aws re/start

명령 작업

Linux 기본 사항

학습 내용

강의 핵심 내용

학습 내용:

- Bash에서 명령과 함께 사용되는 특수 문자의 용도를 설명합니다.
- 일반적으로 사용되는 텍스트 검색 명령과 조작 명령을 설명합니다.
- 리디렉션과 다양한 리디렉트 옵션에 사용하는 일반 구문을 설명합니다.

특수 문자, 와일드카드, 리디렉션

Bash에서 따옴표 사용

- Bash Shell에서 공백은 **구분 기호**(구분자)입니다.
  - 예: `usermod -c devuser jdoe`
    - 이 명령은 /etc/passwd 파일의 `jdoe` 줄에 `devuser` 주석을 추가합니다.
    - 공백은 `usermod` 명령에 전달되는 두 인수를 구분합니다.
- Bash가 인수 안의 공백을 인식하도록 하려면 값을 따옴표(" ")로 묶습니다.
  - 예: `usermod -c "This is a dev user" jdoe`

```
jdoe:x:1002:1002:devuser:/home/jdoe:/bin/bash
```

```
jdoe:x:1002:1002:This is a dev user:/home/jdoe:/bin/bash
```

Bash 메타 문자

메타 문자	설명
* (star)	임의 개수의 임의 문자(와일드카드)
? (hook)	임의의 한 문자(와일드카드)
[characters]	괄호 사이에 일치하는 임의 문자(와일드카드)
`cmd` 또는 $cmd	작은따옴표('')가 아닌, 명령어를 대체하는 백틱(`) 사용
;	명령을 함께 연결
~	사용자 홈 디렉터리 표시
-	이전 작업 디렉터리 표시

참고: Bash에는 많은 메타 문자가 있습니다.

Bash 메타 문자: * 예제

```
[ec2-user@myLinux ~]$ ls
Desktop     myfile  myFilesList.txt  pic.png
documents   myFile  myfile.txt       sales_22082020.txt
[ec2-user@myLinux ~]$ ls documents/
[ec2-user@myLinux ~]$ cp *.txt documents/
[ec2-user@myLinux ~]$ ls documents/
myFilesList.txt  myfile.txt  sales_22082020.txt
[ec2-user@myLinux ~]$
```

Bash 메타 문자: ? 예제

```
[ec2-user@myLinux ~]$ ls
customers_2020.txt  Desktop          sales_2018.txt  sales_2020.txt
customers_2021.txt  sales_2017.txt   sales_2019.txt  sales_2021.txt
[ec2-user@myLinux ~]$ rm sales_201?.txt
[ec2-user@myLinux ~]$ ls
customers_2020.txt  customers_2021.txt  Desktop  sales_2020.txt  sales_2021.txt
[ec2-user@myLinux ~]$
```

Bash 메타 문자: [characters] 괄호의 예제

- 대괄호([ ]): 대괄호 사이의 문자와 일치
- 문자는 숫자, 문자, 특수 문자일 수 있습니다.
- 사용 대상
  - 문자 목록: [aef9]는 a, e, f, 9와만 일치
  - 문자 집합: [a-g]는 a에서 g까지의 모든 문자와 일치

```
[ec2-user@myLinux ~]$ ls
Desktop  log_a.txt  log_b.txt  log_c.txt  log_d.txt  log_e.txt  log_f.txt
[ec2-user@myLinux ~]$ ls log_[abc].txt
log_a.txt  log_b.txt  log_c.txt
[ec2-user@myLinux ~]$ ls log_[a-e].txt
log_a.txt  log_b.txt  log_c.txt  log_d.txt  log_e.txt
[ec2-user@myLinux ~]$
```

Bash 메타 문자: 다른 예제

```
[ec2-user@myLinux ~]$ echo "Current path is ["$(pwd)"]"
Current path is [/home/ec2-user]
[ec2-user@myLinux ~]$ echo "Current path is ["`pwd`"]"
Current path is [/home/ec2-user]
[ec2-user@myLinux ~]$
```

```
[ec2-user@myLinux etc]$ pwd
/etc
[ec2-user@myLinux etc]$ cd ~/Documents/
[ec2-user@myLinux Documents]$ pwd
/home/ec2-user/Documents
[ec2-user@myLinux Documents]$ echo "command1"; echo "command2"
command1
command2
[ec2-user@myLinux Documents]$
```

리디렉션 연산자

연산자	설명
>	명령의 출력을 파일로 전송
<	파일에서 입력된 명령 수신
|	명령을 실행하고 출력을 다른 명령에 대한 입력으로 리디렉션
>>	파일의 기존 내용에 명령이 출력한 값 추가
2>	명령에 따라 생성된 오류를 파일로 리디렉션
2>>	명령에 따라 생성된 오류를 파일의 기존 내용에 추가

주의! > 출력 리디렉터는 기본값으로 경고 없이 기존 파일 내용을 덮어씁니다.

파이프 리디렉터 사용 방법

- 예:
  - `ps -ef | grep sshd`
  - `ls -l /etc | less`

```
[ec2-user@myLinux ~]$ ps -ef | grep sshd
root      3167     1  0 Jun01 ?        00:00:01 /usr/sbin/sshd -D
root      8030  3167  0 06:56 ?        00:00:00 sshd: ec2-user [priv]
ec2-user  8066  8030  0 06:56 ?        00:00:00 sshd: ec2-user@pts/0
root      8737  3167  0 07:32 ?        00:00:00 sshd: ec2-user [priv]
ec2-user  8772  8737  0 07:32 ?        00:00:00 sshd: ec2-user@pts/1
root      8981  3167  0 07:43 ?        00:00:00 sshd: ec2-user [priv]
ec2-user  9016  8981  0 07:43 ?        00:00:00 sshd: ec2-user@pts/2
root      9151  3167  0 07:50 ?        00:00:00 sshd: ec2-user [priv]
ec2-user  9185  9151  0 07:50 ?        00:00:00 sshd: ec2-user@pts/3
ec2-user  9216  9186  0 07:50 pts/3    00:00:00 grep --color=auto sshd
```

리디렉터 > 및 >> 사용 방법

- info.txt 파일 채우기
  - uptime > info.txt
  - hostname >> info.txt
  - ip addr show eth0 >> info.txt

```
[userA@server00 ~]$ uptime > info.txt
[userA@server00 ~]$ hostname >> info.txt
[userA@server00 ~]$ ip addr show enp0s3 >>info.txt
[userA@server00 ~]$ cat info.txt
 22:56:33 up  1:28,  3 users,  load average: 0.00, 0.01, 0.05
server00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:5c:57:3f brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global noprefixroute dynamic enp0s3
       valid_lft 81096sec preferred_lft 81096sec
    inet6 fe80::67ef:dc4a:90a:6b0d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

리디렉션 오류

기타 예제

- `myprogram 2>error.log`
  - myprogram 프로그램을 실행하고 errors.log 파일에 오류 전송

- `find ../ -name 'p*' 2>error.log`
  - ../ 폴더에서 p로 시작하는 파일 검색
  - 오류는 errors.log 파일에 기록됨

```
[ec2-user@myLinux ~]$ find ../ -name 'p*' 2>error.log
../ec2-user/.vnc/passwd
[ec2-user@myLinux ~]$ cat error.log
find: '../mmajor': Permission denied
find: '../jdoe': Permission denied
[ec2-user@myLinux ~]$
```

noclobber 변수

- 출력 리디렉션은 기본값으로 경고 없이 기존 파일을 덮어씁니다.
- 이 동작을 방지하기 위해 noclobber 변수를 설정할 수 있습니다. 대부분의 Linux 디스트리뷰션에서 기본값으로 설정되지 않았습니다.
- 예:
  - `set -o noclobber`
  - `echo "test1" > textfile.txt`
  - `echo "test2" > textfile.txt`

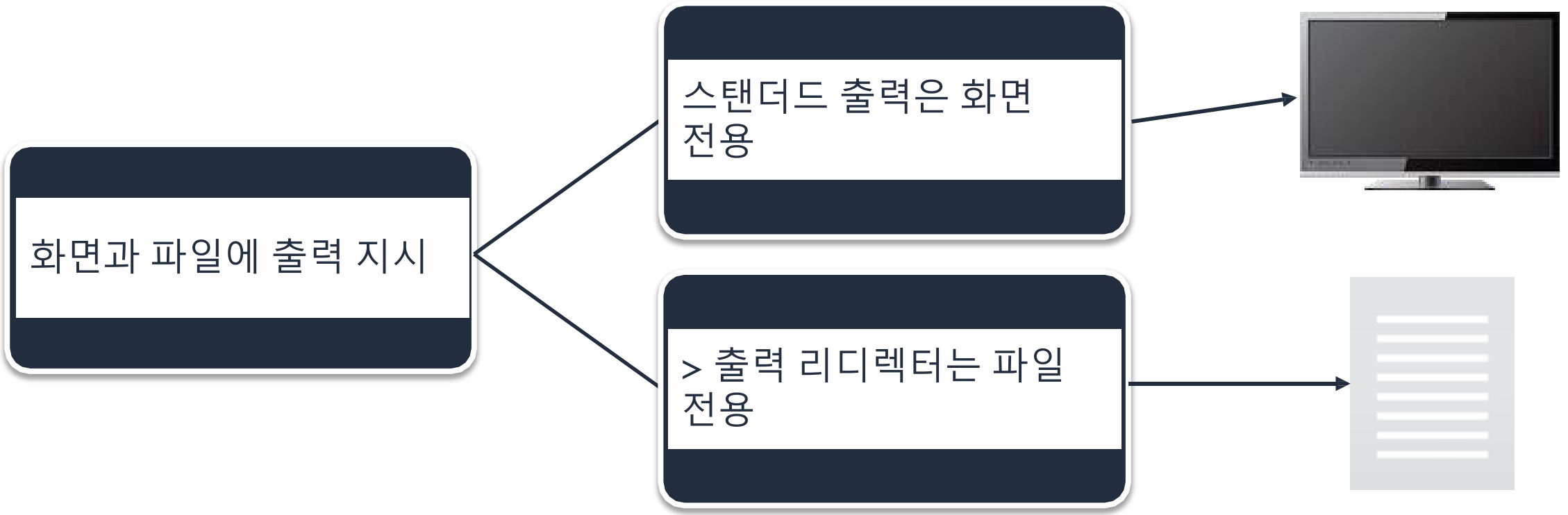
```
[userA@server00 ~]$ set -o noclobber
[userA@server00 ~]$ echo "test1" > textfile.txt
[userA@server00 ~]$ echo "test2" > textfile.txt
bash: textfile.txt: cannot overwrite existing file
```

파이프 리디렉터

```
[ec2-user@myLinux ~]$ ps -au
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       3124  0.0  0.1 121284  1808 tty1     Ss+  Jun01   0:00 /sbin/agetty --
root       3125  0.0  0.2 120932  2124 ttyS0    Ss+  Jun01   0:00 /sbin/agetty --
ec2-user  9575  0.0  0.4 124844  4112 pts/4    Ss+  08:09   0:00 -bash
ec2-user 10586  0.0  0.4 124944  4144 pts/5    Ss+  09:07   0:00 -bash
ec2-user 11205  0.0  0.3 124844  3912 pts/0    Ss+  09:42   0:00 -bash
ec2-user 11674  0.0  0.3 124844  3908 pts/1    Ss   10:06   0:00 -bash
ec2-user 12065  0.0  0.3 164360  3836 pts/1    R+   10:24   0:00 ps -au
[ec2-user@myLinux ~]$ ps -au | grep ec2-user
ec2-user  9575  0.0  0.4 124844  4112 pts/4    Ss+  08:09   0:00 -bash
ec2-user 10586  0.0  0.4 124944  4144 pts/5    Ss+  09:07   0:00 -bash
ec2-user 11205  0.0  0.3 124844  3912 pts/0    Ss+  09:42   0:00 -bash
ec2-user 11674  0.0  0.3 124844  3908 pts/1    Ss   10:06   0:00 -bash
ec2-user 12074  0.0  0.3 164360  3920 pts/1    R+   10:25   0:00 ps -au
ec2-user 12075  0.0  0.0 119416   940 pts/1    S+   10:25   0:00 grep --color=au
to ec2-user
[ec2-user@myLinux ~]$ ps -au | grep ec2-user | awk '{print $1 $2}'
ec2-user9575
ec2-user10586
ec2-user11205
ec2-user11674
ec2-user12083
ec2-user12084
ec2-user12085
```


aws re/start

tee 명령




```
[userA@server00 ~]$ hostname | tee file1.txt
server00
[userA@server00 ~]$ cat file1.txt
server00
```

명령 대체, 연결, 필터링

명령 대체

- 명령줄이나 다른 명령 안에서 명령을 중첩할 수 있습니다. 해당 명령의 결과는 나머지 명령에 따라 표시되거나 사용됩니다.
- 백틱(`, 이전 형식)과 함께 사용합니다.
- $(command, 최신 형식)로 수행할 수 있습니다.

```
[root@server00 ~]# cat demo.sh
#!/bin/bash
DATE=`date`
echo "Today's date is $DATE."
USERS=`who | wc -l`
echo "There are $USERS logged in."
[root@server00 ~]# ./demo.sh
Today's date is Mon Mar 11 00:35:58 GMT 2019.
There are 2 logged in.
[root@server00 ~]#
```

세미콜론으로 명령 연결

세미콜론(;)은 여러 가지 명령을 실행할 때 사용하며, 모두 같은 줄에 작성합니다.

```
[root@server00 ~]# date ; w ; uptime
Mon Mar 11 16:02:49 GMT 2019
 16:02:49 up 15:49,  2 users,  load average: 0.08, 0.04, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
student0 :0       :0               00:27   ?xdm?   5:18   0.37s /usr/libexec/gn
student0 pts/0    :0               00:27    1.00s  0.41s  6.67s /usr/libexec/gn
 16:02:50 up 15:49,  2 users,  load average: 0.08, 0.04, 0.05
```

| grep 사용

- grep은 일반적으로 파이프(|)와 함께 다른 명령 다음에 사용됩니다.
- 예:
  - ps -ef | grep sshd
  - cat /var/log/secure | grep fail

```
[root@server00 ~]# ps -ef | grep -i sshd
root      1221     1  0 00:13 ?        00:00:00 /usr/sbin/sshd -D
root     28208  6128  0 16:04 pts/0    00:00:00 grep --color=auto -i sshd
[root@server00 ~]# rpm -qa | grep samba
samba-common-libs-4.7.1-6.el7.x86_64
samba-common-4.7.1-6.el7.noarch
samba-client-libs-4.7.1-6.el7.x86_64
```

cut 명령

- 문자, 바이트 위치, 구분 기호로 텍스트 줄에서 섹션을 자름
- 해당 정보를 스탠더드 출력으로 표시
- 텍스트 파일에서 관련 정보를 가져와 해당 정보를 표시할 때 사용
- 출력을 새 파일로 파이프 가능

```
[ec2-user@myLinux ~]$ cat names.csv
Alejandro,Rosalez,42,Cherbourg,FR
Carlos Salazar,33, Paris,FR
Li Juan,25,Bordeaux,FR
[ec2-user@myLinux ~]$ cut -d ',' -f 1 names.csv
Alejandro
Carlos Salazar
Li Juan
[ec2-user@myLinux ~]$
```

```
[ec2-user@myLinux ~]$ cut -c 1-2 names.csv
Al
Ca
Li
[ec2-user@myLinux ~]$ cut -b 1-5 names.csv
Aleja
Carlo
Li Ju
[ec2-user@myLinux ~]$
```

텍스트 처리 및 검색

sed 명령

- 비대화형 텍스트 편집기
- 제공된 규칙에 따라 데이터 편집(삽입, 삭제, 검색, 바꾸기 가능)
- 정규 표현식 지원

```
[ec2-user@myLinux ~]$ echo "example.com page" |sed 's/page/website/'
example.com website
[ec2-user@myLinux ~]$ cat example.txt
example.com page
[ec2-user@myLinux ~]$ sed 's/page/website/' example.txt
example.com website
[ec2-user@myLinux ~]$
```


aws re/start

sort 명령

- 파일 내용을 알파벳 순, 역순, 숫자, 월 순서로 정렬
- 예:
  - `sort file.txt`: 알파벳순으로 줄 출력
  - `sort -r file.txt`: 알파벳 역순으로 줄 출력
  - `sort -u file.txt`: 중복 항목 제거(로그 파일에 유용)
  - `sort -M file.txt`: 월순으로 줄 출력

```
[ec2-user@myLinux ~]$ cat names.csv
Alejandro,Rosalez,42,Cherbourg,FR
Carlos Salazar,33, Paris,FR
Li Juan,25,Bordeaux,FR
John Doe,51,Lyon,FR
[ec2-user@myLinux ~]$ sort -c names.csv
sort: names.csv:4: disorder: John Doe,51,Lyon,FR
```

```
[ec2-user@myLinux ~]$ sort names.csv
Alejandro,Rosalez,42,Cherbourg,FR
Carlos Salazar,33, Paris,FR
John Doe,51,Lyon,FR
Li Juan,25,Bordeaux,FR
```

awk 명령

- 데이터를 변환하는 작은 프로그램을 만들 때 사용
- 변수 정의
- 문자열과 산술 연산자 사용
- 제어 흐름과 루프 사용
- 형식이 지정된 보고서 생성
- 구문: awk를 호출하는 두 가지 방법(하나는 명시적 프로그램, 다른 하나는 파일에 있는 프로그램)
  - `awk option -f program-file input-file`
  - `awk option 'program' input-file`
- 옵션:
  - `-F fs` 필드 구분자를 지정할 때(기본 구분자는 임의의 숫자의 공백 또는 탭)
  - `-f source-file` awk 스크립트가 있는 파일을 지정할 때
  - `-v var=value` 변수를 지정할 때

awk 명령(계속)

```
[ec2-user@myLinux ~]$ cat names.csv
Alejandro Rosalez,42,Cherbourg,FR,arosalez@company.com
Carlos Salazar,33,Paris,FR,csalazar@company.com
Li Juan,25,Bordeaux,FR,ljuan@company.com
John Doe,,Lyon,FR,jdoe@company.com
[ec2-user@myLinux ~]$ awk -F , '{print$3}' names.csv
Cherbourg
Paris
Bordeaux
Lyon
[ec2-user@myLinux ~]$ awk -F @ '{print$1}' names.csv
Alejandro Rosalez,42,Cherbourg,FR,arosalez
Carlos Salazar,33,Paris,FR,csalazar
Li Juan,25,Bordeaux,FR,ljuan
John Doe,,Lyon,FR,jdoe
```

```
[ec2-user@myLinux ~]$ awk -F , '/[0-9][0-9]/ {print $1 }' names.csv
Alejandro Rosalez
Carlos Salazar
Li Juan
```


aws re/start

확인 질문

- 언제 ? 와일드카드를 * 와일드카드 대신 사용합니까?
- `Uniq` 명령은 로그 파일을 분석하는 데 어떻게 도움을 줍니까?
- 명령 대체가 어떻게 Bash 스크립트를 더 빠르게 실행할 수 있습니까? 명령을 대체하는 데 또 다른 장점이 있습니까?

요점

- 따옴표(" ")는 공백이 있는 인수에 대한 일반적인 Bash 해석을 개별 인수 두 개로 재정의합니다.
- 메타 문자는 출력, 와일드카드, 연결 명령을 제어하는 강력한 도구입니다.
- Bash의 스탠더드 I/O는 키보드 입력, 모니터 출력입니다.
- 와일드카드는 검색 시 한 개에서 여러 개의 알 수 없는 문자 또는 제한된 특정 값 집합을 지정하는 데 사용됩니다.
- 파이프(|)를 사용하여 한 명령의 출력을 다른 명령으로 보낼 수 있습니다.
- `grep`은 이전 명령으로 파이핑된 출력을 검색하는 데 사용될 수 있습니다.
- `sed`, `sort`, `awk` 명령은 텍스트를 처리하고 검색하는 데 사용됩니다.

aws re/start

감사합니다.

aws re/start