일일 업무 사항 정리

작성자	제품팀 이민성 인턴
업무 일시	2022.08.23

세부 사항

1. 업무 내역 요약 정리

목표 내역	Done & Plan	To-do
1. 리눅스 환경에 대해 이해 2. 리눅스 명령어 숙달 3. 계정 및 디렉토리 관리 공부 4. 쉘 스크립트 언어 이해 및 활용 5. 오라클 데이터베이스 설치과정 복기	오늘은 쉘 스크립트 언어에 대해 더 알아보고 활용하며 공부하였습니다. vscode나 eclipse와 같은 에디터가 아니라 리눅스 환경에서 하다 보니 자동완성도 안되고 오타 하나하나 스스로 수정해야 해서 불편함이 많았습니다. 그리고 기존에 알던 언어와는 다르게 까다로워서 조심해야 되는게 많아서 조금 헷갈리고 어려웠습니다. 예제는 따라해보기는 했지만 이해가 잘 가지는 않습니다. 조금 더 공부해보고 보면 좋을 것 같습니다. 쉘 스크립트 언어를 적응하는데 시간이 꽤 걸릴 것으로 예상됩니다. 그리고 virtual box를 통해 리눅스를 설치하였습니다. 하지만 xshell을 연결할 때 계속 오류가 났는데 원인을 찾지 못하였습니다. 다시 알아보고 고치겠습니다. 이후 shell 언어 연습을 했습니다. 내일은 shell 언어를 더 활용하여 습득하고, 가상머신 설치를 다시 시도해보겠습니다.	- 리눅스 환경에 대한 이해 -- 계정 및 디렉토리 관리 -- 기본 명령어 - 쉘 스크립트 언어에 대한 이해 -- 쉘과 쉘 스크립트 언어 정의 -- 기본 문법 - 설치 -- 과정 복기

2. 내용 세부 (업무 세부 내역 정리 및 기타 사항 정리)

리스트 변수 (배열)

- 선언 : 변수명=(데이터 1 데이터 2 데이터 3)
- 사용 : ${변수명[인덱스번호]} : 인덱스 번호는 0 이 시작이다.
- 코드 예시

```
#!/bin/bash

a=("a" "b" "c") # 변수 a, b, c 선언

echo ${a[1]} # $a 배열의 두 번째 인덱스에 해당하는 b 출력
echo ${a[@]} # $a 배열의 모든 데이터 출력
echo ${a[*]} # $a 배열의 모든 데이터 출력
echo ${#a[@]} # $a 배열의 배열 크기 출력

filelist=($(ls)) # 해당 쉘 스크립트 실행 디렉토리의 파일 리스트를를 배열로 변수 선언
echo ${filelist[*]} # $filelist 모든 데이터 출력
```

```
b
a b c
a b c
3
array.sh hello.sh id.sh tt.sh variable.sh
```

- 예제 3 : 아이디 관련 정보 리스트 변수로 만들고, 각 정보를 출력해보자 (이름, 나이, 직업)

```
#!/bin/bash

id=("leeminsung" 20 "student")
echo ${id[*]} # $id의 모든 데이터 출력
```

```
leeminsung 20 student
```

쉘 스크립트의 기본 문법은 까다롭다. 가변적으로 문법을 바꿔도 동작을 하긴 하지만 환경에 따라 동작을 달리한다.

- 사전에 정의된 지역 변수

$$: 쉘의 프로세스 번호 (pid, 프로세스 아이디)
$0 : 쉘 스크립트 이름
$1~$9 : 명령줄 인수
$* : 모든 명령줄 인수 리스트
$# : 인수의 개수
$? : 최근 실행한 명령어의 종료 값
0 성공, 1 ~ 125 에러, 126 파일이 실행 가능하지 않음, 128 ~ 255 시그널 발생

ls -al -z의 경우
ls가 $$, -al이 $1 , -z가 $2, $#은 2 개

- 예제 4 : 쉘 프로세스 번호, 쉘 스크립트 이름, 명령줄 인수, 모든 명령줄 인수 리스트, 인수 개수 출력해보기

```
#!/bin/bash

echo $$ $0 $1 $* $#
```

```
[testuser@lms sh]$ sh shell.sh
9433 shell.sh 0
[testuser@lms sh]$ sh shell.sh kkk
9462 shell.sh kkk kkk 1
[testuser@lms sh]$ sh shell.sh aaa bbb ccc ddd
9489 shell.sh aaa aaa bbb ccc ddd 4
```

연산자

- expr : 숫자 계산
  1. expr를 사용하는 경우 역 작은 따옴표(')를 사용해야한다.
  2. 연산자 *와 괄호 ( ) 앞에는 역 슬래시를 같이 사용해야한다. ~~wwwwww~~(붙여 쓴다)
  3. 연산자와 숫자, 변수, 기호 사이에는 space를 넣어야한다.

- 예제 5 : expr 명령으로 (30 / 5) - 8 계산해보기

```
#!/bin/bash

num=`expr \( 30 / 5 \) - 8`

echo $num
```

조건문

- 기본 if 문법 : then과 fi 안에만 들어가 있으면 되기 때문에 명령문을 꼭 탭으로 띄워야 하는 것은 아니다.

```
if [조건]
then
    명령문
fi
```

- 예제 6 : 두 인자값을 받아서 두 인자값이 다르면 'different values'를 출력해보자

```
#!/bin/bash

if [$1 != $2] # != 둘 이  다 르 면  이 라 는  의 미
then
   echo "different values"
   exit
fi
```

```
[testuser@lms sh]$ sh if.sh
different values
```

- 조건
  문자 1 == 문자 2 : 문자 1 과 문자 2 가 일치
  문자 1 != 문자 2 : 문자 1 과 문자 2 가 일치 X
  -z 문자 : 문자가 null이면 참 (값이 없으면 true)
  -n 문자 : 문자가 null이 아니면 참

- 수치비교 : <, >는 if 조건시 [[]]를 넣는 경우 정상 작동하기는 하지만, 기본적으로 다음 문법을 사용한다.
  값 1 -eq 값 2 : 값이 같음 (equal)
  값 1 -ne 값 2 : 값이 같지 않음 (not equal)
  값 1 -lt 값 2 : 값 1 이 값 2 보다 작음 (less than)
  값 1 -le 값 2 : 값 1 이 값 2 보다 작거나 같음 (less or equal)
  값 1 -gt 값 2 : 값 1 이 값 2 보다 큼 (greater than)
  값 1 -ge 값 2 : 값 1 이 값 2 보다 크거나 같음 (greater or equal)

- 파일 검사
  -e 파일명 : 파일이 존재하면 참
  -d 파일명 : 파일이 디렉토리면 참
  -h 파일명 : 파일이 심볼릭 링크 파일이면 참
  -f 파일명 : 파일이 일반파일이면 참
  -r 파일명 : 파일이 읽기 가능이면 참
  -s 파일명 : 파일크기가 0 이 아니면 참
  -u 파일명 : 파일이 set-user-id가 설정되면 참
  -w 파일명 : 파일이 쓰기 가능이면 참
  -x 파일명 : 파일이 실행 가능이면 참

- 예제 7 : 쉘 스크립트로 해당 파일이 있는지 없는지 확인해보자

```
#!/bin/bash

if [ -e $1 ]
then
  echo "file exist"
fi
```

```
[testuser@lms sh]$ sh if2.sh cal.sh
file exist
```

- 논리 연산
  조건 1 -a 조건 2 //AND
  조건 1 -o 조건 2 //or
  조건 1 && 조건 2 //양쪽 다 성립
  조건 1 || 조건 2 //한쪽 또는 양쪽 다 성립
  !조건 //조건이 성립하지 않음
  true //조건이 언제나 성립
  false //조건이 언제나 성립하지 않음

- 기본 if/else 구문

  if [조건]

  then

  명령문

  else

  명령문

  fi

- 예제 8 : 두 인자 값을 받아서 두 인자 값이 같으면 'same value', 다르면 'different value'가 출력되도록 해보자

```
#!/bin/bash

if [ $1 -eq $2 ]
then
   echo "same values"
else
   echo "different values"
fi
```

```
[testuser@lms sh]$ sh if3.sh 3 3
same values
```

- ping : 서버에서 여러 가지 컴퓨터가 연결되어 있을 때 연결된 특정 컴퓨터가 정상적으로 동작하는 지, 꺼져 있는지, 비정상적으로 동작하는지 확인하는 명령어이다. 해당 컴퓨터의 ip주소로 ping명령어를 실행한다. 그 주소에 확인 요청을 한다. 정상적인 컴퓨터는 응답을 한다. 응답이 오지 않으면 정상적으로 동작하지 않는다고 판단한다.

  ping -c 1 192.168.0.105 1> /dev/null

  0 : 표준 입력, 1 : 표준 출력, 2 : 표준 에러

  1>/dev/null : 표준 출력 내용은 버려라 (출력하지 말아라)

  -c 1 : 1 번만 체크해보라는 의미

- 코드 예제

```
#!/bin/bash

ping -c 1 192.168.0.105 1> /dev/null
if [ $? == 0 ] # $? : 가장 최근에 쉘 스크립트에서 실행한 명령의 결과값
then
   echo "Gateway ping success!" # 0일 경우 응답이 온 것이라 성공!
else
   echo "Gateway ping failed!" # 응답이 없을 때 나타남
fi
```

```
[testuser@lms sh]$ sh ping.sh
Gateway ping success!
```

- 조건문 한 줄에 작성하기 (if 구문)

  if [조건]; then 명령문; fi

  if [-z $1]; then echo 'Insert arguments'; fi

  if [뒤와] 앞에는 반드시 공백이 있어야함 [ ]에서 &&, ||, <, > 연산자들이 에러가 나는 경우는 [[ ]]를 사용하면 정상 작동하는 경우가 있음

반복문

- 기본 for 구분

  for 변수 in 변수값 1 변수값 2 ....

  do

  명령문

  done

- 예제 8 : 현재 디렉토리에 있는 파일과 디렉토리를 출력해보자

```
for database in $(ls);
do
   echo $database
done
```

```
for database in $(ls); do
   echo $database
done
```

```
for database in $(ls); do echo $database; done
```

```
lists=$(ls)
num=${#lists[@]}
index=0
while [ $num -ge 0 ]
do
   echo ${lists[$index]}
   index=` expr $index + 1 `
   num=` expr $num - 1 `
done
```

- 기본 while 구문

  while [조건문]

  do

  명령문

  done

쉘 스크립트 예제

1. 백업하기

```
#!/bin/bash

if [ -z $1 ] || [ -z $2 ] # || :  or, 두 가지 하나라도 없으면
then
   echo usage : $0 sourcedir targetidir
else
   SRCDIR=$1
   DSTDIR=$2
   BACKUPFILE=backup.$(date +%y%m%d%H%M%S).tar.gz
   if [ -d $DSTDIR ]
   then
      tar -cvzf $DSTDIR/$BACKUPFILE $SRCDIR
   else
      mkdir $DSTDIR
      tar -cvzf $DSTDIR/$BACKUPFILE $SRCDIR
   fi
fi
```

코드에서 || 는 or이라는 뜻으로 두 가지 중 하나라도 없으면 이라는 의미이다.

BACKUPFILE=backup.$(date +%y%m%d%H%M%S).tar.gz 이 문장은 백업할 파일이름을 back~gz로 하겠다는 뜻이다.

코드를 풀어쓰자면

2 개의 인자를 받아서 두 가지 중 하나라도 없으면

$0 : 쉘 이름

sourcedir : 압축할 디렉토리명

targetidr : 압축된 파일을 넣을 디렉토리명

백업할 파일이름을 back~gz로 해주는데

date라는 쉘 명령어를 사용해서 년, 월, 일, 시간, 분, 초를 알 수 있게 해준다. (backup.현재시각.tar.gz)

mkdir : 디렉토리 생성하는 명령어

tar : 압축 명령 (확장자) 묶었다

gz : 압축까지 되었다는 의미

- tar : 압축 명령

주요옵션 :

x 묶음을 해체

c 파일을 묶음

v 묶음, 해제 과정을 화면에 표시

z gunzip을 사용

f 파일 이름을 지정

압축 시 주로 사용하는 옵션

tar -cvzf [압축된 파일 이름] [압축할 파일이나 폴더명]

압축을 풀 때 주로 사용하는 옵션

tar -xvzf [압축 해제할 압축 아카이브 이름]