20211728 김연수 Lab2 보고서

#lab2-0

```
1 #!/bin/sh
2 echo "Hello World"
3 exit 0
```

```
oss@ubuntu:~$ sh lab2-0.sh
Hello World
oss@ubuntu:~$
```

echo를 사용해 "Hello World" 문장을 출력해주었습니다.

#lab2-1

```
1 #!/bin/sh
2 hello=0
3 echo "출력 개수: "
4 read hello
5 i=1
6 while [ $i -le $hello ]
7 do
8   echo "Hello World"
9   i=`expr  $i  +  1`
10 done
11 exit 0
```

```
oss@ubuntu:~$ sh lab2-1.sh
출력 개수:
5
Hello World
Hello World
Hello World
Hello World
Hello World
oss@ubuntu:~$
```

read로 Hello World를 출력할 개수를 받아 변수 hello에 저장한 후 while문을 사용해 변수 hello개수만큼 Hello World를 출력해 주었습니다.

#lab2-2

```
#!/bin/sh

add=`expr $1 + $3`
sub=`expr $1 - $3`

if [ "$2" = "+" ]
then
  echo $add
elif [ "$2" = "-" ]
then
  echo $sub
else
  echo "덧셈, 뺄셈만 가능합니다."
fi

exit 0
```

```
oss@ubuntu:~/lab2$ sh lab2-2.sh 3 + 4
7
oss@ubuntu:~/lab2$ sh lab2-2.sh 9 - 3
6
oss@ubuntu:~/lab2$ sh lab2-2.sh 6 / 2
덧셈, 뺄셈만 가능합니다.
```

더하기와 빼기 공식을 만들고, if문으로 $2와 +, -를 문자열로 비교한 후 앞에서 만든 공식을 사용해 결과를 출력하도록 했습니다. 또한 else를 사용해 +, -가 아닌 문자를 입력 받았을 때를 대비했습니다.

#lab2-3

```
#!/bin/sh

low=18.5
big=23
echo "몸무게 입력"
read weight
echo "키 입력"
read height
height2=$( echo "scale=3; $height / 100 " |bc )
bmi=$( echo "scale=4; $weight / ( $height2 * $height2 )" |bc)
echo "BMI = $bmi"
big_then_low_bmi=$( echo "scale=3; $bmi >= $low" |bc )
small_than_big_bmi=$( echo "scale=3; $bmi <= $big" |bc )

if [ $big_then_low_bmi -eq 1 ]
then
  if [ $small_than_big_bmi -eq 1 ]
  then
    echo "정상 체중입니다."
  else
    echo "과체중입니다."
  fi
else
  echo "저체중입니다."
fi

exit 0
```

```
oss@ubuntu:~/lab2$ sh lab2-3.sh
몸무게 입력
5
키 입력
162
BMI = 1.9051
저체중입니다.
oss@ubuntu:~/lab2$ sh lab2-3.sh
몸무게 입력
55
키 입력
162
BMI = 20.9571
정상 체중입니다.
oss@ubuntu:~/lab2$ sh lab2-3.sh
몸무게 입력
555
키 입력
162
BMI = 211.4769
과체중입니다.
oss@ubuntu:~/lab2$
```

몸무게와 키를 각각 입력 받아 변수에 저장하고 키/100을 변수 height2에 저장한 뒤 공식을 사용해 bmi를 계산해 주었습니다. 이때 if [18.5 -le bmi] 이런 식으로 쓰면 오류가 나서 변수 big_then_low_bmi에 앞에서 구한 bmi가 18.5보다 큰지 계산하게 했습니다. 크면 1, 작으면 0을 반환하기 때문에 그 다음 if문에서 1과 같은지 비교한 후 체중 범위를 echo로 출력해 주었습니다. (small_than_big_bmi도 마찬가지)

#lab2-4

```
#!/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 1
```

```
oss@ubuntu:~$ sh lab2-4.sh
리눅스가 재미있나요? (yes/no)
no
no
oss@ubuntu:~$ sh lab2-4.sh
리눅스가 재미있나요? (yes/no)
Yes
yes
oss@ubuntu:~$ sh lab2-4.sh
리눅스가 재미있나요? (yes/no)
N
no
oss@ubuntu:~$ sh lab2-4.sh
리눅스가 재미있나요? (yes/no)
I don't know
yes or no로 입력해 주세요.
oss@ubuntu:~$
```

yes or no라는 대답을 유도한 후 case를 사용해 yes, Y, y, Yes, YES를 입력받은 경우는 yes를, n이나 N으로 시작하는 답을 받은 경우는 no를 출력하도록 했습니다. *)으로 나머지 모든 대답은 'yes or no로 입력해 주세요'라는 문장이 출력되도록 했습니다.

#lab2-5

```
#!/bin/sh
strrr="$1"
myfunc(){
  echo "함수 안으로 들어왔음"
  str="ls $strrr"
  eval $str
  return
}
echo "프로그램을 시작합니다."
myfunc
echo "프로그램을 종료합니다."
exit 0
```

```
oss@ubuntu:~/lab2$ sh lab2-5.sh
프로그램을 시작합니다.
함수 안으로 들어왔음
files  lab2-0.sh  lab2-1.sh  lab2-2.sh  lab2-3.sh  lab2-4.sh  lab2-5.sh  lab2-6.sh  lab2-7.sh  lab2-8.sh  lab2-9.sh
프로그램을 종료합니다.
oss@ubuntu:~/lab2$ sh lab2-5.sh -l
프로그램을 시작합니다.
함수 안으로 들어왔음
total 44
drwxrwxr-x 2 oss oss 4096 Nov 29 07:42 files
-rw-rw-r-- 1 oss oss   36 Nov 28 06:09 lab2-0.sh
-rw-rw-r-- 1 oss oss  137 Nov 28 06:59 lab2-1.sh
-rw-rw-r-- 1 oss oss   27 Nov 28 07:22 lab2-2.sh
-rw-rw-r-- 1 oss oss  447 Nov 29 23:34 lab2-3.sh
-rw-rw-r-- 1 oss oss  228 Nov 29 00:29 lab2-4.sh
-rw-rw-r-- 1 oss oss  208 Nov 30 00:04 lab2-5.sh
-rw-rw-r-- 1 oss oss   10 Nov 29 07:12 lab2-6.sh
-rw-rw-r-- 1 oss oss   10 Nov 29 07:12 lab2-7.sh
-rw-rw-r-- 1 oss oss   27 Nov 29 08:21 lab2-8.sh
-rw-rw-r-- 1 oss oss   10 Nov 29 07:13 lab2-9.sh
프로그램을 종료합니다.
oss@ubuntu:~/lab2$
```

먼저 변수 strrr에 첫번째 파라미터를 저장하고 함수 myfunc에서 변수 str에 'ls $strrr'을 저장했습니다. 그 후에 eval로 명령문을 출력했습니다.

파라미터를 받지 않은 경우는 명령문으로 ls만 입력받았기 때문에 첫번째와 같이 출력되었고, 파라미터로 -l을 입력받은 경우는 명령문으로 'ls -l'을 입력받은 것이기 때문에 두번째와 같이 출력되었습니다.

#lab2-6

```
#!/bin/sh

read name
if [ ! -d $name ]
then
  eval mkdir $name
  eval cd $name
  for i in 0 1 2 3 4
  do
    eval touch "file$i.txt"
  done
fi

eval mkdir $name
eval cd $name
for i in 0 1 2 3 4
do
  eval touch "file$i.txt"
done
eval tar -cvf $name.tar file0.txt file1.txt file2.txt file3.txt
  file4.txt
eval tar -xvf $name.tar

exit 0
```

```
oss@ubuntu:~/lab2$ sh lab2-6.sh
files
file0.txt
file1.txt
file2.txt
file3.txt
file4.txt
file0.txt
file1.txt
file2.txt
file3.txt
file4.txt
oss@ubuntu:~/lab2$ ls
DB.txt     files      lab2-1.sh  lab2-3.sh  lab2-5.sh  lab2-7.sh  lab2-9.sh
file0.txt  lab2-0.sh  lab2-2.sh  lab2-4.sh  lab2-6.sh  lab2-8.sh
oss@ubuntu:~/lab2$ cd files
oss@ubuntu:~/lab2/files$ ls
file0.txt  file1.txt  file2.txt  file3.txt  file4.txt  files
oss@ubuntu:~/lab2/files$ cd files
oss@ubuntu:~/lab2/files/files$ ls
file0.txt  file1.txt  file2.txt  file3.txt  file4.txt  files.tar
oss@ubuntu:~/lab2/files/files$
```

먼저 만들 파일의 이름을 입력받고, 'mkdir $name'을 사용해 파일을 생성한 후 'cd $name'으로 그 파일에 들어갔습니다. for문을 사용해 0부터 4까지의 file이라는 이름의 텍스트파일을 만들었습니다.

그런 다음 위와 같이 입력받은 이름의 파일을 만들고 또 들어갔습니다. 앞에서와 마찬가지로 for 문을 사용해 0부터 4까지의 file이라는 이름의 텍스트파일을 만들었고 'eval tar -cvf $name.tar file0.txt file1.txt file2.txt file3.txt file4.txt'를 사용해 file0.txt file1.txt file2.txt file3.txt file4.txt을 가지고 있는 $name.tar이라는 이름의 압축 파일을 만들었습니다. 또한 '-xvf $name.tar' 명령으로 압출 파일을 해제했습니다.

#lab2-7

```
#!/bin/sh

read name
if [ ! -d $name ]
then
  mkdir $name
  eval cd $name
  for i in 0 1 2 3 4
  do
    eval mkdir file$i
    eval touch file$i.txt
    ln -s file$i.txt ./file$i
  done
fi
exit 0
```

```
oss@ubuntu:~/lab2$ sh lab2-7.sh
files
oss@ubuntu:~/lab2$ cd files
oss@ubuntu:~/lab2/files$ ls
file0      file1      file2      file3      file4
file0.txt  file1.txt  file2.txt  file3.txt  file4.txt
oss@ubuntu:~/lab2/files$ cd file0
oss@ubuntu:~/lab2/files/file0$ ls
file0.txt
oss@ubuntu:~/lab2/files/file0$ ll
total 8
drwxrwxr-x 2 oss oss 4096 Nov 30 05:25 ./
drwxrwxr-x 7 oss oss 4096 Nov 30 05:25 ../
lrwxrwxrwx 1 oss oss    9 Nov 30 05:25 file0.txt -> file0.txt
oss@ubuntu:~/lab2/files/file0$
```

먼저 만들 파일의 이름을 입력받고 그 이름의 파일을 만들고 그 파일 안에 들어갔습니다. 그리고 for문을 사용해 file0부터 file4까지 파일들을 만들었고 file0.txt부터 file4.txt까지의 텍스트파일을 만들어주었습니다. 또한 'ls -s file$1.txt ./file$1'명령으로 각 폴더에 해당 파일을 링크해주었습니다. (예를 들어 file0이면 그 안에 file0.txt 넣어주기)

#lab2-8

```
#!/bin/sh

info(){
  str="touch DB.txt"
  eval $str
}
info
echo $1 $2 >> DB.txt
exit 0
```

```
oss@ubuntu:~/lab2$ sh lab2-8.sh yeeonsoo 01011112222
oss@ubuntu:~/lab2$ sh lab2-8.sh minji 01022223333
oss@ubuntu:~/lab2$ sh lab2-8.sh sangmin 01088889999
oss@ubuntu:~/lab2$ cat DB.txt
yeeonsoo 01011112222
minji 01022223333
sangmin 01088889999
oss@ubuntu:~/lab2$ echo "* My friends" > DB.txt
oss@ubuntu:~/lab2$ cat DB.txt
* My friends
oss@ubuntu:~/lab2$ sh lab2-8.sh Kim 01055777755
oss@ubuntu:~/lab2$ cat DB.txt
* My friends
Kim 01055777755
oss@ubuntu:~/lab2$
```

먼저 info함수로 DB.txt 파일을 만들었습니다(touch). 또한 'echo $1 $2 >> DB.txt'를 사용해 DB.txt에 입력받은 첫번째, 두번째 파라미터를 저장시켰습니다.

#lab2-9

```
#!/bin/sh
KeyWord=$1
search(){
  strr="grep $KeyWord DB.txt"
  eval $strr
}
search
exit 0
```

```
oss@ubuntu:~/lab2$ sh lab2-9.sh K
Kim 01055777755
oss@ubuntu:~/lab2$ sh lab2-9.sh Kim
Kim 01055777755
oss@ubuntu:~/lab2$ sh lab2-9.sh k
oss@ubuntu:~/lab2$ sh lab2-9.sh Lee
oss@ubuntu:~/lab2$
```

DB.txt에 있는 연락처인지 확인하기 위한 키워드로 첫번째 파라미터를 변수 KeyWord에 저장시켰습니다. Search 함수를 사용해 $KeyWord 를 포함하는 정보가 DB.txt에 있는지 확인한 후 있다면 해당 라인을 출력하도록 했습니다.