

Shell Script 2

ABC Program 5조

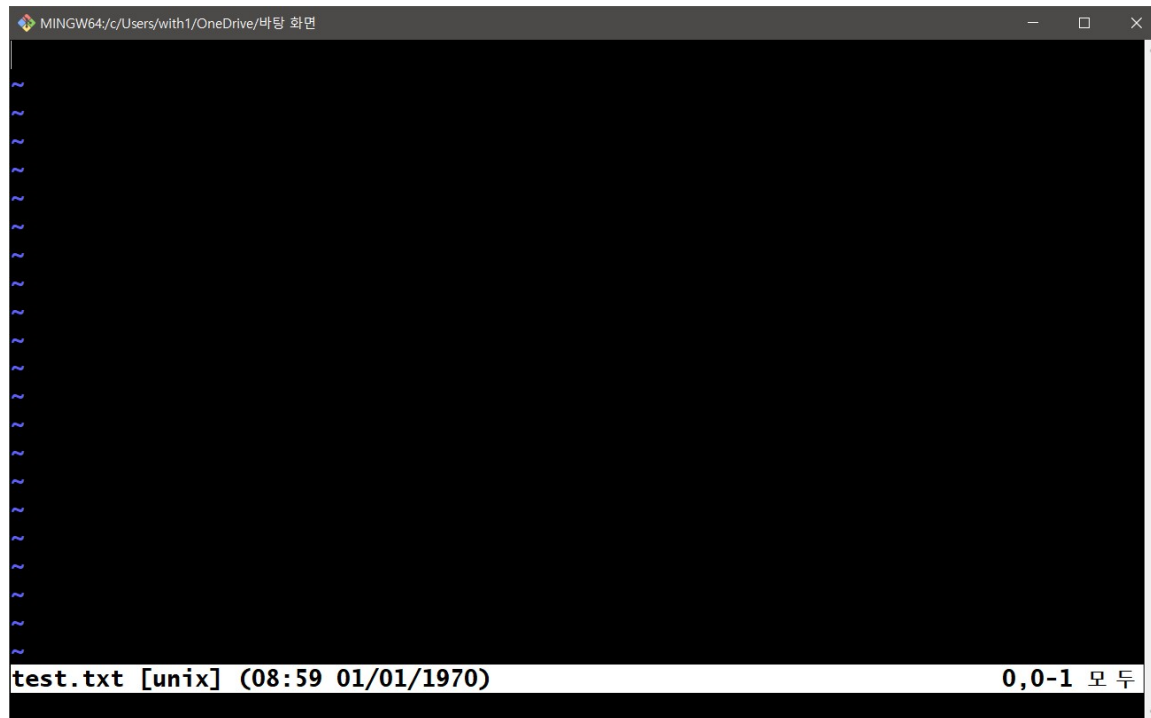
Mentor 정현준

2021/01/15



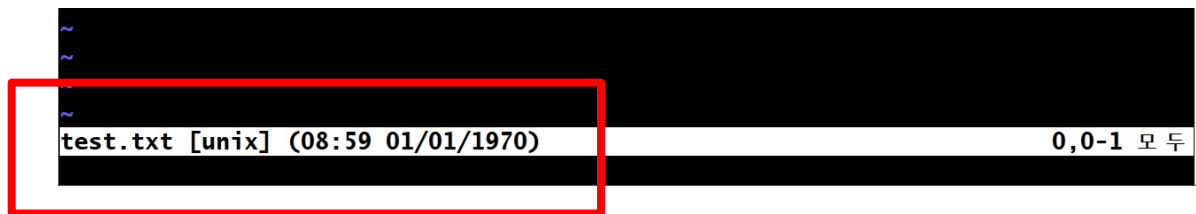
Vim basic

- vi [존재하는 파일 이름] 또는 vi [생성할 파일 이름]



Vim basic

Command 모드



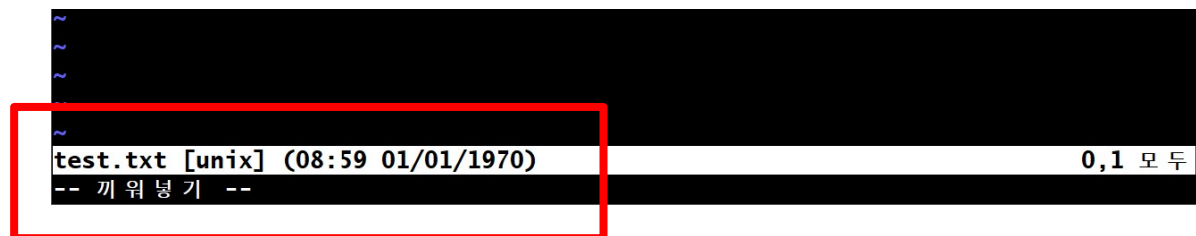
i 입력



Esc 입력

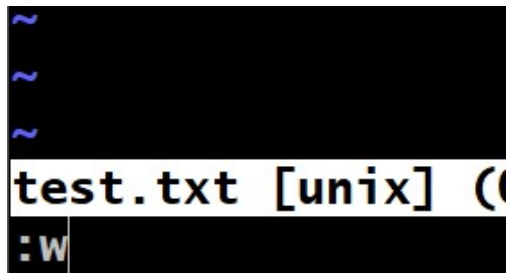


입력 모드



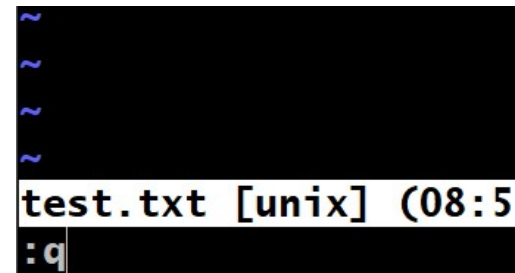
Vim basic

- Command 모드에서 저장/닫기/저장 후 닫기 방법



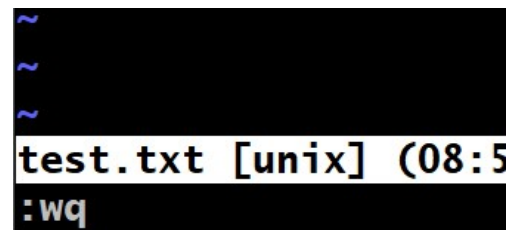
A terminal window showing the Vim editor interface. The status line at the bottom reads "test.txt [unix] (08:50)". The command line shows the command `:w` being entered.

저장



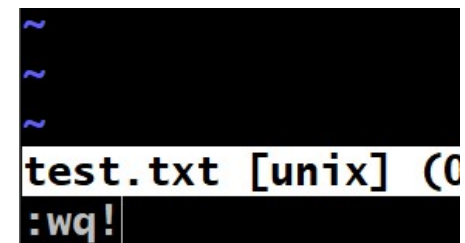
A terminal window showing the Vim editor interface. The status line at the bottom reads "test.txt [unix] (08:50)". The command line shows the command `:q` being entered.

닫기 (저장 x)



A terminal window showing the Vim editor interface. The status line at the bottom reads "test.txt [unix] (08:50)". The command line shows the command `:wq` being entered.

저장 후 닫기



A terminal window showing the Vim editor interface. The status line at the bottom reads "test.txt [unix] (08:50)". The command line shows the command `:wq!` being entered.

명령에 ! 붙이면 강제로 명령 실행

Bash Script

- 파일 상단에 `#!/bin/bash`로 시작합니다.
- Bash shell을 사용한다는 것을 의미합니다.
- .sh 확장자를 사용합니다 (ex: file.sh)
- 실행 할 때는 `source [sh 파일]`을 입력하여 사용합니다.

Variable

- Bash script의 변수는 type이 따로 존재하지 않습니다.
- [변수 이름]=[데이터] 로 선언합니다.
- = 사이에 공백이 반드시 존재해선 안됩니다.
- [변수 이름]=[명령어] 를 사용하면 명령어의 실행 결과가 변수에 들어가게 됩니다.
- 변수의 참조를 위해서는 \${변수 이름}를 사용합니다.

Arithmetic operation

- 괄호 2개 안에서 연산을 합니다.
- $\text{Val1}=\$(1+2*3)$
- $((\text{Val1}++))$
- $\text{Val2}=\$(\{\text{Val1}\}/2)$
- / 연산자는 소수점 나눗셈을 할 수 없습니다.
- bc 명령어와 pipeline을 이용하면 소수점 연산이 가능합니다.
- `echo "55/3" | bc -l`

Array

- `<array 이름>=(값1 값2 값3 ... 값N)`
- Ex) `array=("hello" 5 123 "ABC")`
- 각 element 사이에 **space**를 통해 구분합니다.
- `${array[index]}`를 통해 특정 index의 값 참조가 가능합니다.
- `${array[@]}`를 통해 모든 index의 값을 참조할 수 있습니다.
- `+=(추가할 값들)` 로 array에 값 추가가 가능합니다.
- Ex) `array+=("hello2" "ABC2" 123456)`

if

```
if [ 조건 1 ]  
then  
    ..실행..  
elif [ 조건 2 ]  
then  
    .. 실행 ..  
else  
    .. 실행 ..  
fi
```

연산자	설명 (True가 되는 경우)
! 명제	명제가 거짓일 때
-n 문자열	문자열의 길이가 0보다 클 때
-z 문자열	문자열의 길이가 0일 때
문자열1 = 문자열2	두 문자열이 서로 같을 때
문자열 != 문자열2	두 문자열이 서로 다를 때
정수1 -eq 정수2	두 정수가 서로 같을 때
정수1 -gt 정수2	정수1이 정수2보다 클 때
정수1 -lt 정수2	정수1이 정수2보다 작을 때
-d 디렉토리	해당 디렉토리가 존재할 때
-e 파일	해당 파일이 존재할 때

for

- Java나 Python처럼 foreach 형식으로 많이 사용합니다.

```
for <변수> in <sequence/array>  
do  
    .. 실행 ..  
done
```



foreach 스타일

```
for (( c=1; c<=5; c++ ))  
do  
    .. 실행 ..  
done
```



Three-expression
스타일

while

- 주로 redirection과 함께 파일을 읽을 때 사용합니다.

```
while [ 조건 ]  
do  
    .. 실행 ..  
done
```

```
while read line  
do  
    echo $line  
done < [파일 이름]
```

case

- 주로 bash script 파일의 실행 argument를 처리할 때 사용합니다.

```
case <변수> in
    case1)
        .. 실행 ..
    ;;
    case2)
        .. 실행 ..
    ;;
    ...
    *)
        .. 실행 ..
    ;;
esac
```

Case가 끝나면 ;;를
반드시 써준다.

모든 case에 만족하지
않는 경우를 처리

function

- 함수를 정의할 때 parameter는 따로 쓰지 않습니다.

```
function <함수 이름> (){  
    ... 실행 ...  
    return <return 할 것>  
}
```



```
<함수 이름> <parameters>
```

Argument

- \$0 : script의 이름
- \$1 ~ \$9 (그 이상은 \${10}, \${11} ...) : script로 들어오는 파라미터
- @\$: 모든 argument (배열 형태)
- \$# : argument의 개수
- \$? : 이전에 사용한 커맨드의 return code
- \$\$: 현재 scrip가 사용하고 있는 PID 값
- !! : 마지막으로 사용한 커맨드 전체
- \$_ : 마지막으로 사용한 커맨드의 argument들

다른 파일의 함수 사용하기

- Bash script 파일 안에서 source 명령어를 사용하면 다른 파일에 있는 함수 사용 가능.
- Python의 import나 C언어의 include와 같은 역할을 합니다.
- `source <sh 파일 이름>` 으로 사용.

Pipe

- 프로세스나 실행된 프로그램의 결과를 다른 프로그램으로 넘겨줄 때 사용합니다.
- 두 명령어 사이에 | (shift + W) 키워드로 사용합니다.
- [명령어 1] | [명령어 2] | ... | [명령어 N]

```
cat test.txt | while read line
do
    echo $line
done
```



Cat의 결과를 while로 보내서 사용합니다.

grep

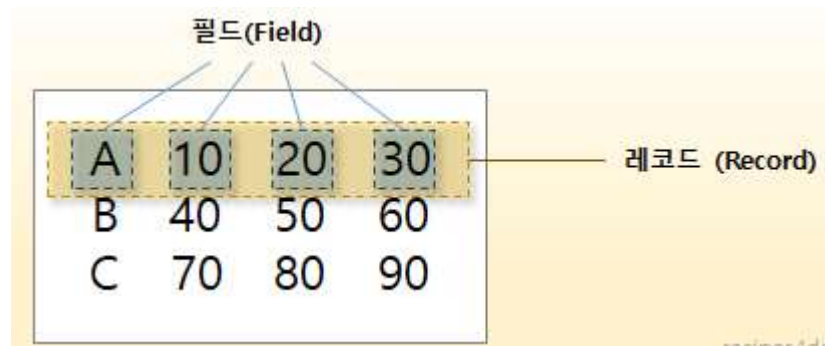
- `grep [option] [pattern] [file]` 로 사용
- 주어진 파일에서 만족하는 문자열 패턴을 찾는데 사용합니다.
- `grep -r [pattern]` 을 이용하면 하위 디렉토리에서 만족하는 문자열 패턴을 모두 찾아냅니다.
- Pattern에는 찾고자 하는 문자열 뿐만 아니라 정규 표현식을 사용할 수 있습니다.

Process management

- ps : 현재 쉘에서 실행중인 프로세스를 보여줍니다.
- ps -ef : 서버에서 실행중인 모든 프로세스를 보여줍니다.
- Grep과 pipe를 사용하면 특정 프로세스의 정보를 가져올 수 있습니다.
- Ex) ps -ef | grep "python"
- kill 명령어를 통해 작업을 강제 종료할 수 있습니다.

awk

- 파일로부터 레코드(record)를 선택하고, 선택된 레코드에 포함된 값을 조작하거나 데이터화 하는 것을 목적으로 사용합니다.
- awk [option] [awk program] [argument]로 사용합니다.



- 각 field는 \$1, \$2, ... \${10}, \${11} ... 방식으로 접근이 가능합니다.
- Awk program으로 주로 print를 많이 사용합니다.

Assignment

- Bash script로 Python 자동화 프로그램을 만들어 봅시다.
 1. example.list 파일에 적힌 파이썬 프로그램을 순차적으로 실행한다.
 2. Case 문을 통해 주어진 argument를 처리한다.
 3. 실행한 Python 프로그램의 결과들은 모두 result.log 파일에 저장한다.
 4. 코딩한 모든 파일을 자신의 Github를 통해 올려본다.

예시 파일) https://github.com/with1015/2020_ABC_Scheduler_Project

- MIT missing course Vim 부분 읽어 오기
- <https://missing.csail.mit.edu/2020/editors/>

Thank you

