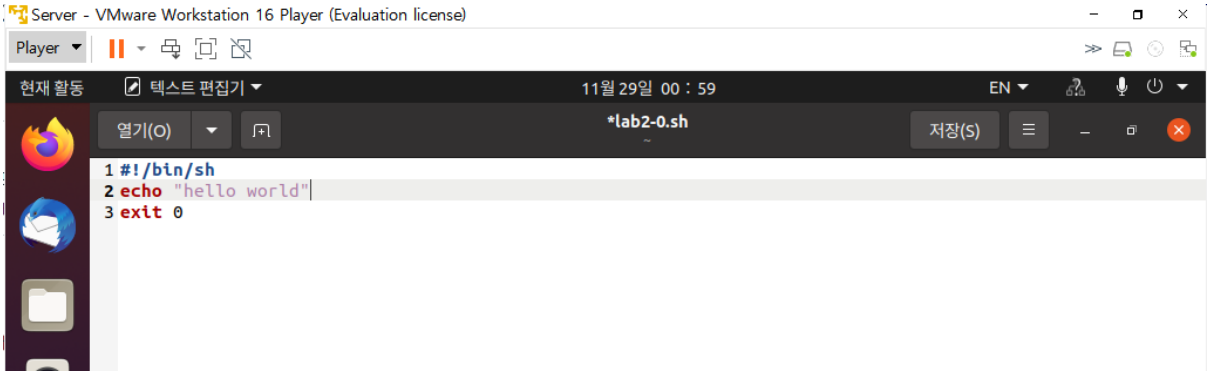

Lab2-0

Echo를 이용해서 hello world를 출력한다.

```
#!/bin/sh
$number=0
read number
time=$(seq 1 $number)
for i in $time
do
  echo "hellow world"
done
exit 0
```

Lab2-1

Read를 통해서 숫자를 입력 받으며 seq를 통해서 1~입력한 수로 time이라는 변수를 설정한다. 그 후 for문에서 i의 범위로 time을 사용하여 입력한 수만큼 hello world를 출력한다.

```
#!/bin/sh
read num1 v num2
if [ $v="-" ]
then
  result=`expr $num1 - $num2`
  echo $result
else
  result=`expr $num1 + $num2`
  echo $result
fi
exit 0
```

Lab2-2

Read를 통하여 2개의 수와 연산자를 받고 if문에서 -인지 +인지 확인한다. 연산자를 확인한 후 그에 맞은 값을 계산하여 출력한다.

```
#!/bin/sh
read weight hight
BMI=`expr $weight \* 1000000 / \( $hight \* $hight \)`
if [ $BMI -lt 1850 ]
then
  echo "저체중입니다."
elif [ $BMI -ge 1850 ] && [ $BMI -lt 2300 ]
then
  echo "정상체중입니다."
else
  echo "과체중입니다."

fi

exit 0


```

Lab2-3

Read를 통하여 몸무게와 키를 입력 받는다. BMI 계산에서 키는 m로 계산되므로 cm로 입력된 키의 값을 실수로 변환시키기 보다 몸무게에 10000만을 곱해준다. 또한 비교 값에 18.5라는 실수도 존재하므로 이를 해결하기 위하여 비교 값과 BMI 값에 100을 곱해준다. 그 후 if문에서 산술 비교 연산자를 이용하여 판단한다.

```
#!/bin/sh
echo "리눅스가 재미있나요? (yes / no)"
read answer
case $answer in
  yes | Y | y | Yes | YES)
    echo "yes";;
  [nN]*)
    echo "no";;
  *)
    echo "yes or no로 입력해 주세요."
    exit 1;;
esac
exit 0
```

Lab2-4

Read를 통하여 받은 값을 case문을 통하여 비교한다. |를 활용하여 여러가지 경우를 입력하고 그 중 하나라도 같다면 그에 대한 결과를 출력한다.

```
#!/bin/sh
funtion () {
  echo "함수 안으로 들어 왔음"
  str="ls"
  eval $str
}
echo "프로그램을 시작합니다."
funtion
echo "프로그램을 종료합니다."
exit 0

```

Lab2-5

funtion이라는 함수를 만들고 그 안에 리눅스 명령어 ls를 실행시키게 만든다. Eval은 문자열을 명령문으로 인식시키는 것이다. Function 함수를 호출하여 이를 실행시킨다.

```
#!/bin/sh
read dir
if [ ! -e $dir ]
then
  mkdir $dir
  goin="cd $dir"
  eval $goin
  for i in 0 1 2 3 4
  do
    str="gedit $dir$i.txt"
    eval $str
  done
  tar -cvf files.tar *
  tar -xvf files.tar
else
  mkdir $dir
  goin="cd $dir"
  eval $goin
  for i in 0 1 2 3 4
  do
    str="gedit $dir$i.txt"
    eval $str

  done
  tar -cvf files.tar *
  tar -xvf files.tar
fi
exit 0
```

Lab2-6

가장 먼저 입력 받은 이름의 파일 혹은 디렉토리가 존재하는지 확인한다. if문에서 만일 입력 받은 이름의 파일이나 폴더가 없다면 mkdir을 통하여 폴더를 생성하고 eval로 생성된 폴더에 들어간다. 그 후 for문을 사용하여 5개의 txt를 생성한다. for문이 끝나고 나면 tar를 이용해서 파일을 압축한다. 파일을 압축할 때 현재의 폴더 내에 있는 모든 것을 압축해야 하므로 *을 사용한다. 그 후 이를 풀어준다.

```
#!/bin/sh
read dir
mkdir $dir
eval "cd $dir"
for i in 0 1 2 3 4 5 6
do
  mkdir $dir$i
  eval "gedit $dir$i.txt"
  eval "ln -s $dir$i.txt $dir$i"
done
exit 0
```

Lab2-7

dir이라는 변수를 입력 받고 mkdir을 이용해서 변수와 같은 이름의 폴더를 만든다. 그 후 eval을 이용해 폴더로 들어간 후 for문을 활용해 txt와 폴더를 생성한다. 또한 ln -s를 이용하여 대응되는 txt와 폴더를 링크로 연결한다.

```
#!/bin/sh
read name number
eval "touch DB.txt"
eval "echo $name $number > DB.txt"

exit 0
```

Lab2-8

이름과 전화번호를 입력 받고 eval로 touch를 사용하여 DB.txt 파일을 만든다. 그 후 >를 사용하여 echo를 통해 결과로 나오는 이름과 전화번호를 DB.txt에 저장한다.

```
#!/bin/sh
read find

eval "grep -n "$find" DB.txt"

exit 0

```

Lab2-9

찾을 문자열을 입력 받고 이를 grep을 통하여 찾는다. 이때 grep -n을 이용하여 찾는다. 그 이유는 -n일 경우 입력 받은 문자열과 동일한 것이 있는 줄을 전부 출력하기 때문다.