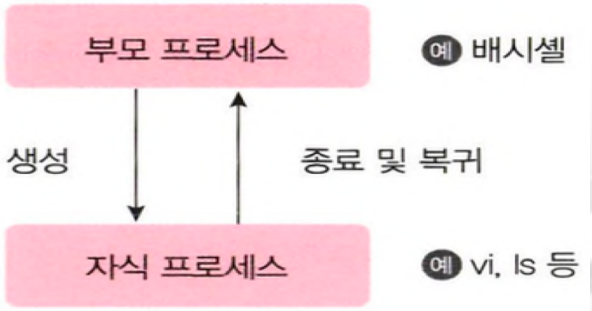
프로세스 관리

프로세스 관리

- ❖ Process
  - ✓ 개요
    - 현재 시스템에서 실행 중인 프로그램
    - 리눅스는 기본적으로 다중 프로세스 시스템이므로 여러 개의 프로세스를 동시에 실행
    - 리눅스 운영에 필요한 다양한 기능을 수행하는 시스템 프로세스도 있고 사용자가 실행한 프로그램인 사용자 프로세스가 있음

프로세스 관리

- ❖ Process
  - ✓ 개요
    - ● 프로세스는 부모 – 자식 관계
      - ◆ 리눅스에서 모든 프로세스는 부모-자식 관계를 가짐
      - ◆ 필요에 따라 부모 프로세스는 자식 프로세스를 생성하고 자식 프로세스는 또 다른 자식 프로세스를 만들 수 있음
      - ◆ 리눅스 시스템을 부팅할 때 스케줄러가 실행한 프로세스인 systemd와 kthreadd 프로세스를 제외하면 모든 프로세스는 부모 프로세스를 가지고 있음
      - ◆ 자식 프로세스는 부모 프로세스에 의해 만들어지는 프로세스로 자식 프로세스는 할 일이 끝나면 부모 프로세스에 결과를 돌려주고 종료되는데 사용자가 vi를 실행하여 셸이 vi 프로세스를 생성할 경우 셸은 부모 프로세스가 되고 vi는 자식 프로세스가 되며 사용자가 vi를 종료하면 vi는 부모 프로세스인 셸로 돌아감



프로세스 관리

- ❖ Process
  - ✓ 프로세스 번호
    - 각 프로세스는 고유한 번호를 가지고 있는데 이것이 PID(Process IDentification number)
    - PID 는 1번부터 시작하고 프로세스가 실행되면서 하나씩 증가하여 부여됨
    - 리눅스가 부팅될 때 PID 1번 systemd 프로세스와 2번 kthreadd 프로세스가 차례로 실행되며 이때 1번 프로세스는 나머지 모든 시스템 프로세스의 부모 프로세스가 되고 2번 프로세스는 모든 스레드의 부모 프로세스가 됨
    - 유닉스에서 1 번 프로세스는 init 프로세스이지만 우분투의 경우 init의 기능을 systemd로 변경
    - 이전 시스템과 호환되도록 1 번 프로세스의 이름은 init을 유지하고 있는데 init은 systemd의 심벌릭 링크

      adam@adam:/var/www/html$ ls -l /sbin/init

      lrwxrwxrwx 1 root root 20 Aug 21 21:11 /sbin/init -> /lib/systemd/systemd

프로세스 관리

- ❖ Process
  - ✓ 프로세스 종류
    - 리눅스의 프로세스 중 사용자가 실행한 일반적인 프로세스는 잠깐 실행되었다가 바로 종료
    - 데몬 프로세스
      - ◆ 데몬(daemon) 프로세스는 특정 서비스를 제공하기 위해 존재하며 리눅스 커널에 의해 실행되므로 서비스라고도 함
      - ◆ 데몬은 평소에는 대기 상태로 있다가 서비스 요청이 들어오면 서비스를 제공
      - ◆ 리눅스에서는 다양한 서비스를 제공하기 위한 데몬이 동작
      - ◆ 원격 접속 서비스를 제공하기 위해 동작하는 sshd 프로세스가 있는데 이를 ssh 서버 데몬이라고 함
    - 고아 프로세스
      - ◆ 자식 프로세스는 종료되면 부모 프로세스로 돌아가지만 자식 프로세스가 아직 실행 중 인데 부모 프로세스가 먼저 종료되면 자식 프로세스는 고아(orphan) 프로세스
      - ◆ 1번 프로세스가 고아 프로세스의 새로운 부모 프로세스가 되어 고아 프로세스가 작업을 마치고 종료될 수 있게 함

프로세스 관리

- ❖ Process
  - ✓ 프로세스 종류
    - ● 좀비 프로세스
      - ◆ 자식 프로세스는 종료될 때 부모 프로세스에 종료 정보(exit status)를 보내고 부모 프로세스가 이 정보를 받으면 자식 프로세스는 프로세스 테이블 목록에서 삭제되지만 자식 프로세스가 실행을 종료했는데도 프로세스 테이블 목록에 남아 있는 경우가 있는데 이러한 자식 프로세스가 좀비 프로세스
      - ◆ 자식 프로세스의 종료 정보를 부모 프로세스가 읽어 가기를 기다리고 있는 것
      - ◆ 부모 프로세스가 자식 프로세스의 종료 정보를 제대로 처리하지 않았기 때문에 이런 일이 발생
      - ◆ 좀비 프로세스는 프로세스 목록에 defunct 프로세스라고 나오기도 함
      - ◆ 좀비 프로세스는 실제로 실행되지는 않지만 동작 중인 프로세스 테이블 목록을 차지하고 있기 때문에 잘못 작성된 프로그램으로 인해 좀비 프로세스가 증가하면 프로세스 테이블의 용량이 부족해서 정상적인 프로세스가 실행되지 않을 수도 있음
      - ◆ 좀비 프로세스는 kill 명령으로 제거할 수 없으며 SIGCHLD 시그널을 부모 프로세스에 보내어 부모 프로세스가 자식 프로세스를 정리하도록 하거나 부모 프로세스 자체를 종료해야만 함
      - ◆ 부모 프로세스가 종료되면 좀비 프로세스는 고아 프로세스가 되고 새로운 부모인 1번 프로세스는 주기적으로 자식 프로세스의 종료 정보를 확인하여 정리

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 현재 실행 중인 프로세스에 대한 정보를 출력하는 명령
      - ◆ 형식: ps [옵션]
      - ◆ 옵션
        - ▪ 유닉스 옵션: 묶어서 사용할 수 있고 -로 시작
          - -e: 시스템에서 실행 중인 모든 프로세스의 정보를 출력
          - -f: 프로세스에 대한 자세한 정보를 출력
          - -u uid: 특정 사용자에 대한 모든 프로세스의 정보를 출력
          - -p pid: pid로 지정한 특정 프로세스의 정보를 출력
        - ▪ BSD 옵션: 묶어서 사용할 수 있고 -로 시작하지 않음
          - a: 터미널에서 실행시킨 프로세스의 정보를 출력
          - u: 프로세스 소유자 이름, CPU 사용량, 메모리 사용량 등 상세 정보를 출력
          - x: 시스템에서 실행 중인 모든 프로세스의 정보를 출력
        - ▪ GNU 옵션: --로 시작
          - --pid PID 목록: 목록으로 지정한 특정 PID 정보를 출력

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 현재 단말기의 프로세스 목록 출력

```
adam@adam:~$ ps
  PID TTY          TIME CMD
 3422 pts/1    00:00:01 bash
11523 pts/1    00:00:00 ps
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 프로세스 상세 정보 출력

```
adam@adam:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
adam        3422    3421  0 Sep26 pts/1    00:00:01 -bash
adam       11526    3422  0 02:02 pts/1    00:00:00 ps -f
```

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)

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 터미널에서 실행한 프로세스 정보 출력

```
adam@adam:~$ ps -a
  PID TTY          TIME CMD
 1343 tty1     00:00:00 bash
11578 pts/0    00:00:00 ps
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 터미널에서 실행한 프로세스 상세 정보 출력

```
adam@adam:/var/www/html$ ps -au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         827  0.0  0.0   5216   808 ?        Ss+  Sep26   0:00 /sbin/agetty -o -p -- ₩u --keep-baud 115200,5760
adam        1338  0.0  0.1 159580  5444 tty2     Ssl+ Sep26   0:00 /usr/libexec/gdm-wayland-session env GNOME_SHELL
adam        1343  0.0  0.3 222824 13744 tty2     Sl+  Sep26   0:00 /usr/libexec/gnome-session-binary --session=ubun
adam        2306  0.0  0.1   8176  4972 pts/0    Ss+  Sep26   0:00 bash
adam        3422  0.0  0.1   8360  5080 pts/1    Ss   Sep26   0:01 -bash
adam       11580  0.0  0.0   9420  1640 pts/1    R+   03:01   0:00 ps -au
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 터미널에서 실행한 프로세스 상세 정보 출력

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

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ STAT 문자의 의미
        - ▪ R: 실행 중
        - ▪ S: 인터럽트가 가능한 대기 상태
        - ▪ T: 작업 제어에 의해 정지된 상태
        - ▪ Z: 좀비 프로세스
        - ▪ STIME: 프로세스의 시작 날짜 나 시간
        - ▪ s: 세션 리더 프로세스
        - ▪ +: 포그라운드 프로세스 그룹
        - ▪ l: 멀티스레드

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 전체 프로세스 목록 출력: ps -ef

```
adam@adam:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Sep26 ?        00:00:05 /sbin/init
root           2       0  0 Sep26 ?        00:00:00 [kthreadd]
root           3       2  0 Sep26 ?        00:00:00 [rcu_gp]
root           4       2  0 Sep26 ?        00:00:00 [rcu_par_gp]
root           5       2  0 Sep26 ?        00:00:00 [slub_flushwq]
root           6       2  0 Sep26 ?        00:00:00 [netns]
root           8       2  0 Sep26 ?        00:00:00 [kworker/0:0H-events_highpri]
root          10       2  0 Sep26 ?        00:00:00 [mm_percpu_wq]
root          11       2  0 Sep26 ?        00:00:00 [rcu_tasks_rude_]
root          12       2  0 Sep26 ?        00:00:00 [rcu_tasks_trace]
```

        - ▪ TTY의 값이 ?인 것은 대부분 데몬으로 시스템이 실행한 프로세스

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 전체 프로세스 목록 출력: ps ax | more

```
adam@adam:~$ ps ax | more
  1 ?        Ss     0:04 /sbin/init
  2 ?        S      0:00 [kthreadd]
  3 ?        S      0:00 [pool_workqueue_release]
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 전체 프로세스 목록 출력: ps aux | more

```
adam@itstudy:~$ ps aux | more
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.3  22432 12568 ?        Ss   Sep16   0:05 /sbin/init
root          2  0.0  0.0      0     0 ?        S    Sep16   0:00 [kthreadd]
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 특정 사용자의 프로세스 목록 출력: -u

```
adam@itstudy:~$ ps -u adam
  PID TTY          TIME CMD
 1220 ?        00:00:00 systemd
 1221 ?        00:00:00 (sd-pam)
 1231 tty1     00:00:00 bash
 1327 ?        00:00:10 sshd
 1328 pts/0    00:00:00 bash
 5398 pts/0    00:00:00 ps
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 특정 사용자의 프로세스 목록 출력: -u

```
adam@itstudy:~$ ps -fu adam
UID          PID    PPID  C STIME TTY          TIME CMD
adam        1220       1  0 Sep16 ?        00:00:00 /usr/lib/systemd/systemd --
user
adam        1221    1220  0 Sep16 ?        00:00:00 (sd-pam)
adam        1231    1060  0 Sep16 tty1     00:00:00 -bash
adam        1327    1281  0 Sep16 ?        00:00:11 sshd: adam@pts/0
adam        1328    1327  0 Sep16 pts/0    00:00:00 -bash
adam        5410    1328 99 06:54 pts/0    00:00:00 ps -fu adam
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 목록 확인
    - ● ps 명령
      - ◆ 특정 프로세스 정보 출력: -p

```
adam@itstudy:~$ ps -fp 1231
UID          PID    PPID  C STIME TTY          TIME CMD
adam        1231    1060  0 Sep16 tty1     00:00:00 -bash
```

프로세스 관리

- Process 관리 명령
  - ✓ 특정 프로세스 검색
    - ps –ef | grep 프로세스이름

```
adam@adam:~$ ps -ef | grep bash
adam        2306    2276  0 Sep26 pts/0    00:00:00 bash
adam        3422    3421  0 Sep27 pts/1    00:00:01 -bash
adam       12584    3422  0 00:22 pts/1    00:00:00 grep --color=auto bash
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 특정 프로세스 검색
    - ● pgrep [옵션] [패턴]
      - ◆ 옵션
        - ▪ -x: 패턴과 정확히 일치하는 프로세스의 정보를 출력
        - ▪ -n: 패턴을 포함하고 있는 가장 최근 프로세스의 정보를 출력
        - ▪ -u 사용자 이름: 특정 사용자에 대한 모든 프로세스를 출력
        - ▪ -l: PID와 프로세스 이름을 출력
        - ▪ -t term: 특정 단말기와 관련된 프로세스의 정보를 출력

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 특정 프로세스 검색
    - ● pgrep [옵션] [패턴]
      - ◆ 실습

```
adam@adam:~$ pgrep -x bash
2306
3422

adam@adam:~$ pgrep -l bash
2306 bash
3422 bash

adam@itstudy:~$ ps -fp $(pgrep -x bash)
UID          PID    PPID  C STIME TTY      STAT   TIME CMD
adam        1231    1060  0 Sep16 tty1     S+     0:00 -bash
adam        1328    1327  0 Sep16 pts/0    Ss     0:01 -bash
```

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 특정 프로세스 검색
    - ● pgrep [옵션] [패턴]
      - ◆ 실습

```
adam@itstudy:~$ ps -fp $(pgrep -u adam bash)
UID          PID    PPID  C STIME TTY      STAT   TIME CMD
adam        1231    1060  0 Sep16 tty1     S+     0:00 -bash
adam        1328    1327  0 Sep16 pts/0    Ss     0:01 -bash
```

프로세스 관리

- Process 관리 명령
  - ✓ 프로세스 종료
    - 프로세스를 종료하는 데는 kill이나 pkill 명령을 사용하는데 프로세스에 시그널을 보내 프로세스를 종료
    - 시그널
      - 프로세스에 무언가 발생했음을 알리는 간단한 메시지
      - 리눅스에서 지원하는 시그널 확인: kill –l

시그널	번호	기본 처리	의미
SIGHUP	1	종료	터미널과 연결이 끊겼을 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.

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 종료
    - kill
      - ◆ 프로세스를 종료하는 명령
      - ◆ 형식: kill [-시그널] PID
      - ◆ 시그널
        - 2: 인터럽트 시그널을 전송
        - 9: 프로세스를 강제로 종료
        - 15: 프로세스와 관련된 파일들을 정리하고 종료하는데 종료되지 않는 프로세스가 있을 수 있음
      - ◆ 실습
        - 터미널에서 man ps 명령 수행
        - 다른 터미널에서 pgrep -x man 이나 ps -ef | grep man 으로 PID 확인
        - 다른 터미널에서 kill PID

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 종료
    - ● pkill
      - ◆ pkill 명령도 kill 명령과 마찬가지로 시그널을 보내지만 PID가 아니라 프로세스의 명령 이름(CMD)으로 프로세스를 찾아 종료
      - ◆ kill 명령과의 차이점은 명령 이름으로 찾아 종료하므로 같은 명령이 여러 개 검색될 경우 한 번에 모두 종료한다는 것
      - ◆ 실습
        - ▪ 여러 터미널을 실행한 후 man ps 명령을 수행
        - ▪ adam@itstudy:~$ ps -fp $(pgrep -x man)

```
UID          PID    PPID  C STIME TTY      STAT   TIME CMD
adam        5676    5666  0 07:06 pts/2    S+     0:00 man ps
adam        5690    5608  0 07:06 pts/1    S+     0:00 man ps
```

        - ▪ adam@itstudy:~$ pkill man
        - ▪ adam@itstudy:~$ pgrep -x man

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 종료
    - killall
      - ◆ killall 명령도 pkill 명령처럼 프로세스의 명령 이름(CMD)으로 프로세스를 찾아 종료
      - ◆ 이름으로 실행 중인 모든 프로세스를 한 번에 종료할 수 있는데 해당 프로세스를 소유하고 있어야 함

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 정보를 주기적으로 출력
    - ● top
      - ◆ 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		

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 정보를 주기적으로 출력
    - ● top
      - ◆ top 명령에는 종료하지 않고 실시간으로 프로세스의 상태를 보여주며 내부적으로 사용하는 명령도 있음

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.

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 프로세스 정보를 주기적으로 출력
    - top

```
top - 01:21:02 up 15:25,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 159 total,   1 running, 158 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3901.8 total,   2222.9 free,    470.1 used,   1401.0 buff/cache
MiB Swap:   3901.0 total,   3901.0 free,      0.0 used.   3431.7 avail Mem
```

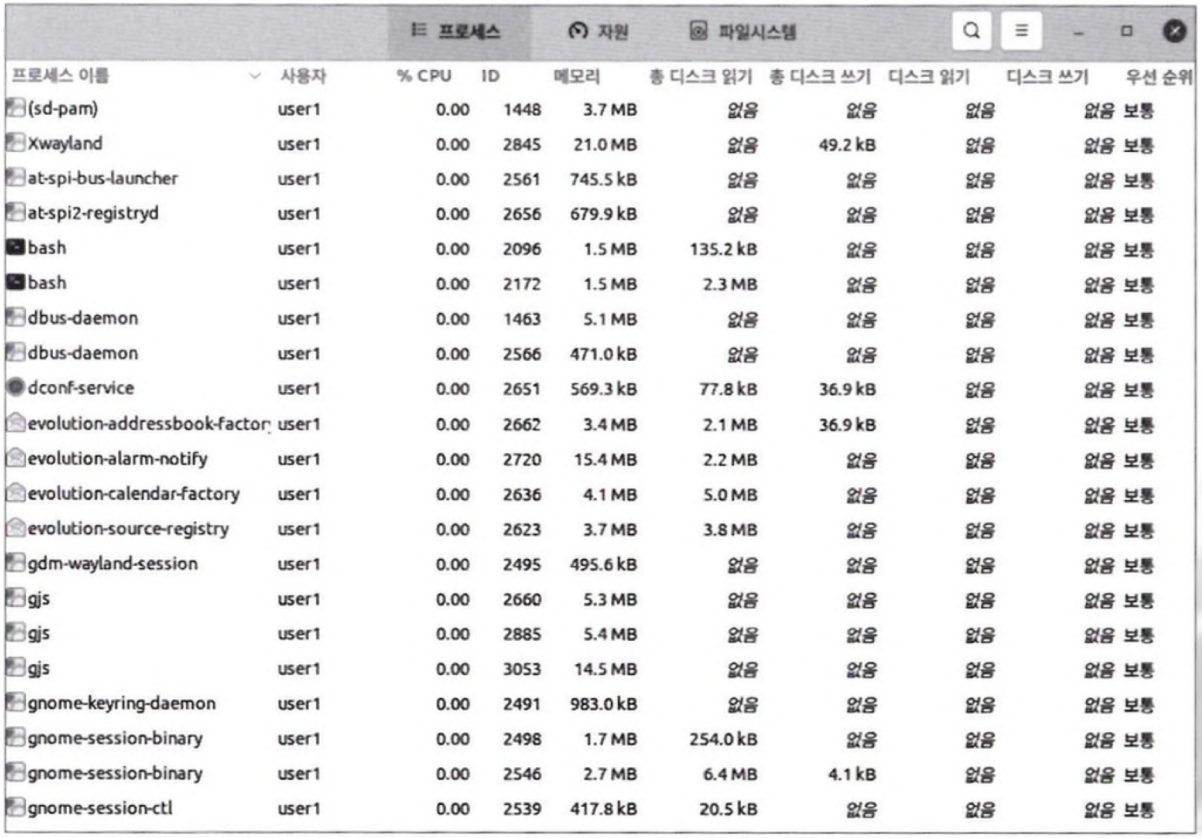
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
14617	root	20	0	0	0	0	I	0.7	0.0	0:02.80	kworker+
1	root	20	0	22432	12568	8600	S	0.0	0.3	0:06.87	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.06	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_wo+
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
12	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
16	root	20	0	0	0	0	S	0.0	0.0	0:00.07	ksoftir+
17	root	20	0	0	0	0	I	0.0	0.0	0:00.96	rcu_pre+
18	root	rt	0	0	0	0	S	0.0	0.0	0:01.28	migrati+
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_in+

프로세스 관리

- ❖ Process 관리 명령
  - ✓ 시스템 감시
    - 우분투의 GUI인 그놈에서 기본으로 제공하는 도구 중에 시스템 감시
    - 시스템 감시 화면은 프로세스 이름, 사용자, CPU 사용량, 메모리 사용량 등의 정보를 보여주며 [프로세스 끝내기 버튼]을 사용하여 프로세스를 종료할 수도 있음
    - 시스템 감시에서는 시스템이나 파일 시스템 관련 정보도 확인할 수 있어 윈도 환경에 익숙한 사용자들은 프로세스를 관리하는 데 편리하게 사용할 수 있음

프로세스 관리

- Process 관리 명령
  - ✓ 시스템 감시



프로세스 | 자원 | 파일시스템

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-factor	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

프로세스 관리

- ❖ 작업 제어
  - ✓ 포그라운드 작업
    - 터미널에서 작업할 때 일반적으로 사용자가 명령을 입력하면 셸이 그 명령을 해석하여 실행하고 결과를 화면에 출력하고 사용자는 화면에 출력된 결과를 보고 다시 명령을 입력하는 대화식으로 작업을 수행
    - 사용자가 입력한 명령이 실행되어 결과가 출력될 때까지 기다리는 방식으로 처리되는 프로세스가 포그라운드 프로세스

```
adam@adam:~$ sleep 100
```

프로세스 관리

- ❖ 작업 제어
  - ✓ 백그라운드 작업
    - 포그라운드 작업은 명령을 한 번에 하나씩 실행하므로 동시에 여러 개의 프로세스를 실행할 수 없지만 작업 제어가 제공하는 백그라운드 기능을 사용하면 다른 프로세스가 실행되는 동안에 뒤에서(백그라운드) 다른 프로세스가 실행될 수 있으므로 한 터미널에서 여러 개의 프로세스를 동시에 실행 가능
    - 백그라운드 방식으로 명령을 실행하면 명령의 처리가 끝나는 것과 관계없이 곧바로 프롬프트가 출력되어 사용자가 다른 작업을 계속할 수 있음
    - 여러 작업을 백그라운드로 실행한 후 터미널에서는 포그라운드 작업을 계속 진행할 수 있음 백그라운드 방식으로 처리되는 프로세스를 백그라운드 프로세스라고 하며 작업 제어에서는 이를 백그라운드 작업이라고 함
    - 백그라운드 작업은 명령의 실행 시간이 오래 걸릴 것으로 예상되거나 명령을 실행한 후 다른 작업을 할 필요가 있을 때 많이 사용
    - 명령을 백그라운드로 실행하려면 명령의 마지막에 & 기호를 추가
    - 백그라운드로 작업을 실행하면 프롬프트가 바로 나옴

```
adam@adam:~$ sleep 100 &
[1] 12598
adam@adam:~$
```

프로세스 관리

- ❖ 작업 제어
  - ✓ 백그라운드 작업
    - 백 그라운드 작업을 수행하게 되면 이후에 결과가 출력되므로 기존의 작업 화면과 백그라운드 작업 결과가 뒤섞인 채 터미널에 출력될 수 있으므로 출력하는 명령을 백그라운드 작업으로 수행하는 경우는 되도록 리다이렉션을 이용해서 출력 방향을 파일로 전환하는 것이 바람직

```
adam@adam:~$ find / -name passwd > pw.dat 2>&1 &
[1] 12598
```

프로세스 관리

- ❖ 작업 제어
  - ✓ 작업 목록 보기
    - jobs
      - ◆ 백그라운드 작업을 모두 보여주는데 특정 작업 번호를 지정하면 해당 작업의 정보만 출력하는 bash의 내부 명령
      - ◆ 형식 jobs [%작업 번호]
        - %번호: 해당 번호의 작업 정보를 출력
        - %+ 또는 %%: 작업 순서가 +인 작업 정보를 출력
        - %-: 작업 순서가 -인 작업 정보를 출력

```
adam@adam:~$ jobs
[1]+  Done                    sleep 100
```

프로세스 관리

- ❖ 작업 제어
  - ✓ 작업 목록 보기
    - ● jobs
      - ◆ 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 • −: + 작업 바로 전에 접근한 작업 • 공백: 그 외의 작업
상태	실행중	작업 상태를 표시한다. • 실행중: 현재 실행되고 있다. • 완료: 작업이 정상적으로 종료되었다. • 종료됨: 작업이 비정상적으로 종료되었다. • 멈춤: 작업이 잠시 중단되었다.
명령	sleep 100 &	백그라운드로 실행 중인 명령이다.

프로세스 관리

❖ 작업 제어

✓ 작업 전환

명령	기능
Ctrl+z	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.

프로세스 관리

❖ 작업 제어

✓ 작업 전환

- adam@itstudy:~$ jobs #백그라운드 작업이 없음
- adam@itstudy:~$ sleep 100 #포그라운드 작업 수행
- ^Z #포그라운드 작업 중지

[1]+ Stopped sleep 100

- adam@itstudy:~$ bg %1 #백그라운드 작업으로 전환

[1]+ sleep 100 &

- adam@itstudy:~$ jobs #백그라운드 작업 확인

[1]+ Running sleep 100 &

- adam@itstudy:~$ fg %1 #포그라운드 작업으로 전환

sleep 100

프로세스 관리

- 작업 제어
  - ✓ 작업 종료
    - 포그라운드 작업은 CTRL + c를 입력하면 대부분 종료되는데 인터럽트 시그널을 포그라운드 프로세스에 전달하며 인터럽트를 받으면 기본적으로 프로세스를 종료하도록 되어있는데 프로그램에서 무시하도록 설정한 경우에는 종료되지 않음
    - 포그라운드 작업을 종료하는 다른 방법은 다른 터미널에서 해당 프로세스의 PID를 찾아 강제로 종료하는 것
    - 실습
      - adam@itstudy:~$ sleep 100
      - ^C
      - adam@itstudy:~$ sleep 100&

      [1] 3296
      - adam@itstudy:~$ kill %1
      - adam@itstudy:~$ jobs

      [1]+ Terminated sleep 100

프로세스 관리

- 작업 제어
  - ✓ 로그아웃 후에도 백그라운드 작업 계속 실행하기
    - nohup 명령 &

프로세스 관리

- ❖ 작업 제어
  - ✓ 실습
    - sleep 명령을 두 개의 백그라운드로 실행
    - 현재 실행 중인 백그라운드 작업이나 정지된 작업을 확인
    - 백그라운드로 실행 중인 sleep 프로세스 중 1번 작업을 종료
    - 백그라운드 작업을 포그라운드 작업으로 전환
    - 포그라운드 작업을 강제 종료

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 한 번만 실행
    - 설치: sudo apt install at
    - 형식: at [옵션] [시각]
    - 옵션
      - -l: 현재 실행 대기 중인 명령의 전체 목록을 출력
      - -r 작업 번호: 현재 실행 대기 중인 명령 중 해당 작업 번호를 삭제
      - -m: 출력 결과가 없더라도 작업이 완료되면 사용자에게 메일로 알려줌
      - -f 파일: 표준 입력 대신 실행할 명령을 파일로 지정
      - -t [[CC]YY]MMDDhhmm[.ss]: 정확한 날짜 및 시간 지정
      - -c 작업 번호: 예약된 작업 내용 확인
      - -q <큐>
        - 큐 지정 (a ~ z, A ~ Z)

        at -q b 01:00
          - 큐마다 실행 우선순위 다름
          - a 기본 큐

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 한 번만 실행
    - ● 시간 설정
      - ◆ 절대 시간
        - ▪ at 14:30
        - ▪ at 2026-01-10 03:00
      - ◆ 상대 시간
        - ▪ at 4pm + 3days: 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
        - ▪ at now + 10 minutes
        - ▪ at now + 2 hours
        - ▪ at now + 1 day
      - ◆ 요일 지정
        - ▪ at 10am next Monday

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 한 번만 실행
    - 작업 확인
      - ◆ ls –l /var/spool/cron/atjobs 명령으로 확인하면 작업과 관련된 파일이 생성되는데 이 파일은 예약된 명령이 수행되면 삭제됨
        - adam@itstudy:~$ sudo ls -l /var/spool/cron/atjobs

        total 4

        -rwx------ 1 adam daemon 3195 Sep 20 01:29 a0000101b727bb
      - ◆ at –l
        - adam@itstudy:~$ at -l

        1 Fri Sep 20 10:35:00 2024 a adam
      - ◆ atq
        - adam@itstudy:~$ atq

        1 Fri Sep 20 10:35:00 2024 a adam
    - 예약된 작업 삭제
      - ◆ at –r 작업번호
      - ◆ atrm 작업번호
    - 특정 스크립트 실행: at -f /root/job.sh 01:00

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 한 번만 실행
    - ● 실습(명령 작성 후 종료는 CTRL + d)

```
adam@adam:~$ at 12:55 pm
warning: commands will be executed using /bin/sh
at Thu Sep 28 12:55:00 2023
at> /usr/bin/ls > ~adam/at.out
at> <EOT>
job 1 at Thu Sep 28 12:55:00 2023

adam@adam:~$ sudo ls -l /var/spool/cron/atjobs
total 4
-rwx------ 1 adam daemon 2972 Sep 28 00:53 a0000101af4a87

adam@adam:~$ at -l
1       Thu Sep 28 12:55:00 2023 a adam
```

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 한 번만 실행
    - ● 실습(명령 작성 후 종료는 CTRL + d)

```
adam@adam:~$ vim at.out
```

```
at.out
darknet
Desktop
Documents
Downloads
ex
hello.c
HelloJava.class
HelloJava.java
hellopython.py
helloworld
if1.sh
Music
name.sh
Pictures
Public
Templates
var1.sh
Videos
```

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 한 번만 실행
    - ● 사용 제한
      - ◆ 시스템 관리자는 일반 사용자들이 at 명령을 사용하도록 허용하거나 사용하지 못하도록 제한할 수 있는데 이와 관련된 파일은 /etc/at.allow와 /etc/at.deny
      - ◆ at 명령의 사용이 허용된 사용자들은 /etc/at.allow 파일에 지정하고 at 명령의 사용이 금지된 사용자들은 /etc/at.deny 파일에 지정
      - ◆ at.deny 파일은 기본적으로 있지만 at.allow 파일은 없으므로 필요할 때에 관리자가 만들어야 하는데 /etc/at.allow 파일과 /etc/at.deny 파일에는 사용자 이름을 한 줄에 하나씩만 입력해야 함
      - ◆ 두 파일의 적용 기준
        - ▪ /etc/at.allow 파일이 있으면 이 파일에 지정된 사용자만 at 명령을 사용할 수 있는데 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
        - ▪ 사용자가 두 파일 모두에 속해 있다면 at 명령을 사용할 수 있음
        - ▪ /etc/at.deny를 빈 파일로 두면 모든 사용자가 at 명령을 사용할 수 있는데 이것이 초기 설정

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - crontab
    - 형식: crontab [-u 사용자ID] [옵션] [파일명]
    - 옵션
      - -e: 사용자의 crontab 파일을 편집
      - -l: crontab 파일의 목록을 출력
      - -r: crontab 파일을 삭제
    - crontab 명령으로 관리하는 파일은 사용자 별로 생성되는데 이 파일에 반복 실행할 작업을 저장
    - crontab 파일에는 여러 개의 작업을 저장할 수 있으며 한 행에 하나의 작업을 설정

프로세스 관리

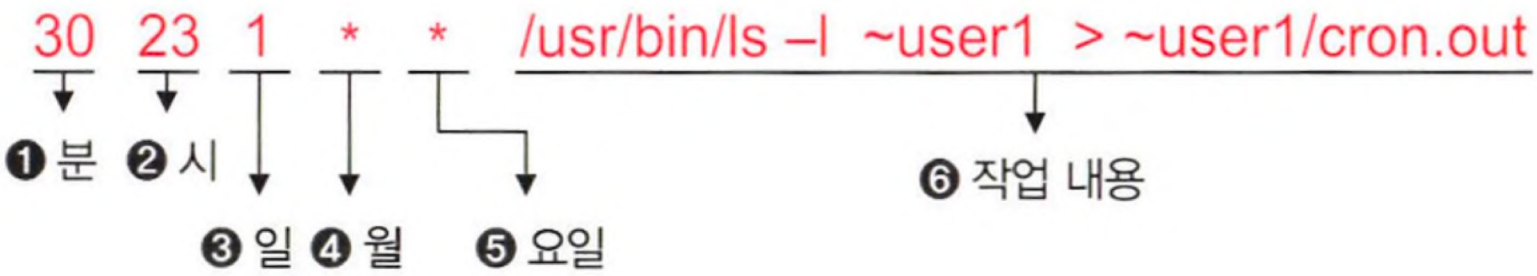
- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - crontab 파일의 한 행은 여섯 항목으로 구성

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

- ◆ 앞에서부터 다섯 항목은 시간과 날짜를 나타내는 숫자이고 마지막 항목은 반복적으로 수행할 명령
- ◆ 요일은 0이 일요일, 1이 월요일, 6이 토요일을 의미
- ◆ 각 항목은 공백 문자 로 구분
- ◆ 항목의 값이 *이면 해당 항목의 모든 값을 의미
- ◆ -: 하이픈 연산자를 사용하면 값의 범위를 지정할 수 있는데 요일 필드에서 1-5를 설정하면 작업이 매주 평일(월요일부터 금요일까지) 실행되는데 첫 번째 및 마지막 값이 범위에 포함됨을 의미
- ◆ ,: 쉼표 연산자를 사용하면 반복에 대한 값 목록을 정의할 수 있는데 시간 필드에 1,3,5가 있는 경우 태스크는 오전 1시, 3시 및 5시에 실행
- ◆ /: 슬래시 연산자를 사용하면 범위와 함께 사용할 수 있는 단계 값을 지정할 수 있는데 분 필드에 1-10/2가 있으면 1,3,5,7,9를 지정하는 것과 마찬가지로 1-10 범위에서 작업이 2분마다 수행되며 값의 범위 대신 별표 연산자를 사용할 수도 있어서 20분마다 실행할 작업을 지정하려면 */20을 사용

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - ● crontab 파일의 한 행은 여섯 항목으로 구성



- ◆ 매월 1일 23시 30분에 지정한 작업을 실행

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - ● 시간 설정 예시
      - ◆ 45 22 * * * — 22시45분에 실행
      - ◆ 0 17 * * 1 — 매주 월요일 17시00분에 실행
      - ◆ 0,10 17 * * 0,2,3 — 매주 일, 화, 수요일 17시00 분과 17시10분에 실행
      - ◆ 0-10 17 1 * * — 매달 1일 17시00분부터17시10분까지 1분 단위로 실행
      - ◆ 0 0 1,15 * 1 — 매달 1일과 15일 그리고 월요일 24시00분에 실행
      - ◆ 42 4 1 * * — 매달 1일 4시42분에 실행
      - ◆ 0 21 * * 1-6 — 월요일 부터 토요일까지 21시00분에 실행
      - ◆ 0,10,20,30,40,50 * * * * — 10분 간격으로 실행
      - ◆ */10 * * * * — 10분 간격으로 실행
      - ◆ * 1 * * * — 1시00분 부터 1시59분 까지 1분 간격으로 실행
      - ◆ 0 */1 * * * — 매시간 0분에 실행(1시간 간격으로 실행)
      - ◆ 0 * * * * — 매시간 0분에 실행(1시간 간격으로 실행)
      - ◆ 5 2-5 * * * — 2시 5분, 3시 5분, 4시 5분, 5시 5분에 실행
      - ◆ 2 8-20/3 * * * — 8시 2분, 11시 2분, 14시 2분, 17시 2분, 20시 2분에 실행
      - ◆ 30 5 1,15 * * — 매달 1일과 15일 5시30분에 실행

프로세스 관리

- 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - 파일 내용 확인: crontab -l
    - 파일 삭제: crontab -r

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - ● 실습
      - ◆ adam@itstudy:~$ crontab -e

no crontab for adam - using an empty one

Select an editor. To change later, run 'select-editor'.

1. /bin/nano   <---- easiest
2. /usr/bin/vim.basic
3. /usr/bin/vim.tiny
4. /bin/ed

Choose 1-4 [1]: 1

crontab: installing new crontab

◆ 58 2 * * * /usr/bin/ls -l > ~adam/cron.out 을 작성하고 저장

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - ● 실습
      - ◆ 내용 확인
        - adam@itstudy:~$ crontab -l

```
# m h  dom mon dow   command

58 2 * * * /usr/bin/ls -l > ~adam/cron.out
```

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - ● 실습
      - ◆ 실행 확인
        - ▪ adam@itstudy:~$ ls -l

        -rw-rw-r-- 1 adam adam 109 Sep 20 02:58 cron.out

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - ● 연습
      - ◆ 현재 날짜와 시간을 확인
      - ◆ at 명령으로 현재 시간보다 5분 후에 ls –l /tmp > ~계정/at.tmp를 실행하도록 설정
      - ◆ at 명령으로 설정한 내용을 확인
      - ◆ 5분 후에 at 명령이 실행되었는지 확인
      - ◆ crontab 명령으로 매주 일요일 18시에 현재 계정의 홈 디렉터리 내용을 출력하여 홈 디렉터리 아래에 weekout 파일로 저장하도록 설정
      - ◆ crontab 명령으로 설정된 내용을 확인
      - ◆ crontab 명령으로 설정된 내용의 실행 결과를 확인

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - crontab 관련 파일
      - ◆ cronid(패키지) = crond(데몬) + crontab(cron table)
      - ◆ /var/spool/cron 디렉토리
        - root 계정용 하나 와 계정 사용자당 1개의 파일을 소유
        - 파일명은 사용자의 계정명이며 cron은 이 이름을 바탕으로 설정 파일에 지정된 작업을 실행 시 사용할 UID를 결정
        - 이 곳에 있는 설정 파일들은 crontab 명령어로 관리
      - ◆ /etc/cron.d: 소프트웨어 패키지를 설치할 때 필요한 주기적 작업을 등록하는 공간으로 사용
      - ◆ /etc/crontab: 관리자가 직접 지정한 작업을 설정하며 임의의 사용자 권한으로 실행할 수 있는데 시스템 관련 작업을 등록하는 곳
      - ◆ /etc/cron.allow: 파일 내 지정된 사용자만 crontab을 등록할 수 있는데 지정하지 않은 사용자는 crontab 명령을 실행할 수 없음
      - ◆ /etc/cron.deny: 허용 파일이 없는 경우에는 이 deny 파일을 사용해서 명령을 실행할 수 없는 사용자를 지정
      - ◆ 데비안, 우분투 배포판은 모든 사용자에게 실행 권한을 부여

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - crontab 로그
      - ◆ 우분투 시스템에서는 기본적으로 cron 로그 기록이 비활성화 되어 cron.log가 syslog에 통합되는데 cron.log를 활성화하고 싶은 경우에는 /etc/rsyslog.d/50-default.conf 파일에 접근하여 cron 로그를 활성화 시켜 주면 됨
      - ◆ syslog 재시작: sudo service rsyslog restart

```
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
#daemon.*                       -/var/log/daemon.log
kern.*                          -/var/log/kern.log
#lpr.*                          -/var/log/lpr.log
mail.*                          -/var/log/mail.log
#user.*                         -/var/log/user.log
```

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - ● crontab 연습
      - ◆ 매주 토요일 오후 1시 1분마다 /etc/init.d/rsyslog restart 하는 작업을 스케줄링

        1 13 * * 6 /usr/bin/systemctl restart rsyslog

      - ◆ 1월~10월까지 2개월마다 1일 12시에 Its system check time. 출력

        0 12 1 1-10/2 * /usr/bin/echo 'Its system check time.

      - ◆ 매월 월,수,금 12시 마다 /var/log/ 모든 파일 지워라.

        0 12 * * 1,3,5 /usr/bin/rm -r /var/log/*

      - ◆ 매일 오후 2시 20분에 시간 동기화를 수행한 후 곧바로 OS reboot

        20 14 * * * /usr/sbin/clock -w && /usr/sbin/shutdown -r now

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - crontab 연습
      - ◆ 3일에서 5일까지 5시,6시,7시에 매 5분마다 time.sh 를 실행

        */05 5,6,7 3-5 * * /bin/bash /root/LABs/time.sh
      - ◆ 매주 금요일 오전 7시 15분에 time.sh 를 실행

        15 7 * * 5 /bin/bash /root/LABs/time.sh

프로세스 관리

- ❖ 작업 예약
  - ✓ 정해진 시간에 반복 실행
    - ● crontab 연습
      - ◆ /var/log 디렉토리의 내용을 년-월-일-backup.tar.gz 압축해서 저장하기
        - ▪ #sudo su
        - ▪ sudo vi /usr/bin/log_backup.sh

          #!/bin/bash

          set $(date)

          fname="$6-$2-$3-backup"

          tar -cvzf /BACKUP/$fname.tar.gz /var/log
        - ▪ sudo chmod 700 /usr/bin/log_backup.sh
        - ▪ sudo mkdir /BACKUP
        - ▪ crontab –e

          00 12 */1 * * /usr/bin/log_backup.sh