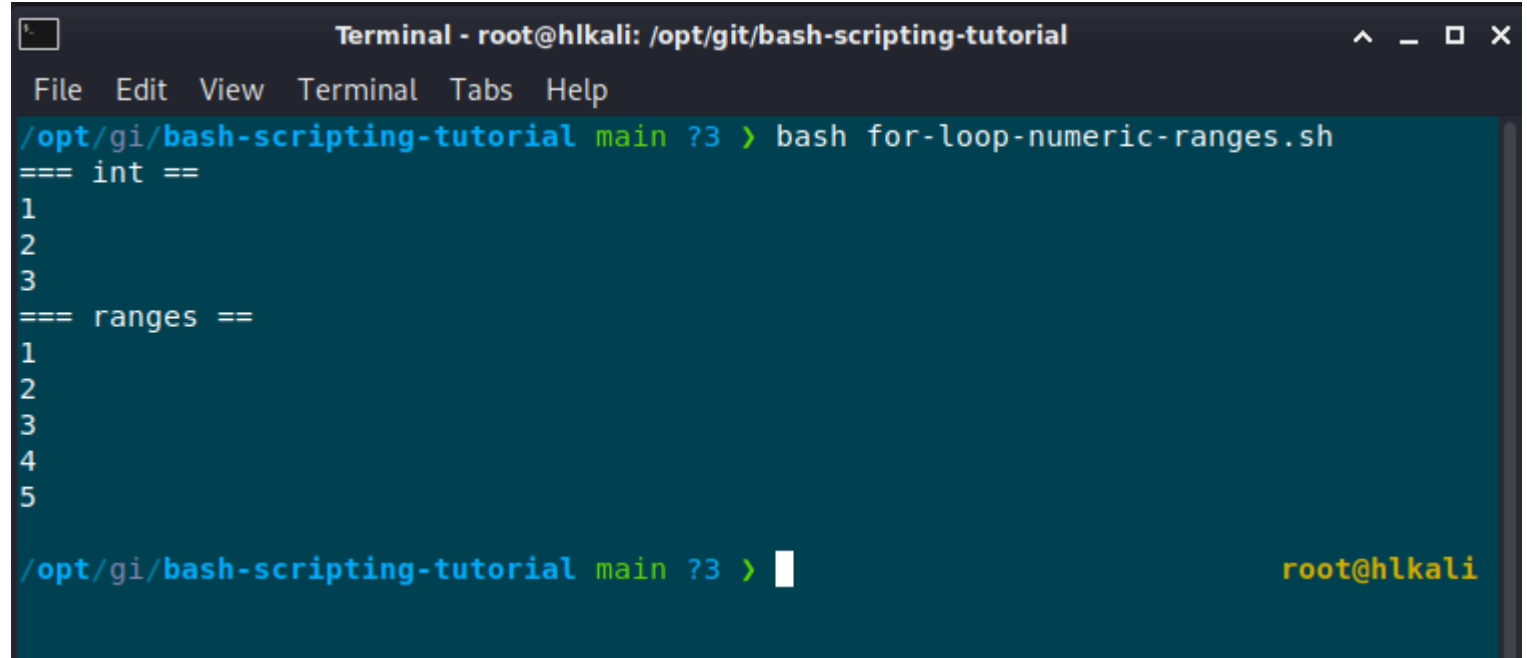
```
Terminal - root@hlkali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?22 > bash functions.sh  
23672  
/opt/git/bash-scripting-tutorial main !1 ?22 > 
```

# for loop – numeric ranges

```
#!/bin/bash
```

```
echo "=== int =="  
for i in 1 2 3; do  
    echo $i  
done
```

```
echo "=== ranges =="  
for i in {1..5}; do  
    echo $i  
done
```

A terminal window titled "Terminal - root@hlkali: /opt/git/bash-scripting-tutorial" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is "/opt/gi/bash-scripting-tutorial main ?3 >". The user has run "bash for-loop-numeric-ranges.sh". The script outputs "=== int ==" followed by the numbers 1, 2, and 3 on separate lines. Then it outputs "=== ranges ==" followed by the numbers 1, 2, 3, 4, and 5 on separate lines. The prompt is now "/opt/gi/bash-scripting-tutorial main ?3 >". The username "root@hlkali" is visible in the bottom right corner.

```
Terminal - root@hlkali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/gi/bash-scripting-tutorial main ?3 > bash for-loop-numeric-ranges.sh  
=== int ==  
1  
2  
3  
=== ranges ==  
1  
2  
3  
4  
5  
/opt/gi/bash-scripting-tutorial main ?3 > root@hlkali
```

<https://www.cyberciti.biz/faq/bash-for-loop/>

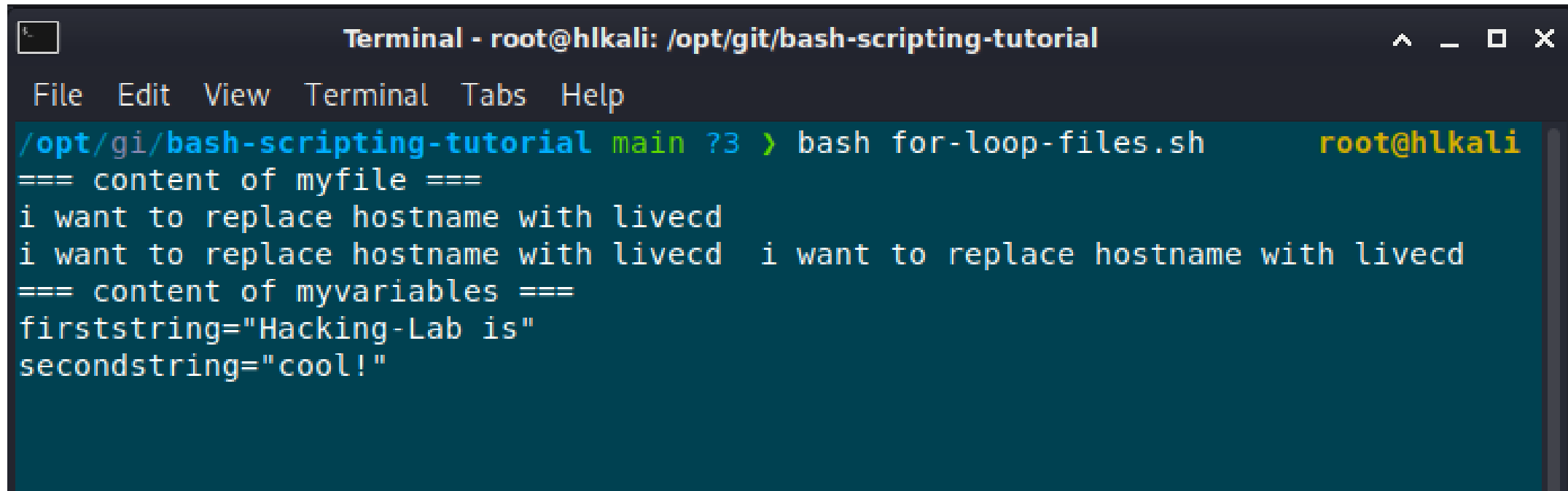
# for loop – files

```
#!/bin/bash
```

```
for i in myfile myvariables; do
    echo "=== content of $i ==="
    cat $i
    echo ""
done
```

myfile is a file in the same folder

myvariables is a file in the same folder

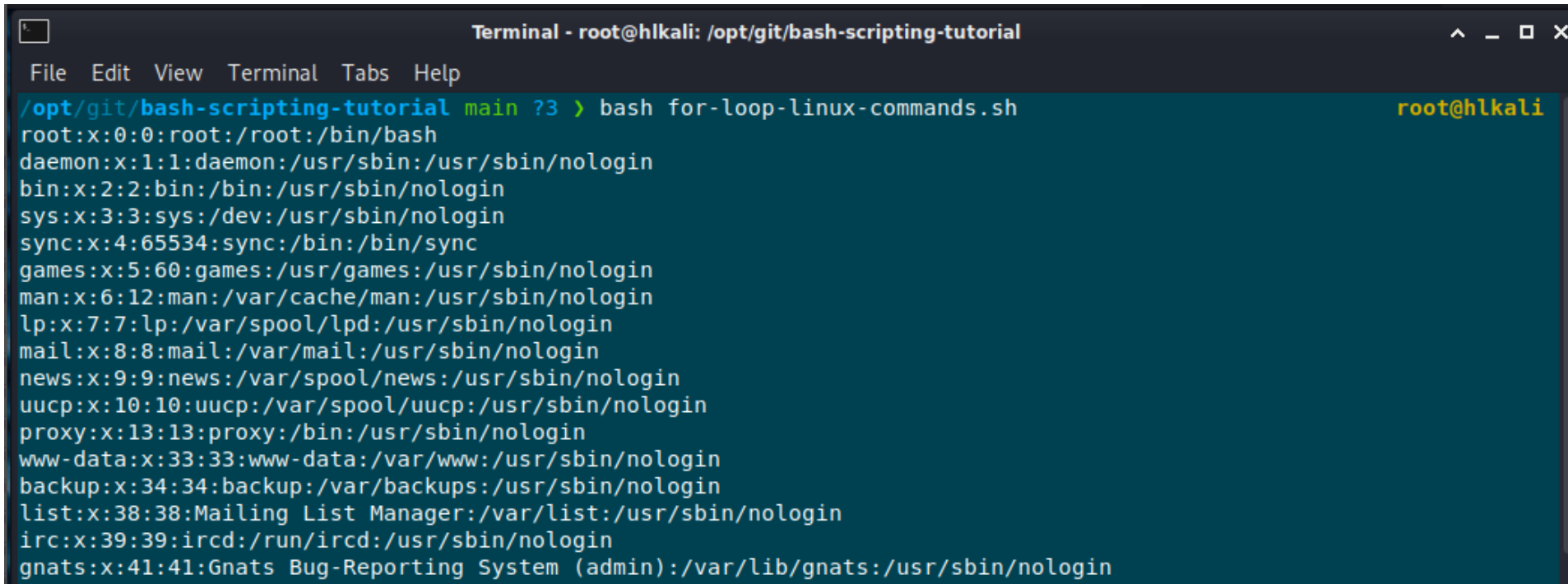
A terminal window titled "Terminal - root@hlkali: /opt/git/bash-scripting-tutorial" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is "/opt/gi/bash-scripting-tutorial main ?3 >". The user has run "bash for-loop-files.sh", and the output is displayed. The output shows the content of 'myfile' and 'myvariables' files. The content of 'myfile' is "i want to replace hostname with livedcd" repeated twice. The content of 'myvariables' is "firststring='Hacking-Lab is'" and "secondstring='cool!'".

```
/opt/gi/bash-scripting-tutorial main ?3 > bash for-loop-files.sh root@hlkali
=== content of myfile ===
i want to replace hostname with livedcd
i want to replace hostname with livedcd i want to replace hostname with livedcd
=== content of myvariables ===
firststring="Hacking-Lab is"
secondstring="cool!"
```

# for loop – commands

```
#!/bin/bash
```

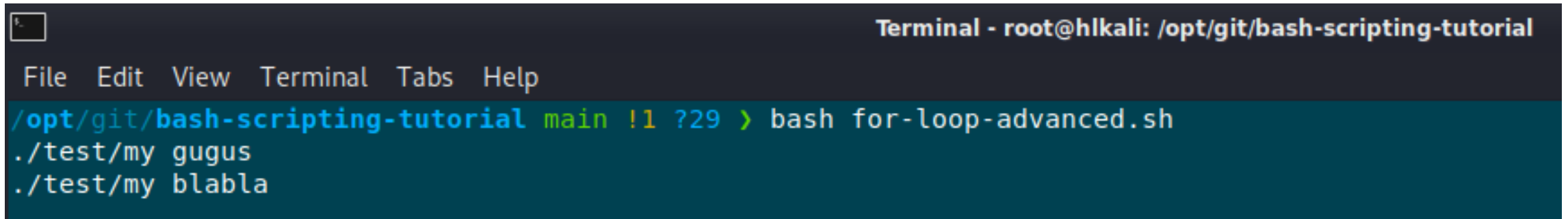
```
for line in "`cat /etc/passwd`; do  
    echo "$line"  
done
```

A terminal window titled "Terminal - root@hlkali: /opt/git/bash-scripting-tutorial" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the execution of a script named "for-loop-linux-commands.sh". The output lists system users and their details: root, daemon, bin, sys, sync, games, man, lp, mail, news, uucp, proxy, www-data, backup, list, ircd, and gnats. The prompt "root@hlkali" is visible in the top right corner of the terminal area.

```
Terminal - root@hlkali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main ?3 > bash for-loop-linux-commands.sh  
root@hlkali  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
ircd:x:39:39:ircd:/run/ircd:/usr/sbin/nologin  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
```

# for loop in context

```
#!/bin/bash
# replace all spaces with underscore
DIR="."
find $DIR -type f | while read file; do
    if [[ "$file" = *[:space:]* ]]; then
        echo $file
        #mv "$file" `echo $file | tr ' ' '_'`
    fi;
done
```

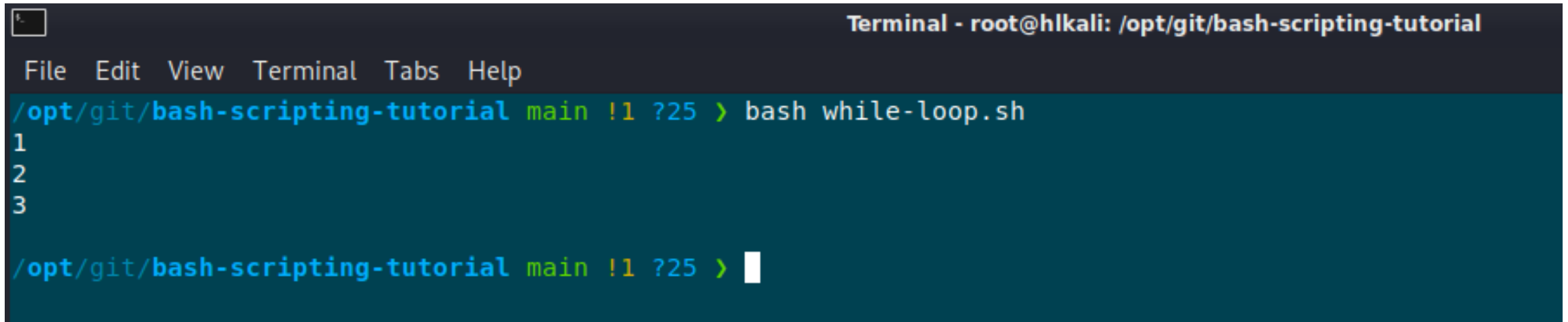
A terminal window with a dark background. The title bar reads "Terminal - root@hikali: /opt/git/bash-scripting-tutorial". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The command prompt shows the user is in the directory "/opt/git/bash-scripting-tutorial" and has run "bash for-loop-advanced.sh". The output of the script is displayed on two lines: "./test/my gugus" and "./test/my blabla".

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?29 > bash for-loop-advanced.sh
./test/my gugus
./test/my blabla
```

# while loop

```
#!/bin/bash
```

```
for i in 1 2 3; do  
    echo $i  
done
```

A terminal window titled "Terminal - root@h1kali: /opt/git/bash-scripting-tutorial" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is "/opt/git/bash-scripting-tutorial main !1 ?25 >". The user enters "bash while-loop.sh". The script outputs "1", "2", and "3" on separate lines. The prompt returns to "/opt/git/bash-scripting-tutorial main !1 ?25 >".

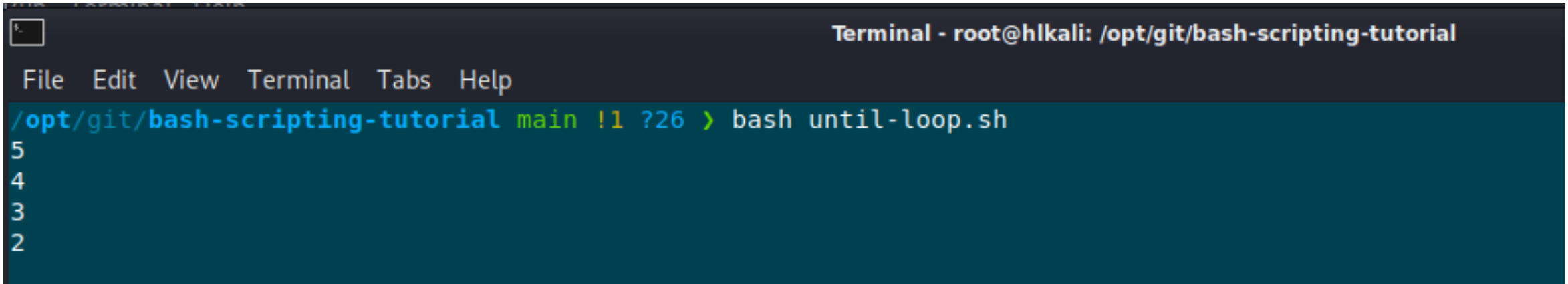
```
Terminal - root@h1kali: /opt/git/bash-scripting-tutorial  
File Edit View Terminal Tabs Help  
/opt/git/bash-scripting-tutorial main !1 ?25 > bash while-loop.sh  
1  
2  
3  
/opt/git/bash-scripting-tutorial main !1 ?25 > 
```



# until loop

```
#!/bin/bash

counter=6
until [ $counter -lt 3 ]; do
    let counter-=1
    echo $counter
done
```

A terminal window titled "Terminal - root@hikali: /opt/git/bash-scripting-tutorial" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is "/opt/git/bash-scripting-tutorial main !1 ?26 >". The user has entered "bash until-loop.sh". The output shows the numbers 5, 4, 3, and 2, each on a new line, indicating the loop executed four times before terminating.

```
Terminal - root@hikali: /opt/git/bash-scripting-tutorial
File Edit View Terminal Tabs Help
/opt/git/bash-scripting-tutorial main !1 ?26 > bash until-loop.sh
5
4
3
2
```

# Redirections

STDOUT	1
STDERR	2

```
echo "Redirect STDOUT to FILE" > /tmp/file  
echo "Redirect STDOUT to FILE" 1> /tmp/file
```

```
echo "Redirect and Append STDOUT to FILE" >> /tmp/file  
echo "Redirect and Append STDOUT to FILE" 1>> /tmp/file
```

```
echo "Redirect STDERR to FILE" 2> /tmp/file  
echo "Redirect and Append STDERR to FILE" 2>> /tmp/file
```

# Redirections

STDOUT	1
STDERR	2

```
echo "Redirect STDOUT to STDERR" 1>&2
```

```
echo "Redirect STDERR to STDOUT" 2>&1
```

```
echo "Redirect STDERR to STDOUT" 2>&1
```

```
echo "Redirect STDOUT and STDERR to FILE" &> /tmp/stdout-and-stderr.log
```

