



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

گزارش کار آزمایشگاه سیستم عامل – شماره ۳

دستور نویسی در سیستم عامل

حسنا اویارحسینی – ۹۸۲۳۰۱۰

استاد درس: جناب آقای مهندس کیخا

نیمسال دوم سال تحصیلی ۱۴۰۰-۰۱

بخش ۱) سوالات متن دستورکار:

- متغیرهای خاص:

متغیر خاص	محتوا
\$0	به کمک این متغیر به اسم فایل که ۰ امین آرگومان است دست میابیم
\$1-\$9	به کمک این متغیر ها به آرگومان های اول تا نهم دست میابیم.
\$#	به کمک این متغیر به تعداد آرگومان های ورودی میرسیم
\$@	این متغیر حاوی تمام آرگومان های ورودی است
\$\$	این متغیر Process id مربوط به shell فعلی را میدهد.
\$USER	نام کاربر فعلی در این متغیر است

- چگونه باید به مقدار ۱۰-امین آرگومان دست یافت؟

از {} استفاده کرده و عدد آرگومان را در آن وارد میکنیم برای مثال: \${10}

- p و -sp چه امکانی را فراهم میکنند؟

-p: گزینه p باعث می شود ورودی به عنوان یک اعلان خوانده شود، به این معنی که قبل

از تلاش برای خواندن ورودی، یک خط جدید آخر اضافه نمی شود.

-s: از گزینه s استفاده میکنیم که هنگام خواندن ورودی را روی ترمینال چاپ نشود.

-sp: ترکیب هر دو option بالا می باشد و هر دو را به صورت همزمان اعمال میکند.

- خروجی کد زیر چیست؟

```
let a=10+8
echo $a
expr 5 \* 4
expr 5 / 4
expr 11 % 2
```

```
a=$( expr 10 - 3 )  
echo $a  
b=$(( a + 3 ))  
echo $b  
((b++))  
echo $b
```

خروجی:

```
18  
20  
1  
1  
7  
10  
11
```

بخش ۲) آزمایش‌ها:

- ۱- دو عددی که به صورت آرگومان به آن داده شده را الف- با هم جمع کند و نتیجه را اعلام کند، ب- عدد بزرگتر را نمایش دهد. ج- اگر کاربر در وارد کردن ورودی ها اشتباه کرده بود راهنمای مناسبی چاپ کند.

```
#!/bin/bash

if [ $# -lt 2 ]
then
    echo "Error: You should enter two number!"
    exit 1
fi

re='^[+]?[0-9]+$'

if ! [[ $1 =~ $re ]]
then
    echo "Error: '$1' is not a number"
    exit 1
fi

if ! [[ $2 =~ $re ]]
then
    echo "Error: '$2' is not a number"
    exit 1
fi

((num1 = $1))
((num2 = $2))
((result = $num1 + $num2))
echo "sum is = $result"

if [ $num1 -gt $num2 ]
then
    echo "larger number = $num1"
else
    echo "larger number = $num2"
fi
```

بخش ج

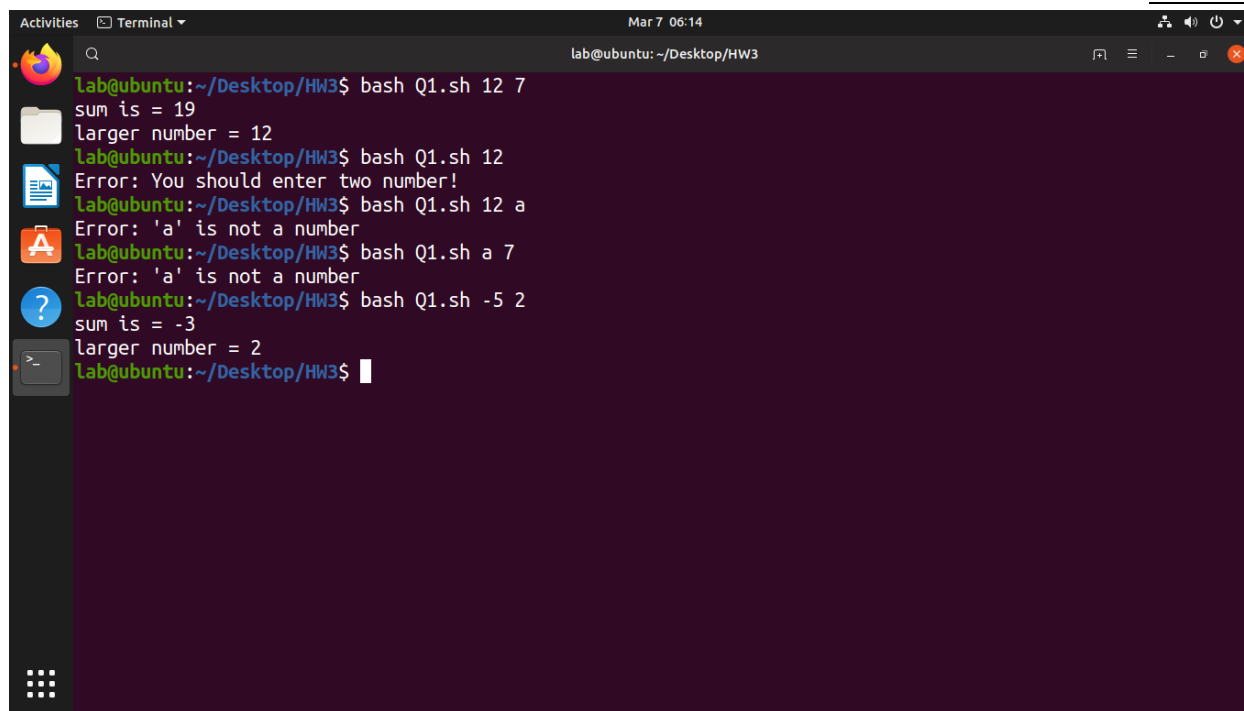
بخش الف

بخش ب

در ابتدا به کمک متغیر `$#` تعداد آرگومان هایی که از سمت ورودی وارد شده اند را بررسی میکنیم اگر کمتر از دو بود خطا می‌دهیم و برنامه را تمام میکنیم اما در غیر این صورت بررسی میکنیم که آیا هر دو آرگومان ورودی عدد صحیح هستند یا خیر برای این کار از REGEX استفاده میکنیم، در

واقع به کمک عبارت $^{+}[-0-9]^+?$ تعیین میکنیم که از ابتدا تا انتهای ورودی میتواند فقط شامل اعداد و در شروع آن + یا - یا هیچ کدام باشد پس از اعتبارسنجی آرگومان ها دو آرگومان را در دو متغیر ریخته و یکبار آنها را جمع میکنیم و حاصل را نمایش میدهیم و بار دیگر به کمک دو دستور `if` بررسی میکنیم که کدام عدد بزرگتر است و آن را نمایش میدهیم.

خروجی:



```
lab@ubuntu: ~/Desktop/HW3
lab@ubuntu:~/Desktop/HW3$ bash Q1.sh 12 7
sum is = 19
larger number = 12
lab@ubuntu:~/Desktop/HW3$ bash Q1.sh 12
Error: You should enter two number!
lab@ubuntu:~/Desktop/HW3$ bash Q1.sh 12 a
Error: 'a' is not a number
lab@ubuntu:~/Desktop/HW3$ bash Q1.sh a 7
Error: 'a' is not a number
lab@ubuntu:~/Desktop/HW3$ bash Q1.sh -5 2
sum is = -3
larger number = 2
lab@ubuntu:~/Desktop/HW3$
```

۲- ماشین حساب:

```
echo "Enter first numbers : "  
read a  
  
re='^[+-]?[0-9]+$'  
if ! [[ $a =~ $re ]]  
then  
    echo "Error: '$a' is not a number"  
    exit 1  
fi  
  
# Input type of operation  
echo "Enter Operation (+, -, X, /):"  
read ch  
  
echo "Enter second numbers : "  
read b  
  
if ! [[ $b =~ $re ]]  
then  
    echo "Error: '$b' is not a number"  
    exit 1  
fi  
  
if [ $b -eq 0 ] && [ $ch = "/" ]  
then  
    echo "Error: zero division!"  
    exit 1  
fi  
  
case $ch in  
+)((res= $a + $b))  
;;  
-)((res= $a - $b))  
;;  
X)((res= $a * $b))  
;;  
/ )res=`echo "scale=2 ; $a / $b" | bc`  
;;  
*)res=`echo INVALID format`  
;;  
esac  
echo "Result : $res"
```

گرفتن ورودی و
اعتبار سنجی

بررسی خطای تقسیم بر صفر

استفاده از case برای
انجام عملیات

برای پیاده سازی ماشین حساب ابتدا عدد و علامت عملگر را از ورودی میگیریم و همانند سوال یک اعداد را اعتبار سنجی میکنیم، سپس اگر عملگر تقسیم استفاده شده بود و عدد دوم یعنی مقسوم علیه صفر بود خطای تقسیم بر صفر را برگردانده و برنامه را تمام میکنیم، در غیر اینصورت به کمک یک case با توجه به مقدار عملگر که در ch وجود دارد عملیات را انجام داده و مقدار آن را در

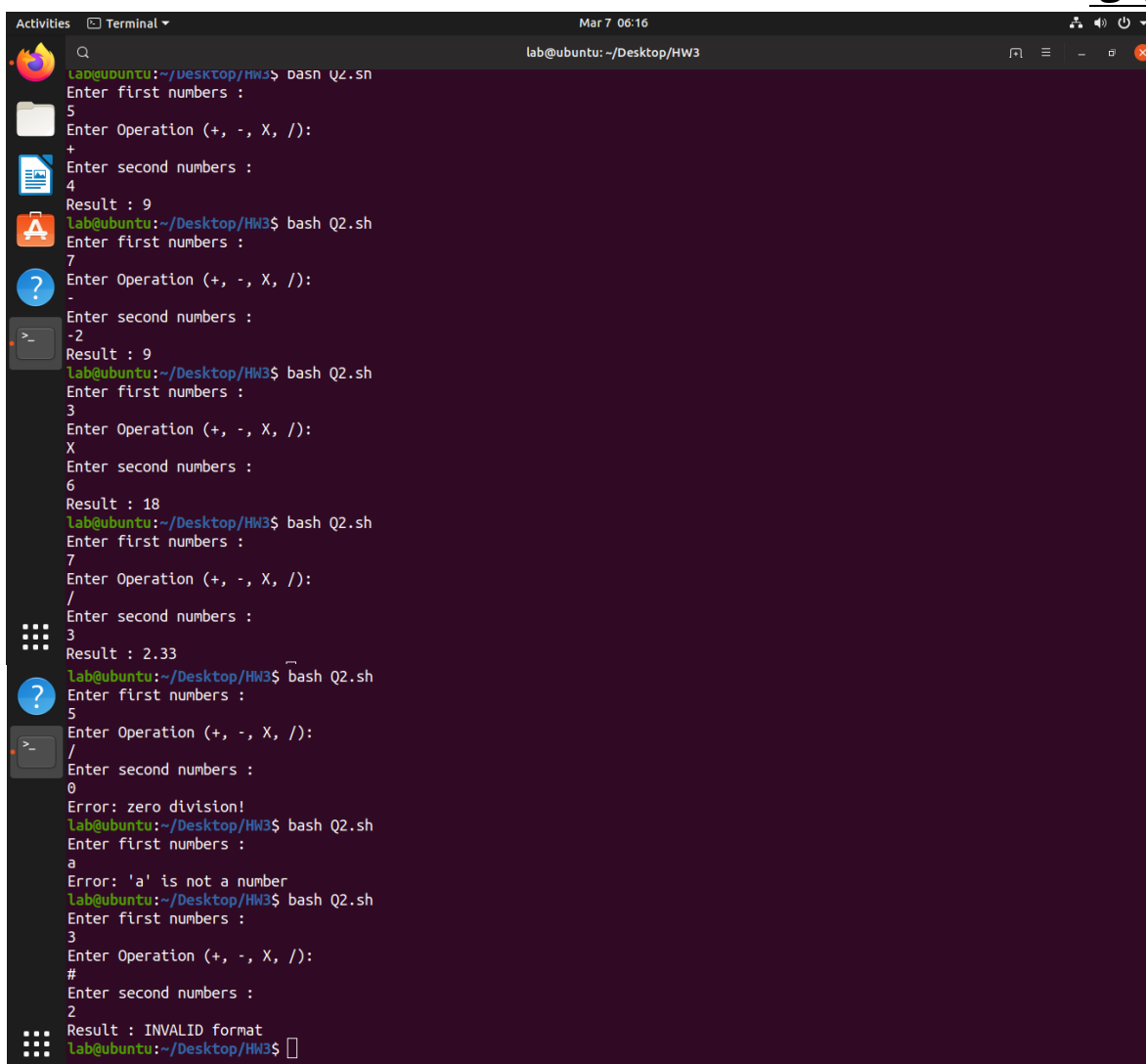
متغیر `res` میریزیم. لازم به ذکر است، برای اینکه در دستور تقسیم اعداد به صورت اعشاری تقسیم شوند و نه صحیح از دستور زیر استفاده میکنیم:

```
res=`echo "scale=2 ; $a / $b" | bc`
```

این دستور ابتدا `a/b` را با دقت دو (scale=2 ; \$a / \$b) به وسیله `echo` به کامند `bc` میدهد و سپس کامند `bc` مقدار این عبارت را حساب کرده و به عنوان خروجی باز میگرداند. کامند `bc` برای انجام محاسبات ساده در ترمینال میباشد.

در نهایت نیز اگر هیچ یک از کیس های معتبر در `case` رخ نداده بود متوجه میشویم که عملگر وارد شده ناصحیح بوده پس `res` را برابر با متن خطا قرار میدهیم. به عنوان آخرین کار نیز محتوای `res` را چاپ میکنیم.

خروجی:



```
lab@ubuntu: ~/Desktop/HW3
lab@ubuntu:~/Desktop/HW3$ bash Q2.sh
Enter first numbers :
5
Enter Operation (+, -, X, /):
+
Enter second numbers :
4
Result : 9
lab@ubuntu:~/Desktop/HW3$ bash Q2.sh
Enter first numbers :
7
Enter Operation (+, -, X, /):
-
Enter second numbers :
-2
Result : 9
lab@ubuntu:~/Desktop/HW3$ bash Q2.sh
Enter first numbers :
3
Enter Operation (+, -, X, /):
X
Enter second numbers :
6
Result : 18
lab@ubuntu:~/Desktop/HW3$ bash Q2.sh
Enter first numbers :
7
Enter Operation (+, -, X, /):
/
Enter second numbers :
3
Result : 2.33
lab@ubuntu:~/Desktop/HW3$ bash Q2.sh
Enter first numbers :
5
Enter Operation (+, -, X, /):
/
Enter second numbers :
0
Error: zero division!
lab@ubuntu:~/Desktop/HW3$ bash Q2.sh
Enter first numbers :
a
Error: 'a' is not a number
lab@ubuntu:~/Desktop/HW3$ bash Q2.sh
Enter first numbers :
3
Enter Operation (+, -, X, /):
#
Enter second numbers :
2
Result : INVALID format
lab@ubuntu:~/Desktop/HW3$
```

۳- معکوس کردن عدد و چاپ مجموع ارقام:

```
function reverse_number(){
    ((temp=$number))
    ((reverse=0))
    while [ $temp -gt 0 ]
    do
        ((reverse=$reverse * 10 + ($temp % 10)))
        ((temp=$temp/10))
    done
    echo "reverse = $reverse"
}

function sum_of_digit(){
    ((sum=0))
    while [ $number -gt 0 ]
    do
        ((sum=$sum + $number%10))
        ((number=$number/10))
    done
    echo "sum of digits = $sum"
}

while [ true ]
do

    echo "Enter number:"
    read number

    re='^[+]?[0-9]+$'
    if ! [[ $number =~ $re ]]
    then
        echo "Error: '$number' is not a number"
        exit 1
    fi

    reverse_number
    sum_of_digit
done
```

معکوس کردن عدد
و چاپ آن

محاسبه مجموع ارقام و
چاپ آن

گرفتن ورودی و
اعتبار سنجی و
فراخوانی توابع

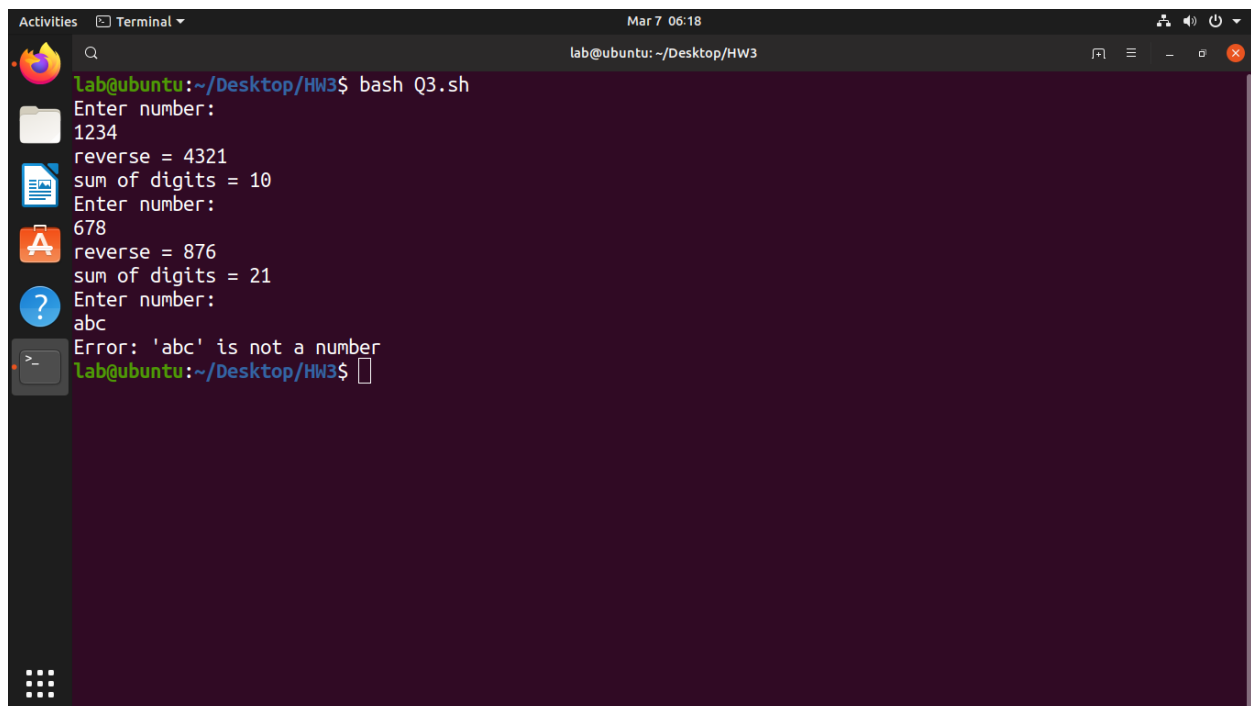
ابتدا به کمک عملیات های ریاضی دو تابع برای محاسبه معکوس عدد و جمع ارقام تعریف میکنیم که هر دو ورودی و خروجی ندارد و از متغیر `number global` استفاده میکنند. تابع `reverse` ابتدا یک کپی از عدد تهیه میکند و در یک حلقه `while` تا وقتی که کپی تهیه

شده صفر نشود یک رقم از سمت راست کم کرده و به سمت چپ متغیر `reverse` اضافه میکند و در نهایت جواب نهایی را چاپ میکند.

در تابع بعدی نیز در یک `while` هر بار یک رقم از `number` را جدا میکنیم و با `sum` قبلی جمع میزنیم این کار را تا جایی ادامه میدهیم که `number` صفر شود. سپس `sum` را چاپ میکنیم.

در بخش آخر نیز در یک `while` همواره یک عدد را به عنوان ورودی میگیریم و اگر معتبر بود معکوس و مجموع ارقامش را چاپ میکنیم.

خروجی:



```
lab@ubuntu: ~/Desktop/HW3
lab@ubuntu:~/Desktop/HW3$ bash Q3.sh
Enter number:
1234
reverse = 4321
sum of digits = 10
Enter number:
678
reverse = 876
sum of digits = 21
Enter number:
abc
Error: 'abc' is not a number
lab@ubuntu:~/Desktop/HW3$
```

۴- خط x تا y فایل:

```
#!/bin/bash
echo "Enter start line:"
read x
echo "Enter end line:"
read y
echo "Enter file name:"
read file_name
if [ $x -gt $y ]
then
    echo "Error: start number should be less than end number"
    exit 1
fi
if [[ ! -f "$file_name" ]]; then
    echo "File doesn't exist"
    exit 1
fi
i=1
while read line
do
    if [ $i -ge $x ]&&[ $i -le $y ]
    then
        echo $line
    fi
    ((i=i+1))
done < $file_name
```

گرفتن
ورودی و
اعتبار
سنجی
خواندن
خطوط

ابتدا شماره خط ابتدا و انتها را میخوانیم و سپس نام فایل را از ورودی میگیریم. بررسی میکنیم اگر فایل وجود نداشت و یا خط ابتدا بیشتر از انتها بود خطا برگردانده و برنامه را پایان میدهیم. در غیر این صورت به کمک یک `while` روی خط های فایل حرکت کرده و خطوطی را که در بازه مشخص شده هستند چاپ میکنیم.

خروجی:

برای تست کردن برنامه یک فایل `test.txt` حاوی ۱۰۰ خط با محتوای ۱ تا ۱۰۰ میسازیم و از آن استفاده میکنیم.

```
Activities Terminal Mar 7 20:07 lab@ubuntu: ~/Desktop/HW3
Enter start line:
4
Enter end line:
13
Enter file name:
test.txt
4
5
6
7
8
9
10
11
12
13
lab@ubuntu:~/Desktop/HW3$ bash Q4.sh
Enter start line:
7
Enter end line:
3
Enter file name:
test.txt
Error: start number should be less than end number
lab@ubuntu:~/Desktop/HW3$ bash Q4.sh
Enter start line:
1
Enter end line:
3
Enter file name:
abc
File doesn't exist
```

۵-رسم شکل:

```
function shape_1(){  
  for i in {1..5}  
  do  
    for j in `seq $i`  
    do  
      echo -n $i  
    done  
    echo  
  done  
}
```

شکل ۱

```
function shape_2(){  
  for i in {1..6}  
  do  
    ((num_of_space = 6-$i))  
    for sapce in `seq $num_of_space`  
    do  
      echo -n " "  
    done  
  
    for j in `seq $i`  
    do  
      echo -n ". "  
    done  
    echo  
  done  
  
  for i in {6..1..-1}  
  do
```

شکل ۲

```
    ((num_of_space = 6-$i))  
    for sapce in `seq $num_of_space`  
    do  
      echo -n " "  
    done  
  
    for j in `seq $i`  
    do  
      echo -n ". "  
    done  
    echo  
  done  
}
```

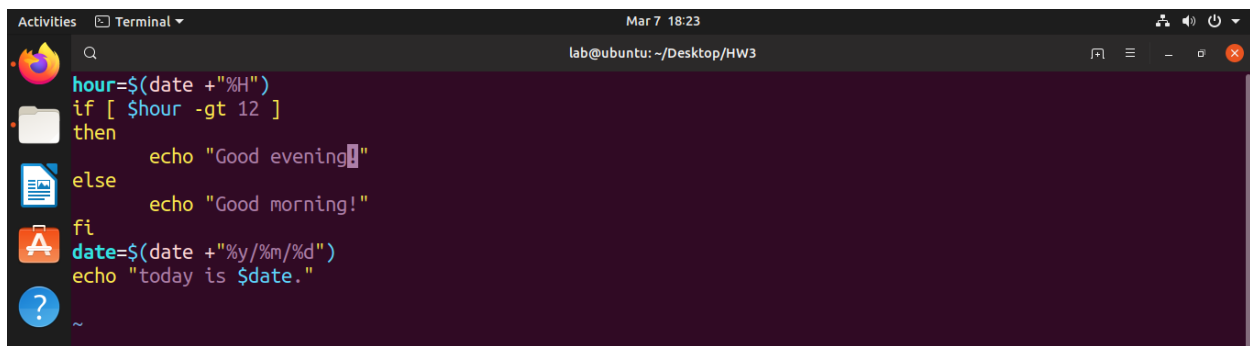
```
function shape_3(){  
  for i in {1..5}  
  do  
    ((num_of_line = $i-1))  
    for j in `seq $num_of_line`  
    do
```

شکل ۳

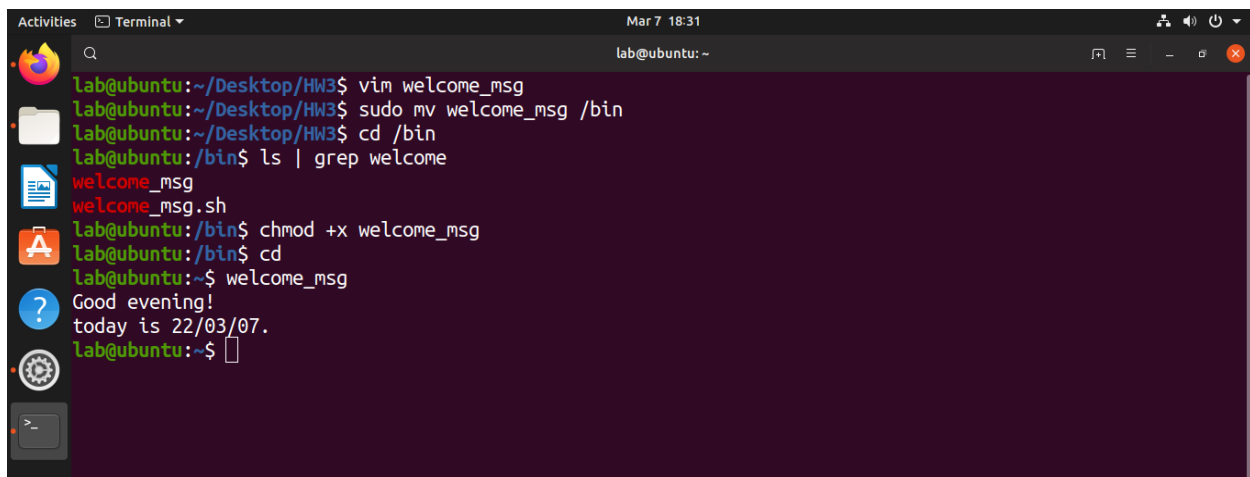
۶- اضافه کردن یک command دلخواه:

ابتدا یک فایل با نام کامند دلخواهمان ایجاد میکنیم و در آن دستوراتی که میخواهیم به کمک این کامند اجرا شوند را مینویسیم، در اینجا با توجه به ساعت فعلی پیام `good morning` یا `good evening` چاپ میشود و سپس تاریخ روز اعلام میشود.

سپس این فایل را (در اینجا `welcome_msg`) به پوشه `/bin` منتقل میکنیم و به کمک دستور `chmod +x` دسترسی `execute` به این فایل میدهیم. با انجام این فرآیند کاند جدید اضافه میشود و به کمک وارد `welcome_msg` در ترمینال این کاند اجرا می‌شود:

A terminal window titled 'Terminal' with the path 'lab@ubuntu: ~/Desktop/HW3'. It displays a shell script with the following content:

```
hour=$(date +%H)
if [ $hour -gt 12 ]
then
    echo "Good evening!"
else
    echo "Good morning!"
fi
date=$(date +%y/%m/%d)
echo "today is $date."
```

A terminal window titled 'Terminal' with the path 'lab@ubuntu: ~'. It shows the steps to create and execute the script:

```
lab@ubuntu:~/Desktop/HW3$ vim welcome_msg
lab@ubuntu:~/Desktop/HW3$ sudo mv welcome_msg /bin
lab@ubuntu:~/Desktop/HW3$ cd /bin
lab@ubuntu:/bin$ ls | grep welcome
welcome_msg
welcome_msg.sh
lab@ubuntu:/bin$ chmod +x welcome_msg
lab@ubuntu:/bin$ cd
lab@ubuntu:~$ welcome_msg
Good evening!
today is 22/03/07.
lab@ubuntu:~$
```