



Linux Process 관련 정리

1. 프로세스 동작 원리

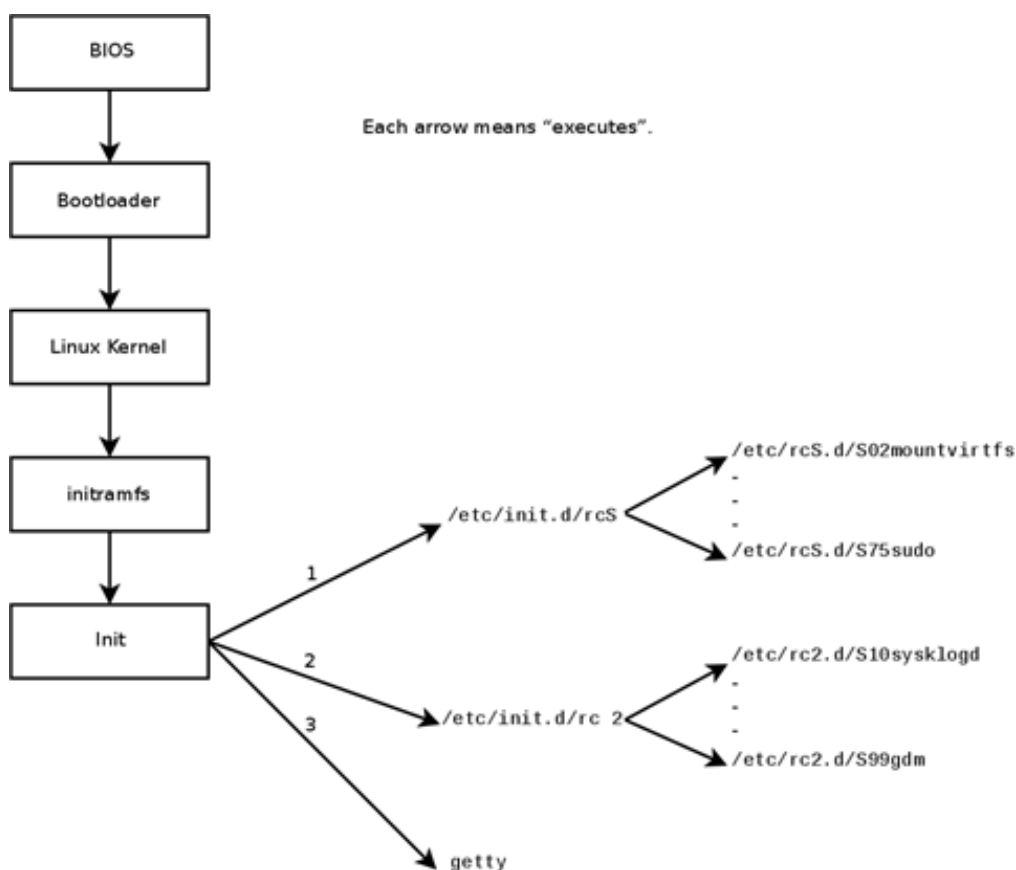
시스템이 구동 될 때, 커널은 /etc 에 위치한 init 이라는 스크립트를 실행함으로써 시스템 서비스들을 차례대로 시작 시킨다.

이 서비스들은 데몬 프로그램(백그라운드)으로 구현되어 있기 때문에 로그인하지 않은 상태에서도 필요 작업들을 수행한다.

프로그램은 프로그램을 실행시킬 수 있는데, 이를 **부모와 자식 프로세스**라고 표현한다.

커널은 이러한 프로세스들을 **구조화시킨 형태로** 유지하기 위해서 **PID** 라는 **프로세스 ID** 정보를 가지고 있다.

👉 예를들어, 위에서 말한 init은 항상 1번 PID 를 할당 받는다.



2. 프로세스 명령어

✓ 정보확인

▼ 1. ps

▼ 설명

- 시스템에서 실행중인 프로세스에 관한 정보를 보여주는 도구
- /proc디렉터리 이하에 프로세스와 연관된 가상 파일시스템의 내용을 토대로 프로세스 정보를 출력

▼ 사용법

```
ps [option]
```

▼ 옵션

• 기본 프로세스 출력

옵션	내용
a	현재 사용자 프로세스 출력
x	터미널과 연관된 프로세스만 출력
-A	모든 프로세스 출력 (-e와 동일)
-e	모든 프로세스 출력
-a	세션 리더와 터미널과 연관되지 않은 프로세스를 제외하고 모든 프로세스를 출력

• 지정한 프로세스 출력

옵션	내용
p	지정한 PID 목록의 정보만 출력
-C	지정한 프로세스의 실행 파일 이름의 정보만 출력
-u	특정 사용자의 프로세스 정보를 출력

• 프로세스 표시 형식

옵션	내용
u	프로세스의 소유자 정보를 함께 출력
l	BSD 형식의 긴 형식으로 출력
e	프로세스 정보와 함께 프로세스의 환경변수 정보도 출력
-l	긴 포맷으로 출력
-0	사용자 정의 형식 지정 가능

• 프로세스 장식

옵션	내용
f	프로세스 계층을 텍스트 형식의 트리구조를 보여줌
-f	전체 포맷으로 출력

▼ 예제

1. ps ax

```
[root@Linux-1 ~]# ps ax
  PID TTY          STAT       TIME COMMAND
    1 ?        Ss         0:01 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
    2 ?        S           0:00 [kthreadd]
    4 ?        S<          0:00 [kworker/0:0H]
    6 ?        S           0:03 [ksoftirqd/0]
    7 ?        S           0:00 [migration/0]
    8 ?        S           0:00 [rcu_bh]
    9 ?        R           0:01 [rcu_sched]
   10 ?        S<          0:00 [lru-add-drain]
```

2. ps aux

```
[root@Linux-1 ~]# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3 125396 3888 ?        Ss   1월24   0:01 /usr/lib/systemd/systemd --switched-root --system -
root         2  0.0  0.0      0     0 ?        S    1월24   0:00 [kthreadd]
root         4  0.0  0.0      0     0 ?        S<   1월24   0:00 [kworker/0:0H]
```

root	6	0.0	0.0	0	0 ?	S	1월24	0:03	[ksoftirqd/0]
root	7	0.0	0.0	0	0 ?	S	1월24	0:00	[migration/0]
root	8	0.0	0.0	0	0 ?	S	1월24	0:00	[rcu_bh]
root	9	0.0	0.0	0	0 ?	R	1월24	0:01	[rcu_sched]
root	10	0.0	0.0	0	0 ?	S<	1월24	0:00	[lru-add-drain]
root	11	0.0	0.0	0	0 ?	S	1월24	0:00	[watchdog/0]
root	13	0.0	0.0	0	0 ?	S	1월24	0:00	[kdevtmpfs]
root	14	0.0	0.0	0	0 ?	S<	1월24	0:00	[netns]
root	15	0.0	0.0	0	0 ?	S	1월24	0:00	[khungtaskd]

3. ps aux | grep apache

```
[root@Linux-1 ~]# ps aux | grep apache
root      9701  0.0  0.1 116972 1024 pts/0    R+   09:12   0:00 grep --color=auto apache
```

4. ps -ef | more

```
[root@Linux-1 ~]# ps -ef | more
UID          PID    PPID  C STIME TTY          TIME CMD
root          1         0  0  1월24 ?        00:00:01 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
root          2         0  0  1월24 ?        00:00:00 [kthreadd]
root          4         2  0  1월24 ?        00:00:00 [kworker/0:0H]
root          6         2  0  1월24 ?        00:00:03 [ksoftirqd/0]
root          7         2  0  1월24 ?        00:00:00 [migration/0]
root          8         2  0  1월24 ?        00:00:00 [rcu_bh]
root          9         2  0  1월24 ?        00:00:01 [rcu_sched]
root         10         2  0  1월24 ?        00:00:00 [lru-add-drain]
root         11         2  0  1월24 ?        00:00:00 [watchdog/0]
root         13         2  0  1월24 ?        00:00:00 [kdevtmpfs]
root         14         2  0  1월24 ?        00:00:00 [netns]
root         15         2  0  1월24 ?        00:00:00 [khungtaskd]
root         16         2  0  1월24 ?        00:00:00 [writeback]
root         17         2  0  1월24 ?        00:00:00 [kintegrityd]
root         18         2  0  1월24 ?        00:00:00 [bioset]
root         19         2  0  1월24 ?        00:00:00 [bioset]
root         20         2  0  1월24 ?        00:00:00 [bioset]
root         21         2  0  1월24 ?        00:00:00 [kblockd]
root         22         2  0  1월24 ?        00:00:00 [md]
```

▼ 2. pstree

▼ 설명

- 리눅스 명령어 pstree는 프로세스의 상관관계(부모-자식 관계)를 트리 형태로 출력해주는 명령어
- 관계를 트리 형태로 출력해주므로 계층 관계를 한 눈에 파악할 수 있다.

▼ 사용법

```
pstree [-a] [-c] [-h] [-n] [-p] [-u]
```

▼ 옵션

옵션	내용
-a	실행한 프로세스의 인자와 옵션까지 모두 표시하는 옵션 입니다.
-c	1개의 프로세스의 중복된 개수로 출력하는 옵션 입니다.
-h	부모 프로세스를 강조하여 출력하는 옵션 입니다.
-n	출력시 PID 순서대로 정렬하여 출력하는 옵션 입니다.
-p	PID를 출력하는 옵션 입니다.
-u	UID를 출력하는 옵션 입니다
-V	버전 정보를 출력하는 옵션 입니다

▼ 예제

1. pstree

```
[root@oignon ~]# pstree
init--abrt-dump-oops
      |
      |--abrttd
      |--acpid
      |--atd
      |--auditd--{auditd}
      |--automount--4*[{automount}]
      |--certmonger
      |--crond
      |--dbus-daemon--{dbus-daemon}
      |--fail2ban-server--4*[{fail2ban-serve}]
      |--gam_server
      |--hald--hald-runner--hald-addon-acpi
      |          |
      |          |--hald-addon-inpu
      |          |
      |          |--{hald}
      |--httpd--8*[httpd]
      |--irqbalance
      |--mcelog
      |--6*[mingetty]
      |--mysqld_safe--mysqld--15*[{mysqld}]
      |--rsyslogd--3*[{rsyslogd}]
      |--sshd--sshd--bash--pstree
      |--udevd--2*[udevd]
      |--vpnservice--vpnservice--83*[{vpnservice}]
      |--vsftpd
```

👉 init에서 파생된 자식 프로세스

2. patree -ap

```
[root@oignon ~]# patree -ap
init,1
|--abrt-dump-oops,1630 -d /var/spool/abrt -rwx /var/log/messages
|--abrttd,1620
|--acpid,1447
|--atd,1697
|--auditd,1312
|   |--{auditd},1313
|--automount,1526 --pid-file /var/run/autofs.pid
|   |--{automount},1527
|   |--{automount},1528
|   |--{automount},1531
|   |--{automount},1534
|--certmonger,1713 -S -p /var/run/certmonger.pid
|--crond,1642
|--dbus-daemon,1408 --system
|   |--{dbus-daemon},1410
|--fail2ban-server,2557 /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock
|   |--{fail2ban-serve},2558
|   |--{fail2ban-serve},2559
|   |--{fail2ban-serve},2561
|   |--{fail2ban-serve},2562
|--gam_server,1556
```

▼ 3. top

▼ 설명

- 시스템에서 실행중인 프로세스에 관한 정보를 보여주는 도구
- /proc디렉터리 이하에 프로세스와 연관된 가상 파일시스템의 내용을 토대로 프로세스 정보를 출력

▼ 사용법

```
top [option]
```

▼ 옵션

- top 실행 후 사용할 수 있는 옵션

옵션	내용
shift + t	실행된 시간이 큰 순서로 정렬
shift + m	메모리 사용량이 큰 순서로 정렬

옵션	내용
shift + p	cpu 사용량이 큰 순서로 정렬
k	Process 종료
c	명령 인자 표시 / 비표시
l	uptime line(첫번째 행)을 표시 / 비표시
space bar	Refresh
u	입력한 유저 소유의 Process만 표시
shift + b	상단의 uptime 및 기타 정보값을 블락선택해 표시
f	화면에 표시될 프로세스 관련 항목 설정
i	idle 또는 좀비 상태의 프로세스는 표시 되지 않음
z	출력 색상 변경
d	설정된 초단위로 Refresh
c	command뒤에 인자값 표시
q	명령어 종료

- **top 실행 전 옵션 : top의 정보들을 서식으로 출력하기 위한 옵션**

옵션	내용
-a	메모리 사용에 따라 정렬
-b	배치 모드에서 시작
-c	명령어 대신 명령어 라인을 보여줌
-d	업데이트 간격을 조정
-h	도움말
-H	모든 개별 쓰레드가 보여짐
-i	좀비(zombie) 또는 Idle 상태의 것들은 무시됨
-m	VIRT 대신 USED를 보고
-M	메모리 유닛(K/M/G)을 보여줌
-n	반복의 최대 수를 지정
-p	지정된 프로세스 ID들만 보여줌
-s	보안 모드로 시작
-S	누적 시간 모드로 시작. 활성화되면 각 프로세스는 CPU를 사용한 시간과 함께 출력
-u	지정된 유효 사용자에게 의한 프로세스만 보여줌
-U	지정된 사용자에게 의한 프로세스만 보여줌. 사용자는 실제, 유효한, 저장된 및 파일시스템 UID를 의미
-v	프로그램 라이브러리 버전을 출력

▼ 예제

1. top

```
top - 09:25:23 up 1 day, 14:25,  2 users,  load average: 0.00, 0.01, 0.05
Tasks:  97 total,   1 running,  96 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,   0.0 sy,   0.0 ni,100.0 id,   0.0 wa,   0.0 hi,   0.0 si,   0.0 st
KiB Mem :  995464 total,   711656 free,   158080 used,   125728 buff/cache
KiB Swap: 2097148 total, 2097148 free,        0 used.  699280 avail Mem

  PID USER  PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
    1 root    20   0 125396   3888   2588 S   0.0   0.4    0:01.50   systemd
    2 root    20   0      0      0      0 S   0.0   0.0    0:00.01   kthreadd
    4 root     0 -20      0      0      0 S   0.0   0.0    0:00.00 worker/0:0H
```

2. `top -d 1 | egrep "PID|systemd"`

```
[root@Linux-2 ~]# top -d 1 | egrep "PID|systemd"
PID USER  PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
    1 root    20   0 125396   3888   2588 S   0.0   0.4    0:01.50   systemd
```


1	root	20	0	125348	3868	2588	S	0.0	0.4	0:01.43	systemd
1	root	20	0	125348	3868	2588	S	0.0	0.4	0:01.43	systemd

✓ 제어

▼ 1. kill

▼ 설명

- kill 명령어는 **프로세스를 강제로 종료**시킨다.
좀비 프로세스나, 응답 없음, 비정상적으로 동작하는 프로세스들의 실행을 끝내게 해준다.
- 프로세스를 포어그라운드로 가져와서 ctrl + C 할 수도 있지만, PID 를 확인하여 kill로 바로 종료시킬 수도 있다.

 **시그널**

- 시그널은 프로세스 사이의 통신 수단인데 어떤 프로세스에 메시지를 보내 프로세스를 제어한다.
- 명령어를 실행함으로써 프로세스가 시작되고 그 프로세스를 제어하기 위하여 사전에 정의된 시그널이 존재한다.
- 몇 가지 시그널이 존재하는데 **불필요한 프로세스, 잘못 실행된 프로세스를 죽이는데 사용할 것은 KILL**이었다.

▼ 사용법

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

▼ 옵션

옵션	내용
pid ...	종료시킬 프로세스 ID나 프로세스 이름을 지정한다.
-s	전달할 시그널의 종류를 지정 🖱 시그널 이름이나 번호를 작성
-l	시그널로 사용할 수 있는 시그널 이름들을 보여준다.
-1	HUP 프로세스를 재할성화한다.
-9	프로세스를 강제로 종료시킨다.

▼ 예제

1. kill pid

```
ps aux | grep sshd
root      1015  0.0  0.4 113000  4340 ?        Ss   02:00   0:00 /usr/sbin/sshd -D
root      1276  0.0  0.6 161820  6236 ?        Ss   02:00   0:00 sshd: root@pts/0
root      9377  0.0  0.1 116972  1024 pts/0    R+   03:21   0:00 grep --color=auto sshd
kill 1015
ps aux | grep sshd
root      1276  0.0  0.6 161820  6236 ?        Ss   02:00   0:00 sshd: root@pts/0
root      9379  0.0  0.1 116972  1020 pts/0    R+   03:21   0:00 grep --color=auto sshd
```

2. kill -9

```
kill -9 1015
```

2. 프로세스를 종료 후 다시 재실행

```
kill -HUP pid
= kill -1 pid
= kill -SIGHUP pid
```

2. kill -l

```
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
```

▼ 2. killall

▼ 설명

- 같은 데몬의 여러 프로세스를 한 번에 종료시킬 때 사용하는 명령
- 프로세스명을 사용한다.
- 기본적인 사용법은 kill 명령과 유사하다.
- 시그널을 지정하지 않으면, 종료 시그널(TERM, SIGTERM)이 전송된다.

▼ 사용법

```
killall [option] 프로세스명
```

▼ 옵션

옵션	내용
-e	종료하려는 프로세스 이름이 매우 긴 경우 사용한다. killall은 15글자가 넘어가면, 그 이후의 단어는 무시하고 앞에 15글자가 같은 프로세스를 모두 종료한다. 따라서 아주 긴 이름을 갖는 프로세스에 유용하다.
-g	그룹을 지정하여 프로세스를 종료시킨다. 같은 프로세스 그룹에 속한 여러 프로세스가 발견되더라도 시그널은 그룹별로 한 번만 보내진다.
-i	프로세스 종료 전 확인 메시지를 출력한다.
-l	알려진 모든 시그널 이름을 출력한다.
-q	어떤 메시지도 출력하지 않는다.
-v	시그널을 제대로 보낸 경우, 상세한 정보를 출력한다.
-V	버전 정보를 출력한다.
-w	프로세스가 종료될 때까지 대기한다.

▼ 예제

1. killall [프로세스명]

```
ping google.com > /dev/null 2>&1 &
[3] 1299
killall ping
[1]  종료됨          ping google.com > /dev/null 2>&1
[2]-  종료됨          ping google.com > /dev/null 2>&1
[3]+  종료됨          ping google.com > /dev/null 2>&1
```

2. killall -l

- 사용 가능한 시그널 리스트 출력

```
killall -l
HUP INT QUIT ILL TRAP ABRT IOT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM TERM
STKFLT CHLD CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM PROF WINCH IO PWR SYS
UNUSED
```

3. killall -q

- -q 옵션으로 불평의 미출력 가능

```
killall -q ping
[1]   종료됨          ping google.com > /dev/null 2>&1
[2]   종료됨          ping google.com > /dev/null 2>&1
[3]-  종료됨          ping google.com > /dev/null 2>&1
[4]+  종료됨          ping google.com > /dev/null 2>&1
killall ping
ping: no process found
```

▼ 3. pkill

▼ 설명

- 프로세스명을 사용해서 특정 프로세스에 시그널을 보내는 명령
- kill 명령과 같이 기본 시그널은 15번 시그널(TERM, SIGTERM)이다.
- 프로세스명과 사용자 및 그룹명 등으로 프로세스를 종료시킬 수 있다.

▼ 사용법

```
pkill [option] [pattern]
```

▼ 옵션

옵션	내용
-u	특정 사용자가 실행시킨 프로세스의 PID에 시그널을 보냄
-U	특정 UID를 갖는 사용자가 실행시킨 프로세스의 PID에 시그널을 보냄
-g	특정 그룹이 실행시킨 프로세스의 PID에 시그널을 보냄
-G	특정 GID를 갖는 그룹이 실행시킨 프로세스의 PID에 시그널을 보냄
-t	특정 터미널에 실행 중인 프로세스의 PID에 시그널을 보냄

▼ 예제

1. pkill -15 pid

```
ping google.com > /dev/null 2>&1 &
[1] 1311
...생략...
ping google.com > /dev/null 2>&1 &
[6] 1316
pkill -15 ping
[1]   종료됨          ping google.com > /dev/null 2>&1
[2]   종료됨          ping google.com > /dev/null 2>&1
[3]   종료됨          ping google.com > /dev/null 2>&1
[4]   종료됨          ping google.com > /dev/null 2>&1
[5]-  종료됨          ping google.com > /dev/null 2>&1
[6]+  종료됨          ping google.com > /dev/null 2>&1
```


2. kill -9 [pattern]

- 지정된 패턴과 일치하는 프로세스를 종료

```
kill -9 firefox
```

3. pkill -u

- pkill이 지정된 사용자에게 의해 실행되는 프로세스와 일치하도록 지시

```
pkill -u mark

# 🗨 여러 명의 사용자를 지정하려면 해당 사용자의 이름을 쉼표로 구분
pkill -u mark,danny
```

4. pkill -n

- 가장 최근에 시작된 프로세스만 표시

```
pkill -9 -n screen
```

✓ 전환 (foreground / background)

foreground

입력한 명령어 실행이 결과가 나올 때까지 기다리는 방식이다.

한 동작을 수행할 동안 다른 동작을 할 수 없는 상태이다.

background

하나의 셸(터미널)에서 여러 개의 프로세스를 동시에 실행할 수 있는 방식이다.

한 동작을 수행하는 동안에도 다른 동작을 할 수 있는 상태이다.

해당 명령어 처리가 오래 걸릴 걸 대비해 백그라운드를 이용하면 여러 작업을 동시에 수행할 수 있다.

▼ 1. jobs

▼ 설명

- 백그라운드로 실행되는 작업 목록(작업 번호, 상태, 명령어)을 보여주는 명령어이다.
- 현재 셸(터미널)에서 작업이 중지된 상태나 백그라운드로 진행 중인 상태를 출력하는 명령어이다.

▼ 사용법

```
jobs [옵션] [작업번호]
```

▼ 옵션

옵션	내용
-l	프로세스 그룹 ID를 state 필드 앞에 출력
-n	프로세스 그룹 중에 대표 프로세스 ID를 출력

옵션	내용
-p	각 프로세스 ID에 대해 한 행씩 출력
-r	실행 중인 작업만 출력
-s	정지한 작업만 출력
command	지정한 명령어를 실행

▼ 예제

1. 기본

test.txt와 test1.txt 파일 만들기 (둘 다 비어있는 파일)

```
[root@Linux-2 ~]# cat > test.txt
^Z
[1]+  Stopped                  cat > test.txt
[root@Linux-2 ~]# jobs
[1]+  Stopped                  cat > test.txt
[root@Linux-2 ~]# cat > test1.txt
^Z
[2]+  Stopped                  cat > test1.txt
[root@Linux-2 ~]# jobs
[1]-  Stopped                  cat > test.txt
[2]+  Stopped                  cat > test1.txt

※ +는 스택의 가장 위에 있다는 뜻이고 -는 바로 그 다음 밑에 있다는 뜻이다.
```

2. -l

필드 값 : 작업 번호, 작업 순서, PID, 상태, 명령

```
[root@Linux-2 ~]# jobs -l
[1]-  8093 정지됨              cat > test.txt
[2]+  8108 정지됨              cat > test1.txt
```

3. -p

PID 출력

```
[root@Linux-2 ~]# jobs -p
8093
8108
```

4. job 번호

job 1번 작업 출력

```
[root@Linux-2 ~]# jobs 1
[1]-  Stopped                  cat > test.txt
```

5. -s

정지한 작업 출력

```
[root@Linux-2 ~]# jobs -s
[1]-  Stopped                  cat > test.txt
[2]+  Stopped                  cat > test1.txt
```

▼ 2. & (후면 처리)

▼ 설명

- 명령어들을 후면에서 처리하고 전면에서는 다른 작업을 할 수 있으며 동시에 여러 작업을 수행할 수 있다.

- 백그라운드는 명령어의 맨 끝에 &를 붙여 사용하면 후면 처리로 명령어가 실행된다.

▼ 사용법

```
(명령어;) &
```

▼ 예제

1. 기본

```
[root@Linux-2 ~]# sleep &
[3] 8419

※ [3]은 백그라운드 프로세스를 돌릴 때마다 부여되는 job 번호를 의미한다.
841는 프로세스 번호를 의미한다.
+ 프로세스는 실행되자마자 비동기적으로 실행되며 시간이 지나면 동시에 프로세스가 끝난다.
```

▼ 3. bg (foreground → background)

▼ 설명

- 프로그라운드 프로세스를 백그라운드로 바꾸어주는 명령어이다.
- 전환을 위해 포어그라운드 프로세스는 정지를 시키고 백그라운드 프로세스로 전환해야 한다.
- 포어그라운드 프로세스 → 프로세스 정지 → 백그라운드 프로세스 전환

▼ 사용법

```
bg [job]
```

▼ 예제

1. 기본

```
Ctrl+Z를 이용해 중지 상태로 변경

(test9.txt와 test8.txt 파일 만들면서 일시정지)
[root@Linux-1 ~]# cat > test9.txt
1
2
^Z
[1]+  Stopped                  cat > test9.txt
[root@Linux-1 ~]# cat > test8.txt
3
4
^Z
```

2. bg

```
명령어를 이용해 백그라운드로 전환

[root@Linux-1 ~]# bg 1
[1]- cat > test9.txt &
[root@Linux-1 ~]# bg
[1]+ cat > test8.txt &
※ +기호가 있는 job 번호가 우선 실행하게 된다. (스택이 높기 때문에)
```

▼ 4. fg (background → foreground)

▼ 설명

- 백그라운드 프로세스를 포어그라운드 프로세스로 전환하는 명령어이다.

▼ 사용법

```
fg [jop]
```



백그라운드 프로세스 일시정지 시키는 방법

👉 백그라운드로 실행시킨 프로세스를 일시중지 시키는 과정이다.
위에서 배운 전면 처리 전환 기법과 함께 쓰면 된다.

- 1. &를 사용해 백그라운드 실행
- 2. jobs 명령을 사용해 프로세스 번호 알아내고, fg 명령어를 실행해 프로세스를 포어그라운드로 가져오기
- 3. Ctrl + z 명령어를 통해 포어그라운드에서 프로세스 일시중지

▼ 예제

1. 기본

```
1. test.txt와 test1.txt 파일 만들면서 일시정지인 상태에서
2. jobs 명령을 사용해 프로세스 번호 알아내기

[root@Linux-2 ~]# jobs
[1]-  Stopped                  cat > test.txt
[2]+  Stopped                  cat > test1.txt

3. fg 명령어를 실행해 프로세스를 포어그라운드로 가져오기

[root@Linux-2 ~]# fg 1
cat > test.txt

4. Ctrl + z 명령어를 통해 포어그라운드에서 프로세스 일시중지

[root@Linux-2 ~]# fg 1
cat > test.txt
^Z
[1]+  Stopped                  cat > test.txt
```

✓ 우선순위

▼ 1. nice

▼ 설명

- nice 명령어는 수정된 스케줄링 우선 순위로 프로그램/프로세스를 실행하는 데 도움이 됩니다.
- 사용자 정의 예약 우선 순위를 사용하여 프로세스를 시작합니다. 이 경우 프로세스에 더 높은 우선 순위를 부여하면 커널이 해당 프로세스에 더 많은 CPU 시간을 할당합니다.

▼ 사용법

```
nice [-n increment] command [arguments]
```

▼ 옵션

옵션	내용
----	----

옵션	내용
-n increment	새 프로세스에 대한 우선 순위를 지정할 수 있으며 증가 값은 -20과 19 사이의 정수 값이 될 수 있습니다.

▼ 예제

1. 예제1:

다음 명령어는 우선 순위 수준 10에서 **find** 명령어를 시작합니다:

```
nice -n 10 find / -name file.txt
```

기본값 **0** 보다 낮은 우선 순위인 **10** 의 우선 순위로 프로그램이 시작됩니다.

2. 예제2:

기본값으로 프로그램 실행합니다:

```
nice yum update
```

3. 예제3:

우선 순위 수준이 다른 여러 명령어들을 시작합니다:

```
nice -n -10 find / -name f1.txt & nice -n 10 yum update
```

▼ 2. renice

▼ 설명

- renice 명령어를 사용하면 이미 실행 중인 프로세스의 예약 우선 순위를 변경하고 수정할 수 있습니다. 리눅스 커널은 프로세스를 예약하고 그에 따라 CPU 시간을 할당합니다.

▼ 사용법

```
renice priority [-p PID | -g pgrp | -u user] [ -t ]
```

▼ 옵션

옵션	내용
-p	-p 옵션은 우선 순위를 변경할 프로세스의 프로세스 ID를 지정하는 데 사용됩니다
-g	-g 옵션은 프로세스 그룹 ID를 지정하는 데 사용됩니다
-u	-u 옵션은 프로세스의 사용자 이름을 지정하는 데 사용됩니다.
-t	-t 옵션은 지정된 프로세스의 모든 프로세스와 모든 하위 프로세스에 영향을 미치는 데 사용됩니다.

▼ 예제

1. 예제1:

PID 가 **1234** 인 프로세스의 우선 순위를 **-5** 로 변경합니다:

```
renice -n -5 -p 1288
1288 (process ID) old priority 0, new priority -5
```

2. 예제2:

사용자 `johndoe`에 속하는 모든 프로세스의 우선 순위를 `10`으로 변경합니다:

```
renice -n 10 -u s1
1001 (user ID) old priority 0, new priority 10
```

3. 예제3:

프로세스 그룹 `1234`의 모든 프로세스 우선 순위를 `5`로 변경합니다:

```
renice -n -5 -g 1288
1288 (process group ID) old priority -5, new priority -5
```

4. 예제4:

`PID`가 `1234`인 프로세스의 모든 하위 프로세스의 우선 순위를 `-10`으로 변경합니다:

```
renice -n -10 -p 1288 -t
1288 (process ID) old priority -10, new priority -10
```

5. 예제5:

시스템에서 실행 중인 모든 프로세스의 우선 순위를 `19`로 변경합니다:

```
renice -n 19 -p 0
0 (process ID) old priority -10, new priority 19
```

▼ 3. nohup

▼ 설명

- nohup(No Hang Up)는 셸/터미널에서 로그아웃한 후에도 프로세스를 실행하는 Linux 시스템의 명령입니다.
- 이 명령은 일반적으로 완료하는 데 오랜 시간이 걸리는 프로세스를 시작하거나 사용자의 SSH 연결이 끊어진 경우에도 계속 실행되는 원격 서버에서 명령을 실행하는 데 사용됩니다.

▼ 사용법

```
nohup command [command-argument ...]
OR
nohup options
```

▼ 옵션

옵션	내용
-f	이 옵션을 선택하면 output이 표준 출력과 표준 오류를 모두 nohup.out 파일로 출력 합니다.
-p	이 옵션을 선택하면 옵션 뒤에 지정된 파일로 출력합니다.
-d	이 옵션을 선택하면 작업 디렉터리가 옵션 뒤에 지정된 디렉터리로 변경니다.
-u	이 옵션을 선택하면 output 출력을 현재 작업 디렉토리 대신 사용자의 홈 디렉토리에 있는 nohup.out 파일로 출력 합니다.

▼ 예제

1. 예제1:

다음은 옵션이 없는 `nohup` 명령어의 예입니다:

```
nohup random_command
```

2. 예제2:

다음은 `-f` 옵션을 사용하는 `nohup` 명령어의 예입니다:

```
nohup -f long_running_command
```

3. 예제3:

다음은 `-p` 옵션을 사용하는 `nohup` 명령어의 예입니다:

```
nohup -p myoutput.log long_running_command
```

4. 예제4:

다음은 `-d` 옵션을 사용하는 `nohup` 명령어의 예입니다:

```
nohup -d /path/to/directory long_running_command
```

5. 예제5:

다음은 `-u` 옵션을 사용하는 `nohup` 명령어의 예입니다:

```
nohup -u long_running_command
```

▼ 4. pgrep

▼ 설명

- pgrep명령어는 지정된 패턴과 일치하는 실행 중인 프로세스의 PID(프로세스 ID)를 검색하고 표시할 수 있는 Linux의 명령줄 유틸리티입니다. grep은 procsps 패키지의 일부입니다.

▼ 사용법

```
pgrep [options] pattern
```

▼ 옵션

옵션	내용
-n	지정된 패턴과 일치하는 모든 프로세스의 최신 프로세스 ID만 표시합니다.
-o	지정된 패턴과 일치하는 모든 프로세스 ID만 표시합니다.
-l	지정된 패턴과 일치하는 프로세스 ID와 함께 프로세스 이름을 표시합니다.
-v	지정된 패턴과 일치하지 않는 모든 프로세스 ID를 표시합니다.
-u	특정 사용자 아래에서 실행 중인 모든 프로세스의 프로세스 ID를 표시합니다.
-g	특정 그룹에 실행 중인 모든 프로세스 ID를 표시합니다.
-f	명령줄 이름 또는 명령줄이 특정 단어로 시작하는 모든 프로세스의 모든 프로세스 ID를 표시합니다.
-d	출력에서 프로세스 ID 및 프로세스 이름을 구분합니다.

▼ 예제

1. 예제1:

`bash` 명령을 실행하는 프로세스의 `PID` 를 찾으려면 다음을 수행합니다:

```
pgrep bash
```

2. 예제2:

`HTTP` 를 실행하는 모든 프로세스를 찾기 위해 다음을 수행합니다:

```
pgrep httpd
```

3. 예제3:

다음은 `-n` 옵션을 사용하는 `pgrep` 명령어의 예입니다:

```
pgrep -n bash
```

4. 예제4:

다음은 `-o` 옵션을 사용하는 `pgrep` 명령어의 예입니다:

```
pgrep -o bash
```

5. 예제5:

다음은 `-l` 옵션을 사용하는 `pgrep` 명령어의 예입니다:

```
pgrep -l bash
```

6. 예제6:

다음은 `-v` 옵션을 사용하는 `pgrep` 명령어의 예입니다:

```
pgrep -v bash
```

7. 예제7:

다음은 `-u` 옵션을 사용하는 `pgrep` 명령어의 예입니다:

```
pgrep -u user1 bash
```

8. 예제8:

다음은 `-g` 옵션을 사용하는 `pgrep` 명령어의 예입니다:

```
pgrep -g group1 bash
```

9. 예제9:

다음은 `-f` 옵션을 사용하는 `pgrep` 명령어의 예입니다:


```
pgrep -f "my_command"
```

10. 예제10:

다음은 `-d` 옵션을 사용하는 `pgrep` 명령어의 예입니다:

```
pgrep -d ':' -l bash
```

3. 오픈소스 프로세스 관리 도구

▼ 1. Monit (unix 및 Linux용 무료 오픈소스 프로세스 감독 도구)

- 데몬 프로세스를 모니터링한다.
- 웹서버가 리소스를 많이 차지할 시 알림
- 메모리, cpu, 파일, 디렉토리, 파일 시스템까지 모니터링하여 타임스태프, 체크섬, 크기 변경하는 것까지 모니터링이 가능하다.
- 오류 상황에서 자동 유지 관리, 수리 및 의미 있는 원인 조치를 실행할 수 있다

▼ 작업순서

1. install

- monit 설치를 위해 yum의 확장된 최신 저장소 버전을 추가

```
yum -y install epel-release
```

- monit 설치

```
yum -y install monit
```

※ 모닛은 오픈소스 기반의 프로세스 도구이기 때문에 데몬으로 실행시켜줘야한다!

```
[root@linux ~]# monit status
New Monit id: a18d0963591101563ae090915dc8a050
Stored in "/root/.monit.id"
Monit: the monit daemon is not running
```

2. 설정 파일 세팅

※ monit의 **설정 파일**은 `/etc/monitrc` 이다.

※ **log 파일**의 위치는 `/var/log/monit.log` 이다.

```
[root@linux ~]# cat /etc/monit.d/logging
# log to monit.log
set logfile /var/log/monit.log
```

※ `monit -h` or `man monit` 명령어를 쓰면 사용법이 나온다.

- 설정 파일에서 건들여야 할 부분은 157번째 줄의 httpd이며 모닛 웹 접

속 시 2812 포트를 통해 들어가게 된다.

```
vi /etc/monitrc
```

```
157 set httpd port 2812 and
158     use address localhost # only accept connection from localhost (drop if you use M/Monit)
159     allow localhost       # allow localhost to connect to the server and
160     allow admin:monit     # require user 'admin' with password 'monit'
161     #with ssl {           # enable SSL/TLS and set path to server certificate
162     #    pemfile: /etc/ssl/certs/monit.pem
163     #}
```



간단하게 설정 파일 설명

설정명	내용
set httpd port	내가 사용할 포트
use address 서버 IP	미 입력시 서버 cmd에서 status 차단
allow 클라이언트 IP/24	원격으로 접속할 ip or 대역
allow admin:monit	전체 권한 관리자 (접속할 id / 패스워드)
allow user:pass read-only	리더 권한 사용자
allow user:pass 허용 ip	허용 ip로부터 접근 가능한 계정, 설정

+ 설정 변경

```
157 set httpd port 2812 and
158     use address 192.168.70.137 # only accept connection from localhost (drop if you use M/Monit)
159     allow 192.168.70.137/24    # allow localhost to connect to the server and
160     allow admin:monit         # require user 'admin' with password 'monit'
161     #with ssl {               # enable SSL/TLS and set path to server certificate
162     #    pemfile: /etc/ssl/certs/monit.pem
163     #}
```

+ 방화벽 해제 및 재가동

- 현재 방화벽을 확인 시 monit 접근 포트 등록 x

```
[root@linux ~]# firewall-cmd --list-all
public (active)
target: default
icmp-block-inversion: no
interfaces: ens32
sources:
services: dhcpv6-client ssh
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

→

```
[root@linux ~]# firewall-cmd --list-all
public (active)
target: default
icmp-block-inversion: no
interfaces: ens32
sources:
services: dhcpv6-client ssh
ports: 2812/tcp
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

```
firewall-cmd --permanent --add-port=2812/tcp
firewall-cmd --reload
```

+ 서비스 (데몬 실행)

```
systemctl start monit
systemctl enable monit
```

👉 monit 프로세서 툴을 사용하기 위해 monit 데몬을 시작하고 enable로 monit을 등록시켜주자

```
[root@linux ~]# systemctl start monit
[root@linux ~]# systemctl enable monit
Created symlink from /etc/systemd/system/multi-user.target.wants/monit.service to /usr/lib/systemd/system/monit.service.
```

3. 실행 및 결과

```
monit status # 모니터링 상황 확인
```

```
[root@linux ~]# monit status
Monit 5.30.0 uptime: 0m

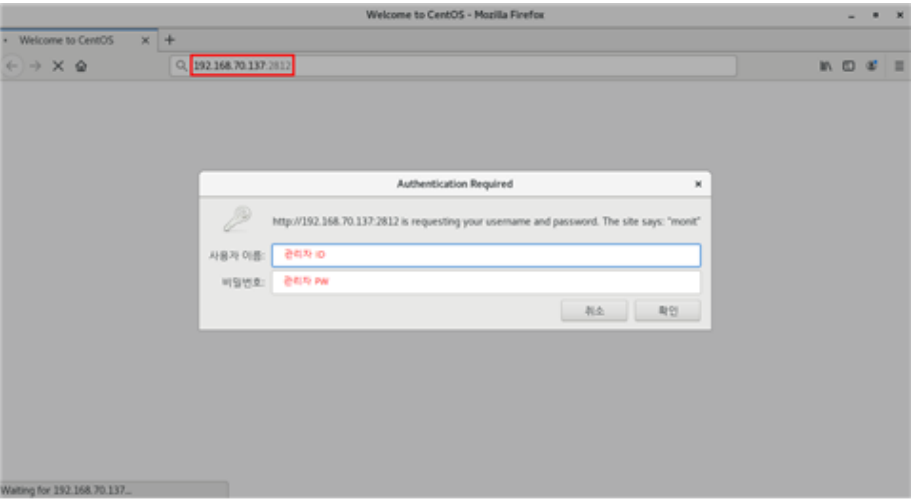
System 'Linux'
status OK
monitoring status Monitored
monitoring mode active
on reboot start
load average [0.08] [0.05] [0.05]
cpu 0.0%usr 0.0%sys 0.0%nice 0.0%lowast 0.0%hardirq 0.0%softirq 0.0%steal 0.0%guest 0.0%guestnice
memory usage 303.2 MB [31.2%]
swap usage 0 B [0.0%]
uptime 24m
boot time Wed, 18 Jan 2023 19:13:47
filedescriptors 1152 [1.3% of 91650 limit]
data collected Wed, 18 Jan 2023 19:38:18
```

monit summary# 모니터링 요약

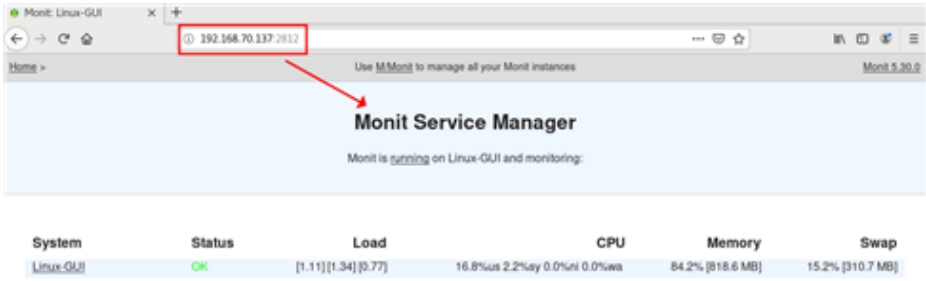
```
[root@Linux ~]# monit summary
Monit 5.30.0 uptime: 0m
```

Service Name	Status	Type
Linux	OK	System

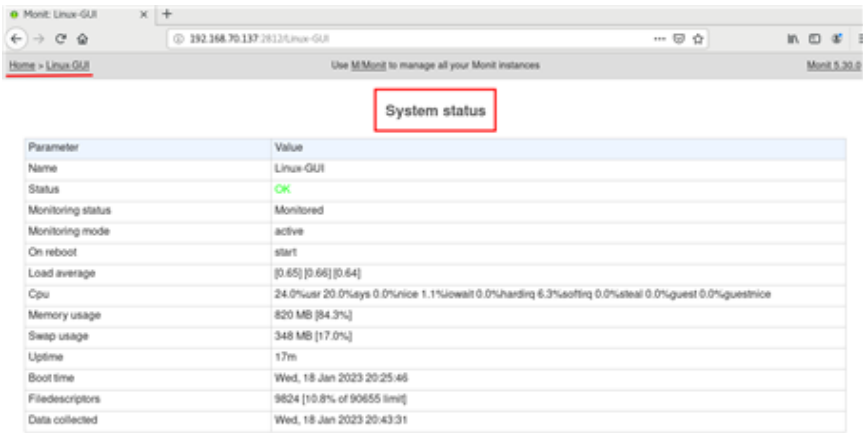
+ GUI 환경에서의 접속



1. 설정 파일에 등록된 IP와 포트번호로 접속 후 관리자 ID, 관리자 PW를 입력해준다.



2. 잘 접속 되었고 프로세스 관련 정보를 확인 할 수 있다. (아래는 status 정보)



▼ 사용법

- 모니터링할 서비스 등록
 - 서비스 등록은 `/etc/monit.d/` 밑에 파일을 생성하기만 하면 된다.

```
[root@Linux-GUI monit.d]# ll
합계 16
-rw-r--r-- 1 root root 264 1월 18 20:54 httpd
-rw-r--r-- 1 root root 51 1월 21 2022 logging
-rw-r--r-- 1 root root 266 1월 18 20:55 mariadb
-rw-r--r-- 1 root root 242 1월 18 20:51 sshd
```

모니터링할 프로세서들을 등록

vi /etc/monit.d/httpd# httpd 안의 내용

```
check process httpd with pidfile /var/run/httpd.pid
group apache
start program = "/usr/bin/systemctl httpd start"
stop program = "/usr/bin/systemctl httpd stop"
if failed host 127.0.0.1 port 80
protocol http then restart
if 5 restarts within 5 cycles then timeout
```

- 파일 등록 후 `monit -t` 하게 되면 오차가 있는지 체크해준다.

```
[root@Linux-GUI monit.d]# monit -t
Control file syntax OK
```

- monit 데몬 재시작

```
systemctl restart monit
```

- GUI 모드에서 프로세스 등록 확인

System	Status	Load	CPU	Memory	Swap
Linux-GUI	OK	[0.77] [0.34] [0.33]	0.0%us 0.0%sy 0.0%id 0.0%wa	87.6% [551.8 MB]	20.7% [424.2 MB]

Process	Status	Uptime	CPU Total	Memory Total	Read	Write
sshd	OK	42m	-	0.3% [3.1 MB]	0 B/s	0 B/s
mysqld	OK	0m	-	8.1% [79.2 MB]	0 B/s	0 B/s
httpd	OK	1m	-	2.1% [20.4 MB]	0 B/s	0 B/s

Filesystem	Status	Space usage	Inodes usage	Read	Write
myMainDisk	OK	16.9% [171.4 MB]	0.1% [340 objects]	0 B/s	0 B/s

👉 스토리지 모니터링은 후에 추가 했지만 이렇게 스토리지 모니터링 가능

👉 스토리지 뿐만 아니라 system Alert, Cpu, 메모 감시, 디스크 공간 사이즈도 감시 가능

- CLI 모드에서 모니터링 확인

```
[root@Linux-GUI monit.d]# monit status
Monit 5.30.0 uptime: 3m

Filesystem 'myMainDisk'
status OK
monitoring status Monitored
monitoring mode active
on reboot start
filesystem type xfs
filesystem flags rw,relatime,attr2,inode64,noquota
permissions 600
uid 0
gid 0
block size 4 kb
space total 1024 MB (of which 0.0% is reserved for root user)
space free for non superuser 862.6 MB [83.2%]
space free total 862.6 MB [83.2%]
inodes total 524288
inodes free 523988 [99.9%]
read bytes 0 B/s [6.4 MB total]
disk read operations 0.0 reads/s [163 reads total]
write bytes 0 B/s [2.1 MB total]
disk write operations 0.0 writes/s [13 writes total]
service time 0.000 ms/operation (of which read 0.000 ms, write 0.000 ms)
data collected Wed, 18 Jan 2023 21:12:21
```

Storage

```
Process 'sshd'
status OK
monitoring status Monitored
monitoring mode active
on reboot start
pid 1134
parent pid 1
uid 0
effective uid 0
gid 0
utime 4m
threads 1
children 0
cpu 0.0%
cpu total 0.0%
memory 0.3% [3.1 MB]
memory total 0.3% [3.1 MB]
security attribute
filedescriptors 22
total filedescriptors 5 [0.5% of 1024 limit]
read bytes 2.3 kB/s [2.3 MB total]
disk read bytes 0 B/s [32.0 MB total]
disk read operations 4.4 reads/s [4424 reads total]
write bytes 24.2 B/s [24.3 kB total]
disk write bytes 0 B/s [0 B total]
disk write operations 0.1 writes/s [184 writes total]
port response time 23.000 ms to 127.0.0.1:22 type TCP/IP protocol SSH
data collected Wed, 18 Jan 2023 21:12:21
```

sshd

```
Process 'mysqld'
status OK
monitoring status Monitored
monitoring mode active
on reboot start
pid 3863
parent pid 5718
uid 27
effective uid 27
gid 27
utime 4m
threads 19
children 0
cpu 0.0%
cpu total 0.0%
memory 8.2% [79.2 MB]
memory total 8.2% [79.2 MB]
security attribute
filedescriptors 34 [3.3% of 1024 limit]
total filedescriptors 34
read bytes 0 B/s [6.5 MB total]
disk read bytes 0 B/s [0 B total]
disk read operations 0.1 reads/s [505 reads total]
write bytes 2.0 B/s [2.2 kB total]
disk write bytes 0 B/s [32 kB total]
disk write operations 0.0 writes/s [16 writes total]
service time 0.152 ms to 127.0.0.1:3306 type TCP/IP protocol DEFAULT
port response time
data collected Wed, 18 Jan 2023 21:12:21
```

mariadb

```
Process 'httpd'
status OK
monitoring status Monitored
monitoring mode active
on reboot start
pid 5377
parent pid 1
uid 0
effective uid 0
gid 0
utime 4m
threads 1
children 0
cpu 0.0%
cpu total 0.0%
memory 2.1% [20.4 MB]
memory total 2.1% [20.4 MB]
security attribute
filedescriptors 8 [0.8% of 1024 limit]
total filedescriptors 8
read bytes 0 B/s [382.6 kB total]
disk read bytes 0 B/s [160 kB total]
disk read operations 0.0 reads/s [4543 reads total]
write bytes 0 B/s [0 B total]
disk write bytes 0 B/s [0 B total]
disk write operations 0.0 writes/s [18 writes total]
data collected Wed, 18 Jan 2023 21:12:21
```

httpd

```
System 'Linux-GUI'
status OK
monitoring status Monitored
monitoring mode active
on reboot start
load average [0.07] [0.18] [0.27]
cpu 2.9%usr 1.0%sys 0.0%nice 0.0%iowait 0.0%hardirq 0.2%softirq 0.0%steal 0.0%guest 0.0%guestnice
memory usage 853.7 MB [87.8%]
swap usage 423.8 MB [20.7%]
uptime 46m
boot time Wed, 18 Jan 2023 20:25:46
filedescriptors 10080 [11.1% of 90655 limit]
data collected Wed, 18 Jan 2023 21:12:21
```

my virtual machine

▼ 2. Cockpit

- GNU/Linux 서버를 위한 사용하기 쉽고 간단한 관리자
- 웹 브라우저를 통해 라이브 Linux 세션을 제공하는 대화형 서버 관리 사용자 인터페이스
- Debian, Ubuntu, Fedora, CentOS, RHEL, Arch Linux를 비롯한 여러 Linux에서 실행 가능
- 시스템 관리자가 컨테이너 시작, 스토리지 관리, 네트워크 구성, 로그 검사와 같은 작업을 쉽고 안정적으로 수행
- 그래프로 볼 수 있어 편리
- shell까지 관리가 가능
- 🌟GUI 환경이라면 간단하게 설치 및 실행하여 편리하게 관리 가능
- 🌟원격 접속 가능

▼ 작업순서

1. install

```
yum -y install cockpit
```

```
Installed:
cockpit.x86_64 0:195.12-1.el7.centos

Dependency Installed:
cockpit-bridge.x86_64 0:195.12-1.el7.centos cockpit-system.noarch 0:195.12-1.el7.centos cockpit-ws.x86_64 0:195.12-1.el7.centos

Complete!
```

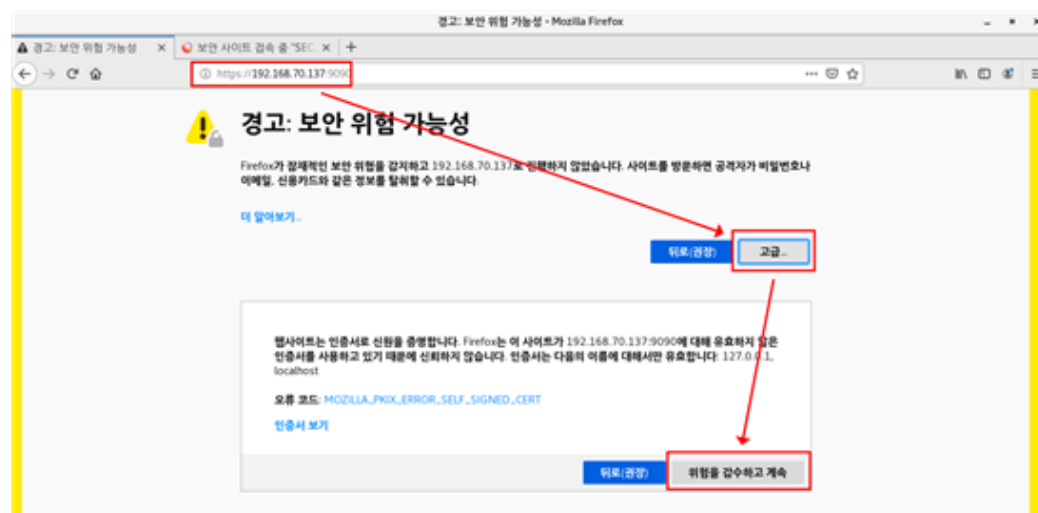
2. 실행

```
systemctl enable --now cockpit.socket
```

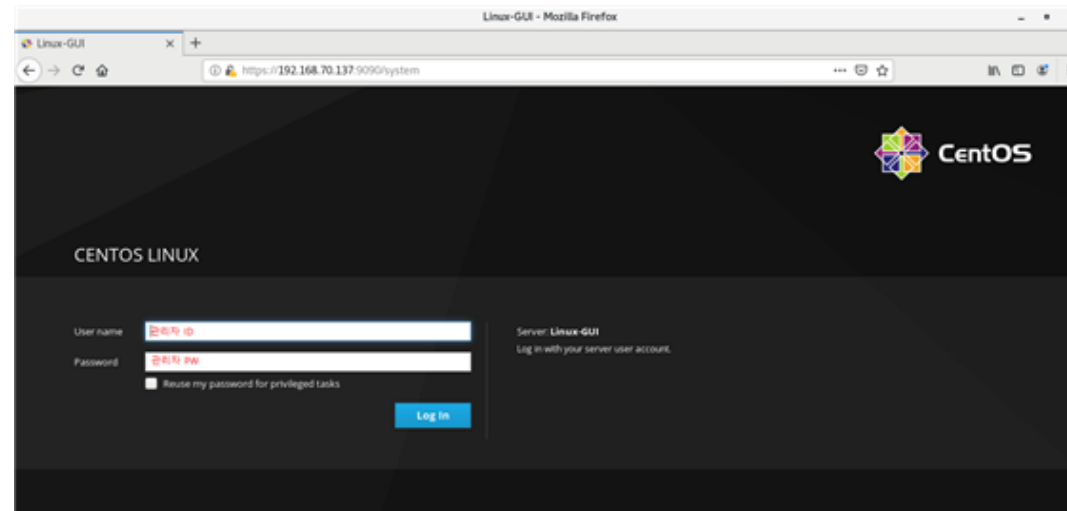
```
[root@Linux-GUI ~]# netstat -na | grep 9090
tcp6      0      0 :::9090          :::*              LISTEN
```

- 👉 cockpit은 기본적으로 **9090 포트**를 사용한다.
- 👉 netstat 명령어를 통해 잘 리스닝 되어있는지 확인하자!

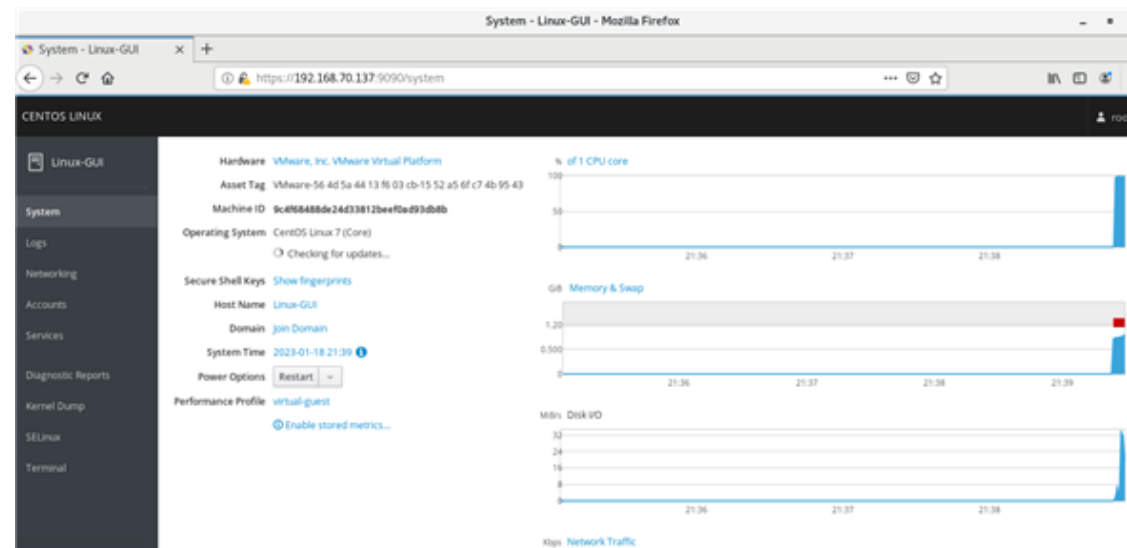
3. 접속



NAT 게이트웨이로 원격 접속 + HTTPS가 공인 인증서가 아니기 때문에 위와 같이 나온다 (무시)



접속



실시간으로 편리하게 CPU, Memory, Disk 사용량을 확인할 수 있다.

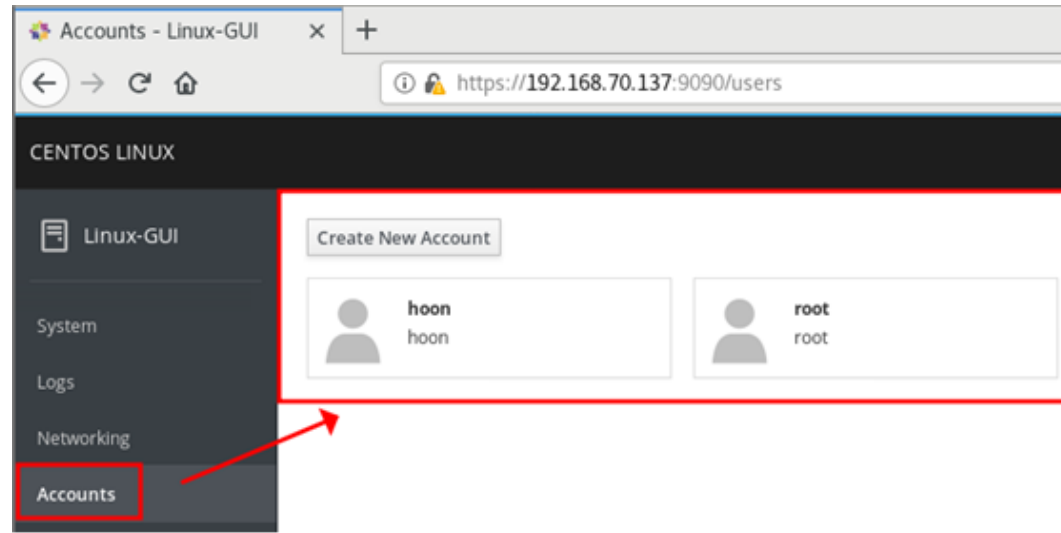
Timestamp	Message	Service
21:07	mdcheck_start timer lacks value setting. Refusing.	systemd
21:07	[fuserfsd/systemd/mdcheck_start timer 12] Failed to parse calendar specification, ignoring: Sun *--1..7 1:00:00	systemd
20:49	Failed to start The nginx HTTP and reverse proxy server.	systemd
20:32	GetManagedObjects() failed: org.freedesktop.DBus.Error.NoReply: Did not receive a reply. Possible causes include: the remote application did not send a reply, ...	pulseaudio
20:32	Failed to open the streaming device "/dev/virtio-ports/org.spice-space.stream.0": 2 - No such file or directory	spice-streaming-agent
20:31	Cannot access vdaagent virtio channel /dev/virtio-ports/com.redhat.spice.0	spice-vdagent
20:26	Failed to open the streaming device "/dev/virtio-ports/org.spice-space.stream.0": 2 - No such file or directory	spice-streaming-agent
20:26	Cannot access vdaagent virtio channel /dev/virtio-ports/com.redhat.spice.0	spice-vdagent
20:26	Failed to start Crash recovery kernel aining.	systemd
20:26	Family 6 Model 140 CPU: only decoding architectural errors	kernel
20:25	piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!	kernel
20:25	sd 0:0:0:0: [sd] Assuming drive cache: write through	kernel
20:25	Warning: Intel Processor - this hardware has not undergone upstream testing. Please consult http://wiki.centos.org/FAQ for more information	kernel
20:25	Detected CPU family 6 model 140 stepping 1	kernel

로그정보 확인

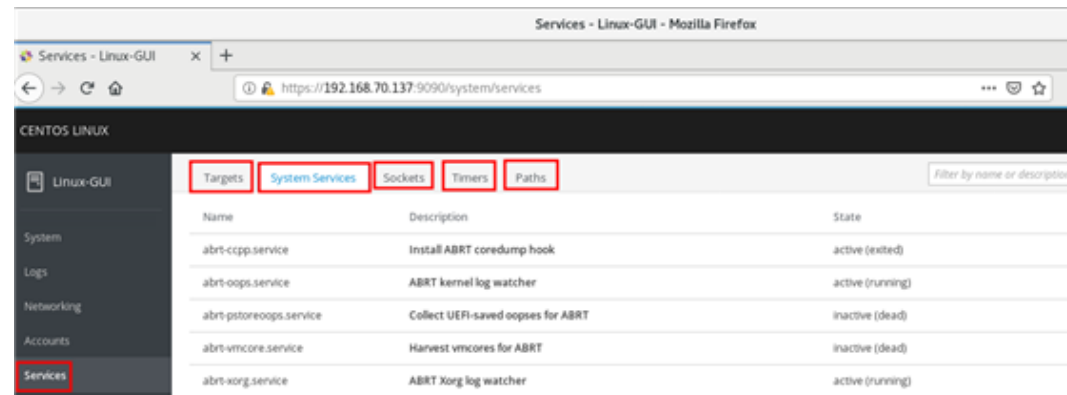
Name	IP Address	Sending	Receiving
ens32	192.168.70.137/24	0 bps	0 bps
virbr0	192.168.122.1/24	No carrier	

Name	IP Address	Sending	Receiving
virbr0-nic			

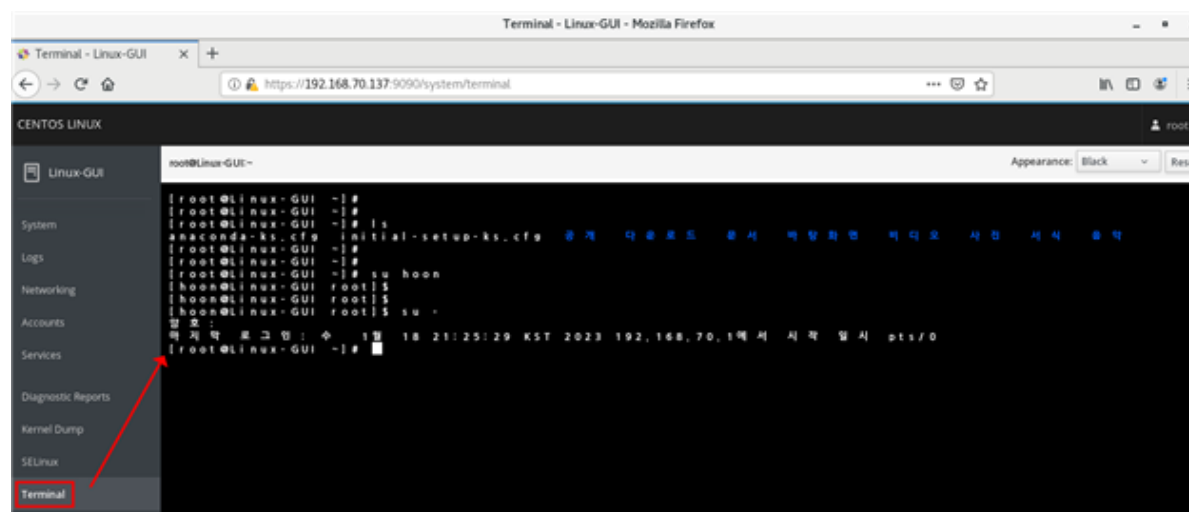
네트워크 정보 확인 (네트워크 환경 추가도 가능하다)



계정 정보도 확인이 가능하다 (추가도 가능)



어떤 종류의 서비스가 올라와 있는지도 확인이 가능



Terminal 접속도 가능하고 여기서 유저도 이동 가능