



우분투 리눅스

시스템 & 네트워크

Chapter 08. 리눅스의 부팅과 종료

목차

00. 개요

01. 리눅스 시스템의 부팅

02. init 프로세스와 런레벨

03. 리눅스 시스템의 종료

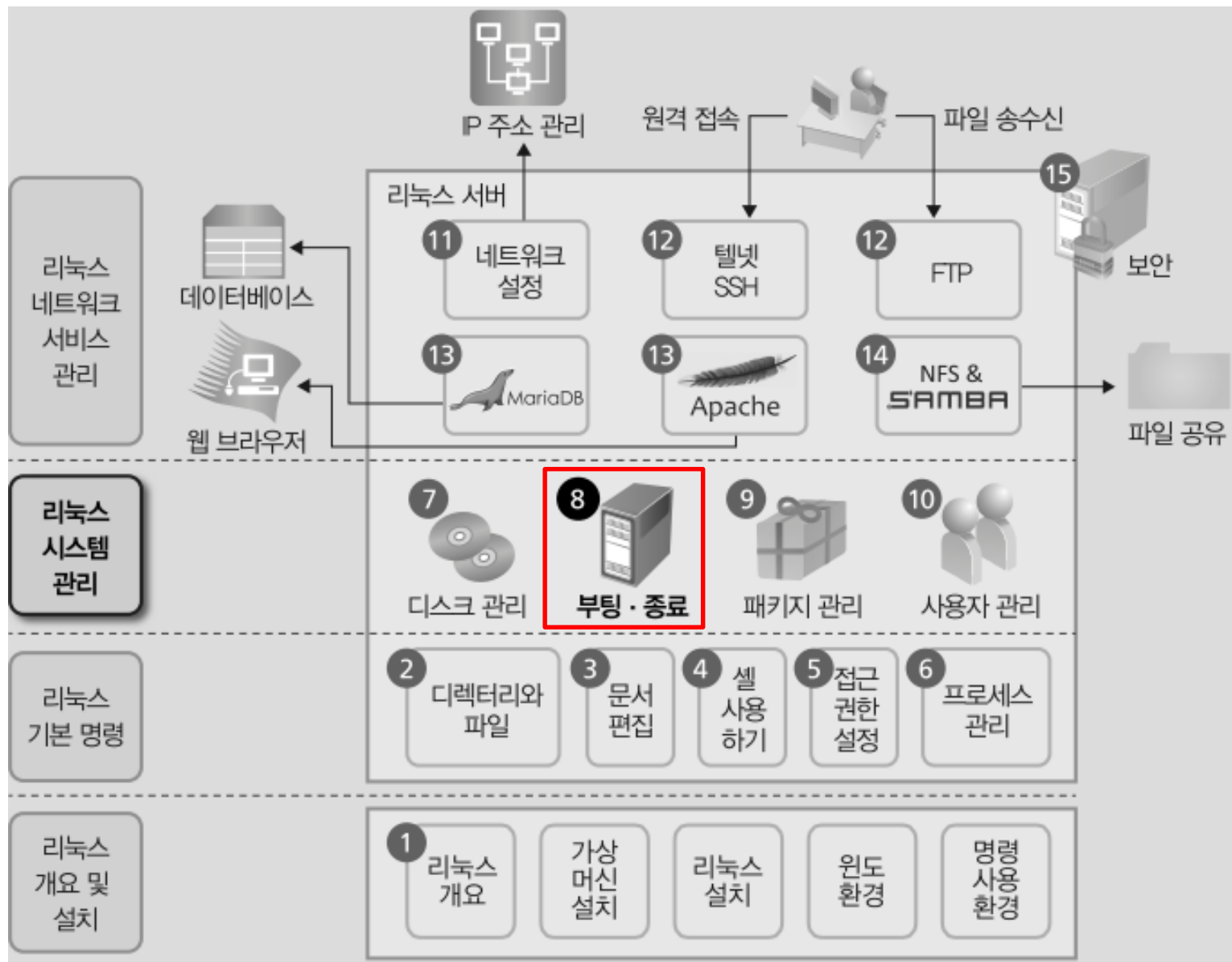
04. 데몬 프로세스

05. 부트 로더

학습목표

- 리눅스 시스템의 부팅 과정을 이해하고 부트 로더의 역할을 설명할 수 있다.
- init 프로세스의 역할을 설명할 수 있다.
- init 프로세스와 관련된 스크립트를 설명할 수 있다.
- 스크립트를 사용하여 서비스를 시작하고 종료할 수 있다.
- 런레벨이 무엇인지 설명하고 런레벨을 변경할 수 있다.
- 리눅스 시스템을 종료할 수 있다.
- 데몬을 이해하고 수퍼 데몬의 역할을 설명할 수 있다.
- 단일 사용자 모드로 부팅할 수 있다.
- 긴급 상황에서 계정의 암호를 복구할 수 있다.

리눅스 실습 스터디 맵



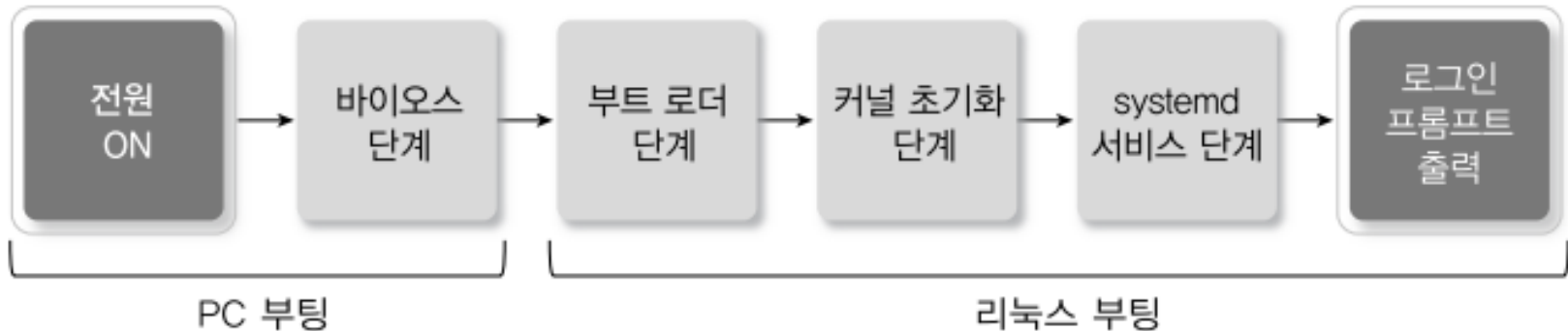
00 개요



[그림 8-1] 8장의 내용 구성

01 리눅스 시스템의 부팅

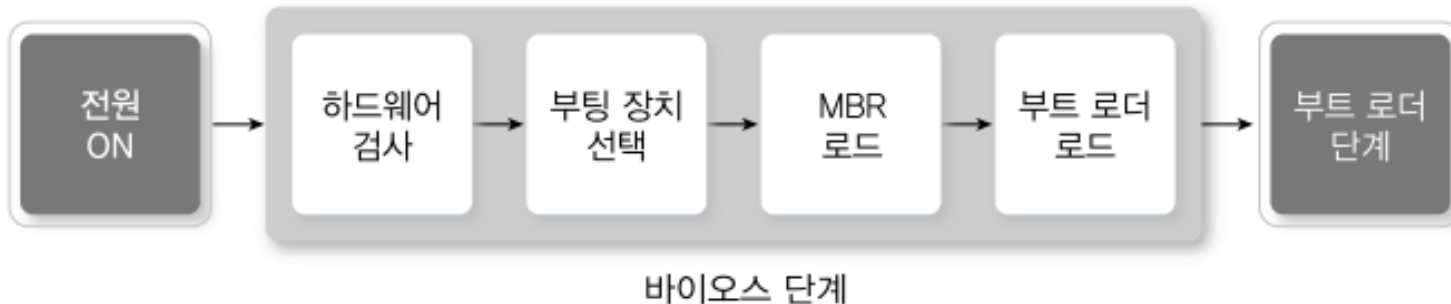
■ 리눅스 시스템의 부팅 과정



[그림 8-2] 리눅스의 부팅 과정

■ 바이오스 단계

- PC의 전원 스위치를 켜서 부팅하면 제일 먼저 바이오스(BIOS, basic input/output system)가 동작
- 바이오스는 PC에 장착된 기본적인 하드웨어(키보드, 디스크 등)의 상태를 확인한 후 부팅 장치를 선택하여 부팅 디스크의 첫 섹터에서 512바이트를 로딩
- 이 512바이트가 마스터 부트 레코드(master boot record, MBR): 2차 부팅 프로그램(부트 로더)의 위치 저장

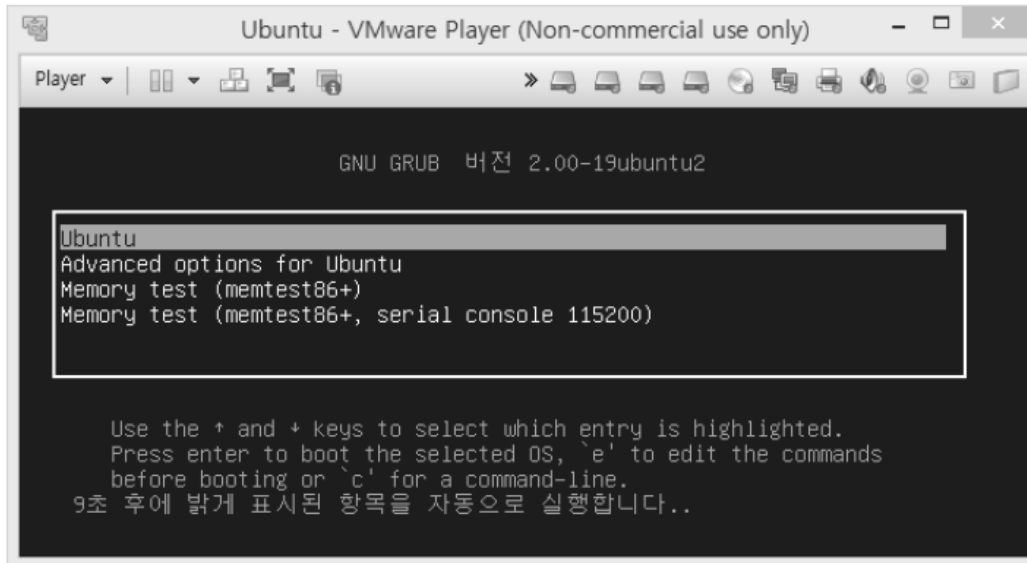


[그림 8-3] 바이오스 단계의 세부 동작

01 리눅스 시스템의 부팅

■ 부트 로더 단계

- 바이오스 단계에서 MBR는 부트 로더를 찾아 메모리에 로딩
- 부트 로더는 여러 운영체제 중에서 부팅할 운영체제를 선택할 수 있도록 메뉴를 제공



[그림 8-4] 부트 로더 시작 화면

- 부트 로더는 리눅스 커널을 메모리에 로딩
- 리눅스 커널은 /boot 디렉터리 아래에 'vmlinuz-버전명'의 형태로 제공
- 리눅스의 대표적인 부트 로더로는 GRUB와 LILO

```
user1@myubuntu:~$ ls /boot/vm*  
/boot/vmlinuz-3.11.0-12-generic  
user1@myubuntu:~$
```

01 리눅스 시스템의 부팅

■ 커널 초기화 단계

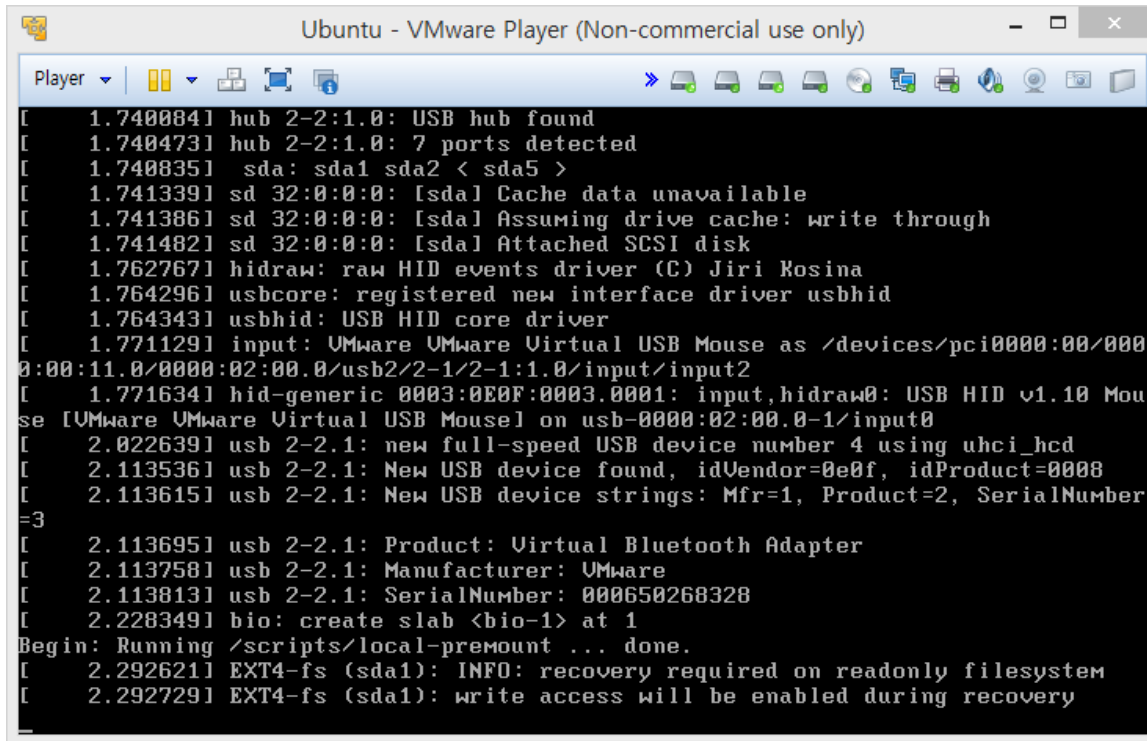
- 커널은 가장 먼저 시스템에 연결된 메모리, 디스크, 키보드, 마우스 등 장치들을 검사
- 장치 검사 등 기본적인 초기화 과정이 끝나면 커널은 fork를 사용하지 않고 생성되는 프로세스와 스레드 생성
 - 이 프로세스들은 메모리 관리 같은 커널의 여러 가지 동작을 수행
 - 이들 프로세스는 일반적인 프로세스와 구분되도록 대괄호([])로 표시하며, 주로 PID 번호가 낮게 배정

```
user1@myubuntu:~$ ps -ef | more
UID          PID    PPID    C   STIME   TTY          TIME      CMD
root          1        0      0  12:46   ?           00:00:02  /sbin/init
root          2        0      0  12:46   ?           00:00:00  [kthreadd]
root          3        2      0  12:46   ?           00:00:00  [ksoftirqd/0]
root          5        2      0  12:46   ?           00:00:00  [kworker/0:0H]
root          7        2      0  12:46   ?           00:00:00  [migration/0]
root          8        2      0  12:46   ?           00:00:00  [rcu_bh]
root          9        2      0  12:46   ?           00:00:00  [rcu_sched]
root         10        2      0  12:46   ?           00:00:00  [watchdog/0]
root         11        2      0  12:46   ?           00:00:00  [khelper]
root         12        2      0  12:46   ?           00:00:00  [kdevtmpfs]
root         13        2      0  12:46   ?           00:00:00  [netns]
(생략)
```

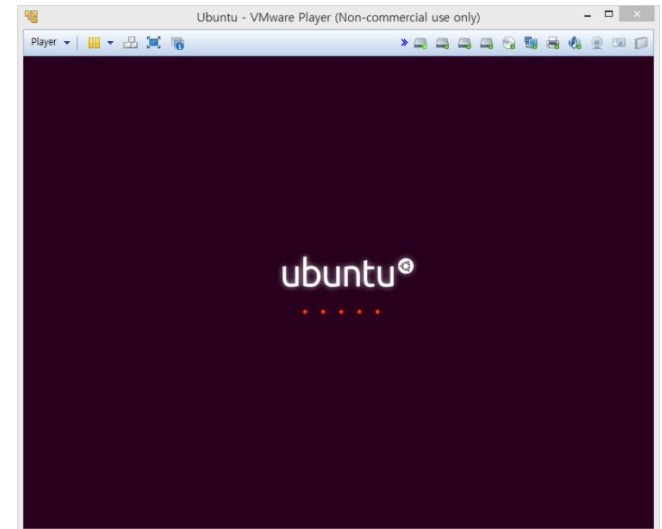

01 리눅스 시스템의 부팅

■ init 실행 단계

- init 실행단계에 이르면 리눅스가 본격적으로 동작하기 시작
- 기본적으로 메시지가 보이지 않도록 하고 대신에 부트 스플래시(boot splash)라고 하는 이미지를 출력
- 부트 스플래시 화면이 진행 중일 때 ctrl+D 키를 누르면 메시지가 출력되는 화면으로 전환



```
[ 1.740084] hub 2-2:1.0: USB hub found
[ 1.740473] hub 2-2:1.0: 7 ports detected
[ 1.740835] sda: sda1 sda2 < sda5 >
[ 1.741339] sd 32:0:0:0: [sda] Cache data unavailable
[ 1.741386] sd 32:0:0:0: [sda] Assuming drive cache: write through
[ 1.741482] sd 32:0:0:0: [sda] Attached SCSI disk
[ 1.762767] hidraw: raw HID events driver (C) Jiri Kosina
[ 1.764296] usbcore: registered new interface driver usbhid
[ 1.764343] usbhid: USB HID core driver
[ 1.771129] input: VMware VMware Virtual USB Mouse as /devices/pci0000:00/0000:00:00:00/0000:02:00:00/usb2/2-1/2-1.0/input/input2
[ 1.771634] hid-generic 0003:0E0F:0003.0001: input,hidraw0: USB HID v1.10 Mouse [VMware VMware Virtual USB Mouse] on usb-0000:02:00:00-1/input0
[ 2.022639] usb 2-2.1: new full-speed USB device number 4 using uhci_hcd
[ 2.113536] usb 2-2.1: New USB device found, idVendor=0e0f, idProduct=0008
[ 2.113615] usb 2-2.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 2.113695] usb 2-2.1: Product: Virtual Bluetooth Adapter
[ 2.113758] usb 2-2.1: Manufacturer: VMware
[ 2.113813] usb 2-2.1: SerialNumber: 000650268328
[ 2.228349] bio: create slab <bio-1> at 1
Begin: Running /scripts/local-premount ... done.
[ 2.292621] EXT4-fs (sda1): INFO: recovery required on readonly filesystem
[ 2.292729] EXT4-fs (sda1): write access will be enabled during recovery
```



[그림 8-5] 우분투 스플래시 화면

[그림 8-6] 부팅 메시지 출력 화면

01 리눅스 시스템의 부팅

■ 부팅 후 메시지 확인

- 부팅시 출력된 메시지는 dmesg 명령이나 more /var/log/boot.log 명령으로 확인 가능

```
user1@myubuntu:~$ dmesg | more
[    0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-3.11.0-12-generic root
=UUID=4c270aab-6780-4f53-87a9-b8824b6d8b50 ro quiet splash
[    0.000000] PID hash table entries: 4096 (order: 2, 16384 bytes)
[    0.000000] Dentry cache hash table entries: 131072 (order: 7, 524288 bytes)
[    0.000000] Inode-cache hash table entries: 65536 (order: 6, 262144 bytes)
[    0.000000] Initializing CPU#0
[    0.000000] xsave: enabled xstate_bv 0x7, cntxt size 0x340
[    0.000000] allocated 2097144 bytes of page_cgroup
(생략)
```

01 리눅스 시스템의 부팅

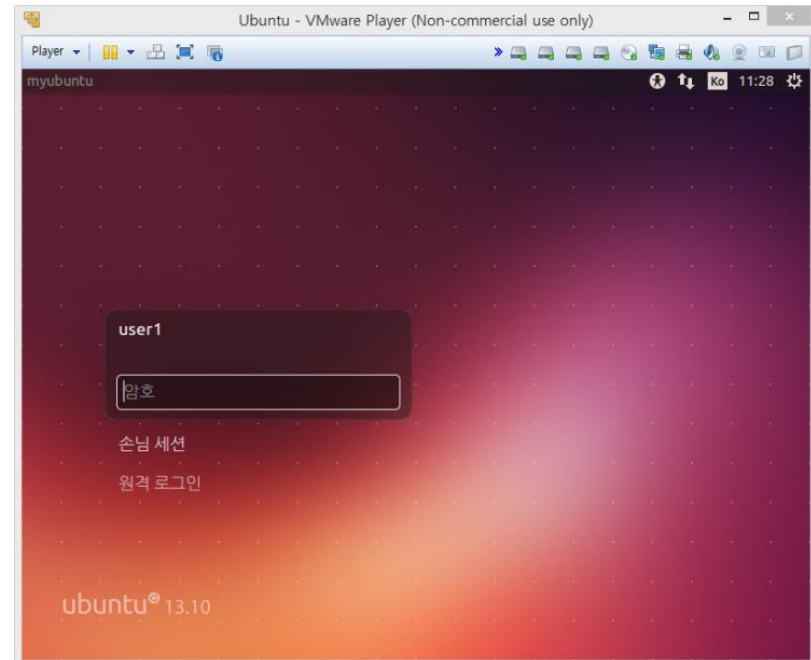
■ 1번 프로세스

- 전통적으로 유닉스에서는 init 프로세스가 처음 생성된 프로세스로서 PID가 1번

```
user1@myubuntu:~$ ps -ef | more
UID          PID  PPID  C  STIME TTY          TIME CMD
root          1    0    0  11:15 ?        00:00:00 /sbin/init
root          2    0    0  11:15 ?        00:00:00 [kthreadd]
root          3    2    0  11:15 ?        00:00:00 [ksoftirqd/0]
(생략)
```

■ 로그인 프롬프트 출력

- 마지막으로 그래픽 로그인 시스템인 GDM(GNOME display manager)을 동작시키고, 로그인 프롬프트 출력



[그림 8-7] 로그인 프롬프트 화면

02 init 프로세스와 런레벨

■ init 프로세스

- init 프로세스는 PID가 1번인 프로세스로 모든 프로세스의 조상 역할
- 우분투는 자체적으로 개발한 upstart를 init 대신 사용하는데, 다만 기존의 유닉스 및 리눅스 환경과의 호환을 위해 실행 파일 이름은 init를 유지
- init의 동작 방식이 바뀜에 따라 전통적으로 init 프로세스와 관련된 설정 파일이었던 /etc/inittab 파일은 이제 우분투에서 사라짐

```
user1@myubuntu:~$ ls /etc/inittab
ls: /etc/inittab에 접근할 수 없습니다: 그런 파일이나 디렉터리가 없습니다
user1@myubuntu:~$
```

02 init 프로세스와 런레벨

■ init 프로세스와 스크립트 파일

- 기존의 init와 새로운 upstart는 모두 프로세스를 실행하거나 종료하기 위해 스크립트 파일을 사용
- /etc/init 디렉터리와 /etc/init.d 디렉터리에 같은 서비스에 대한 파일이 있다면 /etc/init 디렉터리의 스크립트 파일이 우선적으로 적용

■ upstart가 사용하는 작업 파일

- upstart와 관련된 스크립트 파일은 /etc/init 디렉터리에 '작업명.conf' 파일로 구성

```
user1@myubuntu:~$ ls /etc/init
acpid.conf               mountkernfs.sh.conf
alsa-restore.conf        mountnfs-bootclean.sh.conf
alsa-state.conf          mountnfs.sh.conf
alsa-store.conf          mtab.sh.conf
anacron.conf             network-interface-container.conf
appport.conf            network-interface-security.conf
atd.conf                 network-interface.conf
avahi-cups-reload.conf   network-manager.conf
avahi-daemon.conf        networking.conf
(생략)
```

02 init 프로세스와 런레벨

■ 기존 init 프로세스

- init 프로세스가 실행하는 스크립트 파일은 /etc/init.d 디렉터리에 위치

```
user1@myubuntu:~$ ls /etc/init.d
README          dbus            procs           single
acpid           dns-clean      pulseaudio     skeleton
anacron         friendly-recovery rc              speech-dispatcher
apparmor        grub-common   rc.local       ssh
appport         halt          rcS            sudo
atd             irqbalance    reboot         udev
avahi-daemon    kerneloops    resolvconf     umountfs
bluetooth       killprocs     rfkill-restore umountnfs.sh
brltty          kmod          rfkill-store   umountroot
console-font    lightdm       rsync          unattended-upgrades
console-setup  networking    rsyslog        urandom
cron            ondemand      saned          x11-common
cups            postfix       sendsigs
cups-browsed   pppd-dns      setvtrgb
user1@myubuntu:~$
```

02 init 프로세스와 런레벨

■ 기존 init 스크립트와 upstart 스크립트의 관계

- /etc/init.d 디렉터리에 있는 스크립트 중 일부는 /lib/init/upstart-job에 대한 심벌릭 링크
- upstart-job은 해당 서비스의 upstart 스크립트를 찾아서 실행

```
user1@myubuntu:~$ ls -l /etc/init.d | grep upstart
lrwxrwxrwx 1 root root 21 2월 20 21:12 anacron -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 4월 9 2013 atd -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 bluetooth -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 console-font -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 console-setup -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 cron -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 friendly-recovery -> /lib/init/upstar-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 kmod -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 resolvconf -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 rfkill-restore -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 rfkill-store -> /lib/init/upstart-job
lrwxrwxrwx 1 root root 21 2월 20 21:12 setvtrgb -> /lib/init/upstart-job
user1@myubuntu:~$
```

02 init 프로세스와 런레벨

■ upstart 스크립트 시작하고 정지하기

- `initctl [start|stop|restart|reload|status]` 작업명
- `[start|stop|restart|reload|status]` 작업명

initctl

기능 upstart init 데몬을 제어한다.

형식 `initctl [서브 명령] 작업명`

서브 명령

- `start` : 작업을 시작한다.
- `stop` : 작업을 정지한다.
- `reload` : 작업에 SIGHUP 시그널을 보낸다.
- `restart` : 작업을 다시 시작한다.
- `status` : 작업 상태를 출력한다.
- `list` : 알려진 작업의 목록을 현재 상태와 함께 출력한다.

사용 예 `initctl start cups` `initctl stop cups` `initctl list`

- `start, stop, restart, reload, status` 명령은 모두 `initctl` 명령에 대한 심벌릭 링크

```
user1@myubuntu:~$ ls -l /sbin | grep initctl
-rwxr-xr-x 1 root root 179012 10월 10 00:05 initctl
lrwxrwxrwx 1 root root 7 2월 20 21:12 reload -> initctl
lrwxrwxrwx 1 root root 7 2월 20 21:12 restart -> initctl
lrwxrwxrwx 1 root root 7 2월 20 21:12 start -> initctl
lrwxrwxrwx 1 root root 7 2월 20 21:12 status -> initctl
lrwxrwxrwx 1 root root 7 2월 20 21:12 stop -> initctl
user1@myubuntu:~$
```


02 init 프로세스와 런레벨

■ upstart 스크립트 시작하고 정지하기

- initctl list : 전체 작업 목록 출력하기

```
user1@myubuntu:~$ initctl list
avahi-cups-reload stop/waiting
avahi-daemon stop/waiting
mountall-net stop/waiting
mountnfs-bootclean.sh start/running
passwd stop/waiting
rc stop/waiting
rsyslog start/running, process 464
startpar-bridge stop/waiting
(생략)
```

- initctl status : 작업 상태 보기

```
user1@myubuntu:~$ initctl status rsyslog
rsyslog start/running, process 464
user1@myubuntu:~$ status rsyslog
rsyslog start/running, process 464
user1@myubuntu:~$
```

02 init 프로세스와 런레벨

■ upstart 스크립트 시작하고 정지하기

- initctl stop : 작업 종료하기

```
user1@myubuntu:~$ sudo initctl stop rsyslog
rsyslog stop/waiting
user1@myubuntu:~$
```

- initctl start : 작업 시작하기

```
user1@myubuntu:~$ sudo start rsyslog
rsyslog start/running, process 2718
user1@myubuntu:~$
```

- initctl restart : 작업 다시 시작하기

```
user1@myubuntu:~$ sudo restart rsyslog
rsyslog start/running, process 2732
user1@myubuntu:~$
```

02 init 프로세스와 런레벨

■ init 스크립트 시작하고 정지하기

- service 스크립트명 [start|stop|restart|status]
- /etc/init.d/스크립트명 [start|stop|restart|status]

service

기능 시스템 V init 스크립트를 실행한다.

형식 service 스크립트 [서브 명령]

서브 명령 start : 스크립트에 지정한 start 부분을 실행한다.
stop : 스크립트에 지정한 stop 부분을 실행한다.
reload : 스크립트에 지정한 reload 부분을 실행한다.
restart : 스크립트에 지정한 restart 부분을 실행한다.
status : 스크립트에 지정한 status 부분을 실행한다.

사용 예 service cups start service cups stop

02 init 프로세스와 런레벨

■ init 스크립트 시작하고 정지하기

- 서비스 상태 보기 : service 스크립트 status

```
user1@myubuntu:~$ sudo service cups status
cups start/running, process 2757
user1@myubuntu:~$
```

- 서비스 종료하기 : service 스크립트 stop

```
user1@myubuntu:~$ sudo service cups stop
cups stop/waiting
user1@myubuntu:~$
```

- 서비스 시작하기 : service 스크립트 start

```
user1@myubuntu:~$ sudo service cups start
cups start/running, process 2927
user1@myubuntu:~$
```

- 서비스 다시 시작하기 : service 스크립트 restart

```
user1@myubuntu:~$ sudo service cups restart
cups stop/waiting
cups start/running, process 2950
user1@myubuntu:~$
```

02 init 프로세스와 런레벨

■ init 프로세스와 런레벨

- init 프로세스에서 사용하던 런레벨(Run Level)의 개념에 대한 이해 필요
- init는 시스템의 단계를 일곱 개로 정의하여 구분하고 각 단계에 따라 셸 스크립트를 실행하는데, 이 단계들을 런레벨이라고 함
- 기존 유닉스나 페도라와 달리 우분투의 런레벨은 기본 런레벨이 2번임

[표 8-1] 유닉스의 런레벨

런레벨	의미	관련 스크립트의 위치
0	시스템 종료	/etc/rc0.d
1, S, s	단일 사용자 모드	/etc/rc1.d
2	다중 사용자 모드(NFS를 실행하지 않음)	/etc/rc2.d
3	다중 사용자 모드(NFS 포함)	/etc/rc3.d
4	사용하지 않음(예비 번호)	/etc/rc4.d
5	시스템 종료(페도라는 GUI 모드로 부팅)	/etc/rc5.d
6	시스템 재시작	/etc/rc6.d

[표 8-2] 우분투의 런레벨

런레벨	의미	관련 스크립트의 위치
0	시스템 종료	/etc/rc0.d
1, S, s	단일 사용자 모드	/etc/rc1.d, /etc/rcS.d
2	그래피컬 다중 사용자 모드+네트워킹(기본 값)	/etc/rc2.d
3	런레벨 2와 동일	/etc/rc3.d
4		/etc/rc4.d
5		/etc/rc5.d
6	시스템 재시작	/etc/rc6.d

02 init 프로세스와 런레벨

■ init 프로세스와 런레벨

- 런레벨 3, 4, 5번이 2번과 같으므로, rc2.d 디렉터리와 rc3.d, rc4.d, rc5.d 디렉터리의 내용이 모두 같음

```
user1@myubuntu:/etc$ ls rc*.d
```

(생략)

rc2.d:

README	S20speech-dispatcher	S70dns-clean	S99grub-common
S20kerneloops	S50rsync	S70pppd-dns	S99ondemand
S20postfix	S50saned	S75sudo	S99rc.local

rc3.d:

README	S20speech-dispatcher	S70dns-clean	S99grub-common
S20kerneloops	S50rsync	S70pppd-dns	S99ondemand
S20postfix	S50saned	S75sudo	S99rc.local

rc4.d:

README	S20speech-dispatcher	S70dns-clean	S99grub-common
S20kerneloops	S50rsync	S70pppd-dns	S99ondemand
S20postfix	S50saned	S75sudo	S99rc.local

rc5.d:

README	S20speech-dispatcher	S70dns-clean	S99grub-common
S20kerneloops	S50rsync	S70pppd-dns	S99ondemand
S20postfix	S50saned	S75sudo	S99rc.local

(생략)

```
user1@myubuntu:/etc$
```

02 init 프로세스와 런레벨

■ init 프로세스와 런레벨

- 런레벨별로 실행하는 스크립트 파일은 /etc/init.d 디렉터리에 있는 파일에 대한 심벌릭 링크

```
user1@myubuntu:~$ ls -l /etc/rc2.d
```

```
합계 4
```

```
-rw-r--r--  1 root root 677  6월  5  2013 README
lrwxrwxrwx  1 root root  20  2월  20  21:12 S20kerneloops -> ../init.d/kerneloops
lrwxrwxrwx  1 root root  17  2월  24  14:19 S20postfix -> ../init.d/postfix
lrwxrwxrwx  1 root root  27  2월  20  21:12 S20speech-dispatcher -> ../init.d/speech-
dispatcher
lrwxrwxrwx  1 root root  15  2월  20  21:12 S50rsync -> ../init.d/rsync
lrwxrwxrwx  1 root root  15  2월  20  21:12 S50saned -> ../init.d/saned
lrwxrwxrwx  1 root root  19  2월  20  21:12 S70dns-clean -> ../init.d/dns-clean
lrwxrwxrwx  1 root root  18  2월  20  21:12 S70pppd-dns -> ../init.d/pppd-dns
lrwxrwxrwx  1 root root  14  2월  20  21:12 S75sudo -> ../init.d/sudo
lrwxrwxrwx  1 root root  21  2월  20  21:12 S99grub-common -> ../init.d/grub-common
lrwxrwxrwx  1 root root  18  2월  20  21:12 S99ondemand -> ../init.d/ondemand
lrwxrwxrwx  1 root root  18  2월  20  21:12 S99rc.local -> ../init.d/rc.local
user1@myubuntu:~$
```

02 init 프로세스와 런레벨

■ init 프로세스와 런레벨

- 런레벨 변경하기
 - init는 1번 프로세스의 이름이기도 하지만 init 프로세스의 런레벨을 바꾸는 명령으로도 사용
 - init를 명령으로 실행하면 실제로는 /sbin/telinit가 실행
 - init로 런레벨을 변경하기 위해서는 바꾸려는 런레벨을 숫자로 지정

```
user1@myubuntu:~$ sudo init 1
```

- 기본 런레벨 지정하기
 - 우분투에서 기본 런레벨은 /etc/init/rc-sysinit.conf 파일의 DEFAULT_RUNLEVEL 변수에 지정

```
user1@myubuntu:~$ cat /etc/init/rc-sysinit.conf
(생략)
# Default runlevel, this may be overridden on the kernel command-line
# or by faking an old /etc/inittab entry
env DEFAULT_RUNLEVEL=2
(생략)
user1@myubuntu:~$
```

- 현재 런레벨 확인하기 : runlevel 명령

```
user1@myubuntu:~$ runlevel
N 2
user1@myubuntu:~$
```


03 리눅스 시스템의 종료

■ 리눅스를 종료하는 방법

- shutdown 명령을 사용한다.
- halt 명령을 사용한다.
- poweroff 명령을 사용한다.
- 런레벨을 0이나 6으로 전환한다.
- reboot 명령을 사용한다.
- 전원을 끈다 -> 최후의 수단

03 리눅스 시스템의 종료

■ shutdown 명령 사용하기

- 리눅스 시스템을 가장 정상적으로 종료하는 방법

shutdown

기능 리눅스를 종료한다.

형식 shutdown [옵션] [시간] [메시지]

옵션 -k : 실제로 시스템을 종료하는 것이 아니라 사용자들에게 메시지만 전달한다.

-r : 종료 후 재시작한다.

-h : 종료하며 halt 상태로 이동한다.

-f : 빠른 재시작으로 이 과정에서 fsck를 생략할 수도 있다.

-c : 이전에 내렸던 shutdown 명령을 취소한다.

시간 : 종료할 시간(hh:mm, +m, now)

메시지 : 모든 사용자에게 보낼 메시지

사용 예 shutdown -h now shutdown -r +3 "System is going down" shutdown -c

■ shutdown 명령으로 시스템 즉시 종료하기

- h 옵션과 함께 현재 시간으로 지정

```
user1@myubuntu:~$ sudo shutdown -h now
```

03 리눅스 시스템의 종료

■ shutdown 한다는 메시지 보내고 종료하기

- 시스템을 종료할 때 shutdown 명령으로 메시지를 보낼 수 있음
- 사용자들이 메시지를 받고 정리할 시간이 필요하므로 시간을 now로 지정하면 안 되고 특정 시간을 지정
- 예: 2분 후에 종료한다는 메시지 발송

```
user1@myubuntu:~$ sudo shutdown -h +2 "System is going down in 2 min"
```

- 사용자 터미널 출력

```
user1@myubuntu:~$  
Broadcast message from user1@myubuntu  
      (/dev/pts/1) at 23:31 ...  
The system is going down for halt in 2 minutes!  
System is going down in 2 min
```

■ shutdown 명령으로 시스템 재시작하기: -r 옵션 사용

```
user1@myubuntu:~$ sudo shutdown -r +3
```

```
Broadcast message from user1@myubuntu  
      (/dev/pts/1) at 23:32 ...  
The system is going down for reboot in 3 minutes!
```

03 리눅스 시스템의 종료

■ shutdown 명령 취소하기: -c 옵션

```
user1@myubuntu:~$ sudo shutdown -c
```

- 앞의 3분 후 재시작 명령 취소할 경우 메시지 출력

```
shutdown: Shutdown cancelled
```

■ shutdown 메시지만 보내기: -k 옵션

```
user1@myubuntu:~$ sudo shutdown -k 2
```

```
Broadcast message from user1@myubuntu  
(/dev/pts/1) at 23:35 ...  
The system is going down for maintenance in 2 minutes!
```

03 리눅스 시스템의 종료

■ 런레벨 변경하기

- 런레벨을 0으로 바꾸면 시스템이 종료

```
user1@myubuntu:~$ sudo init 0
```

- 재시작하려면 런레벨을 6으로 변경

```
user1@myubuntu:~$ sudo init 6
```

03 리눅스 시스템의 종료

■ 기타 시스템 종료 명령

- 시스템을 종료하거나 재시작하기 위해 사용할 수 있는 명령: halt, poweroff, reboot
- halt와 poweroff는 reboot 명령의 심벌릭 링크

```
user1@myubuntu:~$ ls -l /sbin/reboot
-rwxr-xr-x 1 root root 13896 10월 10 00:05 /sbin/reboot
user1@myubuntu:~$ ls -l /sbin/halt
lrwxrwxrwx 1 root root 6 2월 20 21:12 /sbin/halt -> reboot
user1@myubuntu:~$ ls -l /sbin/poweroff
lrwxrwxrwx 1 root root 6 2월 20 21:12 /sbin/poweroff -> reboot
user1@myubuntu:~$
```

- halt, reboot, poweroff 명령은 /var/log/wtmp 파일에 시스템 종료 기록을 남기고 시스템을 종료하거나 재시작
- 사용할 수 있는 옵션
 - -w : 실질적으로 재시작하거나 종료하지는 않지만 wtmp 파일에 기록을 남긴다.
 - -f : 강제로 명령을 실행하며 shutdown을 호출하지 않는다.
 - -p : 시스템의 전원을 끈다.

04 데몬 프로세스

■ 데몬(daemon)

- 리눅스의 백그라운드에서 동작하면서 특정한 서비스를 제공하는 프로세스
- 리눅스 시스템에서 동작하는 각종 서비스를 제공하는 프로세스들이 바로 데몬

■ 데몬의 동작 방식

- 독자형(standalone)
 - 시스템의 백그라운드에서 서비스별로 항상 동작
 - 자주 호출되는 데몬이 아니라면 시스템의 자원을 낭비할 우려
- 수퍼 데몬에 의한 동작 방식
 - 평소에는 수퍼 데몬만 동작하다가 서비스 요청이 오면 수퍼 데몬이 해당 데몬을 동작 시킴
 - 독자형보다는 서비스에 응답하는 데 시간이 약간 더 걸릴 수 있지만 자원을 효율적으로 사용한다는 장점

■ 수퍼 데몬

- 유닉스에서 수퍼 데몬의 이름은 inetd
- 페도라에서는 보안 기능이 포함된 xinetd를 사용

04 데몬 프로세스

■ 데몬의 조상 : init와 커널 스레드 데몬

■ init 데몬

- 대부분의 프로세스의 조상 프로세스
- pstree 명령으로 확인

```
user1@myubuntu:~$ pstree
init──NetworkManager──dhclient
    │                   ├──dnsmasq
    │                   └──3*[NetworkManager]
    ├──accounts-daemon──2*[accounts-daemon]
    ├──acpid
    ├──at-spi-bus-laun──dbus-daemon
    │                 └──3*[at-spi-bus-laun]
    ├──at-spi2-registr──at-spi2-registr
    │                 ├──atd
    │                 ├──bluetoothd
    │                 ├──colord──2*[colord]
    │                 ├──cron
    │                 └──cups-browsed
```

(생략)

04 데몬 프로세스

■ 커널 스레드 데몬

- 커널의 일부분을 프로세스처럼 관리하는 데몬
- ps 명령으로 확인했을 때 대괄호([])로 둘러싸여 있는 프로세스들
- 예전에는 대부분 k로 시작했으나 요즘은 이를 반드시 준수하지는 않음
- 커널 데몬은 대부분 입출력이나 메모리 관리, 디스크 동기화 등을 수행하며 대체로 PID가 낮은 번호로 할당
- 커널 데몬을 동작시키는 조상 데몬은 커널 스레드 데몬(kthreadd): PID 2번

```
user1@myubuntu:~$ ps -ef | more
UID          PID    PPID  C STIME TTY          TIME CMD
root           1        0  0 23:38 ?        00:00:01 /sbin/init
root           2        0  0 23:38 ?        00:00:00 [kthreadd]
root           3        2  0 23:38 ?        00:00:00 [ksoftirqd/0]
root           5        2  0 23:38 ?        00:00:00 [kworker/0:0H]
root           7        2  0 23:38 ?        00:00:00 [migration/0]
root           8        2  0 23:38 ?        00:00:00 [rcu_bh]
root           9        2  0 23:38 ?        00:00:00 [rcu_sched]
root          10        2  0 23:38 ?        00:00:00 [watchdog/0]
root          11        2  0 23:38 ?        00:00:00 [khelper]
root          12        2  0 23:38 ?        00:00:00 [kdevtmpfs]
(생략)
```

04 데몬 프로세스

■ 주요 데몬

[표 8-3] 리눅스의 주요 데몬

데몬	기능	데몬	기능
atd	특정 시간에 실행하도록 예약한 명령을 실행한다(at 명령으로 예약).	smtpd	메일 전송 데몬이다.
		popd	기본 편지함 서비스를 제공한다.
crond	주기적으로 실행하도록 예약한 명령을 실행한다.	routed	자동 IP 라우터 테이블 서비스를 제공한다.
dhcpcd	동적으로 IP 주소를 부여할 수 있도록 하는 서비스를 제공한다.	smb	삼바 서비스를 제공한다.
		syslogd	로그 기록 서비스를 제공한다.
httpd	웹 서비스를 제공한다.	sshd	원격 보안 접속 서비스를 제공한다.
lpd	프린트 서비스를 제공한다.	in.telnetd	원격 접속 서비스를 제공한다.
nfs	네트워크 파일 시스템 서비스를 제공한다.	ftpd	파일 송수신 서비스를 제공한다.
named	DNS 서비스를 제공한다.	ntpd	시간 동기화 서비스를 제공한다.
sendmail	이메일 서비스를 제공한다.		

05 부트 로더

■ GRUB의 개요

- 'GRand Unified Bootloader'의 약자로, 리눅스의 전통적인 부트 로더인 LILO의 단점을 보완하여 GNU 프로젝트의 일환으로 개발
- GRUB는 LILO에 비해 다음과 같은 장점을 가지고 있음
 - LILO는 리눅스에서만 사용이 가능하지만 GRUB는 윈도우에서도 사용할 수 있다.
 - LILO에 비해 설정과 사용이 편리하다.
 - 부팅 시에 명령을 사용하여 수정이 가능하다.
 - 멀티 부팅 기능을 지원한다.
- GRUB의 가장 최신 버전은 GRUB2로 우분투에서 기본 부트 로더로 사용중

05 부트 로더

■ GRUB2 관련 디렉터리와 파일

- /boot/grub2/grub.cfg 파일: 기존의 menu.lst 파일을 대체하는 기본 설정 파일

```
user1@myubuntu:~$ more /boot/grub/grub.cfg
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#
### BEGIN /etc/grub.d/00_header ###
if [ -s $prefix/grubenv ]; then
    set have_grubenv=true
    load_env
fi
set default="0"
if [ x"$feature_menuentry_id" = xy ]; then
    menuentry_id_option="--id"
else
    menuentry_id_option=""
fi
(생략)
```

05 부트 로더

■ GRUB2 관련 디렉터리와 파일

- /etc/grub.d 디렉터리: GRUB 스크립트를 가지고 있으며 GRUB의 명령이 실행될 때 순서대로 읽혀 grub.cfg 파일이 생성

```
user1@myubuntu:~$ ls /etc/grub.d
00_header          10_linux          20_memtest86+    30_uefi-firmware  41_custom
05_debian_theme    20_linux_xen      30_os-prober     40_custom         README
user1@myubuntu:~$
```

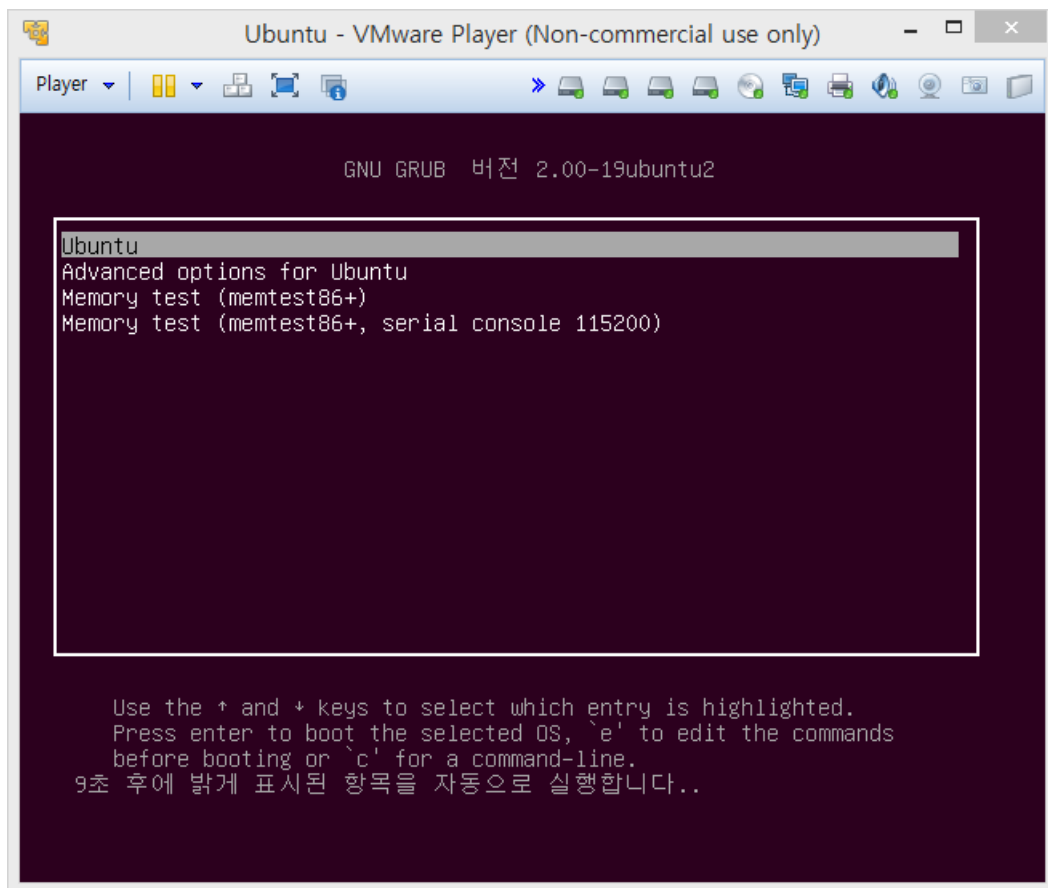
- /etc/default/grub 파일: GRUB 메뉴 설정 내용이 저장

```
user1@myubuntu:~$ more /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'
GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
(생략)
```

05 부트 로더

■ 단일 사용자 모드로 부팅하기

- ① 시스템 재시작하기: 부팅할 때 GRUB 메뉴 초기 화면이 출력

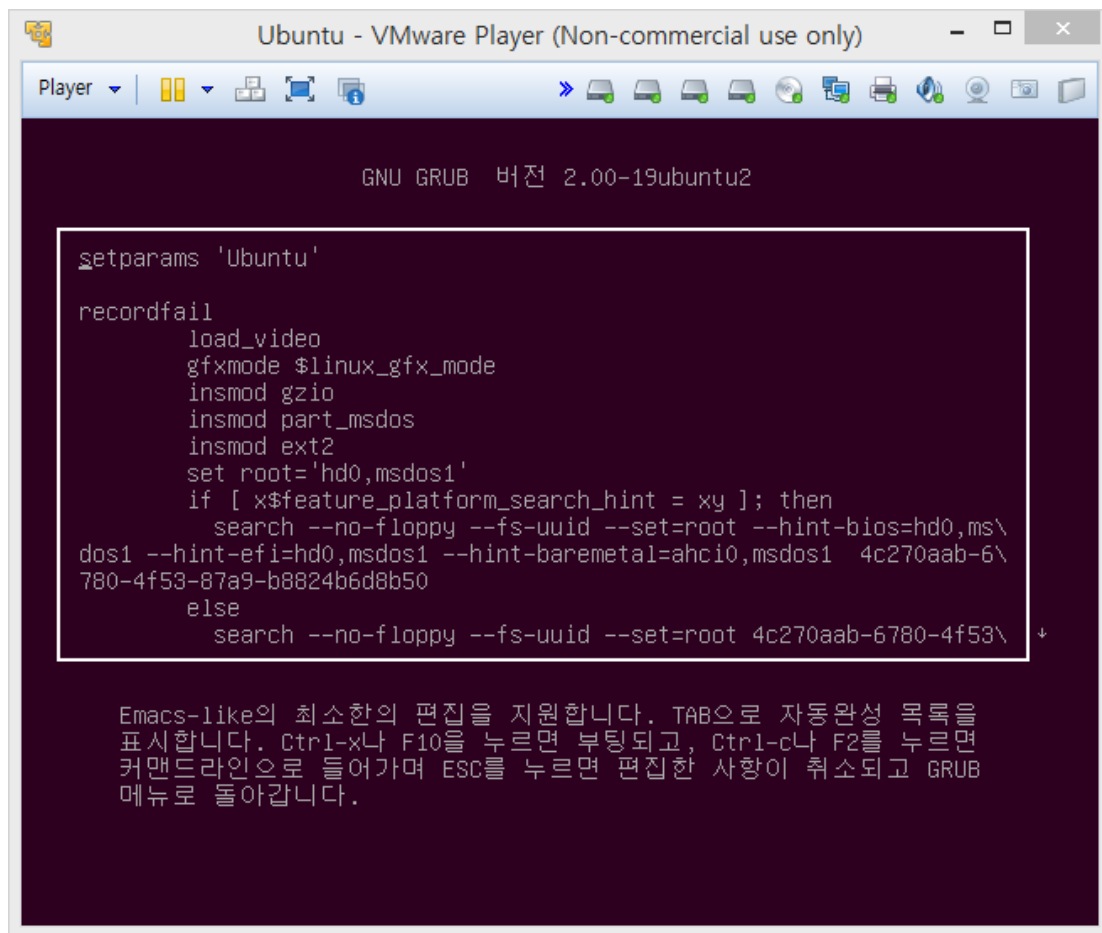


[그림 8-9] GRUB 메뉴 초기 화면

05 부트 로더

■ 단일 사용자 모드로 부팅하기

② GRUB 편집 모드로 전환하기: GRUB Boot Menu가 출력될 때 신속하게 'e' 키를 눌러서 편집 모드로 전환



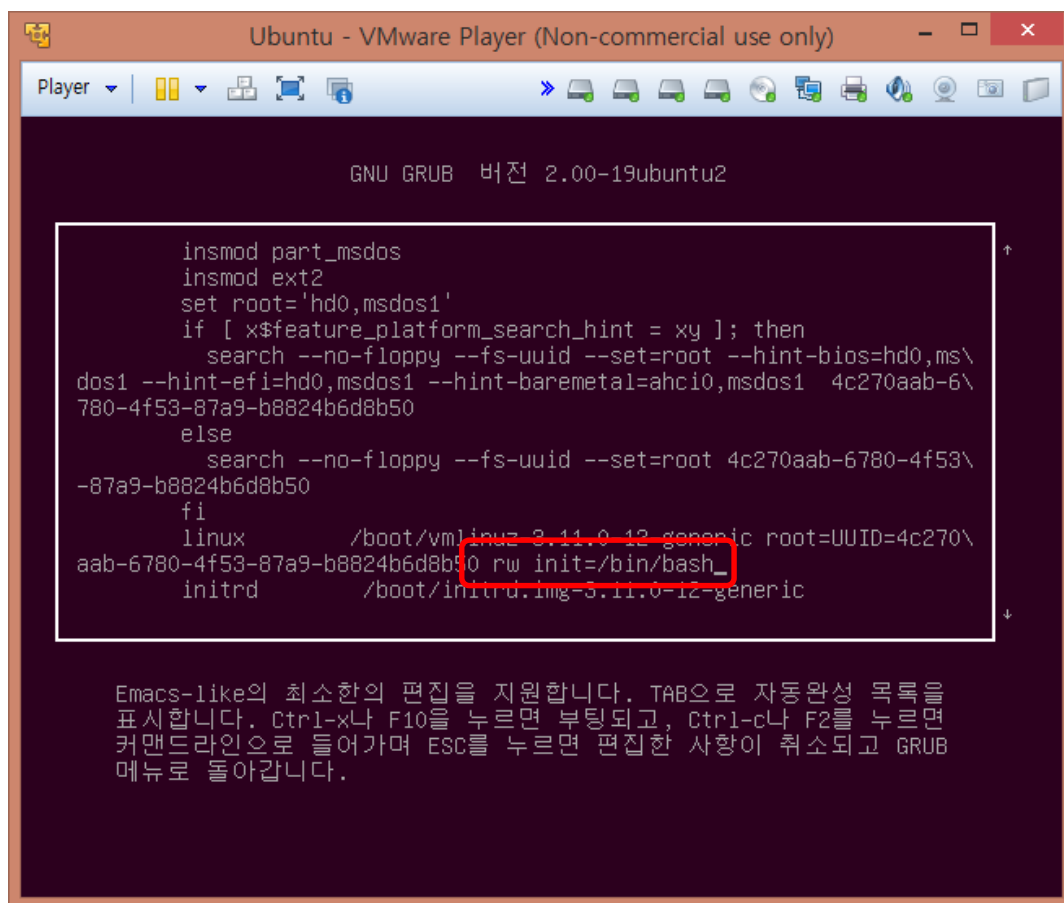
[그림 8-10] GRUB 편집 화면

05 부트 로더

■ 단일 사용자 모드로 부팅하기

③ 단일 사용자 모드로 수정하기

- 리눅스 커널 정보가 있는 행에서 'ro quiet splash \$vt_handoff'를 'rw init=/bin/bash'로 수정

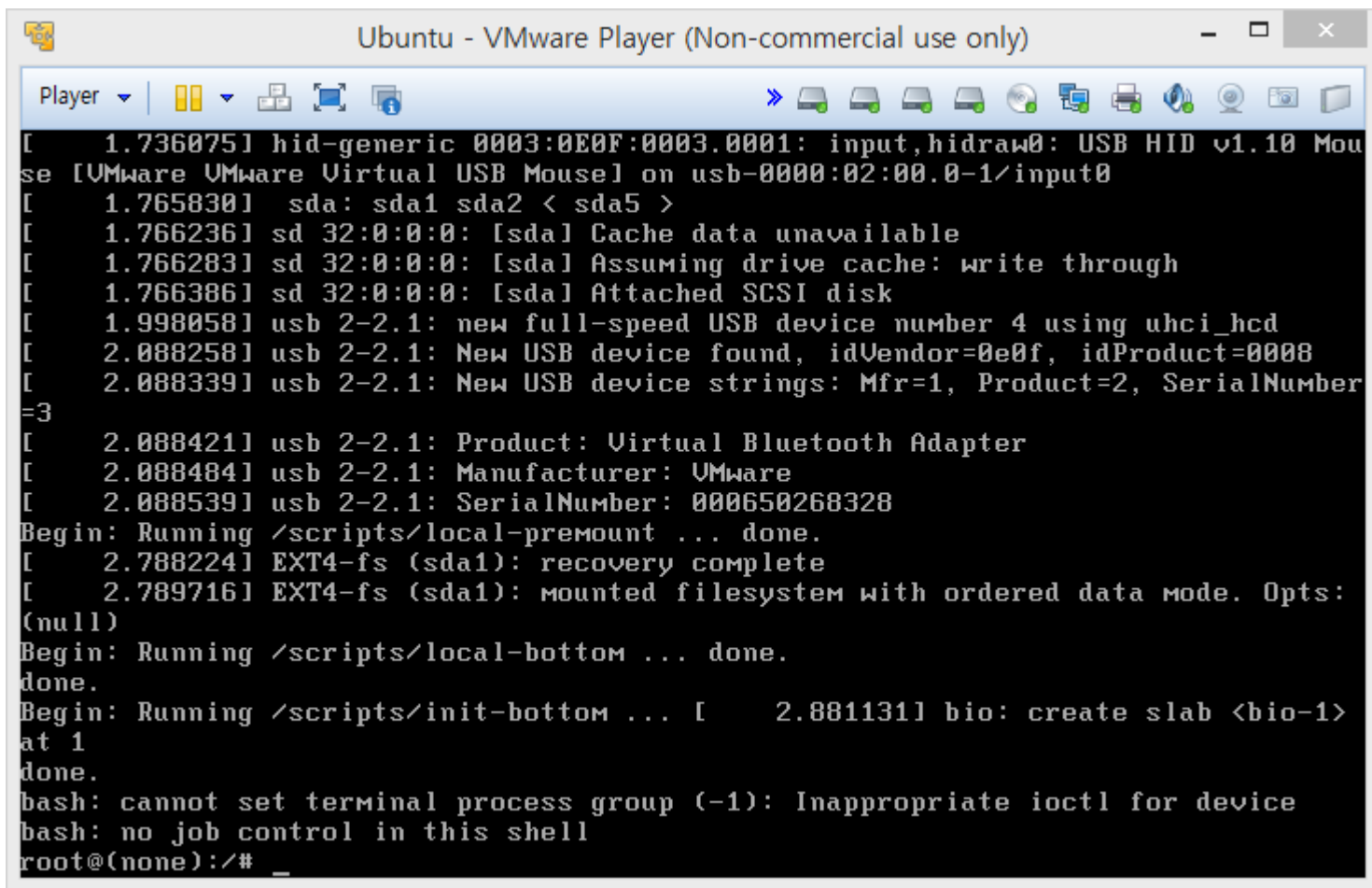


[그림 8-11] 단일사용자 모드로 부팅하기 위해 커널 항목 수정

05 부트 로더

■ 단일 사용자 모드로 부팅하기

④ F10키를 눌러 재시작하면 바로 root 계정으로 동작



```
[ 1.736075] hid-generic 0003:0E0F:0003.0001: input,hidraw0: USB HID v1.10 Mouse [VMware VMware Virtual USB Mouse] on usb-0000:02:00.0-1/input0
[ 1.765830] sda: sda1 sda2 < sda5 >
[ 1.766236] sd 32:0:0:0: [sda] Cache data unavailable
[ 1.766283] sd 32:0:0:0: [sda] Assuming drive cache: write through
[ 1.766386] sd 32:0:0:0: [sda] Attached SCSI disk
[ 1.998058] usb 2-2.1: new full-speed USB device number 4 using uhci_hcd
[ 2.088258] usb 2-2.1: New USB device found, idVendor=0e0f, idProduct=0008
[ 2.088339] usb 2-2.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 2.088421] usb 2-2.1: Product: Virtual Bluetooth Adapter
[ 2.088484] usb 2-2.1: Manufacturer: VMware
[ 2.088539] usb 2-2.1: SerialNumber: 000650268328
Begin: Running /scripts/local-premount ... done.
[ 2.788224] EXT4-fs (sda1): recovery complete
[ 2.789716] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null)
Begin: Running /scripts/local-bottom ... done.
done.
Begin: Running /scripts/init-bottom ... [ 2.881131] bio: create slab <bio-1> at 1
done.
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
root@(none):/#
```

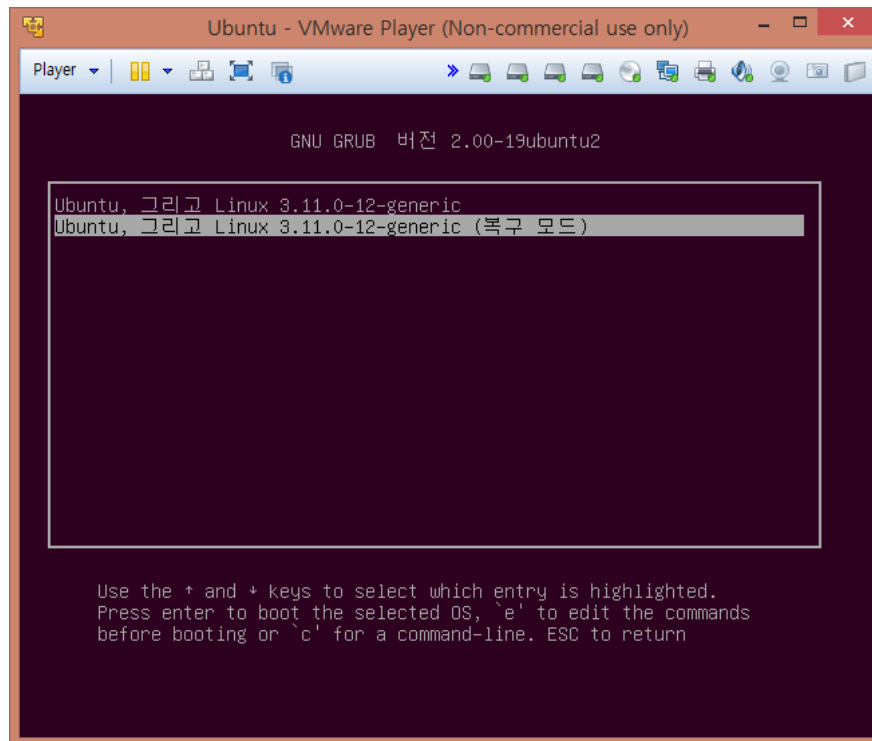
[그림 8-12] root 계정 화면

05 부트 로더

■ 복구 모드로 부팅하기

- 어떤 이유에서든 우분투가 부팅되지 않는다면 복구 모드로 부팅하는 것이 유용
- 복구 모드에서는 root 계정으로 로그인하여 시스템의 복구에 필요한 작업을 수행할 수 있음

① 시스템을 재시작하고 GRUB 메뉴 초기 화면에서 'Advanced options for Ubuntu'를 선택



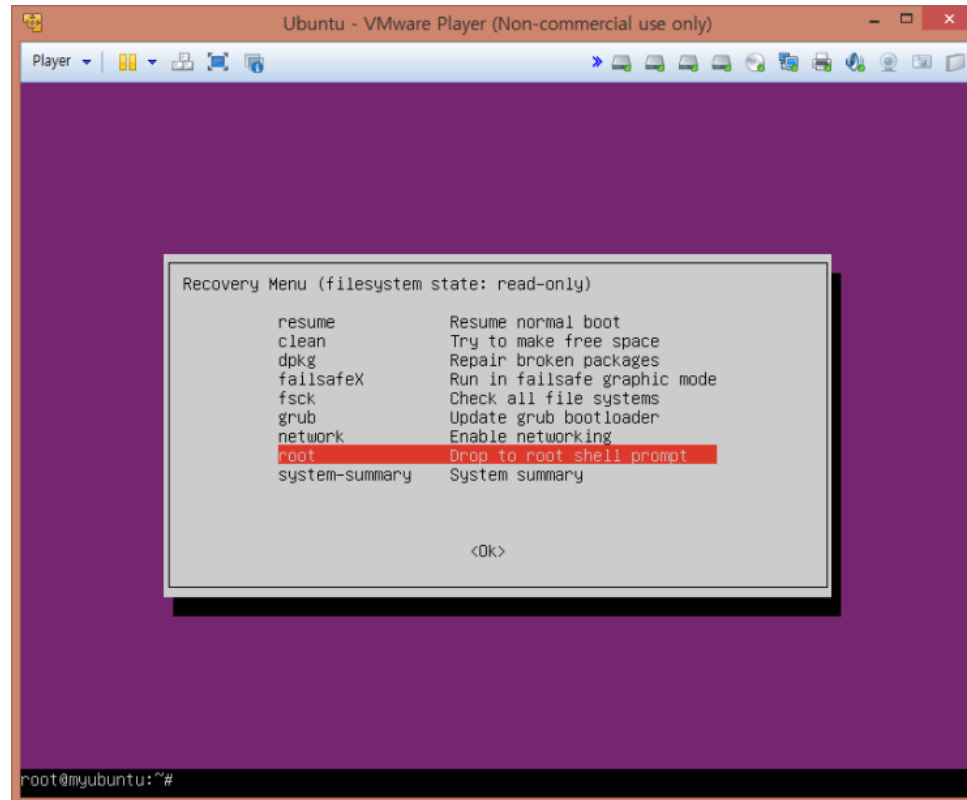
[그림 8-13] 복구 모드 선택 화면

05 부트 로더

■ 복구 모드로 부팅하기

② 복구 메뉴 화면

- Ubuntu, 그리고 Linux 3.11.0-12-generic (복구 모드)'를 선택하면 부팅 과정이 진행되다가 복구 메뉴 화면이 나온다. 이 메뉴에서 'Drop to root shell prompt'를 선택

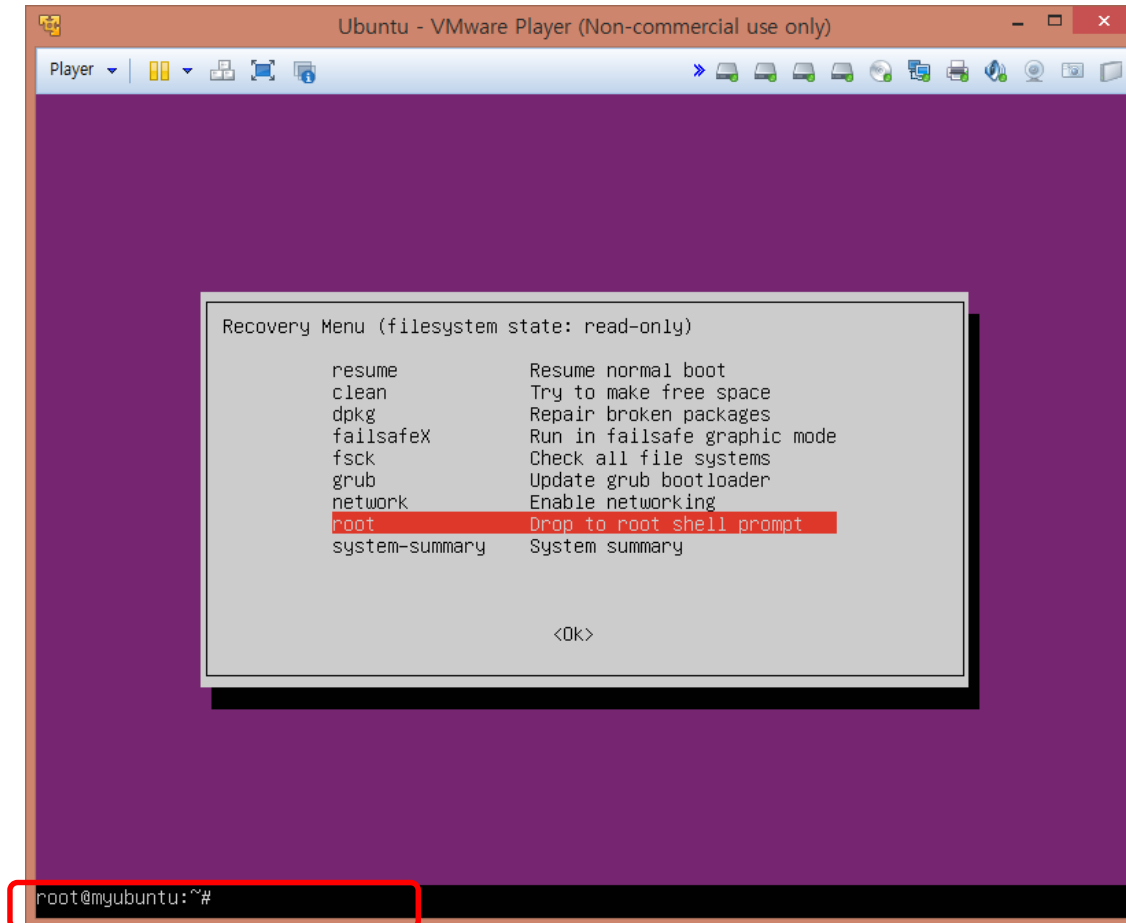


[그림 8-14] 복구 메뉴 선택 화면

05 부트 로더

■ 복구 모드로 부팅하기

③ Drop to root shell prompt'를 선택하면 root 프롬프트가 출력



[그림 8-15] root 프롬프트 출력 화면

05 부트 로더

■ 복구 모드로 부팅하기

④ root 프롬프트가 출력되면 복구 작업 가능

- root 파일 시스템이 읽기 전용으로 마운트되었으므로 읽기, 쓰기가 가능하도록 다시 마운트하고 작업

```
root@myubuntu:~# mount -o remount,rw /
```

⑤ 작업이 완료되면 `reboot -f` 명령으로 리눅스를 재시작



우분투 리눅스

시스템 & 네트워크