과제 내용

- 리눅스 명령어와 vim 에디터에서 매크로 관련하여 조사

1. 리눅스 명령어 : top, ps, jobs, kill 명령어 조사하기

2. vim 에디터에서 매크로 사용방법에 대하여 조사하기 (q , @)

- 조사한 내용을 자신의 github 페이지에 README 파일로 정리함
- 제출할 때는 github 페이지 링크와 pdf 파일을 같이 제시함
- github 페이지의 내용을 pdf 파일로 인쇄하여 같이 반드시 첨부할 것
- 마감기한 이후에 수정하는 것을 방지하기 위한...
- pdf 파일을 제시하지 않을 경우에는 과제 제출로 인정이 절대 안됨!!!
- 정확한 링크 주소를 제대로 안보낼 경우에는 감점 처리함
- 과제 제출 이후에는 commit과 push를 절대 수행하지 말 것
- 마감기한 : 2022년 6월 5일 (일), 23:59:59 까지
- 과제 점수는 10점이며, 지각제출은 불허입니다.

자세한 내용은 강의동영상 및 강의자료를 참고하길 바랍니다.

---

1. 리눅스 명령어 : top, ps, jobs, kill 명령어 조사하기

1.top

1. 옵션

-b : 배치모드로 정보를 출력한다. 실시간 상화 대화형모드로 정보를 화면에 일렬로 출력한다.

-d delay : 지정한 시간(delay 초)의 간격으로 정보를 업데이트하여 출력한다.

-i idle : 토글값이 off일 때, idle 프로세스나 좀비 프로세스 정보를 출력하지 않는다.

-n num : 지정한 시간(num)만큼 업데이트 정보를 출력한다.

-p pid : 지정한 프로세스 ID(pid)의 정보만을 출력한다.

-q : 시간의 간격 없이 계속하여 업데이트 정보를 출력한다.

-s : 몇 개의 대화식 명령을 비활성화한다(시큐어 모드).

-S : 누적된 정보를 출력한다(cumulative 모드).

2. 설명 및 예제

top 명령어는 시스템의 프로세스/메모리 사용 상태를 5초의 간격으로 업데이트하여 출력한다.
화면에 출력되는 기본값은 현재시간, 시스템 업데이트(1) 시간, 시스템에 로그인한 사용자 수,
지난 1분, 지난 5분, 지난 15분간의 시스템 평균 부하를 출력한다.
이 목록 아래에 프로세스 정보, CPU 상태, 메모리와 스왑 상태를 출력한다.
top 명령어를 실행한 후 초기 화면에서 키를 입력하면 사용할 수 있는 단축키 목록을 확인할 수 있다.

3. top 단축키 명령어

명령어	설명
space	정보를 업데이트한다.
^L	스크린을 다시 초기화한다.
F or f	필드를 추가나 제거한다.
O or	출력하는 필드의 정렬 순서를 변경한다.
h or ?	사용 가능한 명령어를 출력한다.
k	프로세스를 종료시킨다.
n or #	출력할 프로세스의 수를 지정한다.
s	출력할 정보의 업데이트 시간을 지정한다.
W	~/.toprc에 설정된 내용을 저장한다.
q	top을 종료한다.

4. top 보기 수정 단축키

출력되는 메인 정보 창 중에 상단의 정보를 수정한다.

명령어	설명
S	cumulative 모드(실시간 정보를 누적 데이터로 보여 줌)를 선택/해제한다.
i	idle 프로세스 정보를 출력한다/해제한다.
I	Irix나 솔라리스 정보를 출력한다/해제한다.
c	명령행에서 실행한 명령어 자체로 출력한다/해제한다.
l	로드 평균 정보를 출력한다/해제한다.
m	메모리 정보를 출력한다/해제한다.
t	요약된 정보만을 출력한다/해제한다.

5.top 정렬 단축키

메인 창에서 실행하는 명령으로 현재 정보를 사용자의 요구대로 정렬한다.

명령어	설명

명령어	설명
r	프로세스의 우선순위를 변경한다.
N	pid 정보를 기준으로 정렬한다.
A	age 정보를 기준으로 정렬한다.
P	CPU 사용량을 기준으로 정렬한다.
M	적재된 메모리 사용량을 기준으로 정렬한다
T	시간/누적시간을 기준으로 정렬한다.
u	지정한 사용자의 정보만을 출력한다.

2. ps

ps 옵션 : 시스템의 동작중인 모든 프로세스의 정보를 표출하는 명령어

1. 설명 및 예제

ps 명령어는 프로세스의 현재 상태를 출력한다. ps로 현재 사용하는 프로세스의 상태를 살펴보자.
아래는 PID, TTY, TIME, CMD 헤더의 필드와 내용을 출력한다.

```
$ ps
  PID TTY TIME CMD
  3134 pts/1        00:00:00 bash
  15027 pts/1       00:00:00 ps
```

2. 전체 프로세스와 관련된 옵션

	전체 프로세스와 관련된 옵션
-A	모든 프로세스를 출력한다.
-N	-A 옵션과 비슷하나 ps 프로세스를 제외하고 출력한다.
-a	세션 리더 및 터미널에 속하지 않는 프로세스를 제외하고 출력한다.
-d	세션 리더를 제외한 모든 프로세스를 출력한다.
-e	커널 프로세스를 제외한 모든 프로세스를 출력한다.
T	현재 터미널에서의 모든 프로세스를 출력한다.
a	현재 터미널의 사용자 고유 프로세스를 출력한다.
r	현재 실행 중인 프로세스를 출력한다.
x	터미널이 없는 프로세스를 출력한다.

전체 프로세스와 관련된 옵션

--deselect	-N 옵션과 같다.

특정 프로세스를 선택하여 보여주는 옵션

-C	지정한 명령어의 이름에 관련된 정보를 출력한다.
-G	그룹 ID에 관련된 정보를 출력한다(이름도 지원).
-U	사용자 ID에 관련된 정보를 출력한다(이름도 지원).
-g	지정한 세션 리더 혹은 그룹명에 관련한 정보를 출력한다.
-p	프로세스 ID를 출력한다.
-s	세션에 속한 프로세스를 지정한다.
-t	tty를 지정한다.
-u	사용자 ID를 지정한다(이름도 지원).
U	지정한 사용자의 프로세스를 출력한다.
p	프로세스 ID를 지정한다.
t	tty를 지정한다.
--Group	실제 그룹 이름이나 ID를 지정한다.
--User	실제 사용자 이름이나 ID를 지정한다.
--group	유효 사용자 이름이나 ID를 지정한다.
--pid	프로세스 ID를 지정한다.
--sid	세션 ID를 지정한다.
--tty	터미널을 지정한다.
--user	유효 사용자 이름이나 ID를 지정한다.
-123	--sid 123과 같은 의미이다.
123	--pid 123과 같은 의미이다.

출력 결과 필드를 제어하는 옵션

-0	PID, TTY, STAT, TIME, COMMAND 등의 필드 목록을 출력한다.
-c	PID, CLS, PRI, TTY, TIME. CMD 등의 필드 목록을 출력한다.
-f	UID, PID, PPID, C, STIME, TTY, TIME, CMD 등의 필드를 CMD 필드의 전체 명령어 형태로 출력한다.
-j	PID, PGID, SID, TTY, TIME, CMD 등의 필드 목록을 출력한다.
-l	F, S, UID, PID, PPID, C, PRI, NI, ADDR, SZ, WCHAN, TTY, TIME, CMD 필드의 상세 정보를 출력한다.
-o	사용자가 정의한 포맷을 지정한다.

	출력 결과 필드를 제어하는 옵션
-y	-l 이나 l 옵션과 함께 ADDR 필드를 RSS 필드로 출력한다.
0	PID, TTY, STAT, IME COMMAND 필드 정보를 출력한다.
X	PID, STACKP, ESP, EIP, TMOUT, ALARM, STAT, TTY, TIME, COMMAND 필드의 정보를 리눅스 i386 레지스터 형식으로 출력한다.
j	PPID, PID, PGID, SID, TTY, TPGID, STAT, UID, TIME, COMMAND 필드의 정보를 작업 제어에 관련된 형식으로 출력한다.
l	F, S, UID, PID, PPID, C, PRI, NI, ADDR, SZ, PSS, WCHAN, TTY, TIME, CMD 필드의 정보를 출력하고 -l 옵션과 함께 PSS 필드를 추가하여 출력한다.
o	사용자 지정 형식으로 출력한다.
s	UID, PID, PENDING, BLOCKED, IGNORED, CAUGHT, STAT, TTY, TIME, COMMAND 필드의 정보를 출력한다.
u	USER, PID, %CPU, %MEM, VSZ, RSS, TTY, STAT, START, TIME, COMMAND 필드의 정보를 출력한다.
v	PID, TTY, STAT, TIME, MAJFL, TRS, DRS, RSS, %MEM, COMMAND 필드의 정보를 출력한다.
--format	사용자 지정 형식으로 출력한다.

	프로그램의 정보
-V	버전 정보를 출력한다.
L	모든 형태의 지시자를 출력한다.
V	버전 정보를 출력한다.
--help	사용법을 출력한다.
--info	디버깅 정보를 출력한다.
--version	버전 정보를 출력한다.

3. jobs

1. 설명 및 예제

jobs는 작업이 중지된 상태, 백그라운드로 진행 중인 작업 상태, 변경되었지만 보고되지 않은 상태 등을 표시한다.
현재 환경의 작업 상태를 아래와 같이 확인할 수 있다.

```
$ jobs
 [1]- 정지됨        vi
 [2]+ 정지됨        tail -f /var/log/messages
```

-l 옵션은 state 필드 앞에 프로세스 ID를 출력한다.

```
$ jobs
 [1]- 4908 Stopped (tty output)        vi
 [2]+ 4987 Stopped                     tail -f /var/log/messages
```

2. 사용법

```
jobs[옵션][jobID...]
jobs -x command[args]
```

-l : 프로세스 그룹 ID를 state 필드 앞에 출력한다.
-n : 프로세스 그룹 중에 대표 프로세스 ID를 출력한다.
-p : 각 프로세스 ID에 대해 한 행씩 출력한다.
command : 지정한 명령어를 실행한다.

3. 옵션법

jobs 명령어로 확인할 수 있는 세션의 상태값은 다음과 같다.

상태	설명
Done	작업이 완료되어 0을 반환하고 종료했음을 뜻한다.
Done	(code) 작업이 정상적으로 완료했으며, 0이 아닌 코드를 반환했음을 뜻한다.
Stopped	작업이 일시 중단됨을 뜻한다.
Stopped	(SIGTSTP) SIGTSTP 신호가 작업을 일시 중단했음을 뜻한다.
Stopped	(SIGSTOP) SIGSTOP 신호가 일시 중단했음을 뜻한다.
Stopped	(SIGTTIN) SIGTTIN 신호가 작업을 일시 중단했음을 뜻한다.
Stopped	(SIGTTOU) SIGTTOU 신호가 작업을 일시 중단했음을 뜻한다.

다음과 같이 하면 v로 시작하는 모든 프로세스 ID를 확인할 수 있다.

```
jobs[옵션][jobID...]
jobs -x command[args]
```

---

kill

---

1. 사용법

```
kill[-s시그널][-a]pid...
kill-l[시그널]
```

	옵션
pid ...	종료시킬 프로세스 ID나 프로세스 이름을 지정한다.
-s	전달할 시그널의 종류를 지정한다. 여기에는 시그널 이름이나 번호를 써준다.
-l	시그널로 사용할 수 있는 시그널 이름들을 보여준다. 이것은 /usr/include/linux/signal.h 파일에서도 알 수 있다.
-1,	-HUP 프로세스를 재활성화한다.
-9	프로세스를 강제로 종료시킨다.

2. 설명 및 예제

kill 명령어는 프로세스에 종료 시그널을 보낸다. 시스템에 얘기치 않은 문제가 생긴 프로세스를 종료시킬 수 있다.
만일 kill 명령으로 종료되지 않는 프로세스는 -9 옵션을 사용해서 강제로 종료하면 된다.
아래와 같이 ps 명령어로 sshd 프로세스를 확인할 수 있다.

```
# ps aux | grep sshd
root    2518 0.0. 0.1 7084 1076 ? Ss Jun07 ...
root    12095 0.0. 0.1 9936 1076 ? Ss Jun07 ...
pirania 2518 0.0. 0.1 9936 1076 ? Ss Jun07 ...
root    2518 0.0. 0.1 3932 684 pts/1 R+ Jun07 ...
```

현재 sshd 프로세스는 root 사용자 권한으로 동작하고 있으며 PID는 2518와 12095이다.
참고로 ps aux 명령에서 볼 수 있는 프로세스 정보에 대한 각각의 필드 내용은 다음과 같다.
좀 더 자세한 정보는 ps 명령어 페이지를 참고하자

	명령어 페이지
Root	USER 프로세스의 사용자
2518	PID 프로세스 ID
0.0	%CPU 마지막 1분 동안 프로세스가 사용한 CPU 점유율
0.1	%MEM 마지막 1분 동안 프로세스가 사용한 메모리의 점유율
7084	VSZ 가상메모리에 있는 프로세스의 KB 단위 크기
1076	RSS 프로세스의 실제 메모리의 크기로 KB 단위

	명령어 페이지
?	TTY 연결되어 있는 터미널
Ss	STAT 실행되고 있는 프로세스 상태
Jun07	START 프로세스가 시작된 날짜
0:00	TIME 프로세스가 소비한 총 시간
/usr/sbin/sshd	COMMAND 사용자가 실행한 명령 이름

ps 명령으로 확인한 PID를 이용하면 원하는 프로세스를 종료시킬 수 있다.

```
# kill 2518
```

만일 kill 명령으로 종료되지 않으면 -9 옵션으로 강제 종료해보자.

```
# kill -9 2518
```

kill -HUP pid 명령을 이용하면 프로세스를 종료 후 다시 재실행한다.

```
# kill -HUP 2518
```

---

2. vim 에디터 매크로 사용방법 조사하기 (q, @)

1. vim 매크로 q

q{레지스터} 로 매크로 기록 시작

q 로 매크로 기록 종료

q<저장할 매크로 문자>

ex) -a 문자에 매크로 내용을 기록하기

qa -> 매크로 작성 -> q

Normal Mode 에서 q 를 입력하면, 하단 상태표시줄에 q 가 표시된다.
이는 앞으로 기록할 레지스터를 지정해주기를 대기하고 있는 상태이다.
레지스터(a-z, 0-9 중 하나)를 정하여 입력하면 상태표시줄에 'Recording @a'(레지스터로 'a'를 입력했다고 가정) 와

같이 실제 명령어를 대기하고 있는 상태가 된다.
일련의 동작들을 입력한 뒤 다시 q 를 입력하면 매크로 기록이 종료된다.

(블로그에 있는 q사용법)

(1) 기본적인 매크로 사용 방법 커맨드 모드 (esc 누른 상태) 에서.

1. 'q' 를 누르고 a~z 사이 문자로 매크로 recording 시작
2. 커맨드 모드로 돌아와서 'q'를 누르면 매크로 recording 끝
3. 매크로를 사용하려면 커맨드 모드에서 @+a~z 를 입력하면 됨

(2) 자주 사용하는 매크로 등록 방법

1. ~/.vimrc 를 연다.
2. let @a = '매크로 문자열' // 이런 식으로 매크로로 동작시킬 문자열을 입력한다.

(3) 매크로 문자열을 편집기에 그대로 출력하는 방법

- 첫 번째 방법: 편집 모드에서 ctrl + r, ctrl + r, 매크로 문자 를 순서대로 입력하면 그대로 출력된다.
- 두 번째 방법: 커맨드 모드에서 "매크로문자p 를 입력하여 래지스터로부터 바로 붙여넣기 한다. (ex: "ap, "bp, "cp, "dp, ...)

첫 번째 방법은 방향키 등의 특수키 문자가 제대로 붙여넣기가 되지 않는다. 분명히 출력 결과물이 같아 보이지만 실제 바이너리를 뜯어보면 다르게 출력됨을 알 수 있다.

2. vim 매크로 @

1. 특정 문자에 저장한 매크로 실행@<저장한 매크로 문자>
2. 매크로 반복실행 반복횟수@<저장한 매크로 문자>
3. 마지막에 수행한 매크로 실행 @@

@{레지스터} 로 저장된 매크로 실행 @@ 로 직전에 실행한 매크로 재실행 {반복횟수}@{레지스터} 또는 {반복횟수}@@ 로 저장된 매크로를 '반복횟수' 만큼 재실행

@{레지스터} 로 특정 레지스터에 저장된 매크로를 실행시킬 수도 있고,
@@ 로 직전에 실행한 매크로를 재실행할 수도 있다.

매크로를 사용하는 방법은 연속된일을 수행할때 사용하는 것같다.

동일한 과정을 100번을 할려고하면

매크로에 저장을 한후 100개의 코드에 사용을 하는 것이다.