1. 쉘 명령어

1) 쉘 명령어 관련 사항

1 쉘(Shell)

- 커널과 사용자간의 인터페이스로서 **중간매개의 역할**을 담당
- 명령어를 해석하고 명령과 관련된 유틸리티나 커널을 호출하고 실행결과를 출력함
- Dos의 command.com과 같이 **기본 제공 OS명령어를 수행하는 곳**
- 윈도우에서 탐색기의 왼쪽 마우스 메뉴의 압축 기능과 같은 기본 명령임
- 최초 유닉스에서는 ksh(콘쉘), 이후 발전된 csh(씨쉘) 등이 사용되었으나, 리눅스에서는 bash(배쉘)이 많이 사용됨
- bash 명령어의 문법은 거의 대부분의 **sh와 호환되어 쓰임**

1) 쉘 명령어 관련 사항

2 쉘 스크립트, 쉘 프로그램

- 하나의 쉘 명령어를 **여러 개 묶어서 실행 가능**
- 쉘 명령어를 나열하여 text file로 기록하여 이를 실행함
- 이러한 **묶음 명령어**를 쉘 스크립트 또는 쉘 프로그램이라고 함

3 쉘 스크립트의 장점

- 타 프로그래밍 언어에 비해 **실행속도가 빠름**
- 컴파일 과정이 필요 없이 빠르게 실행가능
- 시스템 운영이나 유지 · 보수 때 사용하기 용이함

4 쉘 스크립트의 제한 사항

- 다중 산술 작업이나 복잡한 정보 처리작업에 사용하는 방식은 아님
- 유닉스, 리눅스 이외 다른 운영체계에 이식은 어려움

참고

리눅스 쉘 프로그래밍을 잘 하기 위해서는 무엇보다도 많이 사용하는 쉘 명령어를 잘 다룰 수 있어야 한다.

2) 쉘 스크립트 실행

명령어를 묶어서 실행

- 여러 명령어를 한번에 실행하기 위하여 ;(세미콜론)을 사용
- home 디렉토리로 이동한 후 ls명령어를 실행
- pwd;cd /home;ls -l;pwd

2) 쉘 스크립트 실행

명령어를 묶어서 실행

```
sjcu@ubuntu:~$ pwd;cd /home;ls -l;pwd
/home/sjcu
total 72
-rw-r--r-- 1 root root 51810 Dec 19 18:21 aa
-rw-r--r-- 1 root root 476 Dec 20 20:48 a_out.log
-rwxr-xr-x 1 root root 61 Dec 20 19:32 a.sh
-rw-r--r-- 1 root root 232 Jan 17 12:25 boot.log
drwxr-xr-x 3 sjcu sjcu 4096 Jan 16 01:03 sjcu
-rw-r--r-- 1 root root 16 Jan 12 22:21 mytest.log
/home
sjcu@ubuntu:/home$
```

〈출처 : 교수자 제작물〉

2) 쉘 스크립트 실행

쉘 스크립트 파일을 작성 후, 실행

- Vi 등 편집기로 실행할 명령어를 한 줄 씩 작성 후 저장
- 해당 파일명으로 명령어 실행
- 쉘 스크립트 파일을 실행하기 위하여 쉘 파일의 권한과 패스를 고려하여야 하기 때문에 실행 안 됨

2) 쉘 스크립트 실행

쉘 스크립트 파일을 작성 후, 실행

```
sjcu@ubuntu:~$ vi my.sh
cd /home
ls -l
pwd
~
:wq
```

〈출처 : 교수자 제작물〉

```
sjcu@ubuntu:~$ pwd
/home/sjcu
sjcu@ubuntu:~$ vi my.sh
sjcu@ubuntu:~$ my.sh
my.sh: command not found
sjcu@ubuntu:~$
```

〈출처 : 교수자 제작물〉

2) 쉘 스크립트 실행

권한

- ✔ 쉘 스크립트 파일이 실행권한을 가져야 함
  - 예 chmod 755 mysh.sh
- ✔ 또는 sh shell_script_file 형식으로 실행
  - 예 sh mysh.sh

2) 쉘 스크립트 실행

패스

- 쉘 스크립트 파일 내 모든 명령은 절대패스로 명령어 기술(full path)

- 쉘 스크립트 파일보다 일반 명령어 실행이 우선되므로 쉘 스크립트 파일도 패스를 지정해야 함

  예 ./mysh.sh

- 쉘 스크립트 파일에 환경변수를 인지하도록 하는 방법

  예
  - export PATH="/home/s1111111/"
  - #!/bin/sh

3) 자주 쓰는 쉘 프로그래밍 기법

리다이렉션(〉 ,redirection) , 파이프(|, pipe)

명령1 | 명령2	(파이프) 어떤 명령의 결과를 받아 다른 명령을 실행
명령 〉 filename	어떤 명령의 결과를 파일을 새로 생성하여 기록
명령 〉〉 filename	어떤 명령의 결과를 파일 뒤로 계속 첨부기록

예

- cat xinetd.conf 〉 a.file
- cat xinetd.conf 〉〉 a.file
- cat xinetd.conf\|grep telnet 〉 a.file

3) 자주 쓰는 쉘 프로그래밍 기법

리다이렉션(〉,redirection) , 파이프(|, pipe)

명령1 | 명령2	(파이프) 어떤 명령의 결과를 받아 다른 명령을 실행
명령 〉 filename	어떤 명령의 결과를 파일을 새로 생성하여 기록
명령 〉〉 filename	어떤 명령의 결과를 파일 뒤로 계속 첨부기록

예

- ps 〉 ps.out , sort ps.out 〉 ps.sort
- ps \|sort 〉 ps.sort
- ps \| sort 〉 ps.sort , more ps.sort
- ps \| sort \| more

3) 자주 쓰는 쉘 프로그래밍 기법

출력의 echo, 기본 시스템 변수

- ✔ echo $PATH
- ✔ echo “Hello sjcu”

3) 자주 쓰는 쉘 프로그래밍 기법

cut

✔ 텍스트 파일이나 명령어의 결과 중 필요한 부분을 간단히 추출할 때, cut 명령어를 사용

예

옵션 -b, -c 사용사례

- b 옵션은 바이트(byte), -c는 글자(character)를 의미
- 한글은 2바이트가 1글자로 취급되는 것을 주의

3) 자주 쓰는 쉘 프로그래밍 기법

cut

✔ 텍스트 파일이나 명령어의 결과 중 필요한 부분을 간단히 추출할 때, cut 명령어를 사용

형태	설명	사용 예
NN	자릿수만을 의미	cut -b3
N,M,O	N자리와 M자리, O자리를 각각 의미	cut -5,6,7,8,9
N-	N자리부터 마지막까지를 의미	cut -b10-
N-M	N자리부터 M자리까지를 의미	cut -b5-10
-M	처음부터 M자리까지를 의미	cut -b-10

3) 자주 쓰는 쉘 프로그래밍 기법

cut

텍스트 파일이나 명령어의 결과 중 필요한 부분을 간단히 추출할 때,
cut 명령어를 사용

```
sjcu@ubuntu:~$ cat testfile
1234567890abcdefghijk
sjcu@ubuntu:~$ cat testfile|cut -b3
3
sjcu@ubuntu:~$ cat testfile|cut -b5,6,7,8,9
56789
sjcu@ubuntu:~$ cat testfile|cut -b10-
0abcdefghijk
sjcu@ubuntu:~$ cat testfile|cut -b5-10
567890
sjcu@ubuntu:~$ cat testfile|cut -b-10
1234567890
sjcu@ubuntu:~$
```

〈출처 : 교수자 제작물〉

2. 변수 조건문

1) 변수, 매개변수

변수

쉘 프로그램 내에서 간단한 데이터 저장소, 문자형태 변수 등을 저장, 수치계산은 없음

✔ [변수명=“값”] 으로 입력하고 [echo $변수명]으로 사용

1) 변수, 매개변수

✔ 예제 파일로 설명 : test8-1.sh 파일 (변수 실습)

```
#!/bin/sh
str="hello I am variable"
echo $str
echo "$str"
echo '$str'
echo \$str
echo "input text"
read str
echo 'str' is $str
exit 0
```

〈출처 : 교수자 제작물〉

1) 변수, 매개변수

매개변수

실행 명령어 뒤에 공백으로 구분하여 주어지는 값

예 [cp afile bfile]이라고 명령한다면, cp는 명령어 afile은 첫 번째 매개변수, bfile은 두 번째 매개변수임

1) 변수, 매개변수

- ✔ 특별한 매개변수 표현도 있음 [$HOME, $PATH]
- ✔ 예제 파일로 설명 : test8-2.sh 파일

```
#!/bin/sh
echo “shell name $0 ”
echo “parameter 1  $1 ”
echo “parameter 2  $2 ”
echo “parameter 3  $3 ”
echo “home directory  $HOME ”
echo “path  $PATH ”
exit 0
```

〈출처 : 교수자 제작물〉

2) 조건문

if/elif/fi

if문 다음의 나오는 내용의 상태에 따라 elif와 fi사이에 쉘 내용을 수행함

✔ 예제 파일로 설명 : test8-3.sh 파일 (if/elif/fi 실습)

```
#!/bin/sh
echo ‘What is your name?’
read yourname
If [ $yourname = “Tom” ]
then
    echo ‘You are Tom.”
else
   echo ‘ You are not Tom.’
fi
exit 0
```

〈출처 : 교수자 제작물〉

3. 반복문, 조건의 AND-OR

1) 반복문

for in(조건)/do/…/done

조건 내 상황을 하나씩 적용하여 do와 done사이의 스크립트를 실행

✔ 예제 파일로 설명 : test8-4.sh , test8-5.sh 파일

```
#!/bin/sh
for x in 1 2 3 4 5
do
echo $x
done
exit 0
```

```
#!/bin/sh
for file in $(ls *.txt)
do
echo $file
done
exit 0
```

〈출처 : 교수자 제작물〉

1) 반복문

while(조건)/do/…/done

조건 내 상황이 만족되면 계속하여 do와 done사이의 스크립트를 실행하며,
그 외 조건이 되는 경우 반복구문에서 빠져 나옴

✔ 예제 파일로 설명 : test8-6.sh 파일

```
#!/bin/sh
echo -n “Input Password :”
read pawd
while [$pawd != “1234”]
do
        echo -n “Input Password :”
        read pawd
done
echo “OK “
exit 0
```

〈출처 : 교수자 제작물〉

1) 반복문

until(조건)/do/…/done

조건 내 상황이 만족하지 않으면 계속하여 do와 done사이의 스크립트를 실행하며
그 외 조건이 만족되는 경우 반복구문에서 빠져 나옴

✔ 예제 파일로 설명 : test8-7.sh ,test8-8.sh 파일

```
 #!/bin/sh
pawd="0000"
until [$pawd = "1234"]
do
        echo -n "Input Password :"
        read pawd
done
echo "OK "
exit 0
```

```
 #!/bin/sh
loop=10
until ["$loop" -eq 0]
do
       echo "loop $loop"
       loop=$(($loop-1))
done
exit 0
```

〈출처 : 교수자 제작물〉

1) 반복문

break

do/done의 루프 내에서 빠져 나올 경우 사용

✔ 예제 파일로 설명 : test8-9.sh 파일

```
 #!/bin/sh
for var in 1 2 3 4 5 6 7 8 9 10 ; do
    echo “var $var”
    if [ $var = “5” ]; then
      echo “break”
      break;
    fi
done
exit 0
```

〈출처 : 교수자 제작물〉

2) 조건의 AND, OR

“조건을 여러 개 나열할 때,
불리안(boolean)식의 AND(&&)와 OR(||)를 표현”

✔ 예제 파일로 설명 : test8-10.sh,test8-11.sh 파일

```
#!/bin/sh
touch file1
rm -f file2
 if [ -f file1] && echo “next1” && [ -f file2] &&
echo “next2”
#참인 경우 다음문장 수행
then
      echo “if-true”
 else
      echo “if-false”
 fi
exit 0
```

```
#!/bin/sh
touch file1
 if [ -f file1] || echo “next1” || [ -f file2] ||
echo “next2”
#참인 경우 다음문장 수행
then
      echo “if-true”
 else
      echo “if-false”
 fi
exit 0
```

〈출처 : 교수자 제작물〉

4. 쉘 명령어, 변수 조건문 실습

1) 실습하기

실습내용

(1) 쉘 명령어
(2) 변수 조건문

5. 반복문, 조건의 AND-OR 실습

1) 실습하기

실습내용

(1) 반복문, 조건의 AND-OR

학습활동

* 일시정지 버튼을 클릭하고 학습활동에 참여해 보세요.

Q 오늘은 리눅스에서 간단한 쉘 프로그래밍을 배웠습니다.
배운 내용을 응용하여 간단한 수치연산을 계산하는 방법을 만들어 봅시다.

※ 학습활동에 대한 해설

Q 오늘은 리눅스에서 간단한 쉘 프로그래밍을 배웠습니다.
배운 내용을 응용하여 간단한 수치연산을 계산하는 방법을 만들어 봅시다.

A

- 리눅스 쉘에서 수치연산을 하기 위하여 expr이라는 명령을 사용합니다.
- 쉘을 다음과 같이 작성합니다.

```
a=`expr 4 + 5`
echo $a
```

- 즉, expr이라는 명령을 실행하여 결과를 a에 집어넣고 출력하는 것입니다.
- 이때 expr을 감싸는 ` 기호는 키보드 숫자 1왼쪽에 있는 따옴표 같은 기호입니다.

학습활동에 대한 해설

Q 오늘은 리눅스에서 간단한 쉘 프로그래밍을 배웠습니다.
배운 내용을 응용하여 간단한 수치연산을 계산하는 방법을 만들어 봅시다.

A

- 이때 4 + 5 사이도 칸을 띄어 줘야 계산이 됩니다.
- expr을 이용하여 연산을 쉘 프로그램에서 수행할 수 있습니다.