

Chapter 06 프로세스 관리

목차

00 개요

01 프로세스의 개념

02 프로세스 관리 명령

03 포그라운드/백그라운드 프로세스와 작업제어

04 작업 예약

학습목표



- 프로세스가 무엇인지 설명할 수 있다.
- 프로세스의 목록을 확인하고 특정 프로세스를 검색할 수 있다.
- 프로세스를 강제로 종료할 수 있다.
- 프로세스 관리 도구로 전체 프로세스의 상태를 확인할 수 있다.
- 포그라운드와 백그라운드 작업의 차이를 설명할 수 있다.
- 명령을 자동으로 실행하는 방법을 직접 설정할 수 있다.

00 개요

00. 개요

■ 리눅스 학습 맵에서 6장의 위치

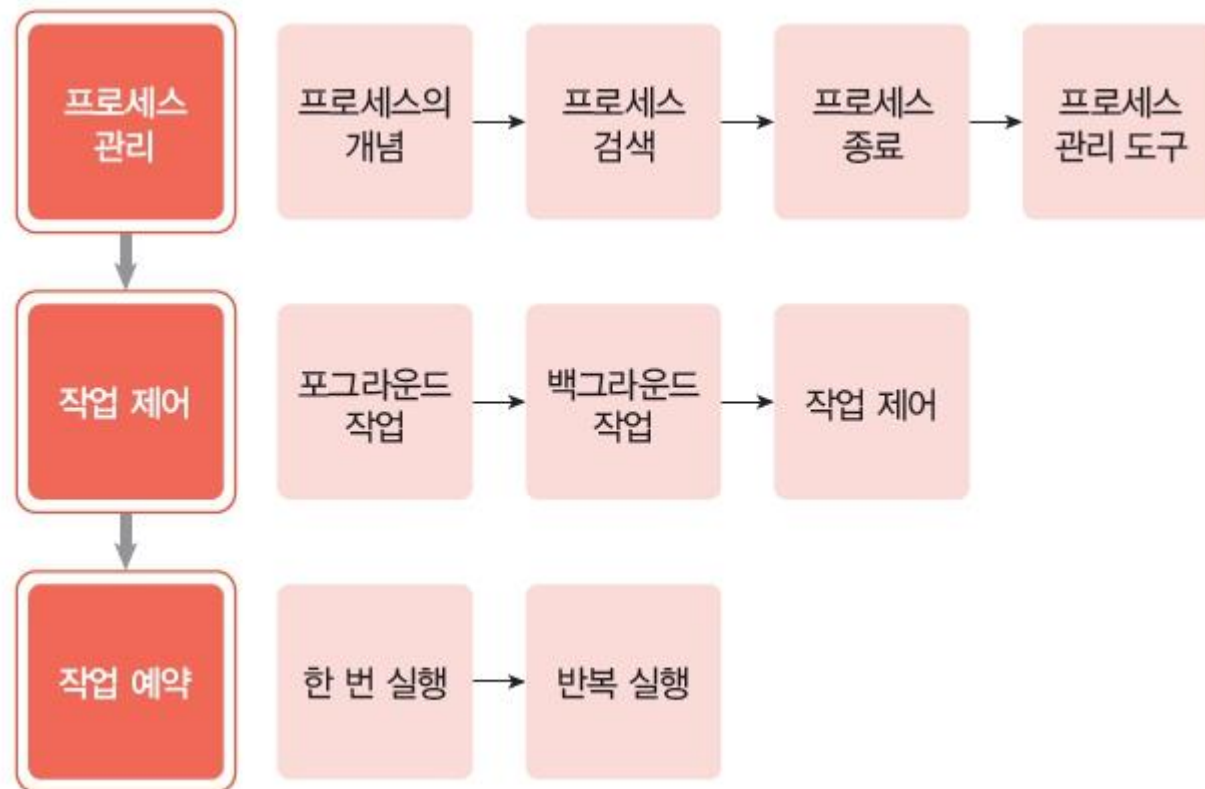
- 6장은 제2단계 중 마지막 항목으로 프로세스가 무엇인지 이해하고, 이를 관리하는 명령의 사용법과 작업 제어, 작업 예약 방법을 익힌다.



00. 개요

■ 6장의 내용 구성

- 프로세스의 개념 이해하기
- 프로세스를 검색하는 방법 이해하기
- 프로세스를 종료하는 방법 학습하기
- 프로세스 관리도구 사용법 익히기
- 포그라운드와 백그라운드 작업을 제어하는 방법 익히기
- 작업을 예약하는 방법 익히기



01 프로세스의 개념

01. 프로세스의 개념

■ 프로세스

- 현재 시스템에서 실행 중인 프로그램
- 리눅스는 다중 프로세스 시스템으로 동시에 여러 프로세스 실행

■ 프로세스의 부모-자식 관계

- 리눅스에서 모든 프로세스는 부모-자식 관계가 있음
- 부모프로세스가 자식프로세스 생성

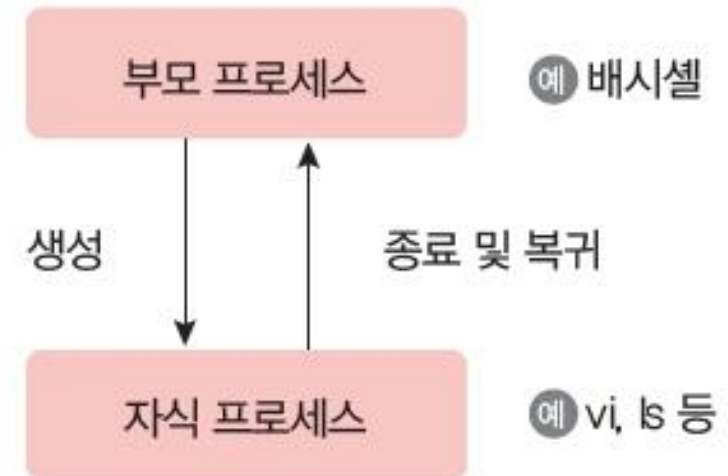


그림 6-1 부모 프로세스와 자식 프로세스의 관계

01. 프로세스의 개념

■ 프로세스 번호

- 각 프로세스는 고유한 번호를 가짐: PID
- 1번 프로세스는 system, 2번 프로세스는 kthreadd

■ 프로세스의 종류

- 데몬 프로세스: 특정 서비스를 제공하기 위해 존재하며 리눅스 커널에 의해 실행
- 고아 프로세스: 자식 프로세스가 아직 실행 중인데 부모 프로세스가 먼저 종료되면 자식 프로세스는 고아 프로세스가 됨
- 좀비 프로세스
 - 자식 프로세스가 실행을 종료했는데도 프로세스 테이블 목록에 남아 있는 경우가 있는데 이러한 자식 프로세스를 좀비 프로세스라고 함
 - 좀비 프로세스는 프로세스 목록에 defunct 프로세스라고 나오기도 함
 - 좀비 프로세스가 증가하면 프로세스 테이블의 용량이 부족해서 정상적인 프로세스가 실행되지 않을 수도 있음

02 프로세스 관리 명령

02. 프로세스 관리 명령

■ 프로세스 목록 확인

ps

- **기능** 현재 실행 중인 프로세스에 대한 정보를 출력한다.
- **형식** ps [옵션]
- **옵션** <유닉스 옵션>
 - e: 시스템에서 실행 중인 모든 프로세스의 정보를 출력한다.
 - f: 프로세스에 대한 자세한 정보를 출력한다.
 - u uid: 특정 사용자에게 대한 모든 프로세스의 정보를 출력한다.
 - p pid: pid로 지정한 특정 프로세스의 정보를 출력한다.<BSD 옵션>
 - a: 터미널에서 실행시킨 프로세스의 정보를 출력한다.
 - u: 프로세스 소유자 이름, CPU 사용량, 메모리 사용량 등 상세 정보를 출력한다.
 - x: 시스템에서 실행 중인 모든 프로세스의 정보를 출력한다.<GNU 옵션>
 - pid PID 목록: 목록으로 지정한 특정 PID 정보를 출력한다.
- **사용 예**
 - ps
 - ps -ef
 - ps aux

02. 프로세스 관리 명령

■ ps 명령의 옵션 유형

- 유닉스(SVR4) 옵션: 묶어서 사용할 수 있고, -으로 시작한다(예: -ef).
- BSD 옵션: 묶어서 사용할 수 있고, -으로 시작하지 않는다(예 aux).
- GNU 옵션: - 두 개로 시작한다(예 --pid).

02. 프로세스 관리 명령

■ 현재 단말기의 프로세스 목록 출력하기: ps

- 현재 셸이나 터미널에서 실행한 사용자 프로세스의 정보를 출력

```
user1@myubuntu:~$ ps
  PID TTY          TIME CMD
 1584 pts/0        00:00:00 bash
 1653 pts/0        00:00:00 ps
```

02. 프로세스 관리 명령

■ 프로세스 상세 정보 출력하기: -f 옵션

- PPID와 C(CPU 사용량), 시작 시간 등의 정보가 추가로 출력

```
user1@myubuntu:~$ ps -f
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
user1	1584	1583	0	00:00	pts/0	00:00:00	-bash
user1	1660	1584	0	00:16	pts/0	00:00:00	ps -f

표 6-1 ps -f의 출력 정보

항목	의미	항목	의미
UID	프로세스를 실행한 사용자 ID	STIME	프로세스의 시작 날짜나 시간
PID	프로세스 번호	TTY	프로세스가 실행된 터미널의 종류와 번호
PPID	부모 프로세스 번호	TIME	프로세스 실행 시간
C	CPU 사용량(% 값)	CMD	실행되고 있는 프로그램 이름(명령)

02. 프로세스 관리 명령

■ 터미널에서 실행시킨 프로세스 정보 출력하기: a 옵션

```
user1@myubuntu:~$ ps a
  PID TTY          STAT TIME  COMMAND
  1586 tty2        Ssl+   0:00 /usr/libexec/gdm-wayland-session env GNOME_SHELL_SESSION_
MODE=ubuntu /usr/bin/gnome-session --sessio
  1589 tty2        Sl+    0:00 /usr/libexec/gnome
=ubuntu
  2183 pts/0        Ss     0:00 bash
  2951 pts/0        R+     0:00 ps a
```

표 6-2 STAT에 사용되는 문자의 의미

문자	의미	비고
R	실행 중(running)	
S	인터럽트가 가능한 대기(sleep) 상태	
T	작업 제어에 의해 정지된(stopped) 상태	
Z	좀비 프로세스(defunct)	
STIME	프로세스의 시작 날짜나 시간	
s	세션 리더 프로세스	BSD 형식
+	포그라운드 프로세스 그룹	
l(소문자 L)	멀티스레드	

02. 프로세스 관리 명령

■ 터미널에서 실행시킨 프로세스 상세 정보 출력하기: a 옵션과 u 옵션

```
user1@myubuntu:~$ ps au
USER          PID  %CPU  %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
user1        1586   0.0   0.1 172388  6036 tty2      Ssl+ 10:12   0:00 /usr/libexec/gdm-
wayland-session env GNOME_SHELL_SESSION_MODE=
user1        1589   0.0   0.3 231444 15432 tty2      Sl+   10:12   0:00 /usr/libexec/
gnome-session-binary --systemd --session=ubuntu
user1        2183   0.0   0.1  21036  5044 pts/0    Ss   10:12   0:00
user1        2957   0.0   0.0  22468  3492 pts/0    R+   11:12   0:00
```

표 6-3 ps au의 출력 정보

항목	의미
USER	사용자 계정 이름
%CPU	퍼센트로 표시한 CPU 사용량
%MEM	퍼센트로 표시한 물리적 메모리 사용량
VSZ	사용 중인 가상 메모리의 크기(KB)
RSS	사용 중인 물리적 메모리의 크기(KB)
START	프로세스 시작 시간

02. 프로세스 관리 명령

- 전체 프로세스 목록 출력하기(유닉스 옵션): - e 옵션과 - f 옵션
 - -e 옵션은 시스템에서 실행 중인 모든 프로세스를 출력

```
user1@myubuntu:~$ ps -e | more
```

PID	TTY	TIME	CMD
-----	-----	------	-----

1	?	00:00:01	systemd
---	---	----------	---------

2	?	00:00:00	kthreadd
---	---	----------	----------

3	?	00:00:00	rcu_gp
---	---	----------	--------

4	?	00:00:00	rcu_par_gp
---	---	----------	------------

6	?	00:00:00	kworker/0:0H-events_highpri
---	---	----------	-----------------------------

(생략)

02. 프로세스 관리 명령

- 전체 프로세스 목록 출력하기(유닉스 옵션): - e 옵션과 - f 옵션
 - 전체 프로세스의 더 자세한 정보를 확인하려면 -e 옵션과 -f 옵션을 함께 사용

```
user1@myubuntu:~$ ps -ef | more
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	12월18	?	00:00:01	/sbin/init splash
root	2	0	0	12월18	?	00:00:00	[kthreadd]
root	3	2	0	12월18	?	00:00:00	[rcu_gp]
root	4	2	0	12월18	?	00:00:00	[rcu_par_gp]
root	6	2	0	12월18	?	00:00:00	[kworker/0:0H-events_highpri]

(생략)

02. 프로세스 관리 명령

■ 전체 프로세스 목록 출력하기(BSD 옵션): ax 옵션

- 시스템에서 실행 중인 모든 프로세스를 출력

```
user1@myubuntu:~$ ps ax | more
```

PID	TTY	STAT	TIME	COMMAND
1	?	Ss	0:01	/sbin/init splash
2	?	S	0:00	[kthreadd]
3	?	I<	0:00	[rcu_gp]
4	?	I<	0:00	[rcu_par_gp]
6	?	I<	0:00	[kworker/0:0H-events_highpri]

(생략)

02. 프로세스 관리 명령

■ 전체 프로세스 목록 출력하기(BSD 옵션): aux 옵션

- aux 옵션은 -ef 옵션처럼 시스템에서 실행 중인 모든 프로세스에 대한 자세한 정보를 출력

```
user1@myubuntu:~$ ps aux | more
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.2	164520	10692	?	Ss	12월18	0:01	/sbin/init splash
root	2	0.0	0.0	0	0	?	S	12월18	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	12월18	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	12월18	0:00	[rcu_par_gp]

02. 프로세스 관리 명령

■ 특정 사용자의 프로세스 목록 출력하기: -u 옵션

```
user1@myubuntu:~$ ps -u user1
```

PID	TTY	TIME	CMD
1446	?	00:00:00	systemd
1448	?	00:00:00	(sd-pam)
1456	?	00:00:00	pipewire
1457	?	00:00:00	pipewire-media-
1458	?	00:00:00	pulseaudio
1463	?	00:00:00	dbus-daemon
1465	?	00:00:00	gvfsd
1470	?	00:00:00	gvfsd-fuse

(생략)

02. 프로세스 관리 명령

■ 특정 사용자의 프로세스 목록 출력하기: -u, -f 옵션

- 더욱 상세한 정보를 보고 싶으면 -f 옵션을 함께 사용(-u 옵션이 -f 뒤에 와야함)

```
user1@myubuntu:~$ ps -fu user1
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
user1	1446	1	0	00:00	?	00:00:00	/lib/systemd/systemd --user
user1	1448	1446	0	00:00	?	00:00:00	(sd-pam)
user1	1456	1446	0	00:00	?	00:00:00	/usr/bin/pipewire
user1	1457	1446	0	00:00	?	00:00:00	/usr/bin/pipewire-media-sess
user1	1458	1446	0	00:00	?	00:00:00	/usr/bin/pulseaudio --daemon

02. 프로세스 관리 명령

■ 특정 프로세스 정보 출력하기: -p 옵션

- -p 옵션과 함께 특정 PID를 지정하면 해당 프로세스의 정보를 출력

```
user1@myubuntu:~$ ps -p 1584
```

PID	TTY	TIME	CMD
-----	-----	------	-----

1584	pts/0	00:00:00	bash
------	-------	----------	------

```
user1@myubuntu:~$ ps -fp 1584
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
user1	1584	1583	0	00:00	pts/0	00:00:00	-bash

02. 프로세스 관리 명령

■ 특정 프로세스 정보 검색하기: ps와 grep 활용

- 'ps -ef | grep 명령'의 형태로 연결해서 사용

```
user1@myubuntu:~$ ps -ef | grep bash
user1      2096      2095  0 13:18 pts/0    00:00:00 -bash
user1      2111      2096  0 13:18 pts/0    00:00:00 grep --color=auto bash
```


02. 프로세스 관리 명령

■ 특정 프로세스 정보 검색하기: pgrep

pgrep

- **기능** 지정한 패턴과 일치하는 프로세스의 정보를 출력한다.
- **형식** pgrep [옵션] [패턴]
- **옵션**
 - x: 패턴과 정확히 일치하는 프로세스의 정보를 출력한다.
 - n: 패턴을 포함하고 있는 가장 최근 프로세스의 정보를 출력한다.
 - u 사용자 이름: 특정 사용자에 대한 모든 프로세스를 출력한다.
 - l: PID와 프로세스 이름을 출력한다.
 - t term: 특정 단말기와 관련된 프로세스의 정보를 출력한다.
- **사용 예** pgrep bash

02. 프로세스 관리 명령

■ pgrep 사용 예

```
user1@myubuntu:~$ pgrep -x bash
2096
```

bash 패턴을 지정하여 검색

```
user1@myubuntu:~$ pgrep -l bash
2096 bash
```

-l 옵션은 PID와 명령 이름을 출력

```
user1@myubuntu:~$ ps -fp $(pgrep -x bash)
UID          PID    PPID  C  STIME TTY          STAT      TIME CMD
user1        2096    2095  0  13:18 pts/0        00:00:00 -bash
```

pgrep과 ps 명령을 연결하면
자세한 정보 검색 가능

```
user1@myubuntu:~$ ps -fp $(pgrep -u user1 bash)
UID          PID    PPID  C  STIME TTY          STAT      TIME CMD
user1        2096    2095  0  13:18 pts/0        00:00:00 -bash
```

-u 옵션으로 사용자명을 지정
하면 해당 사용자의 프로세스
정보만 검색

02. 프로세스 관리 명령

■ 시그널

- 프로세스에 무언가 발생했음을 알리는 간단한 메시지
- 무엇이 발생했는지를 나타내는, 미리 정의된 상수를 사용
- 시그널은 번호로 구분되며 이름을 가지고 있음
- 시그널을 받은 프로세스는 기본적으로 종료

```
user1@myubuntu:~$ kill -l
```

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL	5) SIGTRAP
6) SIGABRT	7) SIGBUS	8) SIGFPE	9) SIGKILL	10) SIGUSR1
11) SIGSEGV	12) SIGUSR2	13) SIGPIPE	14) SIGALRM	15) SIGTERM
16) SIGSTKFLT	17) SIGCHLD	18) SIGCONT	19) SIGSTOP	20) SIGTSTP
21) SIGTTIN	22) SIGTTOU	23) SIGURG	24) SIGXCPU	25) SIGXFSZ
26) SIGVTALRM	27) SIGPROF	28) SIGWINCH	29) SIGIO	30) SIGPWR
31) SIGSYS	34) SIGRTMIN	35) SIGRTMIN+1	36) SIGRTMIN+2	37) SIGRTMIN+3
38) SIGRTMIN+4	39) SIGRTMIN+5	40) SIGRTMIN+6	41) SIGRTMIN+7	42) SIGRTMIN+8
43) SIGRTMIN+9	44) SIGRTMIN+10	45) SIGRTMIN+11	46) SIGRTMIN+12	47) SIGRTMIN+13
48) SIGRTMIN+14	49) SIGRTMIN+15	50) SIGRTMAX-14	51) SIGRTMAX-13	52) SIGRTMAX-12
53) SIGRTMAX-11	54) SIGRTMAX-10	55) SIGRTMAX-9	56) SIGRTMAX-8	57) SIGRTMAX-7
58) SIGRTMAX-6	59) SIGRTMAX-5	60) SIGRTMAX-4	61) SIGRTMAX-3	62) SIGRTMAX-2
63) SIGRTMAX-1	64) SIGRTMAX			

02. 프로세스 관리 명령

■ 시그널

표 6-4 주요 시그널

시그널	번호	기본 처리	의미
SIGHUP	1	종료	터미널과 연결이 끊겼을 때 발생한다.
SIGINT	2	종료	인터럽트로 사용자가 <code>Ctrl+C</code> 를 입력하면 발생한다.
SIGQUIT	3	종료, 코어덤프	종료 신호로 사용자가 <code>Ctrl+\</code> 을 입력하면 발생한다.
SIGKILL	9	종료	이 시그널을 받은 프로세스는 무시할 수 없으며 강제로 종료된다.
SIGALRM	14	종료	알람에 의해 발생한다.
SIGTERM	15	종료	kill 명령이 보내는 기본 시그널이다.

02. 프로세스 관리 명령

■ 프로세스 종료하기: kill

kill

- **기능** 지정한 시그널을 프로세스에게 보낸다.
- **형식** kill [-시그널] PID...
- **시그널** 2: 인터럽트 시그널을 보낸다(**Ctrl**+C).
9: 프로세스를 강제로 종료한다.
15: 프로세스와 관련된 파일들을 정리하고 종료한다. 종료되지 않는 프로세스가 있을 수 있다.
- **사용 예**
kill 1001
kill -15 1001
kill -9 1001

02. 프로세스 관리 명령

■ 프로세스 종료 예

```
user1@myubuntu:~$ man ps
```

```
PS(1)
```

```
User Commands
```

터미널 1번

```
PS(1)
```

```
NAME
```

```
ps - report a snapshot of the current processes.
```

```
(생략)
```

```
user1@myubuntu:~$ ps -fp $(pgrep -x man)
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
user1	2290	2172	0	14:48	pts/1	00:00:00	man ps

```
user1@myubuntu:~$ kill 2290
```

터미널 2번:

- man 프로세스 검색
- kill 명령으로 프로세스 강제 종료

```
Manual page ps(1) line 1 (press h for help or q to quit)종료됨
```

```
user1@myubuntu:~$
```

터미널 1번: man 프로세스 종료

02. 프로세스 관리 명령

■ 프로세스 종료하기: pkill

- PID가 아니라 프로세스의 명령 이름(CMD)으로 프로세스를 찾아 종료

```
user1@myubuntu:~$ ps -fp $(pgrep -x man)
```

UID	PID	PPID	C	STIME	TTY	STAT	TIME	CMD
user1	2394	2386	0	14:51	pts/2	S+	0:00	man pkill
user1	2411	2172	0	14:51	pts/1	S+	0:00	man pkill

```
user1@myubuntu:~$ pkill man
user1@myubuntu:~$ pgrep -x man
```

02. 프로세스 관리 명령

■ 프로세스 종료하기: killall

- pkill 명령처럼 프로세스의 명령 이름(CMD)으로 프로세스를 찾아 종료
- 해당 이름으로 실행 중인 모든 프로세스를 한 번에 종료
- 해당 프로세스를 소유하고 있어야 함

281p. 따라해보기 : 프로세스 찾아서 종료시키기

- ① 터미널 1에서 sh 프로세스를 실행
- ② 터미널 1에서 more 명령을 실행
- ③ 터미널 1에서 실행한 프로세스를 다른 터미널에서 검색: -t 옵션이나 a 옵션을 사용
- ④ 터미널 1에서 실행한 more 명령을 종료
 - 터미널 1에서 more 종료를 확인
- ⑤ -9 시그널로 터미널 1에서 실행한 sh 프로세스를 종료
 - 터미널 1에서 sh의 종료를 확인

02. 프로세스 관리 명령

■ 프로세스 관리 도구: top 명령

- 현재 실행 중인 프로세스의 정보를 주기적으로 출력
- 프로세스의 자세한 요약 정보를 상단에 출력하고 각 프로세스의 정보를 하단에 출력

표 6-5 top 명령의 출력 정보

항목	의미	항목	의미
PID	프로세스 ID	SHR	프로세스가 사용하는 공유 메모리의 크기
USER	사용자 계정	%CPU	퍼센트로 표시한 CPU 사용량
PR	우선순위	%MEM	퍼센트로 표시한 메모리 사용량
NI	Nice 값	TIME+	CPU 누적 이용 시간
VIRT	프로세스가 사용하는 가상 메모리의 크기	COMMAND	명령 이름
RES	프로세스가 사용하는 메모리의 크기		

02. 프로세스 관리 명령

```
user1@myubuntu:~$ top
```

```
top - 14:59:03 up 2:42, 2 users, load average: 0.00, 0.00, 0.00
```

```
Tasks: 299 total, 1 running, 298 sleeping, 0 stopped, 0 zombie
```

```
%Cpu(s): 0.0 us, 5.9 sy, 0.0 ni, 94.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
```

```
MiB Mem : 3893.3 total, 2711.7 free, 595.0 used, 586.6 buff/cache
```

```
MiB Swap: 923.2 total, 923.2 free, 0.0 use
```

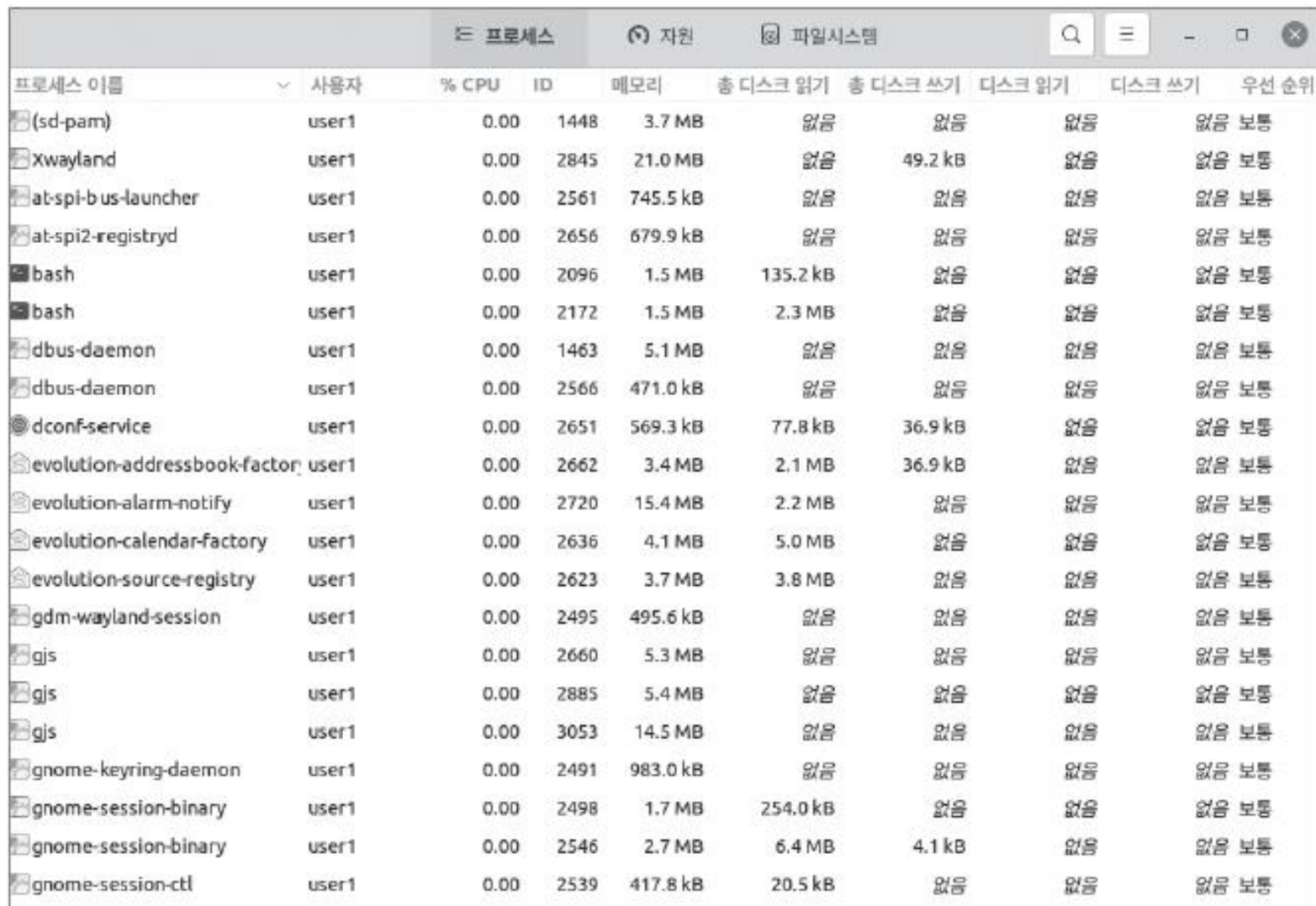
PID	USER	PR	NI	VIRT	RES	SHR	S	%C
1	root	20	0	164520	10736	7580	S	0
2	root	20	0	0	0	0	S	0
3	root	0	-20	0	0	0	I	0

표 6-6 top 명령의 내부 명령

내부 명령	기능
<code>Enter</code> , <code>Space Bar</code>	화면을 즉시 다시 출력한다.
<code>h, ?</code>	도움말 화면을 출력한다.
<code>k</code>	프로세스를 종료한다. 종료할 프로세스의 PID를 물어본다.
<code>n</code>	출력하는 프로세스의 개수를 바꾼다.
<code>u</code>	사용자에 따라 정렬하여 출력한다.
<code>M</code>	사용하는 메모리 크기에 따라 정렬하여 출력한다.
<code>p</code>	CPU 사용량에 따라 정렬하여 출력한다.
<code>q</code>	top 명령을 종료한다.

02. 프로세스 관리 명령

- 프로세스 관리 도구:
시스템 감시(GUI)
 - [프로그램 표시]-[시스템 감시]와 같은 순서로 동작



The screenshot shows the 'System Monitor' window with the 'Processes' tab selected. The window displays a list of running processes with columns for Process Name, User, % CPU, ID, Memory, Total Disk Read, Total Disk Write, Disk Read, Disk Write, and Priority. The processes listed include (sd-pam), Xwayland, at-spi-bus-launcher, at-spi2-registryd, bash, dbus-daemon, dconf-service, evolution-addressbook-factory, evolution-alarm-notify, evolution-calendar-factory, evolution-source-registry, gdm-wayland-session, gjs, gnome-keyring-daemon, gnome-session-binary, and gnome-session-ctl.

프로세스 이름	사용자	% CPU	ID	메모리	총 디스크 읽기	총 디스크 쓰기	디스크 읽기	디스크 쓰기	우선 순위
(sd-pam)	user1	0.00	1448	3.7 MB	없음	없음	없음	없음	보통
Xwayland	user1	0.00	2845	21.0 MB	없음	49.2 kB	없음	없음	보통
at-spi-bus-launcher	user1	0.00	2561	745.5 kB	없음	없음	없음	없음	보통
at-spi2-registryd	user1	0.00	2656	679.9 kB	없음	없음	없음	없음	보통
bash	user1	0.00	2096	1.5 MB	135.2 kB	없음	없음	없음	보통
bash	user1	0.00	2172	1.5 MB	2.3 MB	없음	없음	없음	보통
dbus-daemon	user1	0.00	1463	5.1 MB	없음	없음	없음	없음	보통
dbus-daemon	user1	0.00	2566	471.0 kB	없음	없음	없음	없음	보통
dconf-service	user1	0.00	2651	569.3 kB	77.8 kB	36.9 kB	없음	없음	보통
evolution-addressbook-factory	user1	0.00	2662	3.4 MB	2.1 MB	36.9 kB	없음	없음	보통
evolution-alarm-notify	user1	0.00	2720	15.4 MB	2.2 MB	없음	없음	없음	보통
evolution-calendar-factory	user1	0.00	2636	4.1 MB	5.0 MB	없음	없음	없음	보통
evolution-source-registry	user1	0.00	2623	3.7 MB	3.8 MB	없음	없음	없음	보통
gdm-wayland-session	user1	0.00	2495	495.6 kB	없음	없음	없음	없음	보통
gjs	user1	0.00	2660	5.3 MB	없음	없음	없음	없음	보통
gjs	user1	0.00	2885	5.4 MB	없음	없음	없음	없음	보통
gjs	user1	0.00	3053	14.5 MB	없음	없음	없음	없음	보통
gnome-keyring-daemon	user1	0.00	2491	983.0 kB	없음	없음	없음	없음	보통
gnome-session-binary	user1	0.00	2498	1.7 MB	254.0 kB	없음	없음	없음	보통
gnome-session-binary	user1	0.00	2546	2.7 MB	6.4 MB	4.1 kB	없음	없음	보통
gnome-session-ctl	user1	0.00	2539	417.8 kB	20.5 kB	없음	없음	없음	보통

그림 6-3 시스템 감시 화면

03 포그라운드, 백그라운드 프로세스와 작업제어

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 작업 제어

- 한 터미널에서 동시에 여러 프로세스를 실행하고 관리
- C셸에서 처음 개발되었고 4.1BSD 커널에 포함
- 본셸에도 도입되어 대부분의 유닉스 셸과 리눅스 셸에 포함하여 제공
- 작업 제어 도구가 관리하는 프로세스를 작업이라고 함

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 포그라운드 작업

- 사용자가 입력한 명령이 실행되어 결과가 출력될 때까지 기다리는 방식으로 처리되는 프로세스를 포그라운드 프로세스
- 작업 제어에서는 이를 포그라운드 작업이라고 함

```
user1@myubuntu:~$ sleep 100
```

→ 포그라운드 작업

→ sleep 명령이 끝날 때까지 기다려야 한다.

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 백그라운드 작업

- 백그라운드 방식으로 명령을 실행하면 명령의 처리가 끝나는 것과 관계없이 곧바로 프롬프트가 출력되어 사용자가 다른 작업을 계속할 수 있음
- 한 터미널에서 여러 개의 프로세스를 동시에 실행 가능
- 백그라운드 방식으로 처리되는 프로세스를 백그라운드 프로세스라고 함
- 작업 제어에서는 이를 백그라운드 작업이라고 함
- 명령을 백그라운드로 실행하려면 명령의 마지막에 &(앰퍼샌드) 기호를 추가
- 백그라운드로 작업을 실행하면 프롬프트가 바로 나옴

```
user1@myubuntu:~$ sleep 100 & → 백그라운드 작업
```

```
user1@myubuntu:~$ → 프롬프트가 바로 나와 다른 명령을 실행시킬 수 있다.
```

```
user1@myubuntu:~$ find / -name passwd > pw.dat 2>&1 & → pw.dat에 결과와 오류를 저장한다.
```


03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 작업 제어

- 작업 전환, 작업 일시 중지, 작업 종료를 의미
- 작업 전환: 포그라운드 작업 <-> 백그라운드 작업 간 전환
- 작업 일시 중지: 작업을 잠시 중단
- 작업 종료: 프로세스 종료처럼 작업 종료

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 작업 목록 보기: jobs

jobs

- **기능** 백그라운드 작업을 모두 보여준다. 특정 작업 번호를 지정하면 해당 작업의 정보만 보여준다.
- **형식** jobs [%작업 번호]
- **%작업번호** %번호: 해당 번호의 작업 정보를 출력한다.
%+ 또는 %+: 작업 순서가 +인 작업 정보를 출력한다.
%-: 작업 순서가 -인 작업 정보를 출력한다.
- **사용 예** jobs %1
jobs

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ jobs 명령 실행 예

```
user1@myubuntu:~$ jobs
[1]-  실행중                sleep 100 &
[2]+  실행중                find / -name passwd > pw.dat 2>&1 &
```

표 6-7 jobs 명령의 출력 정보

항목	출력 예	의미
작업 번호	[1]	작업 번호로서 백그라운드로 실행할 때마다 순차적으로 증가한다([1], [2], [3],...).
작업 순서	+	작업 순서를 표시한다. <ul style="list-style-type: none">• +: 가장 최근에 접근한 작업• -: + 작업 바로 전에 접근한 작업• 공백: 그 외의 작업
상태	실행중	작업 상태를 표시한다. <ul style="list-style-type: none">• 실행중: 현재 실행되고 있다.• 완료: 작업이 정상적으로 종료되었다.• 종료됨: 작업이 비정상적으로 종료되었다.• 멈춤: 작업이 잠시 중단되었다.
명령	sleep 100 &	백그라운드로 실행 중인 명령이다.

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 작업 전환하기

표 6-8 작업 전환 명령

명령	기능
<code>Ctrl+z</code> 또는 <code>stop %작업 번호</code>	포그라운드 작업을 중지한다(종료하는 것이 아니라 잠시 중단하는 것이다).
<code>bg %작업 번호</code>	작업 번호가 지시하는 작업을 백그라운드 작업으로 전환한다.
<code>fg %작업 번호</code>	작업 번호가 지시하는 작업을 포그라운드 작업으로 전환한다.

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 포그라운드 -> 백그라운드로 작업 전환

- 작업 중지(ctrl+z) -> bg %작업번호

```
user1@myubuntu:~$ jobs
```

```
user1@myubuntu:~$ sleep 100
```

```
^Z
```

```
[1]+  멈춤                sleep 100
```

```
user1@myubuntu:~$ bg %1
```

```
[1]+  sleep 100 &
```

```
user1@myubuntu:~$ jobs
```

```
[1]+  실행중                sleep 100 &
```

→ 백그라운드 작업이 없다.

→ 포그라운드로 실행한다.

→ **Ctrl**+z로 일시 중지한다.

→ 일시 중지된 상태다.

→ 백그라운드로 전환한다.

→ 백그라운드로 실행 중이다.

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 백그라운드 -> 포그라운드로 작업 전환

- fg %작업번호

```
user1@myubuntu:~$ jobs
```

```
[1]+  실행중
```

```
sleep 100 &
```

```
user1@myubuntu:~$ fg
```

→ 포그라운드로 전환한다.

```
sleep 100
```

→ 포그라운드로 실행 중이다.



03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 작업 종료하기: ctrl+c

- 포그라운드 작업은 ctrl+c를 입력하면 대부분 종료
- 또는 PID를 검색하여 kill 명령으로 강제 종료

```
user1@myubuntu:~$ sleep 100
```

```
^C
```

```
user1@myubuntu:~$
```

→ 포그라운드로 실행 중이다.

→ 강제 종료한다.

- 백그라운드 작업은 kill 명령으로 강제 종료: kill %작업번호

```
user1@myubuntu:~$ sleep 100&
```

```
[1] 53864
```

```
user1@myubuntu:~$ kill %1
```

```
user1@myubuntu:~$
```

```
[1]+  종료됨
```

```
sleep 100
```

→ 백그라운드로 실행 중이다.

→ 강제 종료한다.

→ 종료 메시지가 출력된다.

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ 로그아웃 후에도 백그라운드 작업 계속 실행하기

- 백그라운드 작업을 실행한 터미널이 종료되거나 사용자가 로그아웃하면 실행 중이던 백그라운드 작업도 함께 종료
- 로그아웃한 다음에도 작업이 완료될 때까지 백그라운드작업을 실행해야 할 경우에 nohup 명령을 사용

nohup

- **기능** 로그아웃한 후에도 백그라운드 작업을 계속 실행한다.
- **형식** nohup 명령&

03. 포그라운드, 백그라운드 프로세스와 작업 제어

■ nohup 사용 예

```
user1@myubuntu:~$ nohup find / -name passwd &  
[1] 53629
```

user1@myubuntu:~\$ nohup: 입력 무시 및 'nohup.out' 에 출력 추가

기본적으로 nohup.out 파일에
실행 결과 저장

```
user1@myubuntu:~$ more nohup.out  
/usr/bin/passwd  
/usr/share/doc/passwd  
/usr/share/bash-completion/completions/passwd  
/usr/share/lintian/overrides/passwd  
find: '/lost+found': 허가 거부  
find: '/run/gdm3': 허가 거부  
find: '/run/udisks2': 허가 거부  
find: '/run/cups/certs': 허가 거부  
(생략)
```

```
user1@myubuntu:~$ nohup find / -name passwd > pw.dat 2>&1 &  
[1] 53878  
user1@myubuntu:~$ exit
```

실행 결과를 저장할 파일 지정

```
user1@myubuntu:~$ more pw.dat  
nohup: 입력 무시  
/usr/bin/passwd  
/usr/share/doc/passwd  
/usr/share/bash-completion/completions/passwd  
/usr/share/lintian/overrides/passwd  
find: '/lost+found': 허가 거부  
find: '/run/gdm3': 허가 거부
```

292p. 따라해보기 : 작업 관리하기

① 백그라운드 작업 생성

- ① 파일을 하나 복사하고 vi 명령을 실행
- ② ctrl+z를 실행하여 vi 작업을 일시 정지
- ③ vi 작업이 일시 정지되면 Stopped로 표시됨

```
user1@myubuntu:~/linux_ex/ch6$ vi hosts
```

```
[1]+  멈춤 vi hosts
```

② jobs 명령으로 백그라운드 작업을 확인

- ③ 정지 중인 작업을 fg 명령을 사용하여 복구
- ④ q!로 vi를 종료

04 작업 예약

04. 작업 예약

■ 정해진 시간에 한 번 실행: at 명령

at

- **기능** 예약한 명령을 정해진 시간에 실행한다.
- **형식** at [옵션] [시간]
- **옵션**
 - l: 현재 실행 대기 중인 명령의 전체 목록을 출력한다(atq 명령과 동일).
 - r 작업 번호: 현재 실행 대기 중인 명령 중 해당 작업 번호를 삭제한다(atrm과 동일).
 - m: 출력 결과가 없더라도 작업이 완료되면 사용자에게 메일로 알려준다.
 - f 파일: 표준 입력 대신 실행할 명령을 파일로 지정한다.
- **사용 예**
 - at -m 0730 tomorrow
 - at 10:00 pm
 - at 8:15 am May 30

04. 작업 예약

■ at 명령 설치하기

```
user1@myubuntu:~$ sudo apt install at
```

[sudo] user1의 암호:

패키지 목록을 읽는 중입니다... **at 설치**

의존성 트리를 만드는 중입니다... 완료

상태 정보를 읽는 중입니다... 완료



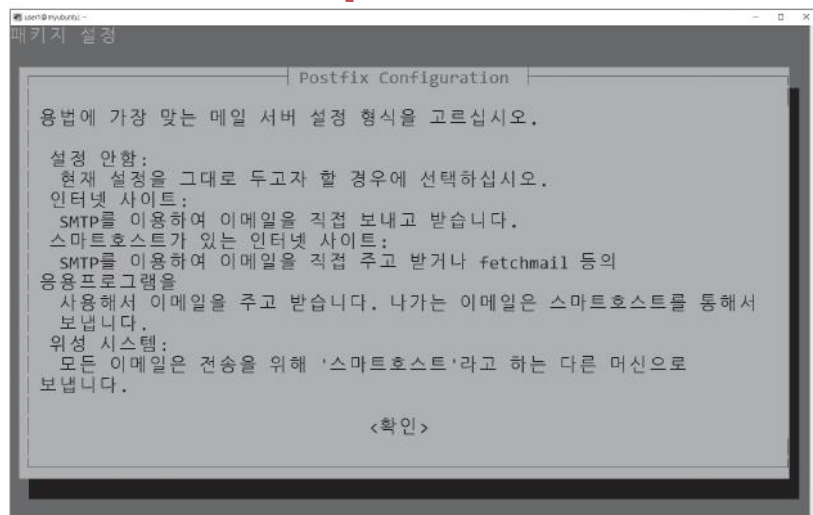
메일 설치

```
user1@myubuntu:~$ sudo apt install mailutils
```

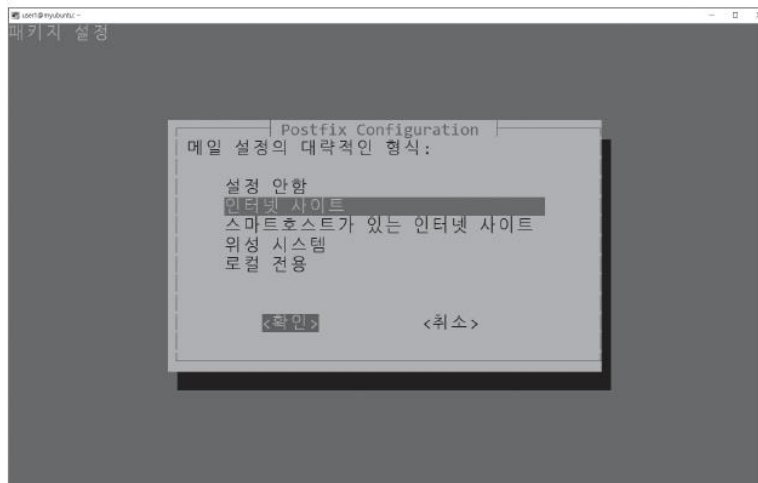
패키지 목록을 읽는 중입니다... 완료

의존성 트리를 만드는 중입니다... 완료

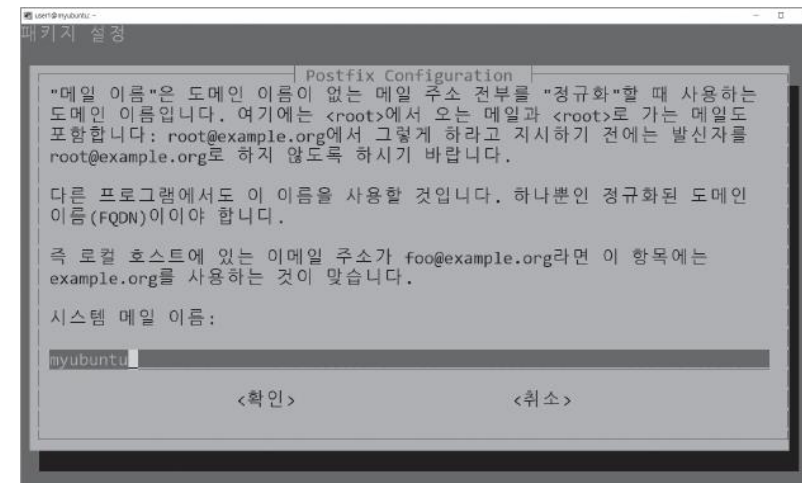
상태 정보를 읽는 중입니다... 완료



(a) 메일 서버 설정 형식 설명



(b) 메일 서버 설정 형식 선택



(c) 시스템 메일 이름 설정(myubuntu)

04. 작업 예약

■ at 명령 설정하기

```
user1@myubuntu:~/linux_ex/ch6$ at 17:00
warning: commands will be executed using /bin/sh
at>
```

시간을 지정하여 at 명령
실행

■ 시간 설정 방식

- at 4pm + 3 days: 지금부터 3일 후 오후 4시에 작업을 수행
- • at 10am Jul 31: 7월 31일 오전 10시에 작업을 수행
- • at 1am tomorrow: 내일 오전 1시에 작업을 수행
- • at 10:00am today: 오늘 오전 10시에 작업을 수행

04. 작업 예약

■ at 명령 설정 예

```
user1@myubuntu:~/linux_ex/ch6$ at 04:30 pm
warning: commands will be executed using /bin/sh
at> /usr/bin/ls -l ~user1 > ~user1/at.out
at> <EOT>
job 2 at Sun Dec 19 16:50:00 2021
```

→ 시간을 지정한다.

→ 실행할 명령을 지정한다.

→ **Ctrl**+d 입력으로 종료한다.

→ 작업 예약을 완료한다.

04. 작업 예약

■ at 작업 파일 확인하기

- at로 생성된 작업 파일은 /var/spool/cron/atjobs 디렉터리에 저장
- 작업 번호로 파일이 생성
- atjobs 디렉터리의 내용은 daemon 그룹의 사용자만 확인할 수 있으므로 sudo 명령을 사용해야 함

```
user1@myubuntu:~/linux_ex/ch6$ ls -l /var/spool/cron/atjobs
ls: '/var/spool/cron/atjobs' 디렉터리를 열 수 없음: 허가 거부
user1@myubuntu:~/linux_ex/ch6$ sudo ls -l /var/spool/cron/atjobs
합계 4
-rwx----- 1 user1 daemon 2982 12월 19 16:38 a0000201a10c56
```


04. 작업 예약

■ at 작업 목록 확인하기: -l 옵션

```
user1@myubuntu:~/linux_ex/ch6$ at -l
2          Sun Dec 19 16:50:00 2021 a user1
```

■ at 작업 목록 확인하기: atq 명령

atq

- **기능** 현재 사용자의 등록된 작업 목록을 보여준다. 슈퍼유저일 경우 모든 사용자의 작업 목록을 보여준다.
- **형식** atq

```
user1@myubuntu:~/linux_ex/ch6$ atq
2          Sun Dec 19 16:50:00 2021 a user1
```

04. 작업 예약

■ at 작업 삭제하기: -d 옵션과 atrm 명령

```
user1@myubuntu:~/linux_ex/ch6$ at 1am tomorrow
warning: commands will be executed using /bin/sh
at> /usr/bin/ls > ~user1/at1.out
at> <EOT>
job 3 at Mon Dec 20 01:00:00 2021
user1@myubuntu:~/linux_ex/ch6$ at 10pm today
warning: commands will be executed using /bin/sh
at> /usr/bin/ls /tmp > ~user1/at2.out
at> <EOT>
job 4 at Sun Dec 19 22:00:00 2021
```

작업 2개 예약



atrm

- **기능** 지정된 작업 번호의 작업을 삭제한다.
- **형식** atrm 작업 번호

```
user1@myubuntu:~/linux_ex/ch6$ atq
3      Mon Dec 20 01:00:00 2021 a user1
4      Sun Dec 19 22:00:00 2021 a user1
2      Sun Dec 19 16:50:00 2021 a user1
```

작업 목록 확인



```
user1@myubuntu:~/linux_ex/ch6$ at -d 3
user1@myubuntu:~/linux_ex/ch6$ atrm 4
user1@myubuntu:~/linux_ex/ch6$ atrm 2
user1@myubuntu:~/linux_ex/ch6$ atq
user1@myubuntu:~/linux_ex/ch6$
```

작업 삭제

04. 작업 예약

■ at 명령 사용 제한하기

- /etc/at.allow: at 명령의 사용이 허용된 사용자들 저장, 필요할 때 만들어야함
- /etc/at.deny: at 명령의 사용이 금지된 사용자들 저장, 기본적으로 파일이 있음
- 적용 기준
 - /etc/at.allow 파일이 있으면 이 파일에 지정된 사용자만 at 명령을 사용 가능. /etc/at.deny 파일 무시
 - /etc/at.allow 파일이 없으면 /etc/at.deny 파일에 지정된 사용자를 제외한 모든 사용자가 at 명령을 사용 가능
 - 만약 두 파일이 모두 없다면 root만 at 명령을 사용 가능
 - 사용자가 두 파일 모두에 속해 있다면 at 명령 사용 가능
 - /etc/at.deny를 빈 파일로 두면 모든 사용자가 at 명령을 사용 가능(초기 설정)

04. 작업 예약

■ 정해진 시간에 반복 실행

crontab

- **기능** 사용자의 crontab 파일을 관리한다.
- **형식** `crontab [-u 사용자ID] [옵션] [파일명]`
- **옵션**
 - e: 사용자의 crontab 파일을 편집한다.
 - l: crontab 파일의 목록을 출력한다.
 - r: crontab 파일을 삭제한다.
- **사용 예**
 - `crontab -l`
 - `crontab -u user1 -e`
 - `crontab -r`

04. 작업 예약

■ crontab 파일 형식

- crontab 명령으로 관리하는 파일은 사용자별로 생성
- crontab 파일에는 여러 개의 작업을 저장할 수 있으며 한 행에 하나의 작업을 설정
- crontab 파일의 한 행은 다음과 같이 여섯 항목으로 구성

분(0~59)	시(0~23)	일(1~31)	월(1~12)	요일(0~6)	작업 내용
---------	---------	---------	---------	---------	-------

- 요일은 0이 일요일, 1이 월요일, 6이 토요일을 의미
- 각 항목은 공백문자로 구분
- 항목의 값이 *이면 해당 항목의 모든 값을 의미

04. 작업 예약

■ crontab 항목 설정 예



그림 6-5 crontab 파일의 형식

- ④번의 월 항목은 매월을 의미
- ⑤번의 요일은 모든 요일을 의미
- 따라서 이 명령은 매월 1일 23시 30분에 지정한 작업을 실행하라는 뜻

04. 작업 예약

■ crontab 파일 생성하고 편집하기: crontab -e

- crontab 편집기로는 기본적으로 VISUAL 또는 EDITOR 환경 변수에 지정된 편집기를 사용

```
user1@myubuntu:~/linux_ex/ch6$ EDITOR=vi;export EDITOR
```

- 작업 생성 예

```
user1@myubuntu:~/linux_ex/ch6$ crontab -e
(생략)
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
30 23 1 * * /usr/bin/ls -l ~user1 > ~user1/cron.out
~
~
:wq
```

04. 작업 예약

■ crontab 파일 확인하기

- /var/spool/cron/crontabs 디렉터리에 사용자 이름으로 생성

```
user1@myubuntu:~/linux_ex/ch6$ sudo ls -l /var/spool/cron/crontabs
합계 4
-rw----- 1 user1 crontab 1142 12월 19 16:49 user1
```

■ crontab 파일 내용 확인하기: crontab -l

```
user1@myubuntu:~/linux_ex/ch6$ crontab -l
(생략)
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow    command
30 23 1 * * /usr/bin/ls -l ~user1 > ~user1/cron.out
```


04. 작업 예약

■ crontab 파일 삭제하기: crontab -r

- 본인의 작업 삭제

```
user1@myubuntu:~/linux_ex/ch6$ crontab -r
user1@myubuntu:~/linux_ex/ch6$ crontab -l
no crontab for user1
```

- 시스템 관리자가 특정 사용자의 crontab을 삭제하려면 'crontab -u user1 -r'과 같이 삭제하려는 사용자 ID를 지정

04. 작업 예약

■ crontab 명령 사용 제한하기: /etc/cron.allow와 /etc/cron.deny

- /etc/cron.allow 파일이 있으면 이 파일에 지정된 사용자만 crontab 명령을 사용 가능
- /etc/cron.allow 파일이 없고 /etc/cron.deny 파일이 있으면 이 파일에 사용자 계정이 없어야 crontab 명령을 사용 가능
- /etc/cron.allow 파일과 /etc/cron.deny 파일이 모두 없으면 시스템 설정에 따라 시스템 관리자만 crontab 명령을 사용할 수도 있고 모든 사용자가 사용할 수도 있음
- 만일 두 파일이 모두 있다면 /etc/cron.allow 파일의 내용이 적용되고, /etc/cron.deny는 무시

305p. 따라해보기 : crontab 설정하기

- ① `crontab -e` 명령으로 crontab 파일을 편집: 실습시간 내에 확인이 가능하도록 날짜와 시간 설정
- ② `crontab -l` 명령으로 설정 내용을 확인
- ③ crontab에 설정한 시간이 경과한 후 작업 결과를 확인

Thank You !