

Name: Lương Toàn Bách

ID: 21521845

Class: KHTN2021

OPERATING SYSTEM LAB 2'S REPORT

SUMMARY

		Task	Status	Page
Section 2.4	Section 2.4.1	Using shell from cmd	Done	1
		Using shell from script file	Done	2
		Execute script	Done	3
	Section 2.4.2	Using variables	Done	4
		Metacharacters of shell	Done	5
		Environment variable	Done	6
		Parameter variable	Done	7
	Section 2.4.3	test command or []	Done	8
		Compare number and string	Done	9
	Section 2.4.4	if command	Done	10
		elif command	Done	11
		Problem with variables	Done	12
		for command	Done	13
		while command	Done	14
		until command	Done	15
		case command	Done	16

Self-scores: 10

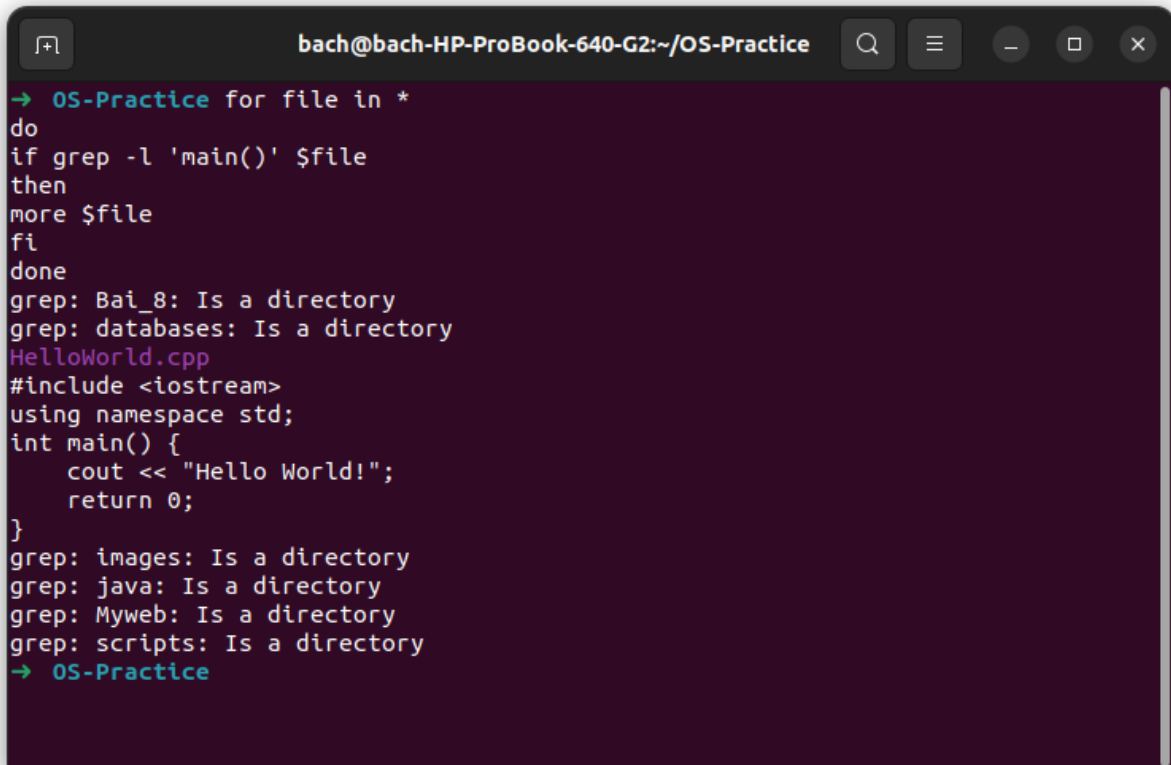
Note: Export file to **PDF and name the file by following format:
LAB X – <Student ID>.pdf*

Section 2.4

2.4.1. Using shell like a programming language

2.4.1.1. Using shell from cmd

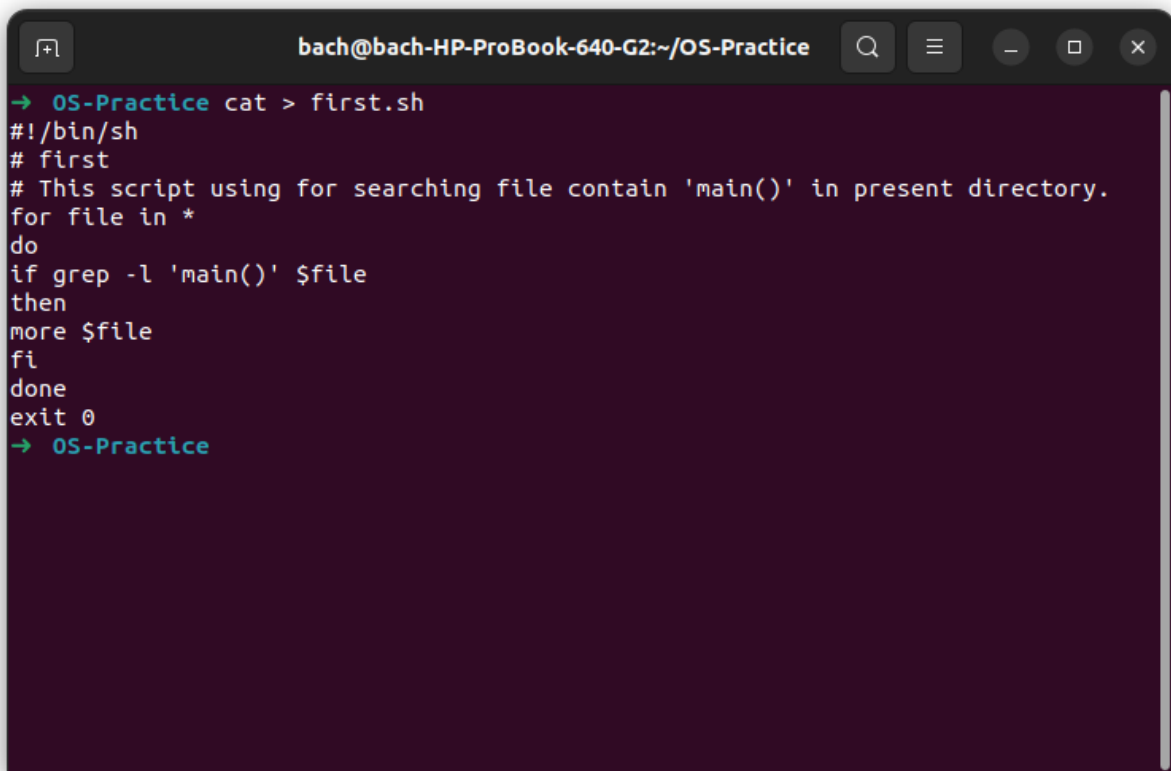
- The following picture showing how to use shell to search file contain 'main()' in present directory:

A terminal window with a dark purple background. The title bar shows the user 'bach' at host 'bach-HP-ProBook-640-G2' in directory '~/OS-Practice'. The terminal displays a shell script being executed. The script uses a 'for' loop to iterate over files in the current directory. For each file, it checks if it contains the string 'main()' using the 'grep' command. If found, it uses 'more' to display the file's content. The output shows that 'Bai_8' and 'databases' are directories, while 'HelloWorld.cpp' is a file containing C++ code with a 'main()' function. Other directories like 'images', 'java', 'Myweb', and 'scripts' are also listed. The prompt returns to the shell after the script finishes.

```
→ OS-Practice for file in *
do
if grep -l 'main()' $file
then
more $file
fi
done
grep: Bai_8: Is a directory
grep: databases: Is a directory
HelloWorld.cpp
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!";
    return 0;
}
grep: images: Is a directory
grep: java: Is a directory
grep: Myweb: Is a directory
grep: scripts: Is a directory
→ OS-Practice
```

2.4.1.2. Using shell from script file

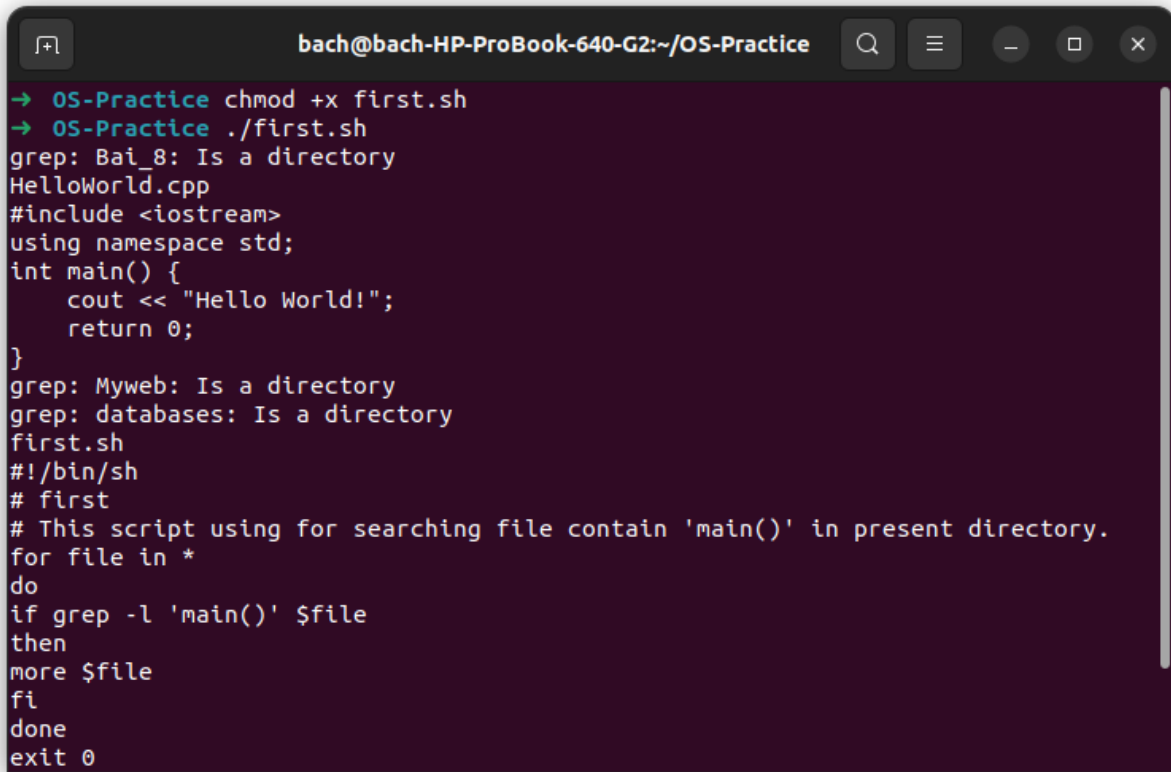
- The following picture represent the script that have the same usage with the commands in 2.4.1 section:



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice cat > first.sh
#!/bin/sh
# first
# This script using for searching file contain 'main()' in present directory.
for file in *
do
if grep -l 'main()' $file
then
more $file
fi
done
exit 0
→ OS-Practice
```

2.4.1.3. Execute script

- Using chmod command to make first.sh excuteable.
- Run file first.sh

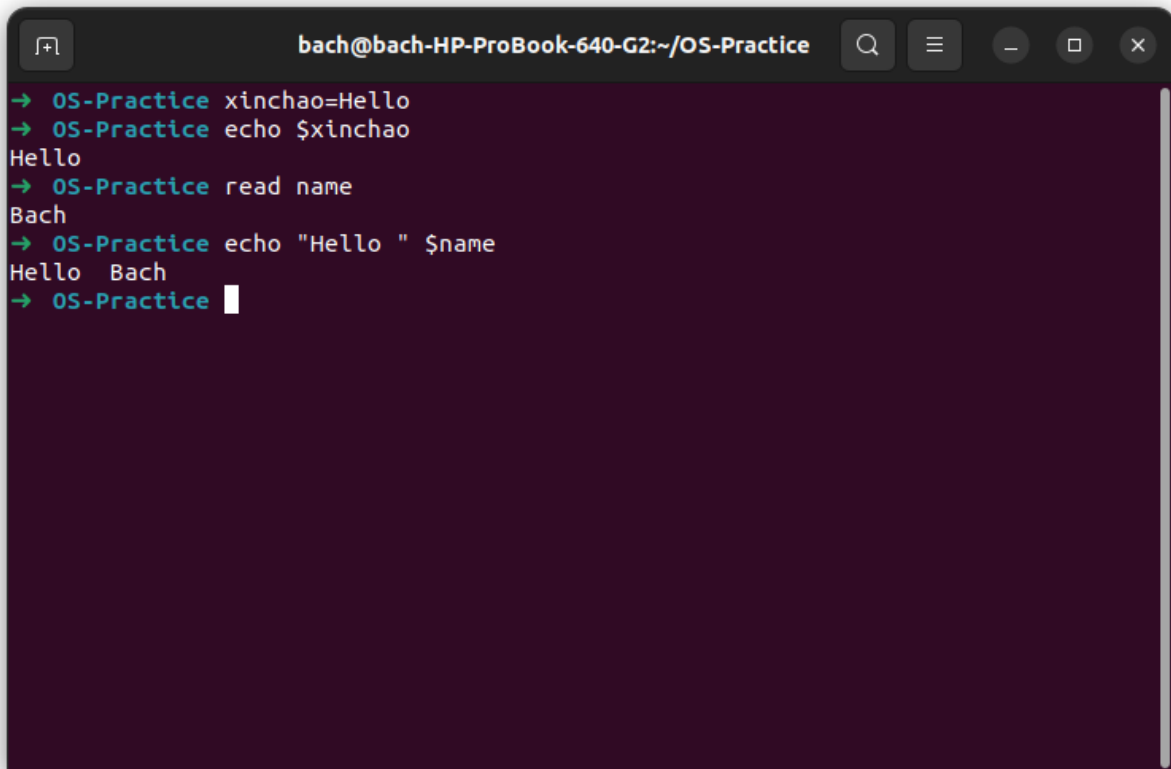


```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice chmod +x first.sh
→ OS-Practice ./first.sh
grep: Bai_8: Is a directory
HelloWorld.cpp
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!";
    return 0;
}
grep: Myweb: Is a directory
grep: databases: Is a directory
first.sh
#!/bin/sh
# first
# This script using for searching file contain 'main()' in present directory.
for file in *
do
if grep -l 'main()' $file
then
more $file
fi
done
exit 0
```

2.4.2. Shell syntax

2.4.2.1. Using variables

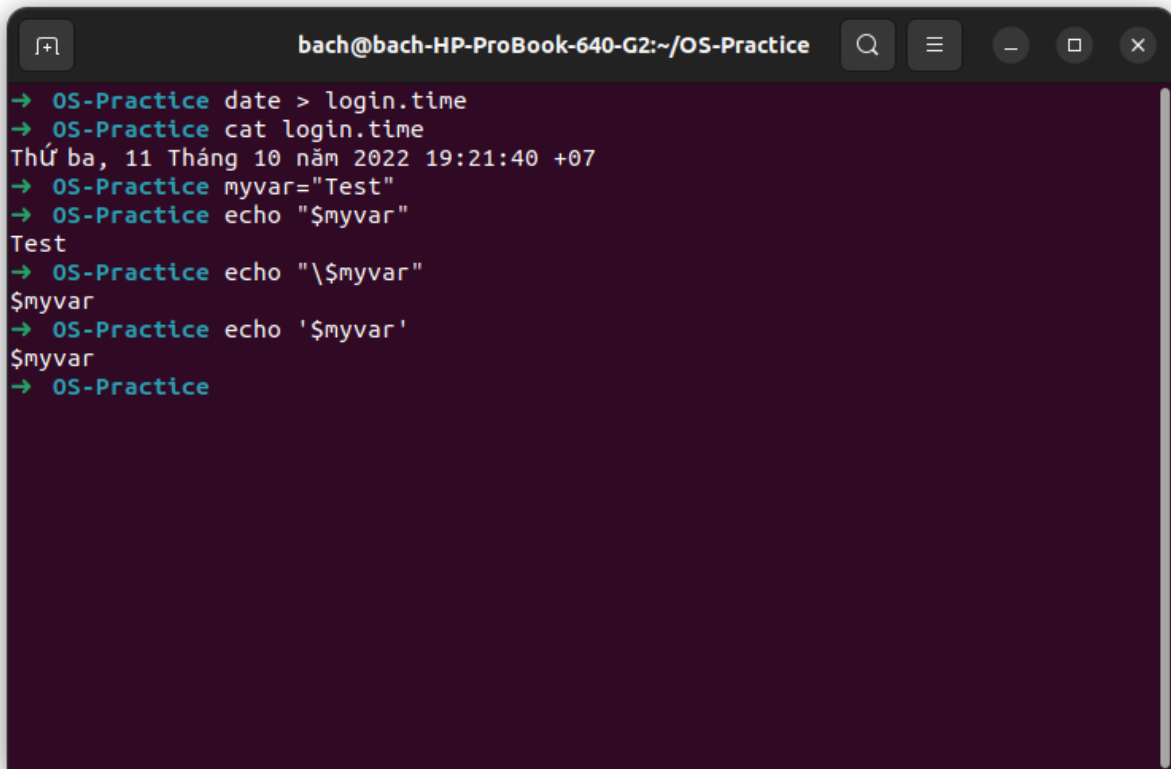
- Set variable xinchao value “Hello”
- Print value of variable xinchao using echo command.
- Using read command to get the input from the user.
- Print out the “Hello “ \$name



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice xinchao=Hello
→ OS-Practice echo $xinchao
Hello
→ OS-Practice read name
Bach
→ OS-Practice echo "Hello " $name
Hello  Bach
→ OS-Practice
```

2.4.2.2. Metacharacters of shell

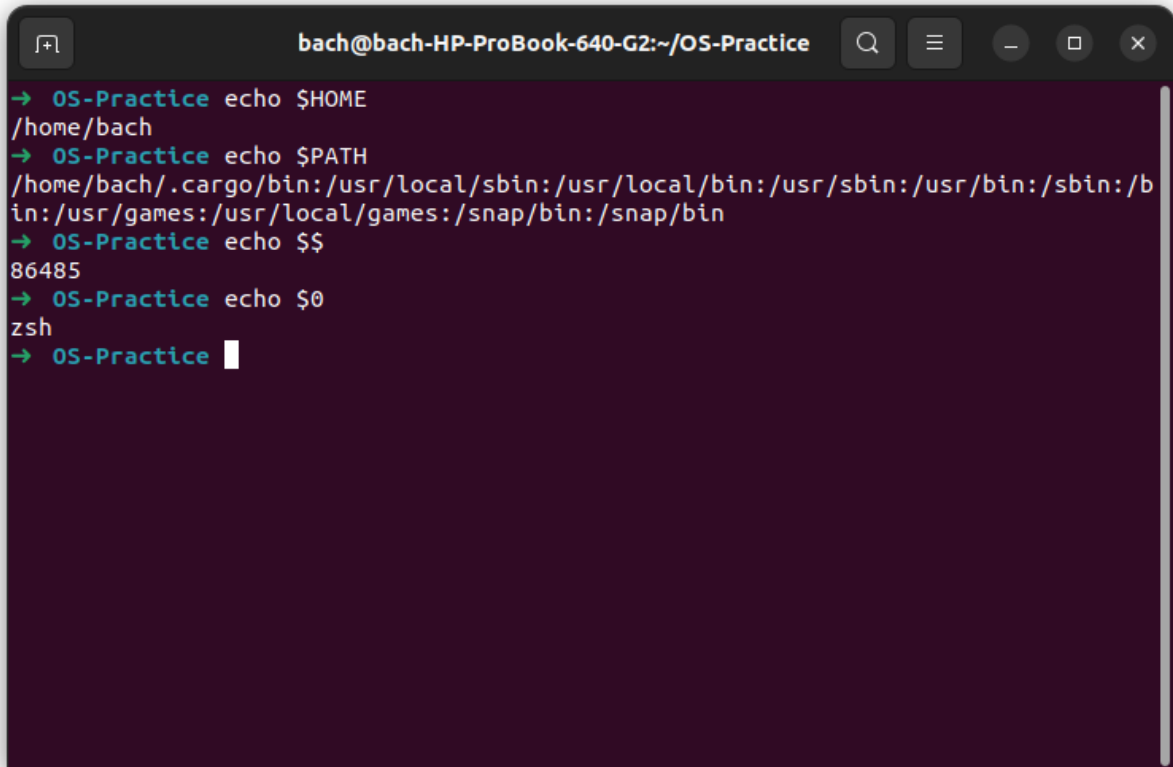
- Using operate “>” to redirect output of date command(which provide present time) to login.time file
- As you can see variable in “” can be get value with prefix \$ and disable that ability with / before \$.
- ‘’ is stronger than “” that it can not get value of variable.



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice date > login.time
→ OS-Practice cat login.time
Thứ ba, 11 Tháng 10 năm 2022 19:21:40 +07
→ OS-Practice myvar="Test"
→ OS-Practice echo "$myvar"
Test
→ OS-Practice echo "\$myvar"
$myvar
→ OS-Practice echo '$myvar'
$myvar
→ OS-Practice
```

2.4.2.3. Environment variables

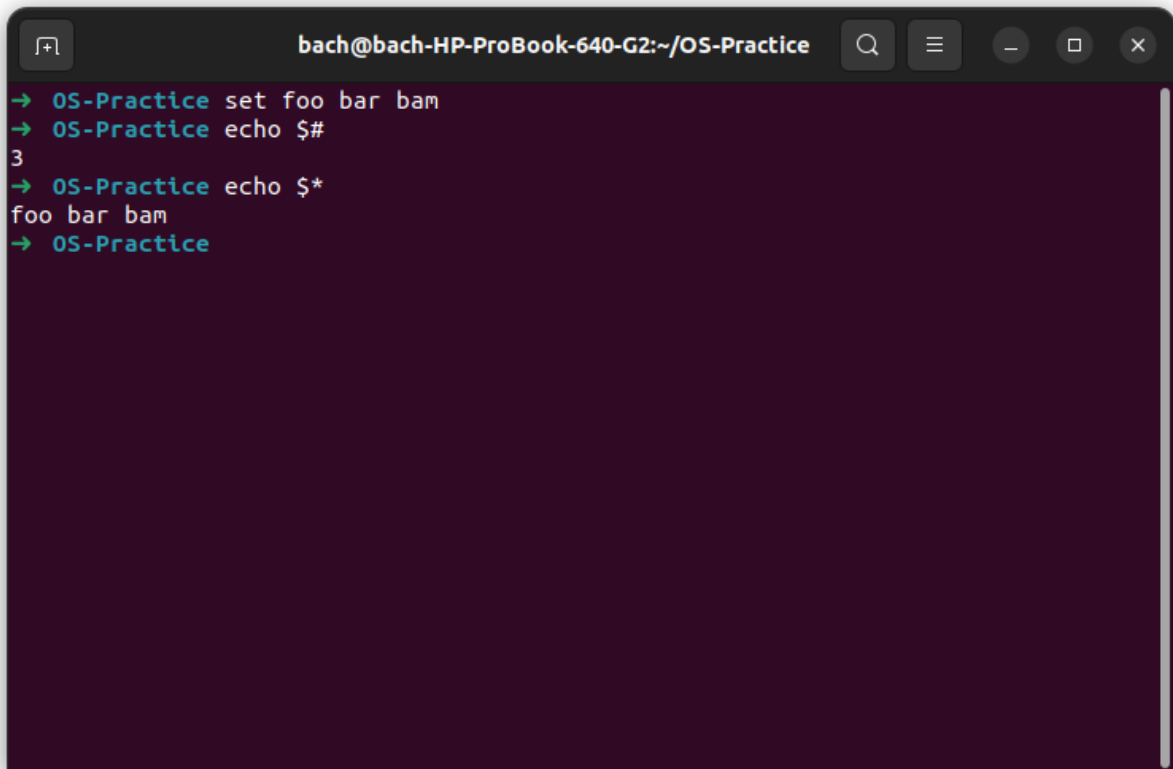
- Enviroment variables
 - \$HOME contain value of home directory.
 - \$PATH contain list of directory separate by “:”.
 - \$0 contain name of program cmd.
 - \$\$ contain process id.

A terminal window with a dark purple background. The title bar shows 'bach@bach-HP-ProBook-640-G2:~/OS-Practice'. The terminal contains the following text:

```
→ OS-Practice echo $HOME
/home/bach
→ OS-Practice echo $PATH
/home/bach/.cargo/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/b
in:/usr/games:/usr/local/games:/snap/bin:/snap/bin
→ OS-Practice echo $$
86485
→ OS-Practice echo $0
zsh
→ OS-Practice
```

2.4.2.4. Parameter Variables

- Set 3 parameters variables foo, bar, bam.
- \$# return the number of parameter variables.
- List all parameter variables on the screen using \$* and echo command.

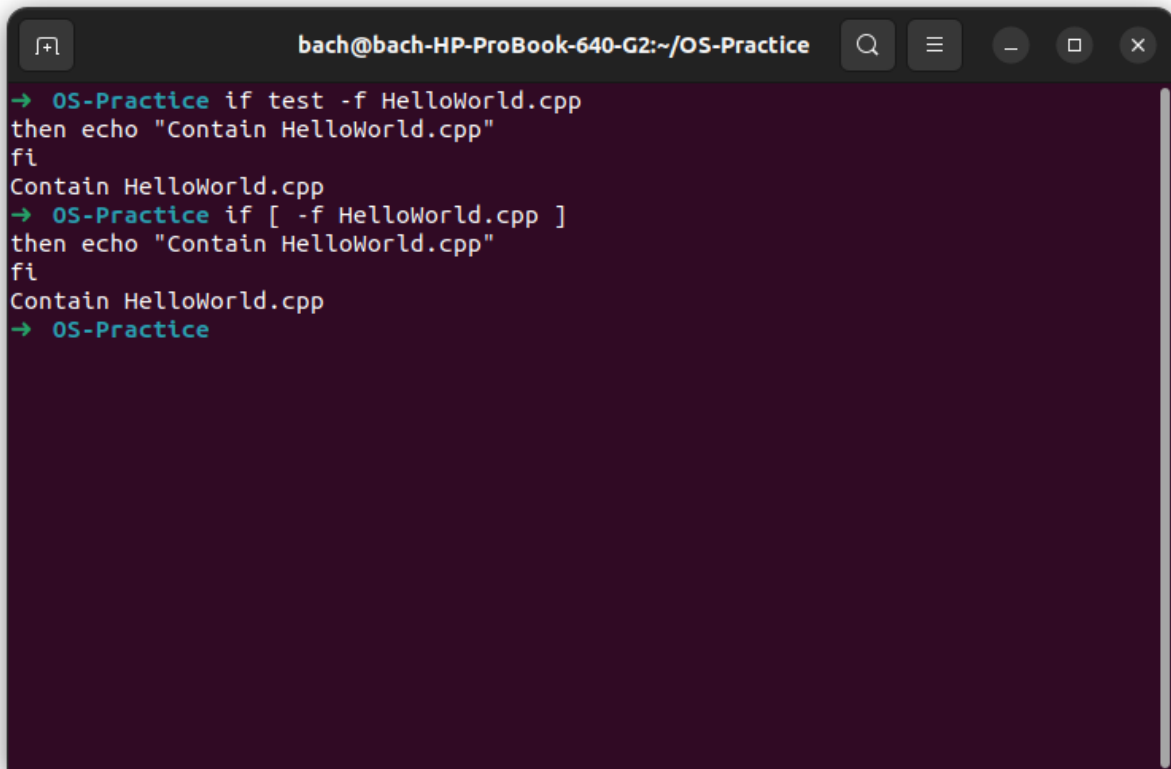


```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice set foo bar bam
→ OS-Practice echo $#
3
→ OS-Practice echo $*
foo bar bam
→ OS-Practice
```

2.4.3. Condition Structure.

2.4.3.1. test command or [].

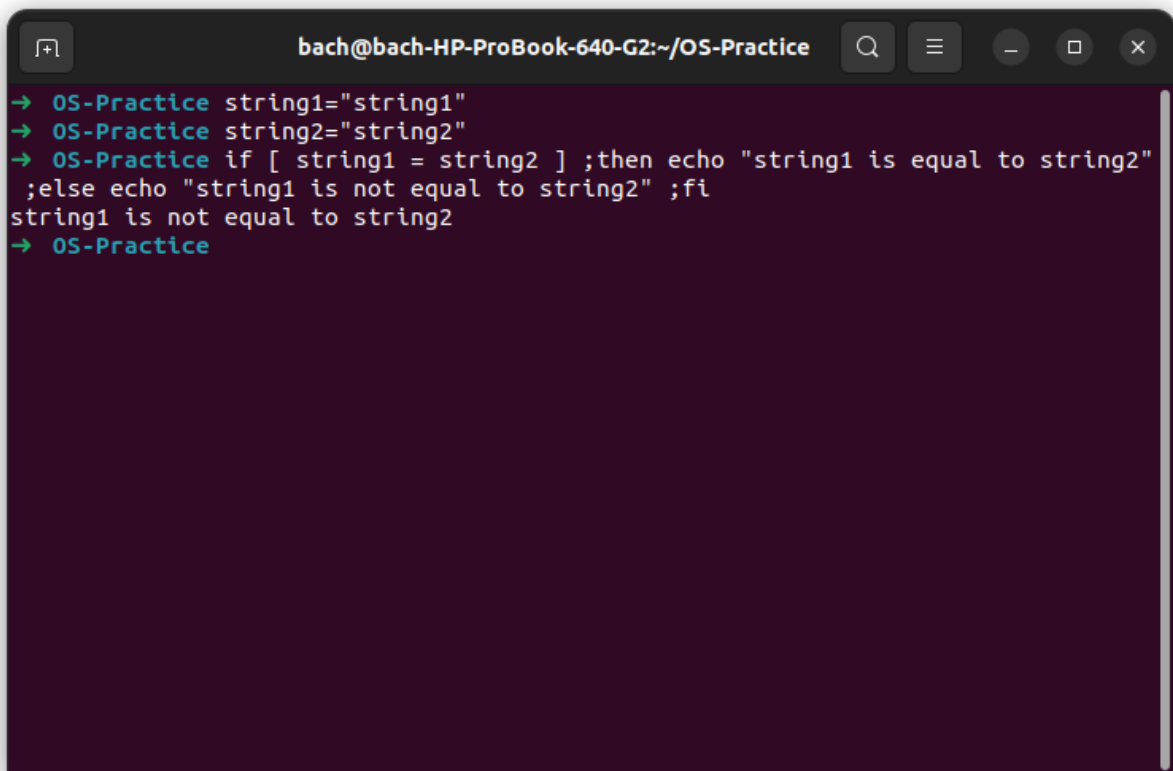
- Using test command with flag -f to check HelloWorld.cpp file exist or not.
- Using [] instead of test with same usage.



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice if test -f HelloWorld.cpp
then echo "Contain HelloWorld.cpp"
fi
Contain HelloWorld.cpp
→ OS-Practice if [ -f HelloWorld.cpp ]
then echo "Contain HelloWorld.cpp"
fi
Contain HelloWorld.cpp
→ OS-Practice
```

2.4.3.2. Compare number and string

- set variable string1 to “string1”
- set variable string2 to “string2”
- using if to check value contained in two variables are different or not.

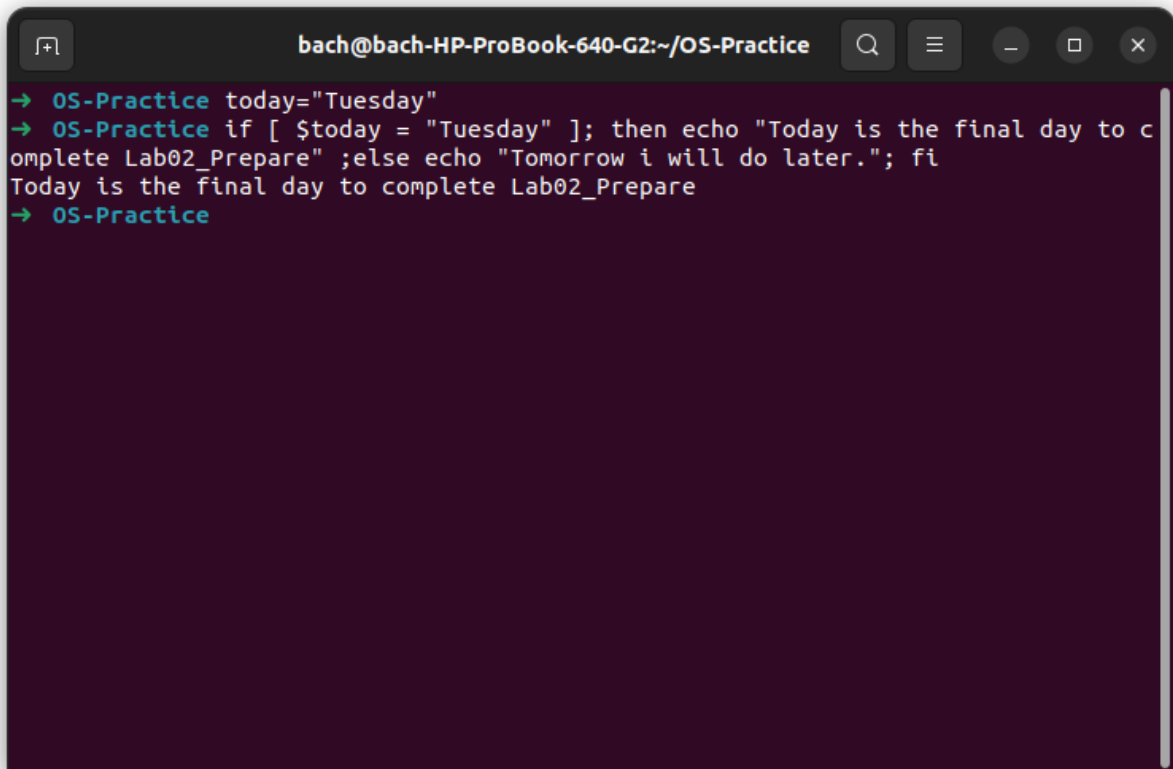


```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice string1="string1"
→ OS-Practice string2="string2"
→ OS-Practice if [ string1 = string2 ] ;then echo "string1 is equal to string2"
;else echo "string1 is not equal to string2" ;fi
string1 is not equal to string2
→ OS-Practice
```

2.4.4. Control Structure

2.4.4.1. if command

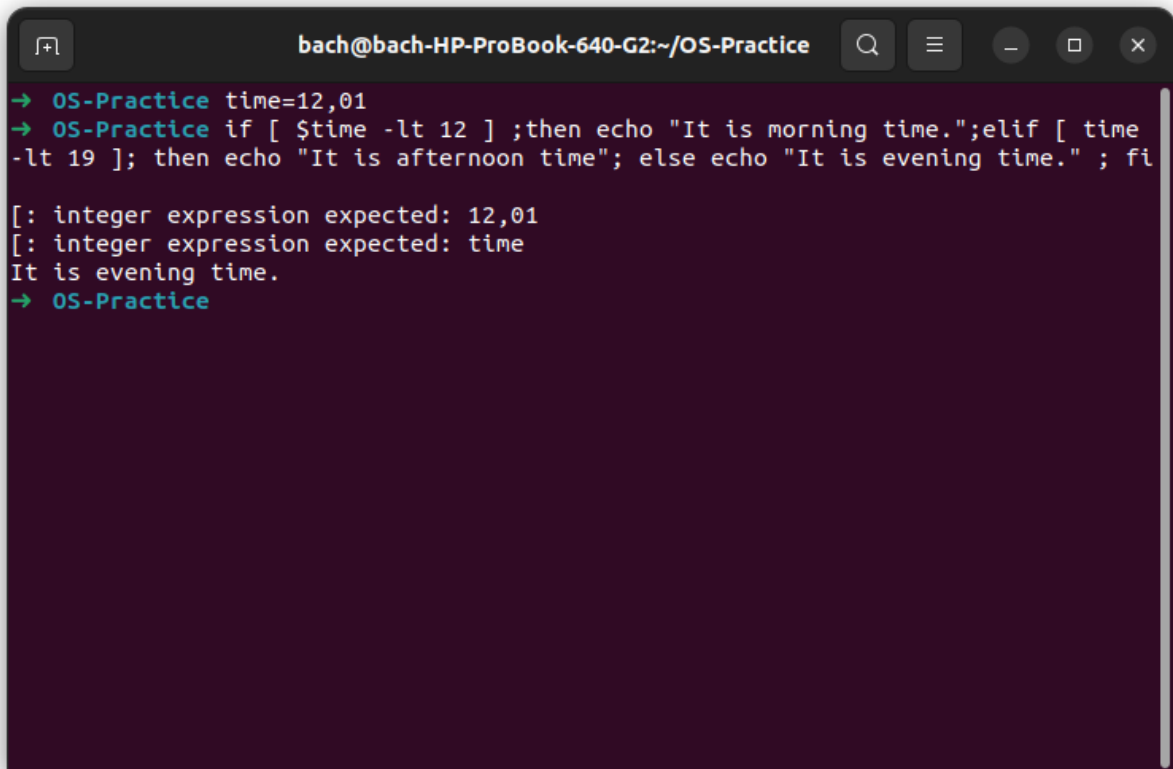
- set variable today equal to “Tuesday”.
- using if command to check is today Tuesday or not.



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice today="Tuesday"
→ OS-Practice if [ $today = "Tuesday" ]; then echo "Today is the final day to c
omplete Lab02_Prepare" ;else echo "Tomorrow i will do later."; fi
Today is the final day to complete Lab02_Prepare
→ OS-Practice
```

2.4.4.2. elif command

- set variable time equals to 12,01
- if time < 12 then it is morning time.
- elif time < 19 then it is afternoon time.
- else it is evening time.



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice time=12,01
→ OS-Practice if [ $time -lt 12 ] ;then echo "It is morning time.";elif [ time
-lt 19 ]; then echo "It is afternoon time"; else echo "It is evening time." ; fi

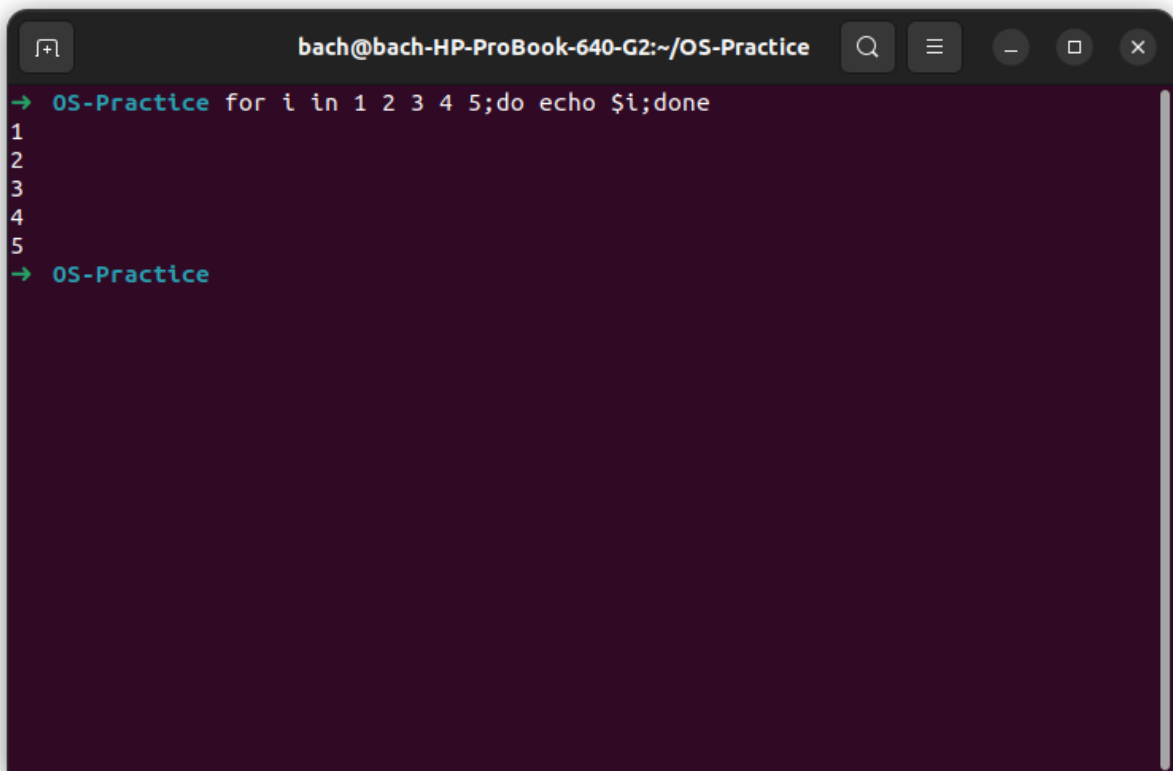
[: integer expression expected: 12,01
[: integer expression expected: time
It is evening time.
→ OS-Practice
```

2.4.4.3. Problems with variables

- while using read command to take input to user there is a case that user input an empty string so that the operation of if command will get trouble.

2.4.4.4. for command

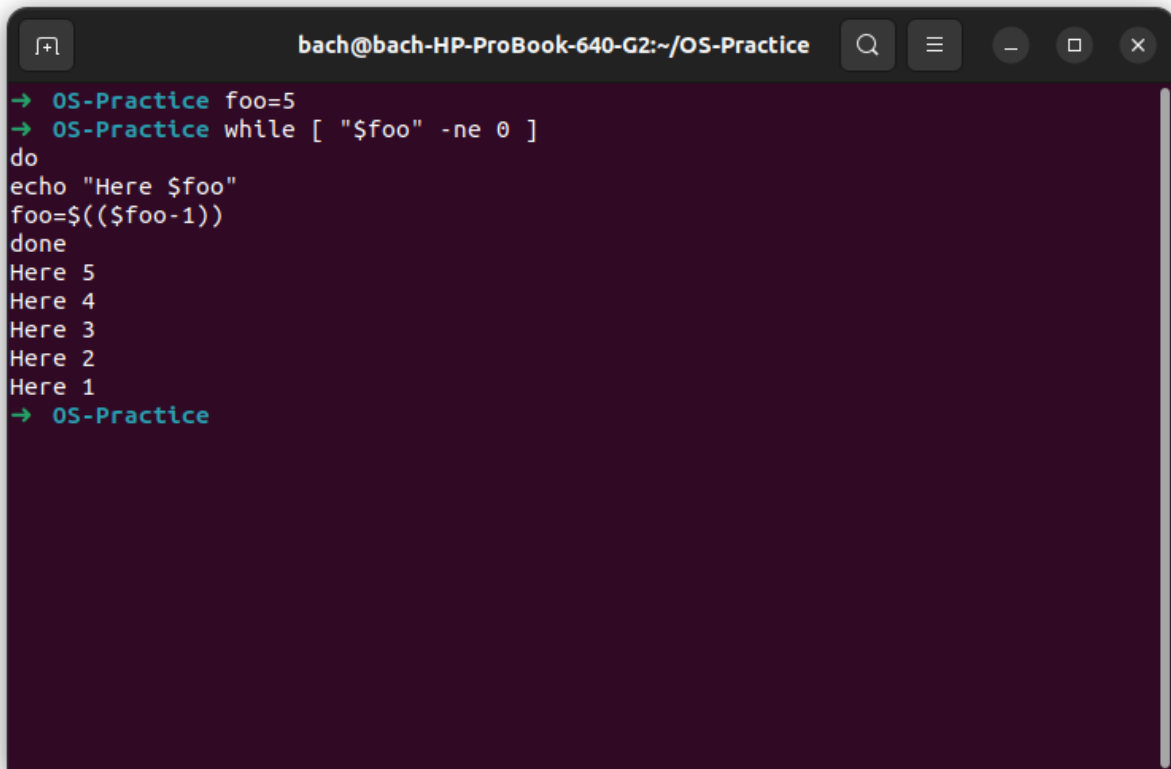
- using for variable i to iterate over all value from 1 to 5.
- Then print out the screen using echo.



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice for i in 1 2 3 4 5;do echo $i;done
1
2
3
4
5
→ OS-Practice
```

2.4.4.5. while command

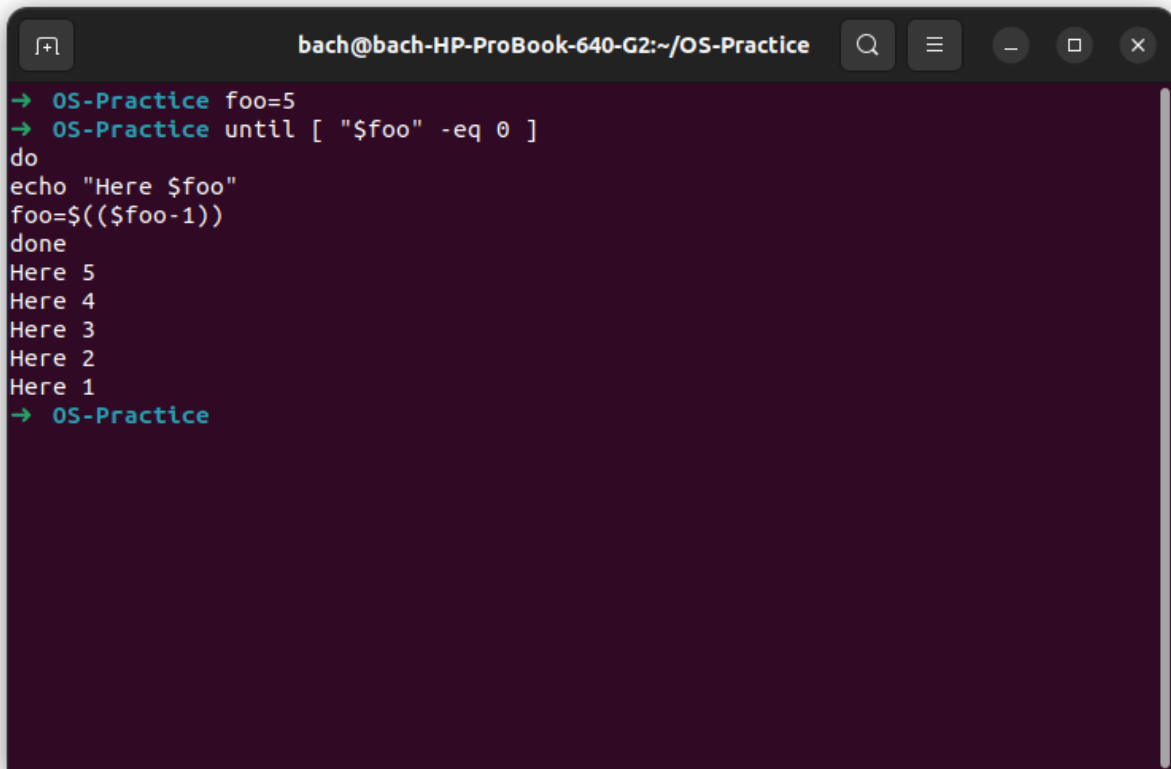
- set variable foo equals to 5
- if foo is not equals to 0 print out Here \$foo
- Subtract foo by 1 then again.

A terminal window with a dark purple background. The title bar shows the user 'bach' on a machine named 'bach-HP-ProBook-640-G2' in the directory '~/OS-Practice'. The terminal contains a script that initializes 'foo' to 5 and enters a 'while' loop that prints 'Here \$foo' and decrements 'foo' until it reaches 1. The output shows the script's execution from 'Here 5' down to 'Here 1'.

```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice foo=5
→ OS-Practice while [ "$foo" -ne 0 ]
do
echo "Here $foo"
foo=$((foo-1))
done
Here 5
Here 4
Here 3
Here 2
Here 1
→ OS-Practice
```

2.4.4.6. until command

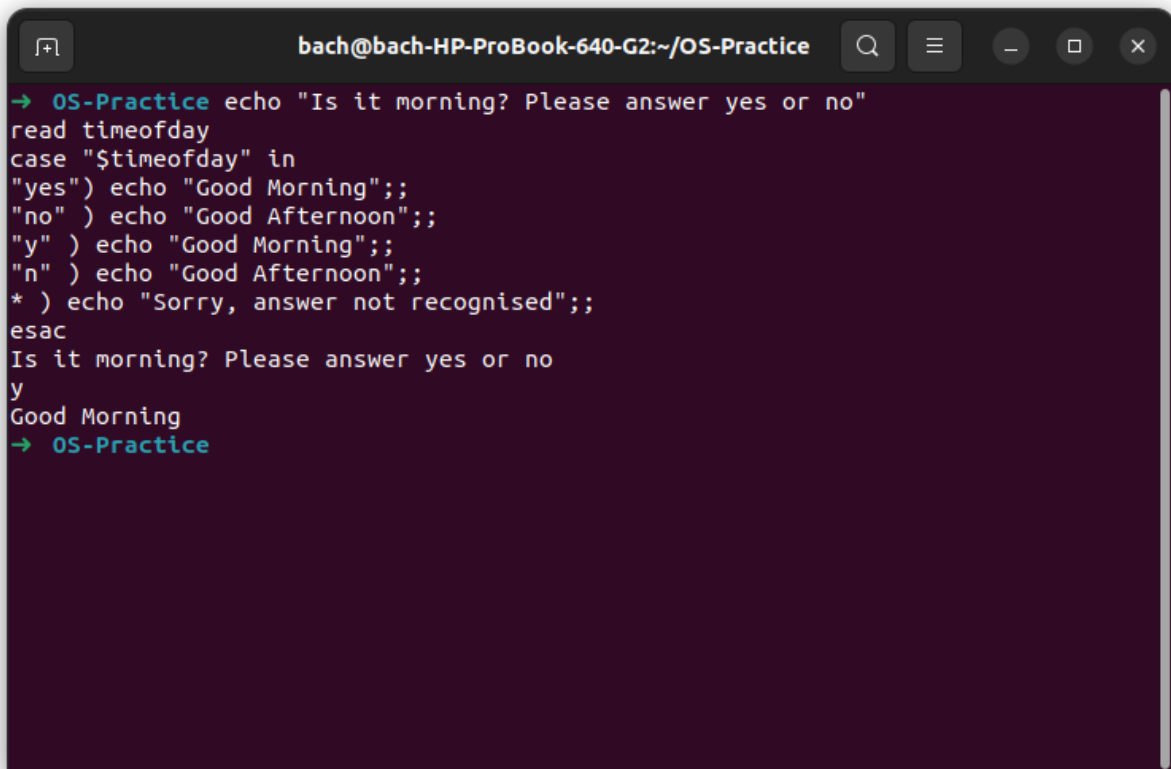
- Using same method example in 2.4.4.5 but now we using until command

A terminal window with a dark purple background. The title bar shows the user 'bach' on a machine named 'bach-HP-ProBook-640-G2' in the directory '~/OS-Practice'. The terminal contains a shell script that uses a 'do-until' loop to print the values of a variable 'foo' from 5 down to 1. The script is as follows:

```
→ OS-Practice foo=5
→ OS-Practice until [ "$foo" -eq 0 ]
do
echo "Here $foo"
foo=$((foo-1))
done
Here 5
Here 4
Here 3
Here 2
Here 1
→ OS-Practice
```

2.4.4.7. case command

- Using read command to get the input of timeofday variable.
- Due to what user input the program will answer difference.

A terminal window with a dark background and light-colored text. The window title is "bach@bach-HP-ProBook-640-G2:~/OS-Practice". The terminal shows a shell script being executed. The script prompts the user with "Is it morning? Please answer yes or no". The user enters "y", and the script outputs "Good Morning". The prompt "→ OS-Practice" is visible at the end of the line.

```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice echo "Is it morning? Please answer yes or no"
read timeofday
case "$timeofday" in
"yes") echo "Good Morning";;
"no" ) echo "Good Afternoon";;
"y" ) echo "Good Morning";;
"n" ) echo "Good Afternoon";;
* ) echo "Sorry, answer not recognised";;
esac
Is it morning? Please answer yes or no
y
Good Morning
→ OS-Practice
```