원리를 알면 IT가 맛있다
IT COOKBOOK

IT CookBook

CentOS 리눅스

시스템 & 네트워크

[강의교안 이용 안내]

CentOS 리눅스

시스템 & 네트워크

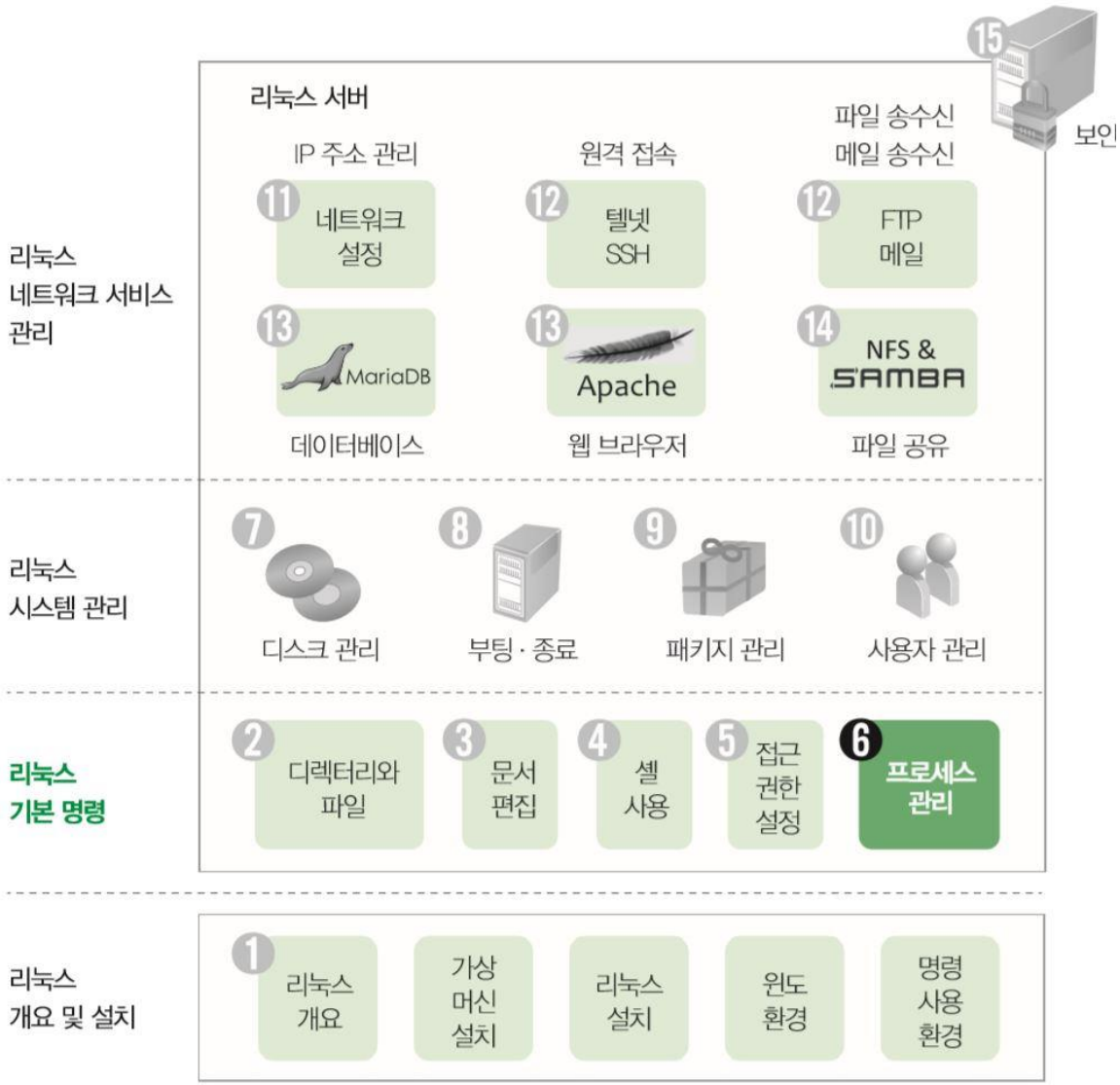
Chapter 06. 프로세스 관리

목차

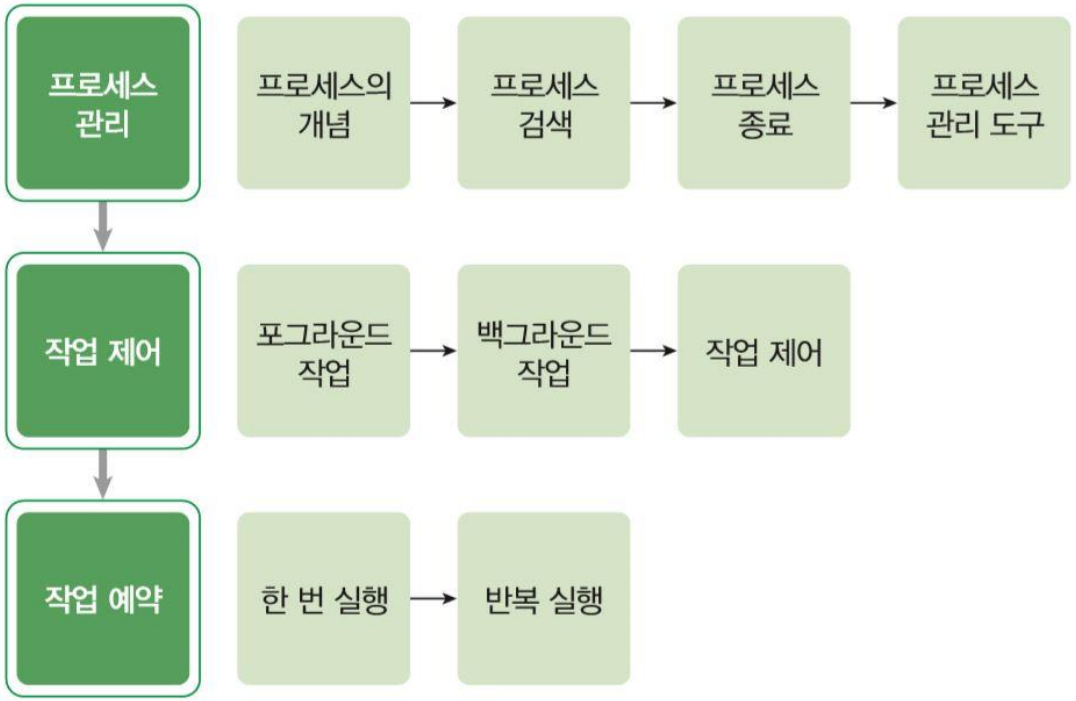
학습목표

- 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.목차


15
보안
리눅스 서버
IP 주소 관리
원격 접속
파일 송수신
메일 송수신
리눅스
네트워크 서비스
관리
11 네트워크 설정
12 텔넷 SSH
12 FTP 메일
13 MariaDB
13 Apache
14 NFS & SAMBA
데이터베이스
웹 브라우저
파일 공유
리눅스
시스템 관리
7 디스크 관리
8 부팅 · 종료
9 패키지 관리
10 사용자 관리
리눅스
기본 명령
2 디렉터리와 파일
3 문서 편집
4 셸 사용
5 접근 권한 설정
6 프로세스 관리
리눅스
개요 및 설치
1 리눅스 개요
가상 머신 설치
리눅스 설치
윈도 환경
명령 사용 환경

00.목차


프로세스 관리
프로세스의 개념
프로세스 검색
프로세스 종료
프로세스 관리 도구
작업 제어
포그라운드 작업
백그라운드 작업
작업 제어
작업 예약
한 번 실행
반복 실행

01.프로세스의 개념

프로세스의 부모-자식 관계

- 프로세스 : 현재 시스템에서 실행 중인 프로그램
- 모든 프로세스는 부모-자식 관계를 가지고 있음
- 부모 프로세스는 자식 프로세스를 생성하고, 자식 프로세스는 또 다른 자식 프로세스를 만들 수 있음
- systemd와 kthreadd 프로세스를 제외하면 모든 프로세스는 부모 프로세스를 가지고 있음

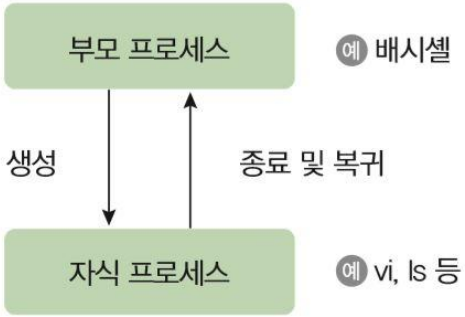


그림 6-1 부모 프로세스와 자식 프로세스의 관계

- 자식 프로세스는 할 일이 끝나면 부모 프로세스에 결과를 돌려주고 종료
  - 사용자가 vi를 실행하여 셸이 vi 프로세스를 생성할 경우, 셸은 부모 프로세스가 되고 vi는 자식 프로세스가 됨
  - 사용자가 vi를 종료하면 다시 부모 프로세스인 셸로 돌아감

01.프로세스의 개념

프로세스의 번호

- PID
  - 각 프로세스는 고유한 번호를 가지고 있음 그 번호를 PID 라고 함
  - PID는 1번부터 시작하고 프로세스가 실행되면서 하나씩 증가하여 부여

프로세스의 종류

- 데몬 프로세스
  - 특정 서비스를 제공하기 위해 존재하며 리눅스 커널에 의해 실행
- 고아 프로세스
  - 자식 프로세스가 아직 실행 중인데 부모 프로세스가 먼저 종료된 자식 프로세스는 고아 프로세스
  - 1번 프로세스가 고아 프로세스의 새로운 부모 프로세스가 되어 고아 프로세스의 작업 종료 지원
- 좀비 프로세스
  - 자식 프로세스가 실행을 종료했는데도 프로세스 테이블 목록에 남아 있는 경우
  - 좀비 프로세스는 프로세스 목록에 defunct 프로세스라고 나오기도 함
  - 좀비 프로세스가 증가하면 프로세스 테이블의 용량이 부족해서 일반 프로세스가 실행되지 않을 수도 있음

02.프로세스 관리 명령

프로세스 목록 확인

- ps 명령 : 실행 중인 프로세스의 목록을 보는 명령

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 사용자에 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

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

02.프로세스 관리 명령

프로세스 목록 확인

현재 단말기의 프로세스 목록 출력하기 : ps

- ps 명령을 옵션 없이 사용하면 현재 셸이나 터미널에서 실행한 사용자 프로세스의 정보를 출력

```
[user1@localhost ~]$ ps
   PID TTY          TIME CMD
  3141 pts/1    00:00:00 bash
  6990 pts/1    00:00:00 ps
```

PID : 프로세스 번호
TTY : 현재 터미널 번호
TIME : 해당 프로세스가 사용한 CPU 시간의 양
CMD : 프로세스가 실행 중인 명령

프로세스 상세 정보 출력하기 : -f 옵션

- -f 옵션은 프로세스의 상세한 정보를 출력
- PPID와 터미널 번호, 시작 시간 등의 정보가 추가로 출력

```
[user1@localhost ~]$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
user1       3141    3136  0 11:02 pts/1    00:00:00 -bash
user1       8246    3141  0 20:05 pts/1    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

프로세스 목록 확인

- 터미널에서 실행시킨 프로세스 정보 출력하기 : a 옵션
  - a 옵션은 터미널에서 실행시킨 프로세스의 정보를 출력

```
[user1@localhost ~]$ ps a
   PID TTY      STAT   TIME COMMAND
  1108 tty1     Ssl+   0:00 /usr/libexec/gdm-wayland-session gnome-session --aut
ostart /usr/share/gdm/greeter/autostart
  1172 tty1     Sl+    0:00 /usr/libexec/gnome-session-binary --autostart /usr/s
hare/gdm/greeter/autostart
  1232 tty1     Sl+    0:21 /usr/bin/gnome-shell
(생략)
  2727 tty2     Sl     0:00 /usr/libexec/ibus-engine-hangul --ibus
  3089 pts/0    Ss+    0:00 bash
  3141 pts/1    Ss     0:00 -bash
  8276 pts/1    R+     0:00 ps a
```

  - 출력 내용 중 STAT는 프로세스의 상태를 나타냄

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	

문자	의미	비고
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

프로세스 목록 확인

- 터미널에서 실행시킨 프로세스 상세 정보 출력하기 : a 옵션과 u 옵션
  - a 옵션과 u 옵션을 함께 사용하면 터미널에서 실행한 프로세스의 상세 정보를 출력
  - 단순히 a 옵션이나 -f 옵션을 사용한 것과 비교해보면 CPU와 메모리 사용량 등 추가적인 정보가 출력

```
[user1@localhost ~]$ ps au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
gdm         1108  0.0  0.5 315704  9548 tty1     Ssl+ 10:58   0:00 /usr/libexec/g
dm-wayland-session gnome-session --autostart /usr/share/gdm/greeter/autostart
gdm         1172  0.0  0.7 560144 14400 tty1     Sl+  10:58   0:00 /usr/libexec/g
nome-session-binary --autostart /usr/share/gdm/greeter/autostart
gdm         1232  0.0 12.3 2918928 227524 tty1   Sl+  10:58   0:21 /usr/bin/gnome
-shell
(생략)
user1       2727  0.0  0.5 281896  9536 tty2     Sl   10:59   0:00 /usr/libexec/i
bus-engine-hangul --ibus
user1       3089  0.0  0.2  22720  4932 pts/0    Ss+  11:01   0:00 bash
user1       3141  0.0  0.2  22848  5120 pts/1    Ss   11:02   0:00 -bash
user1       8341  0.0  0.2  55608  3904 pts/1    R+   20:13   0:00 ps au
```

표 6-3 ps au의 출력 정보

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

02.프로세스 관리 명령

프로세스 목록 확인

- 전체 프로세스 목록 출력하기(유닉스 옵션) : - e 옵션과 - f 옵션
  - -e 옵션은 시스템에서 실행 중인 모든 프로세스를 출력
  - -e 옵션을 실행하면 출력 내용이 위로 스크롤 되어 프로세스 목록을 제대로 확인하기가 어렵기 때문에 |(파이프)와 more나 less 명령을 함께 사용

```
[user1@localhost ~]$ ps -e | more
   PID TTY          TIME CMD
     1 ?        00:00:06 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-kblockd
(생략)
    21 ?        00:00:00 kcompactd0
    22 ?        00:00:00 ksmd
    23 ?        00:00:00 khugepaged
    24 ?        00:00:00 crypto
    25 ?        00:00:00 kintegrityd
--More--
```

02.프로세스 관리 명령

프로세스 목록 확인

- 전체 프로세스 목록 출력하기(유닉스 옵션) : - e 옵션과 - f 옵션
  - 전체 프로세스의 더 자세한 정보를 확인하려면 -e 옵션과 -f 옵션을 함께 사용해야 함
  - 두 옵션을 함께 사용할 때는 -ef를 입력

```
[user1@localhost ~]$ ps -ef | more
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 10:57 ?        00:00:06 /usr/lib/systemd/systemd --swi
tched-root --system --deserialize 17
root           2       0  0 10:57 ?        00:00:00 [kthreadd]
root           3       2  0 10:57 ?        00:00:00 [rcu_gp]
root           4       2  0 10:57 ?        00:00:00 [rcu_par_gp]
root           6       2  0 10:57 ?        00:00:00 [kworker/0:0H-kblockd]
(생략)
root          21       2  0 10:57 ?        00:00:00 [kcompactd0]
root          22       2  0 10:57 ?        00:00:00 [ksmd]
root          23       2  0 10:57 ?        00:00:00 [khugepaged]
root          24       2  0 10:57 ?        00:00:00 [crypto]
--More--
```

02.프로세스 관리 명령

프로세스 목록 확인

- 전체 프로세스 목록 출력하기(BSD 옵션) : ax 옵션과 aux 옵션
  - ax 옵션은 –e 옵션과 마찬가지로 시스템에서 실행 중인 모든 프로세스를 출력

```
[user1@localhost ~]$ ps ax | more
   PID TTY      STAT   TIME COMMAND
     1 ?        Ss     0:06 /usr/lib/systemd/systemd --switched-root --system --
deserialize 17
     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-kblockd]
     8 ?        I<     0:00 [mm_percpu_wq]
(생략)
    21 ?        S      0:00 [kcompactd0]
    22 ?        SN     0:00 [ksmd]
    23 ?        SN     0:00 [khugepaged]
    24 ?        I<     0:00 [crypto]
--More--
```

02.프로세스 관리 명령

프로세스 목록 확인

- 전체 프로세스 목록 출력하기(BSD 옵션) : ax 옵션과 aux 옵션
  - aux 옵션은 –ef 옵션처럼 시스템에서 실행 중인 모든 프로세스에 대한 자세한 정보를 출력

```
[user1@localhost ~]$ ps aux | more
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.7 179072 13940 ?        Ss   10:57   0:06 /usr/lib/syste
md/systemd --switched-root --system --deserialize 17
root           2  0.0  0.0      0     0 ?        S    10:57   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   10:57   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   10:57   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   10:57   0:00 [kworker/0:0H-
kblockd]
(생략)
root          21  0.0  0.0      0     0 ?        S    10:57   0:00 [kcompactd0]
root          22  0.0  0.0      0     0 ?        SN   10:57   0:00 [ksmd]
root          23  0.0  0.0      0     0 ?        SN   10:57   0:00 [khugepaged]
--More--
```

02.프로세스 관리 명령

프로세스 목록 확인

- 특정 사용자의 프로세스 목록 출력하기 : -u
  - -u 옵션을 사용하면 특정 사용자가 실행한 프로세스의 목록을 확인할 수 있음

```
[user1@localhost ~]$ ps -u user1
   PID TTY          TIME CMD
  2262 ?        00:00:00 systemd
  2267 ?        00:00:00 (sd-pam)
  2273 ?        00:00:01 pulseaudio
  2279 ?        00:00:00 gnome-keyring-d
  2284 ?        00:00:00 dbus-daemon
  2291 tty2     00:00:00 gdm-wayland-ses
  2298 tty2     00:00:00 gnome-session-b
  2360 tty2     00:01:17 gnome-shell
  2375 ?        00:00:00 gvfsd
(생략)
```

02.프로세스 관리 명령

프로세스 목록 확인

- 특정 사용자의 프로세스 목록 출력하기 : -u
  - 더욱 상세한 정보를 보고 싶으면 -f 옵션을 함께 사용
  - 이 경우 -u 옵션이 -f 뒤에 와야 함

```
[user1@localhost ~]$ ps -fu user1
UID          PID    PPID  C STIME TTY          TIME CMD
user1       2262       1  0 10:59 ?        00:00:00 /usr/lib/systemd/systemd --use
user1       2267    2262  0 10:59 ?        00:00:00 (sd-pam)
user1       2273    2262  0 10:59 ?        00:00:01 /usr/bin/pulseaudio --daemoniz
user1       2279       1  0 10:59 ?        00:00:00 /usr/bin/gnome-keyring-daemon
user1       2284    2262  0 10:59 ?        00:00:00 /usr/bin/dbus-daemon --session
user1       2291    2244  0 10:59 tty2     00:00:00 /usr/libexec/gdm-wayland-sessi
user1       2298    2291  0 10:59 tty2     00:00:00 /usr/libexec/gnome-session-bin
user1       2360    2298  0 10:59 tty2     00:01:17 /usr/bin/gnome-shell
user1       2375    2262  0 10:59 ?        00:00:00 /usr/libexec/gvfsd
(생략)
```

02.프로세스 관리 명령

프로세스 목록 확인

- 특정 프로세스 정보 출력하기 : -p 옵션
  - -p 옵션과 함께 특정 PID를 지정하면 해당 프로세스의 정보를 출력할 수 있음
  - -f 옵션을 함께 사용하는 것이 좋음

```
[user1@localhost ~]$ ps -p 3141
   PID TTY          TIME CMD
  3141 pts/1    00:00:00 bash
[user1@localhost ~]$ ps -fp 3141
UID          PID    PPID  C STIME TTY          TIME CMD
user1       3141    3136  0 11:02 pts/1    00:00:00 -bash
```

02.프로세스 관리 명령

특정 프로세스 정보 검색

특정 프로세스 정보 검색하기 : ps

- ps 명령과 grep 명령을 |(파이프)로 연결하여 특정 프로세스에 대한 정보를 검색할 수 있음
- 'ps -ef | grep 명령'의 형태로 연결해서 사용

```
[user1@localhost ~]$ pgrep -l bash
10291 bash
```

특정 프로세스 정보 검색하기 : pgrep

- pgrep 명령 : ps와 grep을 하나로 통합하여 만든 명령
- 기본적으로 인자로 지정한 패턴과 일치하는 프로세스를 찾아 PID를 알려줌
- 옵션의 지정에 따라 검색 내용이 다양

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

특정 프로세스 정보 검색

특정 프로세스 정보 검색하기 : pgrep

- bash 패턴을 지정하여 검색한 예

```
[user1@localhost ~]$ pgrep -l bash
10291 bash
```

- pgrep의 경우 -l 옵션을 지정해도 단지 PID와 명령 이름만 출력

```
[user1@localhost ~]$ pgrep -x bash
10291
```

- 더 자세한 정보를 검색하려면 pgrep 명령을 ps 명령과 연결하여 사용

```
[user1@localhost ~]$ ps -fp $(pgrep -x bash)
UID          PID    PPID  C STIME TTY          TIME CMD
user1      10291   10286  0 21:32 pts/1    00:00:00 -bash
```

- -u 옵션으로 사용자명을 지정하여 검색

```
[user1@localhost ~]$ ps -fp $(pgrep -u user1 bash)
UID          PID    PPID  C STIME TTY          TIME CMD
user1      10291   10286  0 21:32 pts/1    00:00:00 -bash
```

02.프로세스 관리 명령

프로세스 종료

- 시그널: 프로세스에 무언가 발생했음을 알리는 간단한 메시지
- 프로세스를 종료하는 데는 kill이나 pkill 명령을 사용 : 시그널을 보내 프로세스를 종료

```
[user1@localhost ~]$ 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
```

- 리눅스에서 지원하는 시그널의 목록은 kill -l 명령으로 알 수 있음

표 6-4 주요 시그널

시그널	번호	기본 처리	의미
SIGHUP	1	종료	터미널과의 연결이 끊어졌을 때 발생한다.
SIGINT	2	종료	인터럽트로 사용자가 Ctrl+c를 입력하면 발생한다.
SIGQUIT	3	종료, 코어덤프	종료 신호로 사용자가 Ctrl+\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 종료

- 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
  ```

- kill 명령은 인자로 지정한 프로세스에 시그널을 전달
- 프로세스는 각 시그널을 받았을 때 어떻게 처리할 것인지 동작이 지정되어 있음
- kill 명령에서 시그널을 지정하지 않을 경우 15번 시그널로 간주
- 15번 시그널은 일반적으로 프로세스 종료이지만, 시그널을 무시하거나 다른 동작을 하도록 지정되어 있다면 프로세스가 종료되지 않을 수 있음
- 9번 시그널은 강제 종료이기 때문에 무조건 종료되지만 좀비 프로세스의 경우 9번 시그널을 받아도 종료되지 않을 수 있음

02.프로세스 관리 명령

프로세스 종료

프로세스 종료하기 : kill

- 터미널 1에서 man ps 명령을 실행

```
[user1@localhost ~]$ man ps
PS(1)                          User Commands                          PS(1)

NAME
       ps - report a snapshot of the current processes.
(생략)
```

- 터미널 0에서 pgrep과 ps 명령으로 man 프로세스의 PID를 확인

```
[user1@localhost ~]$ ps -fp $(pgrep -x man)
UID          PID    PPID  C STIME TTY          TIME CMD
user1      10675   10291  0 22:05 pts/1    00:00:00 man ps
```

- man 프로세스의 PID를 지정하여 kill 명령으로 man 프로세스를 종료

```
[user1@localhost ~]$ kill 10675
```

- kill 명령으로 man 프로세스가 종료된 것을 확인

```
Manual page ps(1) line 1 (press h for help or q to quit)종료됨
[user1@localhost ~]$
```

```
[user1@localhost ~]$ pgrep -x man
```

02.프로세스 관리 명령

프로세스 종료

프로세스 종료하기 : pkill

- pkill 명령도 kill 명령과 마찬가지로 시그널을 보냄
- PID가 아니라 프로세스의 명령 이름(CMD)으로 프로세스를 찾아 종료
- 명령 이름으로 찾아 종료 하므로 같은 명령이 여러 개 검색될 경우 한 번에 모두 종료
- 터미널 1과 터미널 2에서 각각 man pkill을 실행

```
[user1@localhost ~]$ ps -fp $(pgrep -x man)
UID          PID    PPID  C STIME TTY      STAT   TIME CMD
user1      10909   10880  0 22:16 pts/2    S+     0:00 man kill
user1      10932   10291  0 22:16 pts/1    S+     0:00 man kill
```

- pkill 명령으로 man 프로세스를 모두 종료

```
[user1@localhost ~]$ pkill man
[user1@localhost ~]$ pgrep -x man
```

프로세스 종료하기 : killall

- killall 명령도 pkill 명령처럼 프로세스의 명령 이름(CMD)으로 프로세스를 찾아 종료
- 이 이름으로 실행 중인 모든 프로세스를 한 번에 종료

02.프로세스 관리 명령

[따라해보기] 프로세스 찾아서 종료시키기

① 터미널 1에서 sh 프로세스를 실행

```
[user1@localhost ~]$ sh
sh-4.4$
```

② 터미널 1에서 more 명령을 실행

```
sh-4.4$ more /etc/services
# /etc/services:
#
# Network services, Internet style
(생략)
```

02.프로세스 관리 명령

- **[따라해보기] 프로세스 찾아서 종료시키기**
  - ③ 터미널 1에서 실행한 프로세스를 터미널 0에서 찾음 (-t 옵션이나 a 옵션을 사용)

```
[user1@localhost ~]$ ps -ft pts/1
UID          PID   PPID  C STIME TTY          TIME CMD
user1      10291  10286  0 21:32 pts/1    00:00:00 -bash
user1      11088  10291  0 22:29 pts/1    00:00:00 sh
user1      11089  11088  0 22:29 pts/1    00:00:00 more /etc/services
```

```
[user1@localhost ~]$ ps a
  PID TTY      STAT   TIME COMMAND
(생략)
 10291 pts/1    Ss     0:00 -bash
 10547 pts/0    Ss     0:00 -bash
 10880 pts/2    Ss+    0:00 -bash
 10924 pts/2    S      0:00 less
 11088 pts/1    S      0:00 sh
 11089 pts/1    S+     0:00 more /etc/services
 11100 pts/0    R+     0:00 ps a
```

02.프로세스 관리 명령

[따라해보기] 프로세스 찾아서 종료시키기

④ 터미널 1에서 실행한 more 명령을 종료

```
[user1@localhost ~]$ kill 11089        ➜ 터미널 1에서 more 종료를 확인한다.
```

⑤ -9 시그널로 터미널 1에서 실행한 sh 프로세스를 종료

```
[user1@localhost ~]$ kill -9 11088     ➜ 터미널 1에서 sh의 종료를 확인한다.
```

02.프로세스 관리 명령

프로세스 관리 도구

top 명령

- 현재 실행 중인 프로세스에 대한 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

프로세스 관리 도구

top 명령

- top 명령의 실행 화면

```
[user1@localhost ~]$ top
top - 22:36:30 up 13:32,  4 users,  load average: 0.00, 0.00, 0.00
Tasks: 254 total,   2 running, 252 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  1.7 sy,  0.0 ni, 96.7 id,  0.0 wa,  0.3 hi,  0.3 si,  0.0 st
MiB Mem :   1806.1 total,    108.6 free,   1054.2 used,    643.2 buff/cache

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
11135 user1     20   0   62436   4764   3904 R   0.7   0.3   0:00.71 top
    1 root      20   0  179072  13936   9196 S   0.0   0.8   0:08.42 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_+
    6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/+
    8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percp+
    9 root      20   0       0      0      0 S   0.0   0.0   0:01.21 ksoftirq+
   10 root      20   0       0      0      0 R   0.0   0.0   0:01.62 rcu_sched
   11 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migratio+
   12 root      rt   0       0      0      0 S   0.0   0.0   0:00.04 watchdog+
   13 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
   16 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
   17 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kauditd
   18 root      20   0       0      0      0 S   0.0   0.0   0:00.05 khungtas+
   19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 oom_reap+
```

02.프로세스 관리 명령

프로세스 관리 도구

top 명령

- top 명령은 종료하지 않고 실시간으로 프로세스의 상태를 보여주며 내부적으로 사용하는 명령도 있음

표 6-6 top 명령의 내부 명령

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

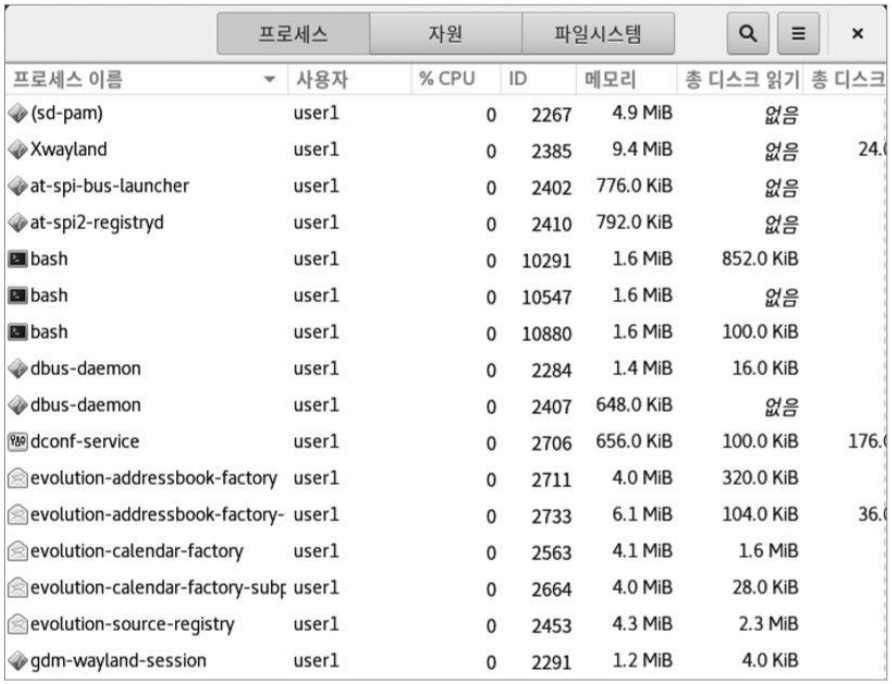
02.프로세스 관리 명령

프로세스 관리 도구

시스템 정보

- [현재 활동]-[프로그램 표시]-[유틸리티]-[시스템 정보]와 같은 순서로 동작
- 시스템 정보 화면은 프로세스 이름, 사용자, CPU 사용량, 메모리 사용량 등의 정보를 보여줌
- [프로세스 끝내기(P)] 버튼을 사용하여 프로세스를 종료할 수도 있음

그림 6-2 시스템 정보 동작시키기



프로세스 이름	사용자	% CPU	ID	메모리	총 디스크 읽기	총 디스크
(sd-pam)	user1	0	2267	4.9 MiB	없음	
Xwayland	user1	0	2385	9.4 MiB	없음	24.(
at-spi-bus-launcher	user1	0	2402	776.0 KiB	없음	
at-spi2-registryd	user1	0	2410	792.0 KiB	없음	
bash	user1	0	10291	1.6 MiB	852.0 KiB	
bash	user1	0	10547	1.6 MiB	없음	
bash	user1	0	10880	1.6 MiB	100.0 KiB	
dbus-daemon	user1	0	2284	1.4 MiB	16.0 KiB	
dbus-daemon	user1	0	2407	648.0 KiB	없음	
dconf-service	user1	0	2706	656.0 KiB	100.0 KiB	176.(
evolution-addressbook-factory	user1	0	2711	4.0 MiB	320.0 KiB	
evolution-addressbook-factory-	user1	0	2733	6.1 MiB	104.0 KiB	36.(
evolution-calendar-factory	user1	0	2563	4.1 MiB	1.6 MiB	
evolution-calendar-factory-subp	user1	0	2664	4.0 MiB	28.0 KiB	
evolution-source-registry	user1	0	2453	4.3 MiB	2.3 MiB	
gdm-wayland-session	user1	0	2291	1.2 MiB	4.0 KiB	

그림 6-3 시스템 정보 화면

03.포그라운드·백그라운드 프로세스와 작업 제어

포그라운드 작업과 백그라운드 작업

포그라운드 작업

- 포그라운드 프로세스 : 사용자가 입력한 명령이 실행되어 결과가 출력될 때까지 기다려야 하는 방식으로 처리되는 프로세스
- 이를 작업 제어에서는 포그라운드 작업이라고 함
- 일반적으로 사용자가 명령을 실행하는 방식
- 프롬프트가 출력되지 않아 다른 명령을 입력할 수 없으므로 기다려야 함

```
[user1@localhost ~]$ sleep 100    ➜ 포그라운드 작업
                                  ➜ sleep 명령이 끝날 때까지 기다려야 한다.
```

03.포그라운드·백그라운드 프로세스와 작업 제어

포그라운드 작업과 백그라운드 작업

백그라운드 작업

- 백그라운드 프로세스 : 명령을 실행하면 명령의 처리가 끝나는 것과 관계없이 곧바로 프롬프트가 출력되어 사용자가 다른 작업을 계속할 수 있음
- 작업 제어에서는 백그라운드 작업이라고 함
- 명령의 실행 시간이 많이 걸릴 것으로 예상되거나 명령을 실행한 후 다른 작업을 할 필요가 있을 때 많이 사용

```
[user1@localhost ~]$ sleep 100 &  ➞ 백그라운드 작업
[user1@localhost ~]$              ➞ 프롬프트가 바로 나와 다른 명령을 실행시킬 수 있다.
```

- 기존의 작업 화면과 백그라운드 작업 결과가 뒤섞인 채 터미널 화면에 출력될 수 있음
- 백그라운드로 처리할 때는 다음과 같이 출력과 오류 방향을 전환하고, 실행 결과와 오류 메시지는 파일로 저장하는 방법을 사용하여 문제를 해결

```
[user1@localhost ~]$ find / -name passwd > pw.dat 2>&1 & ➞ pw.dat에 결과와 오류를 저장한다.
```

작업 제어

- 작업 제어는 작업 전환과 작업 일시 중지, 작업 종료를 의미
- 작업 전환 : 포그라운드 작업->백그라운드 작업, 백그라운드 작업->포그라운드 작업으로 전환
- 작업 일시 중지: 작업을 잠시 중단
- 작업 종료 : 프로세스를 종료하는 것처럼 작업을 종료

03.포그라운드·백그라운드 프로세스와 작업 제어

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

- Ex) jobs 명령을 실행한 결과

```
[user1@localhost ~]$ jobs
[1]-  Running                 sleep 100 &
[2]+  Running                 find / -name pwsswd > 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. • +: 가장 최근에 접근한 작업 • -: + 작업보다 바로 전에 접근한 작업 • 공백: 그 외의 작업
상태	실행 중	작업 상태를 표시한다. • Running: 현재 실행 중이다. • Done: 작업이 정상적으로 종료된다. • Terminated: 작업이 비정상적으로 종료된다. • Stopped: 작업이 잠시 중단된다.
명령	sleep 100 &	백그라운드로 실행 중인 명령이다.

03.포그라운드·백그라운드 프로세스와 작업 제어

작업 제어

작업 전환하기

표 6-8 작업 전환 명령

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

- 실행 중인 작업을 백그라운드로 전환하려면 우선 [Ctrl] + z로 작업을 중지

```
[user1@localhost ~]$ jobs                    ➞ 백그라운드 작업이 없다.
[user1@localhost ~]$ sleep 100               ➞ 포그라운드로 실행한다.
^Z                                           ➞ Ctrl+z로 일시 중지한다.
[1]+  Stopped                 sleep 100      ➞ 일시 중지된 상태이다.
[user1@localhost ~]$ bg %1                   ➞ 백그라운드로 전환한다.
[1]+ sleep 100 &
[user1@localhost ~]$ jobs
[1]+  Running                 sleep 100 &    ➞ 백그라운드로 실행 중이다.
```

- 백그라운드로 실행 중인 작업을 다시 포그라운드로 전환하려면 'fg %작업 번호' 명령을 사용

```
[user1@localhost ~]$ jobs
[1]+  Running                 sleep 100 &
[user1@localhost ~]$ fg              ➞ 포그라운드로 전환한다.
sleep 100                            ➞ 포그라운드로 실행 중이다.
```

03.포그라운드·백그라운드 프로세스와 작업 제어

작업 제어

- 작업 종료하기 : [Ctrl]+c
  - 포그라운드 작업은 [Ctrl]+c를 입력하면 대부분 종료

```
[user1@localhost ~]$ sleep 100      ➜ 포그라운드로 실행 중이다.
^C                                  ➜ 강제 종료한다.
[user1@localhost ~]$
```

  - 백그라운드 작업은 kill 명령으로 강제 종료: PID 또는 '%작업 번호'

```
[user1@localhost ~]$ sleep 100&          ➜ 백그라운드로 실행 중이다.
[1] 3328
[user1@localhost ~]$ kill %1             ➜ 강제 종료한다.
[user1@localhost ~]$
[1]+  종료됨              sleep 100      ➜ 종료 메시지가 출력된다.
```

03.포그라운드·백그라운드 프로세스와 작업 제어

작업 제어

로그아웃 후에도 백그라운드 작업 계속 실행하기

- nohup 명령 : 로그아웃한 다음에도 작업이 완료될 때까지 백그라운드 작업을 실행해야 할 경우에 사용

nohup

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

- Ex) nohup 명령 사용 예

```
[user1@localhost ~]$ nohup find / -name passwd &
[1] 3359
[user1@localhost ~]$ nohup: ignoring input and appending output to 'nohup.out'
exit
```

- Ex) nohup 명령 사용 예

```
[user1@localhost ~]$ more nohup.out
find: '/boot/grub2': 허가 거부
find: '/boot/loader/entries': 허가 거부
find: '/boot/lost+found': 허가 거부
(생략)
/usr/bin/passwd
/usr/share/licenses/passwd
/usr/share/doc/passwd
/usr/share/bash-completion/completions/passwd
(생략)
```

03.포그라운드·백그라운드 프로세스와 작업 제어

작업 제어

로그아웃 후에도 백그라운드 작업 계속 실행하기

- 지정한 파일에 결과와 오류 메시지를 출력

```
[user1@localhost ~]$ nohup find / -name passwd > pw.dat 2>&1 &
[1] 3480
[user1@localhost ~]$ exit
```

- 다시 로그인하여 파일 내용을 살펴

```
[user1@localhost ~]$ more pw.dat
nohup: ignoring input
find: '/boot/grub2': 허가 거부
find: '/boot/loader/entries': 허가 거부
find: '/boot/lost+found': 허가 거부
find: '/boot/efi/EFI/centos': 허가 거부
find: '/proc/tty/driver': 허가 거부
(생략)
```

```
/usr/bin/passwd
/usr/share/licenses/passwd
/usr/share/doc/passwd
/usr/share/bash-completion/completions/passwd
(생략)
```

03.포그라운드·백그라운드 프로세스와 작업 제어

[따라해보기] 작업 관리하기

① 실습 디렉터리를 만들고 이동

```
[user1@localhost ~]$ mkdir linux_ex/ch6
[user1@localhost ~]$ cd linux_ex/ch6
[user1@localhost ch6]$
```

② 다음과 같이 백그라운드 작업을 만든다.

- 파일을 하나 복사하고 vi 명령을 실행
- [Ctrl]+ z를 실행하여 vi를 정지된 작업으로 만듬

```
[user1@localhost ch6]$ cp /etc/hosts .
[user1@localhost ch6]$ vi hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
^Z
```

- vi 작업이 일시 정지되면 다음과 같이 Stopped로 표시

```
[user1@localhost ch6]$ vi hosts

[1]+  Stopped                 vim hosts
```

③ jobs 명령으로 백그라운드 작업을 확인

```
[user1@localhost ch6]$ jobs
[1]+  Stopped                 vim hosts
```

03.포그라운드·백그라운드 프로세스와 작업 제어

[따라해보기] 작업 관리하기

④ 정지 중인 작업을 fg 명령을 사용하여 복구

```
[user1@localhost ch6]$ fg
```

⑤ q!로 vi를 종료

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
~
~
~
:q!
```

04.작업 예약

정해진 시간에 한 번 실행

- 특정한 시간에 작업을 실행하도록 예약할 수 있는 두 가지 방법
  - 정해진 시간에 한 번만 수행
  - 정해진 시간에 반복 수행

- at 명령: 정해진 시간에 한 번만 명령을 실행할 때 사용
- at 명령 설정하기

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

정해진 시간에 한 번 실행

at 명령 설정하기

- at 명령을 사용하여 정해진 시간에 명령을 실행하도록 예약하려면 at 명령 뒤에 시간을 명시

```
[user1@localhost ch6]$ at 04:30 pm
at>
```

- 시간을 지정하는 형식
  - at 4pm + 3 days : 지금부터 3일 후 오후 4시에 작업을 수행한다.
  - at 10am Jul 31 : 7월 31일 오전 10시에 작업을 수행한다.
  - at 1am tomorrow : 내일 오전 1시에 작업을 수행한다.
  - at 10:00am today : 오늘 오전 10시에 작업을 수행한다.
- at로 실행할 명령은 기본적으로 표준 입력으로 지정: 명령의 입력을 마치려면 [Ctrl]+d 입력

```
[user1@localhost ch6]$ at 04:30 pm          → 시간을 지정한다.
warning: commands will be executed using /bin/sh
at> /usr/bin/ls  -l ~user1 > ~user1/at.out  → 실행할 명령을 지정한다.
at> <EOT>                                   → Ctrl+d 입력으로 종료한다.
job 2 at Fri Nov  8 16:30:00 2019           → 작업 예약을 완료한다.
```

04.작업 예약

정해진 시간에 한 번 실행

- at 작업 파일 확인하기
  - at로 생성된 작업 파일은 /var/spool/at 디렉터리에 저장

```
[root@localhost ~]# ls -l /var/spool/at
합계 4
-rwx------. 1 user1 user1 3270 11월  8 16:19 a00002019015c2
drwx------. 2 root  root     6  5월 11 22:12 spool
[root@localhost ~]# exit
[user1@localhost ch6]$
```

  - root 사용자만 /var/spool/at 디렉터리 내용 확인 가능

04.작업 예약

정해진 시간에 한 번 실행

- at 작업 목록 확인하기 : -l 옵션, atq
  - at 명령으로 설정된 작업의 목록은 -l 옵션으로 확인
  - 출력 형식 : 각 작업당 한 줄 로 작업 번호, 날짜, 시간, 작업 구분 순

```
[user1@localhost ch6]$ at -l
2       Fri Nov  8 16:30:00 2019 a user1
```

atq 명령

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

  - -l 옵션 외에 atq 명령도 실행 대기 중인 작업의 목록을 출력

  - Ex) at -l'과 같은 형식으로 출력됨

```
[user1@localhost ch6]$ atq
2       Fri Nov  8 16:30:00 2019 a user1
```

04.작업 예약

정해진 시간에 한 번 실행

- at 작업 삭제하기 : -d 옵션과 atrm 옵션
  - at 명령으로 설정한 작업이 실행되기 전에 삭제하려면 -d 옵션을 사용하고 삭제할 작업 번호를 지정
  - atrm은 at -d와 같은 기능을 수행

atrm

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

  - Ex) at 명령으로 작업 두 개를 예약

```
[user1@localhost ch6]$ at 1am tomorrow
warning: commands will be executed using /bin/sh
at> /usr/bin/ls > ~user1/at1.out
at> <EOT>
job 3 at Sat Nov  9 01:00:00 2019
[user1@localhost 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 Fri Nov  8 22:00:00 2019
```

04.작업 예약

정해진 시간에 한 번 실행

- at 작업 삭제하기 : -d 옵션과 atrm 옵션
  - Ex) 설정된 작업을 atq 명령

```
[user1@localhost ch6]$ atq
3       Sat Nov  9 01:00:00 2019 a user1
4       Fri Nov  8 22:00:00 2019 a user1
```

  - Ex) 3번은 at –d 명령으로, 4번 작업은 atrm 명령으로 삭제

```
[user1@localhost ch6]$ at -d 3
[user1@localhost ch6]$ atrm 4
[user1@localhost ch6]$ atq
[user1@localhost ch6]$
```

04.작업 예약

정해진 시간에 한 번 실행

at 명령 사용 제한하기

- 사용 제한과 관련된 파일 : /etc/at.allow와 /etc/at.deny
  - /etc/at.allow 파일과 /etc/at.deny 파일에는 한 줄에 사용자 이름을 하나씩만 기록
  - /etc/at.allow 파일이 있으면 이 파일에 있는 사용자만 at 명령을 사용할 수 있음, 이 경우에 /etc/at.deny 파일은 무시
  - /etc/at.allow 파일이 없으면 /etc/at.deny 파일에 지정된 사용자를 제외한 모든 사용자가 at 명령을 사용할 수 있음
  - 만약 두 파일이 모두 없다면 root만 at 명령을 사용할 수 있음
  - 한 사용자가 두 파일 모두에 속해 있다면 그 사용자는 at 명령을 사용할 수 있음, /etc/at.allow 파일이 적용되기 때문.
  - /etc/at.deny를 빈 파일로 두면 모든 사용자가 at 명령을 사용할 수 있는데, 이것이 초기 설정

- 만약 at.deny 파일에 user1 사용자가 기록되어 있다면 at 명령을 실행했을 때 사용 권한이 없다는 메시지가 출력

```
[user1@localhost ch6]$ at
You do not have permission to use at.
```

04.작업 예약

■ 정해진 시간에 반복 실행

- crontab 명령 : 지정 시간이 되면 반복적으로 실행하도록 설정

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

- crontab 파일 형식

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

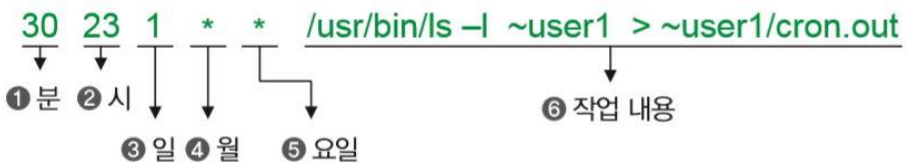


그림 6-4 crontab 파일의 형식

- ❹번의 월 항목은 매월을, ❺번의 요일은 모든 요일을 의미

04.작업 예약

정해진 시간에 반복 실행

- crontab 파일 생성하고 편집하기 : crontab –e
  - crontab 파일의 생성과 편집은 crontab -e 명령으로 수행
  - crontab 편집기로는 기본 적으로 VISUAL 또는 EDITOR 환경 변수에 지정된 편집기를 사용

```
[user1@localhost ch6]$ crontab -l
30 23 1 * * /usr/bin/ls -l ~user1 > ~user1/cron.out
```

  - crontab –e 명령으로 수행할 작업을 하나 작성

```
[user1@localhost ch6]$ crontab -e
30 23 1 * * /usr/bin/ls -l ~user1 > ~user1/cron.out
~
~
:wq
```

  - crontab -e 명령으로 편집한 파일을 저장하면 자동적으로 /var/spool/cron 디렉터리에 사용자 이름으로 생성

```
[user1@localhost ch6]$ ls -l /var/spool
합계 0
(생략)
drwx------. 2 root root 19 11월  8 17:07 cron/
(생략)
```

04.작업 예약

정해진 시간에 반복 실행

- crontab 파일 내용 확인하기 : crontab –l

```
[user1@localhost ch6]$ crontab -l
30 23 1 * * /usr/bin/ls -l ~user1 > ~user1/cron.out
```

- crontab 파일 삭제하기 : crontab –r

```
[user1@localhost ch6]$ crontab -r
[user1@localhost ch6]$ crontab -l
no crontab for user1
```

- crontab 명령 사용 제한하기
  - /etc/cron.allow, /etc/cron.deny 파일로 crontab 명령 사용 권한을 제한
  - cron.deny 파일은 기본적으로 있지만 cron.allow 파일은 관리자가 만들어야 함
  - 두 파일이 적용되는 기준
    - /etc/cron.allow 파일이 있으면 이 파일 안에 있는 사용자만 crontab 명령을 사용할 수 있음
    - /etc/cron.allow 파일이 없고 /etc/cron.deny 파일이 있으면 이 파일에 사용자 계정이 없어야 crontab 명령을 사용할 수 있음
    - /etc/cron.allow 파일과 /etc/cron.deny 파일이 모두 없다면 시스템 관리자만 crontab 명령을 사용할 수 있음
  - 두 파일이 모두 없는데 일반 사용자가 crontab 명령을 사용하려고 하면 다음과 같은 메시지가 출력

```
[user1@localhost ch6]$ crontab -e
You (user1) are not allowed to use this program (crontab)
See crontab(1) for more information
```

04.작업 예약

[따라해보기] crontab 설정하기

① crontab -e 명령으로 crontab 파일을 편집

- 명령의 실행 결과를 확인하기 위해 이 실습을 하고 있는 날짜와 시간을 먼저 확인하고 설정

```
[user1@localhost ch6]$ date
2019. 11. 08. (금) 17:12:41 KST
[user1@localhost ch6]$ crontab -e
20 17 8 * * /usr/bin/ls -l /tmp > /home/user1/linux_ex/ch6/tmp.out
~
~
:wq
[user1@localhost ch6]$
```

② crontab -l 명령으로 설정 내용을 확인

```
[user1@localhost ch6]$ crontab -l
20 17 8 * * /usr/bin/ls -l /tmp > /home/user1/linux_ex/ch6/tmp.out
```

04.작업 예약

[따라해보기] crontab 설정하기

③ crontab에 설정한 시간이 경과한 후 작업 결과를 확인

```
[user1@localhost ch6]$ ls
hosts  tmp.out
[user1@localhost ch6]$ cat tmp.out
합계 0
drwxrwxrwt. 2 user1 user1 20 11월  2 09:13 VMwareDnD
drwx------. 3 root  root  17 11월  8 14:45 systemd-private-7768011a91c94c8c978c7
ab280743286-ModemManager.service-KvleS2
drwx------. 3 root  root  17 11월  8 14:45 systemd-private-7768011a91c94c8c978c7
ab280743286-bluetooth.service-IgAEUR
drwx------. 3 root  root  17 11월  8 14:46 systemd-private-7768011a91c94c8c978c7
ab280743286-bolt.service-S5M9sI
drwx------. 3 root  root  17 11월  8 14:45 systemd-private-7768011a91c94c8c978c7
ab280743286-chronyd.service-76ux9G
drwx------. 3 root  root  17 11월  8 14:46 systemd-private-7768011a91c94c8c978c7
ab280743286-colord.service-VV1uSG
drwx------. 3 root  root  17 11월  8 14:46 systemd-private-7768011a91c94c8c978c7
ab280743286-fwupd.service-LnRfLQ
```

```
drwx------. 3 root  root  17 11월  8 14:45 systemd-private-7768011a91c94c8c978c7
ab280743286-rtkit-daemon.service-MuoBuy
drwx------. 2 user1 user1  6 11월  8 15:16 tracker-extract-files.1000
drwx------. 2 root  root   6 11월  8 14:45 vmware-root_827-4256676229
drwx------. 2 root  root   6 11월  3 10:58 vmware-root_831-4248090624
```

Thank you