우분투 리눅스

시스템 & 네트워크

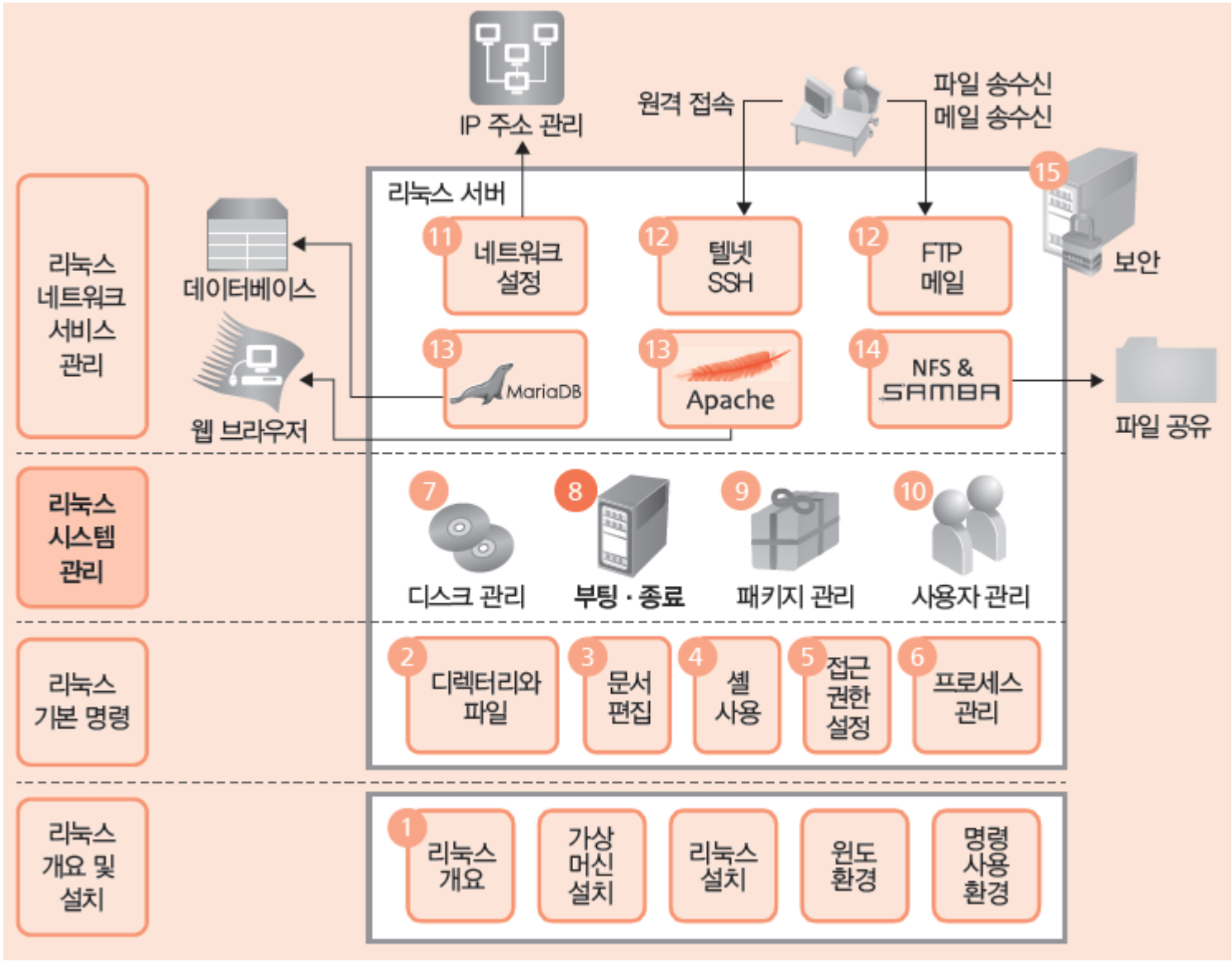
Chapter 08. 리눅스의 부팅과 종료

목차

학습목표

- 리눅스 시스템의 부팅 과정을 이해하고 부트 로더의 역할을 설명할 수 있다.
- systemd 프로세스의 역할을 설명할 수 있다.
- systemd의 유닛을 설명할 수 있다.
- systemctl 명령으로 유닛을 시작·종료하고 상태를 확인할 수 있다.
- 런레벨을 이해하고 변경할 수 있다.
- 리눅스 시스템을 종료할 수 있다.
- 데몬을 이해하고 슈퍼데몬의 역할을 설명할 수 있다.
- 단일 사용자 모드로 부팅할 수 있다.
- root 계정의 암호를 복구할 수 있다.

리눅스 실습 스터디 맵


IP 주소 관리
원격 접속
파일 송수신
메일 송수신
리눅스 서버
15
보안
리눅스 네트워크 서비스 관리
데이터베이스
웹 브라우저
11 네트워크 설정
12 텔넷 SSH
12 FTP 메일
13 MariaDB
13 Apache
14 NFS & SAMBA
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 개요

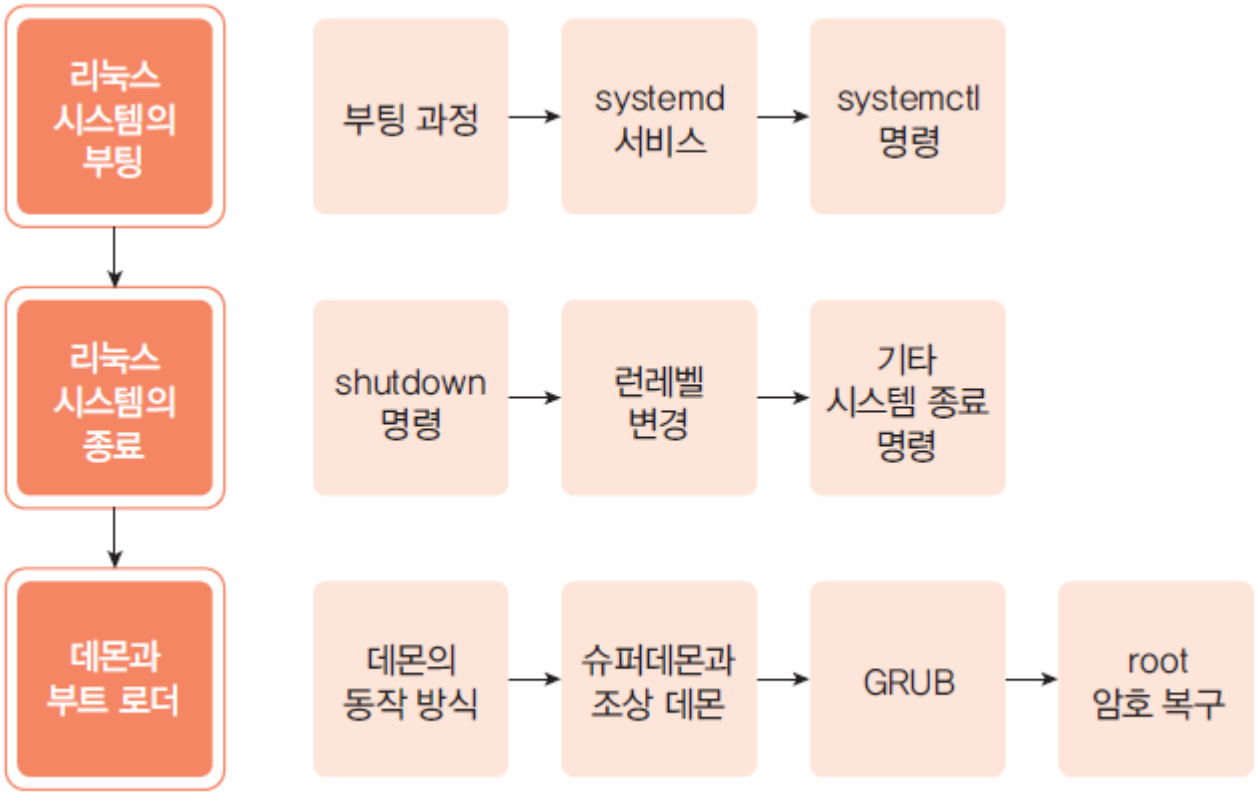


그림 8-1 8장의 내용 구성

01 리눅스 시스템의 부팅

■ 리눅스 시스템의 부팅 과정

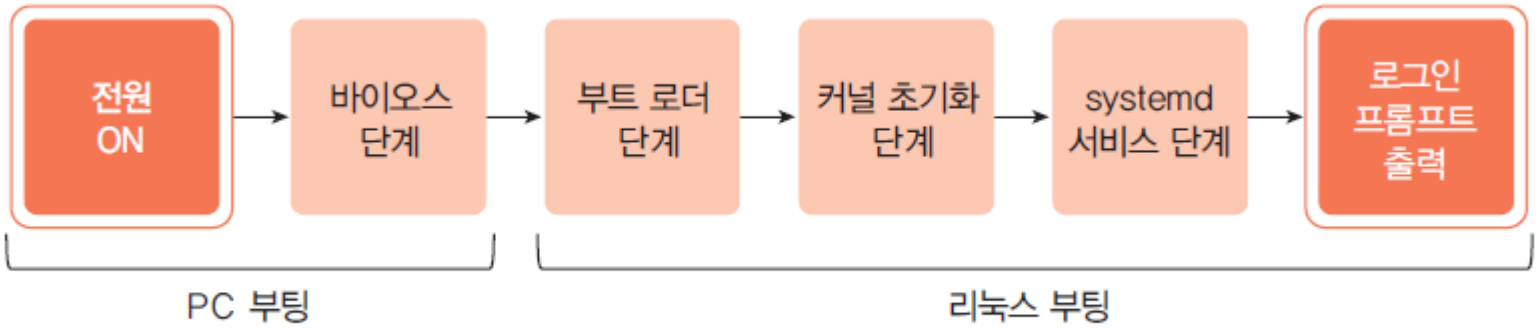


그림 8-2 리눅스의 부팅 과정

■ 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

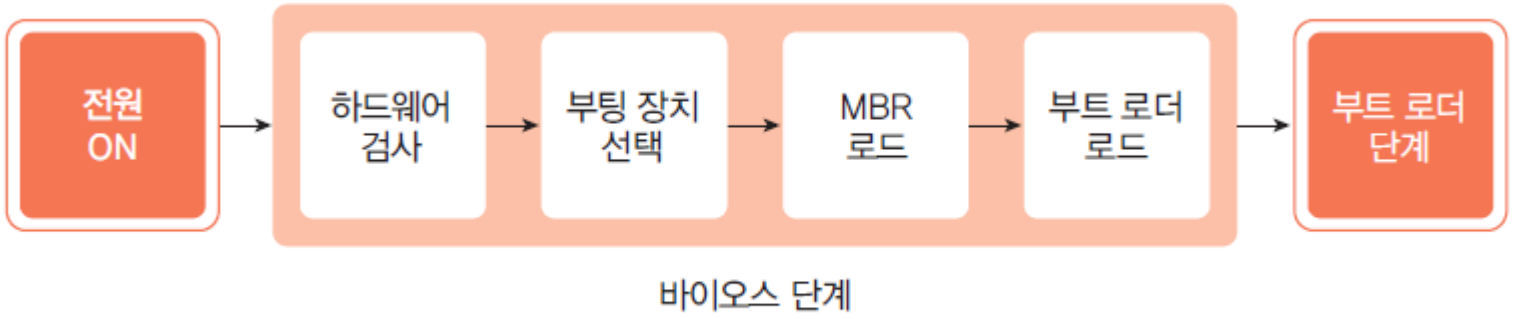


그림 8-3 바이오스 단계의 세부 동작

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

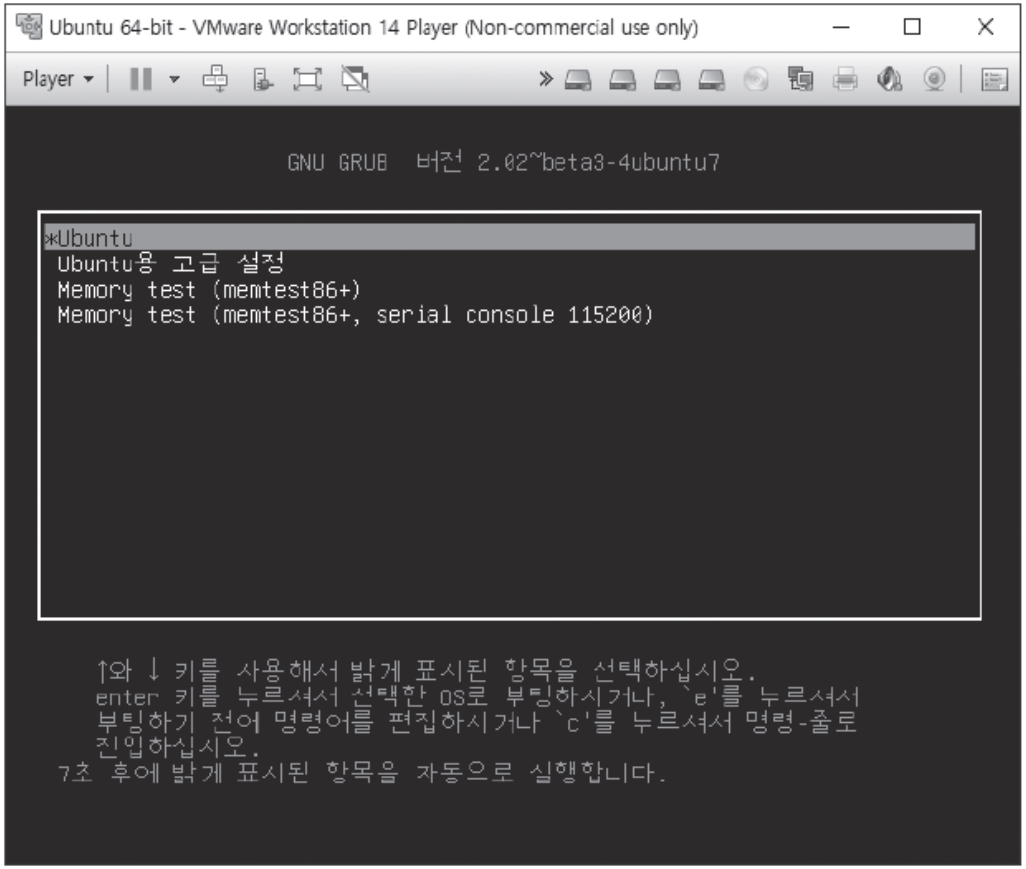


그림 8-4 부트 로더 시작 화면

01 리눅스 시스템의 부팅

부트 로더 단계

- 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-4.13.0-16-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 18:17 ?        00:00:01 /sbin/init splash
root           2       0  0 18:17 ?        00:00:00 [kthreadd]
root           3       2  0 18:17 ?        00:00:00 [kworker/0:0]
root           4       2  0 18:17 ?        00:00:00 [kworker/0:0H]
root           6       2  0 18:17 ?        00:00:00 [mm_percpu_wq]
root           7       2  0 18:17 ?        00:00:00 [ksoftirqd/0]
root           8       2  0 18:17 ?        00:00:00 [rcu_sched]
root           9       2  0 18:17 ?        00:00:00 [rcu_bh]
root          10       2  0 18:17 ?        00:00:00 [migration/0]
root          11       2  0 18:17 ?        00:00:00 [watchdog/0]
root          12       2  0 18:17 ?        00:00:00 [cpuhp/0]
(생략)
```

01 리눅스 시스템의 부팅

systemd 서비스 단계

- 우분투에서 systemd 서비스는 기존의 init 스크립트를 대체한 것으로 다양한 서비스를 동작
- 각 서비스가 시작하는 과정은 화면에 메시지로 출력

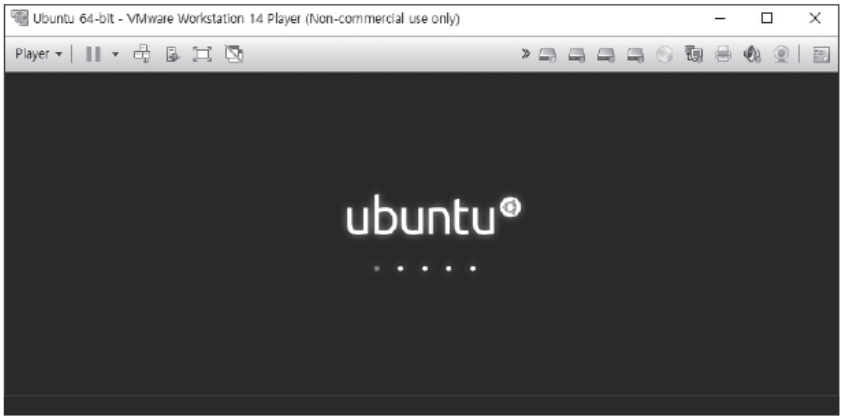


그림 8-5 우분투의 부트 스플래시 화면

- 부트 스플래시 화면이 바로 종료되고 [그림 8-6]과 같이 메시지가 출력되는 화면으로 전환

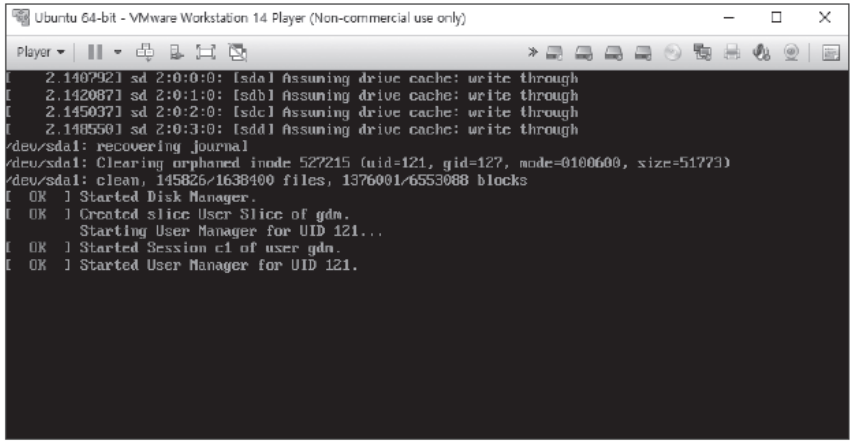


그림 8-6 부팅 메시지 출력 화면

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] random: get_random_bytes called from start_kernel+0x42/0x4e1 with
crng_init=0
[    0.000000] Linux version 4.13.0-16-generic (buildd@lcy01-02) (gcc version 7.2.0
(Ubuntu 7.2.0-8ubuntu2)) #19-Ubuntu SMP Wed Oct 11 18:35:14 UTC 2017 (Ubuntu 4.13.
0-16.19-generic 4.13.4)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.13.0-16-generic root=UUID
=7009cb18-dbd5-4ffc-af86-599cee765454 ro quiet splash
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Centaur CentaurHauls
[    0.000000] Disabled fast string operations
[    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point reg
isters'
(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 18:39 ?        00:00:01 /sbin/init splash
root           2       0  0 18:39 ?        00:00:00 [kthreadd]
root           3       2  0 18:39 ?        00:00:00 [kworker/0:0]
root           4       2  0 18:39 ?        00:00:00 [kworker/0:0H]
(생략)
```

로그인 프롬프트 출력

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

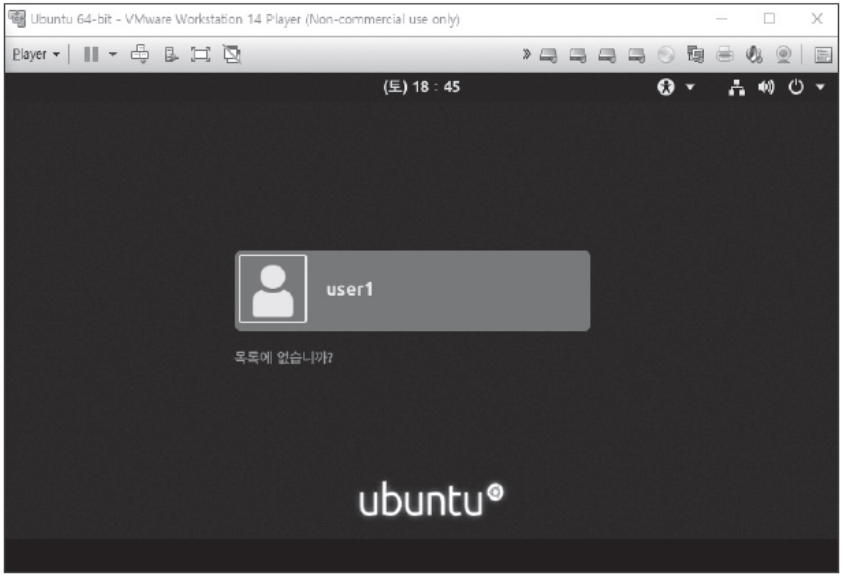


그림 8-7 로그인 프롬프트 화면

02 systemd 서비스

init 프로세스

- init 프로세스는 스크립트를 순차적으로 실행하여 다른 프로세스를 동작
- 우분투는 14.10 버전까지 자체적으로 개발한 upstart를 기존의 init 대신 사용해왔으나 현재는 upstart를 기본적으로 설치하지 않음
- upstart를 사용하려고 하면 다음과 같은 메시지가 출력

```
user1@myubuntu:~$ upstart
프로그램 'upstart'을(를) 설치하지 않습니다. 다음을 입력해 설치할 수 있습니다:
sudo apt install upstart
user1@myubuntu:~$
```

02 systemd 서비스

init 프로세스

- init가 systemd로 대체

```
user1@myubuntu:~$ man init
SYSTEMD(1)                          systemd                          SYSTEMD(1)

NAME
       systemd, init - systemd system and service manager

SYNOPSIS
       systemd [OPTIONS...]

       init [OPTIONS...] {COMMAND}

DESCRIPTION
       systemd is a system and service manager for Linux operating systems.
       When run as first process on boot (as PID 1), it acts as init system
       that brings up and maintains userspace services.
(생략)
```

02 systemd 서비스

init 프로세스

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

```
user1@myubuntu:~$ ls /etc/init
anacron.conf       rfkill-restore.conf  thermald.conf
gpu-manager.conf   rfkill-store.conf    whoopsie.conf
user1@myubuntu:~$
```

- init와 관련된 스크립트 파일은 /etc/init.d 디렉터리에 있으며 아직 일부 서비스의 스크립트 파일이 남아 있음

```
user1@myubuntu:~$ ls /etc/init.d
acpid             cups-browsed        lvm2-lvmetad      saned
alsa-utils        dbus                lvm2-lvmpolld     speech-dispatcher
anacron           dns-clean           network-manager   ssh
apparmor          gdm3                networking        thermald
apport            grub-common         plymouth          udev
atd               hwclock.sh          plymouth-log      ufw
avahi-daemon      irqbalance          postfix           unattended-upgrades
bluetooth         kerneloops          pppd-dns          uuidd
console-setup.sh  keyboard-setup.sh   procps            whoopsie
cron              kmod                rsync             x11-common
cups              lvm2                rsyslog
user1@myubuntu:~$
```

02 systemd 서비스

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

표 8-1 우분투의 런레벨

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

02 systemd 서비스

init 프로세스와 런레벨

- 런레벨 2, 3, 4번이 동일하다는 것은 /etc/rc2.d, /etc/rc3.d, /etc/rc4.d 디렉터리의 내용이 모두 같다는 것으로 알 수 있음

```
user1@myubuntu:~$ ls /etc/rc*.d
(생략)
/etc/rc2.d:
S01acpid            S01cron           S01kerneloops     S01saned
S01anacron          S01cups           S01lvm2-lvmetad   S01speech-dispatcher
S01apport           S01cups-browsed   S01lvm2-lvmpolld  S01ssh
S01atd              S01dbus           S01plymouth       S01thermald
S01avahi-daemon     S01gdm3           S01postfix        S01unattended-upgrades
S01bluetooth        S01grub-common    S01rsync          S01uuidd
S01console-setup.sh S01irqbalance     S01rsyslog        S01whoopsie

/etc/rc3.d:
S01acpid            S01cron           S01kerneloops     S01saned
S01anacron          S01cups           S01lvm2-lvmetad   S01speech-dispatcher
S01apport           S01cups-browsed   S01lvm2-lvmpolld  S01ssh
(생략)
user1@myubuntu:/etc$
```

02 systemd 서비스

init 프로세스와 런레벨

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

```
user1@myubuntu:~$ ls -l /etc/rc2.d
합계 0
lrwxrwxrwx 1 root root 15 11월  8 23:10 S01acpid -> ../init.d/acpid
lrwxrwxrwx 1 root root 17 11월  8 23:10 S01anacron -> ../init.d/anacron
lrwxrwxrwx 1 root root 16 11월  8 23:10 S01apport -> ../init.d/apport
lrwxrwxrwx 1 root root 13 11월 17 15:55 S01atd -> ../init.d/atd
(생략)
user1@myubuntu:~$
```

02 systemd 서비스

systemd의 기본 개념

- systemd는 init 방식에 비해 다음과 같은 장점을 가지고 있음
  - 소켓 기반으로 동작하여 inetd와 호환성을 유지한다.
  - 셸과 독립적으로 부팅이 가능하다.
  - 마운트 제어가 가능하다.
  - fsck 제어가 가능하다.
  - 시스템 상태에 대한 스냅숏을 유지한다.
  - SELinux와 통합이 가능하다.
  - 서비스에 시그널을 전달할 수 있다.
  - shutdown 전에 사용자 세션의 안전한 종료가 가능하다.

02 systemd 서비스

systemd 유닛

- 전체 시스템을 시작하고 관리하는 데 유닛(unit)이라 부르는 구성 요소를 사용
- systemd는 관리 대상의 이름을 '서비스명.유닛 종류'의 형태로 관리
- 각 유닛은 같은 이름과 종류로 구성된 설정 파일과 동일한 이름을 사용
- 유닛과 관련한 보다 자세한 내용은 'man systemd.유닛명'으로 확인

표 8-2 systemd 유닛의 종류

유닛	기능	예
service	가장 명백한 유닛으로 데몬을 시작·종료·재시작·로딩한다.	atd.service
socket	소켓을 관리하는 유닛으로 AF_INET, AF_INET6, AF_UNIX 소켓 스트림과 데이터그램, FIFO를 지원한다.	dbus.socket
device	리눅스 장치 트리에 있는 장치를 관리한다.	sys-module-fuse.device
mount	디렉터리 계층 구조의 마운트 포인트를 관리한다.	boot.mount
automount	디렉터리 계층 구조에서 자동 마운트 포인트를 관리한다.	proc-sys-fs-binfmt_misc.automount
target	유닛을 그루핑한다(예: multi-user.target → 런레벨 5에 해당하는 유닛).	basic.target multi-user.target
swap	스왑 장치를 관리한다.	dev-mapper-fedora\x2dswap.swap
path	경로를 관리한다.	cups.path
timer	타이머와 관련된 기능을 관리한다.	dnf-makecache.timer
slice	프로세스 그룹의 자원을 계층적으로 관리한다.	system-getty.slice
scope	외부에서 생성된 프로세스를 관리한다.	init.scope

02 systemd 서비스

systemd 관련 명령

- systemd 기반으로 서비스를 시작하거나 종료할 때 사용하는 명령: systemctl

systemctl

- **기능** systemd를 제어한다.
- **형식** systemctl [옵션] [명령] [유닛명]
- **옵션** -a: 상태와 관계없이 유닛 전체를 출력한다.
  -t 유닛 종류: 지정한 종류의 유닛만 출력한다.
- **명령** start: 유닛을 시작한다.
  stop: 유닛을 정지한다.
  reload: 유닛의 설정 파일을 다시 읽어온다.
  restart: 유닛을 재시작한다.
  status: 유닛 상태를 출력한다.
  enable: 부팅 시 유닛이 시작하도록 설정한다.
  disable: 부팅 시 유닛이 시작하지 않도록 설정한다.
  is-active: 유닛이 동작하고 있는지 확인한다.
  is-enabled: 유닛이 시작되었는지 확인한다.
  isolate: 지정한 유닛 및 이와 관련된 유닛만 시작하고 나머지는 정지한다.
  kill: 유닛에 시그널을 전송한다.
- **사용 예**
```
systemctl
systemctl -a
systemctl start atd.service
```

02 systemd 서비스

동작 중인 유닛 출력하기

- 옵션이나 명령 없이 systemctl 명령만 사용하면 현재 동작 중인 유닛이 출력

```
user1@myubuntu:~$ systemctl
UNIT                          LOAD   ACTIVE SUB     DESCRIPTION
(생략)
apparmor.service              loaded active exited  AppArmor initialization
keyboard-setup.service        loaded active exited  Set the console keyboard l
kmod-static-nodes.service     loaded active exited  Create list of required st
lvm2-monitor.service          loaded active exited  Monitoring of LVM2 mirrors
rescue.service                loaded active running Rescue Shell
setvtrgb.service              loaded active exited  Set console scheme
systemd-journal-flush.service loaded active exited  Flush Journal to Persisten
systemd-journald.service      loaded active running Journal Service
systemd-modules-load.service  loaded active exited  Load Kernel Modules
systemd-random-seed.service   loaded active exited  Load/Save Random Seed
systemd-remount-fs.service    loaded active exited  Remount Root and Kernel Fi
systemd-sysctl.service        loaded active exited  Apply Kernel Variables
(생략)
```

02 systemd 서비스

전체 유닛 출력하기: -a 옵션

- systemctl 명령에 -a 옵션을 지정하면 전체 유닛이 출력

```
user1@myubuntu:~$ systemctl -a
  UNIT                      LOAD      ACTIVE   SUB    DESCRIPTION
(생략)
  atd.service               loaded    inactive dead   Deferred execution schedu
● auditd.service            not-found inactive dead   auditd.service
  avahi-daemon.service      loaded    inactive dead   Avahi mDNS/DNS-SD Stack
  bluetooth.service         loaded    inactive dead   Bluetooth service
● connman.service           not-found inactive dead   connman.service
● console-screen.service    not-found inactive dead   console-screen.service
  console-setup.service     loaded    inactive dead   Set console font and keym
  cron.service              loaded    inactive dead   Regular background progra
  cups-browsed.service      loaded    inactive dead   Make remote CUPS printers
  cups.service              loaded    inactive dead   CUPS Scheduler
  dbus.service              loaded    inactive dead   D-Bus System Message Bus
(생략)
```

02 systemd 서비스

특정 유닛 출력하기: -t 옵션

- 특정 종류의 유닛만 출력하려면 -t 옵션을 사용
- service 유닛만 출력한 예

```
user1@myubuntu:~$ systemctl -t service
UNIT                            LOAD   ACTIVE SUB     DESCRIPTION
apparmor.service                loaded active exited  AppArmor initialization
keyboard-setup.service          loaded active exited  Set the console keyboard l
kmod-static-nodes.service       loaded active exited  Create list of required st
lvm2-monitor.service            loaded active exited  Monitoring of LVM2 mirrors
rescue.service                  loaded active running Rescue Shell
setvtrgb.service                loaded active exited  Set console scheme
systemd-journal-flush.service   loaded active exited  Flush Journal to Persisten
systemd-journald.service        loaded active running Journal Service
systemd-modules-load.service    loaded active exited  Load Kernel Modules
systemd-random-seed.service     loaded active exited  Load/Save Random Seed

(생략)
```

02 systemd 서비스

유닛 서비스 시작하기: start 명령

- 유닛 서비스를 시작하려면 start 명령을 사용
- cron 유닛을 시작한 후 is-active 명령으로 동작 여부를 확인해보면 active 상태임을 알 수 있음

```
user1@myubuntu:~$ sudo systemctl start cron
user1@myubuntu:~$ systemctl is-active cron
active
user1@myubuntu:~$
```

02 systemd 서비스

유닛 상태 확인하기: status 명령

- 유닛의 상태를 확인하려면 status 명령을 사용
- cron.service의 상태를 출력한 예

```
user1@myubuntu:~$ systemctl status cron
● cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabl
   Active: active (running) since Sat 2017-11-18 21:33:50 KST; 1min 50s ago
     Docs: man:cron(8)
 Main PID: 1829 (cron)
    Tasks: 1 (limit: 19660)
   CGroup: /system.slice/cron.service
           └─1829 /usr/sbin/cron -f

11월 18 21:33:50 myubuntu systemd[1]: Started Regular background program
processin
11월 18 21:33:50 myubuntu cron[1829]: (CRON) INFO (pidfile fd = 3)
(생략)
```

02 systemd 서비스

유닛 서비스 정지하기: stop 명령

- 유닛 서비스를 정지하려면 stop 명령을 사용
- cron 유닛을 정지한 후 다시 status 명령으로 상태를 확인해보면 inactive(dead)

```
user1@myubuntu:~$ sudo systemctl stop cron
user1@myubuntu:~$ systemctl status cron
● cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabl
   Active: inactive (dead) since Sat 2017-11-18 21:36:38 KST; 8s ago
     Docs: man:cron(8)
  Process: 1829 ExecStart=/usr/sbin/cron -f $EXTRA_OPTS (code=killed, signal=TERM)
 Main PID: 1829 (code=killed, signal=TERM)

11월 18 21:33:50 myubuntu systemd[1]: Started Regular background program processin
11월 18 21:33:50 myubuntu cron[1829]: (CRON) INFO (pidfile fd = 3)
(생략)
```

02 systemd 서비스

systemd와 런레벨

- 런레벨은 현재 시스템의 상태를 나타내는 한 자리 숫자(문자 S, s 포함)
- 이에 대응하는 systemd의 target 유닛은 [표 8-3]과 같이 제공
- 이 파일들은 /lib/systemd/system 디렉터리에 있음

표 8-3 런레벨과 target 유닛의 관계

런레벨	target 파일(심벌릭 링크)	target 원본 파일
0	runlevel0.target	poweroff.target
1	runlevel1.target	rescue.target
2	runlevel2.target	multi-user.target
3	runlevel3.target	
4	runlevel4.target	
5	runlevel5.target	graphical.target
6	runlevel6.target	reboot.target

- 현재 런레벨 확인하기 : runlevel 명령

```
user1@myubuntu:~$ runlevel
N 5
user1@myubuntu:~$
```

02 systemd 서비스

기본 target 지정하기

- 부팅할 때 동작할 기본 런레벨은 기본 target으로 바뀌었고, 다음과 같은 형식으로 지정

```
systemctl set-default <name of target>.target
```

- 이 명령은 /etc/systemd/system 디렉터리 아래에 심벌릭 링크인 default.target이 가리키는 target 파일을 변경
- 현재 target인 graphical.target에서 multi-user.target으로 바꾸는 예

```
user1@myubuntu:~$ sudo systemctl set-default multi-user.target
Created symlink /etc/systemd/system/default.target -> /lib/systemd/system/multi-
user.target.
user1@myubuntu:~$ ls -l /etc/systemd/system/default.target
lrwxrwxrwx 1 root root 37 11월 18 21:47 /etc/systemd/system/default.target -> /
lib/systemd/system/multi-user.target
user1@myubuntu:~$
```

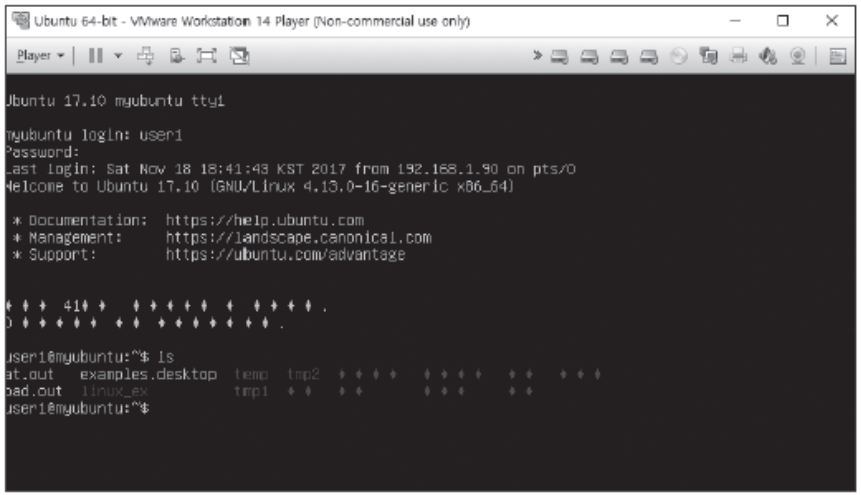


그림 8-8 multi-user.target 모드로 부팅한 화면

02 systemd 서비스

target 변경하기

- systemd에서 런레벨을 변경하는 것도 isolate 명령으로 간단히 해결
- multi-user.target(런레벨 3)으로 변경하려면 다음 명령 중 하나를 입력

```
systemctl isolate multi-user
```

```
systemctl isolate runlevel3
```

- graphical.target(런레벨 5)으로 변경하려면 다음 명령 중 하나를 사용

```
systemctl isolate graphical
```

```
systemctl isolate runlevel5
```

02 systemd 서비스

telinit, init 명령으로 런레벨 변경하기

- init는 init 프로세스의 런레벨을 바꿀 때 사용하기도 함
- 현재 init는 systemd에 대한 심벌릭 링크

```
user1@myubuntu:~$ ls -l /sbin/init
lrwxrwxrwx 1 root root 20 10월 26 20:56 /sbin/init -> /lib/systemd/systemd
user1@myubuntu:~$
```

- init 명령만 입력하면 다음과 같이 출력

```
user1@myubuntu:~$ init
init: required argument missing.
user1@myubuntu:~$
```

02 systemd 서비스

telinit, init 명령으로 런레벨 변경하기

- init --help로 사용법을 알아보면 다음과 같이 출력

```
user1@myubuntu:~$ init --help
init [OPTIONS...] {COMMAND}

Send control commands to the init daemon.

     --help      Show this help
     --no-wall   Don't send wall message before halt/power-off/reboot

Commands:
  0               Power-off the machine
  6               Reboot the machine
  2, 3, 4, 5      Start runlevelX.target unit
  1, s, S         Enter rescue mode
  q, Q            Reload init daemon configuration
  u, U            Reexecute init daemon
user1@myubuntu:~$
```

02 systemd 서비스

telinit, init 명령으로 런레벨 변경하기

- 런레벨을 바꾸는 명령으로 telinit

```
user1@myubuntu:~$ ls -l /sbin/telinit
lrwxrwxrwx 1 root root 14 10월 26 20:56 /sbin/telinit -> /bin/systemctl
user1@myubuntu:~$
```

- telinit 명령을 실행하면 init 명령과 같은 결과가 출력

```
user1@myubuntu:~$ telinit --help
telinit [OPTIONS...] {COMMAND}

Send control commands to the init daemon.

     --help        Show this help
     --no-wall     Don't send wall message before halt/power-off/reboot

Commands:
  0               Power-off the machine
  6               Reboot the machine
  2, 3, 4, 5      Start runlevelX.target unit
  1, s, S         Enter rescue mode
  q, Q            Reload init daemon configuration
  u, U            Reexecute init daemon
user1@myubuntu:~$
```

02 systemd 서비스

단일 사용자 모드로 전환하기: rescue.target(런레벨 1)

- 시스템에 문제가 있을 경우 시스템을 rescue.target 유닛(런레벨 1, 런레벨 S)으로 변경하여 점검
- 윈도의 안전 모드 같은 것으로, 다중 사용자 모드에서 시스템 관리자만 사용할 수 있는 단일 사용자 모드로 전환하는 것

```
systemctl isolate rescue
```

```
systemctl isolate runlevel1
```

```
init 1
```

```
telinit S
```

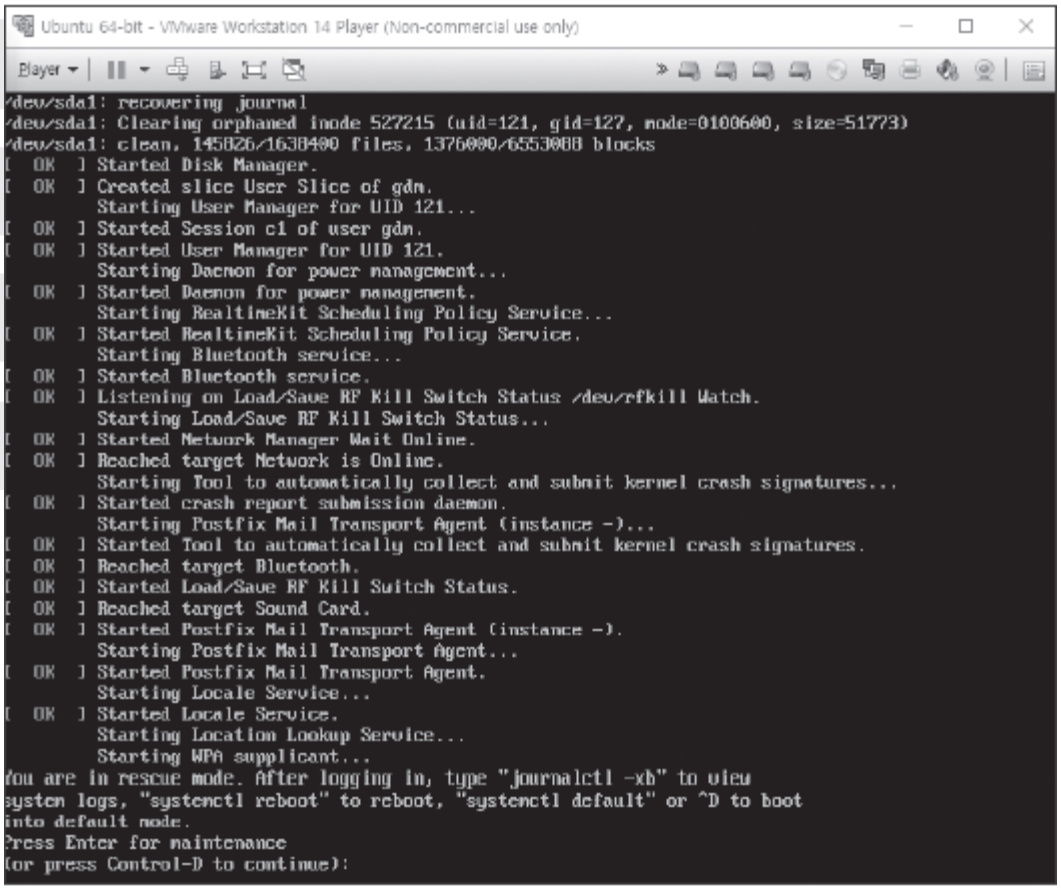


그림 8-9 런레벨 1로 바뀐 화면

03 리눅스 시스템의 종료

리눅스를 종료하는 방법

- shutdown 명령을 사용한다.
- halt 명령을 사용한다.
- poweroff 명령을 사용한다.
- 런레벨을 0이나 6으로 전환한다(target을 전환한다).
- reboot 명령을 사용한다.
- 전원을 끈다.

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 halt나 power-off 상태로 이동한다.
  -c: 이전에 했던 shutdown 명령을 취소한다.
  시간: 종료할 시간이다(hh:mm, +m, now).
  메시지: 모든 사용자에게 보낼 메시지이다.
- **사용 예** shutdown -h now shutdown -c
  shutdown -r +3 "System is going down"

shutdown 명령으로 시스템 즉시 종료하기

- -h 옵션과 함께 현재 시간으로 지정

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

03 리눅스 시스템의 종료

셧다운한다는 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 root@myubuntu on pts/0 (Sat 2017-11-18 22:04:49 KST):

System is going down in 2 min.
The system is going down for poweroff at Sat 2017-11-18 22:06:49 KST!
```

shutdown 명령으로 시스템 재시작하기: -r 옵션 사용

```
user1@myubuntu:~$ sudo shutdown -r +3
Shutdown scheduled for Sat 2017-11-18 22:09:21 KST, use 'shutdown -c' to cancel.
user1@myubuntu:~$
```

- 3분 후에 시스템 재시작

```
Broadcast message from root@myubuntu on pts/0 (Sat 2017-11-18 22:06:21 KST):

The system is going down for reboot at Sat 2017-11-18 22:09:21 KST!
```

03 리눅스 시스템의 종료

shutdown 명령 취소하기: -c 옵션

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

- 앞의 3분 후 재시작 명령 취소할 경우 메시지 출력

```
Broadcast message from root@myubuntu on pts/0 (Sat 2017-11-18 22:06:40 KST):

The system shutdown has been cancelled
```

shutdown 메시지만 보내기: -k 옵션

```
user1@myubuntu:~$ sudo shutdown -k 2
Shutdown scheduled for Sat 2017-11-18 22:09:27 KST, use 'shutdown -c' to cancel.
user1@myubuntu:~$
```

- -k 다음에 2를 지정하면 다른 사용자의 터미널에는 다음과 같은 메시지가 출력

```
Broadcast message from root@myubuntu on pts/0 (Sat 2017-11-18 22:07:27 KST):

The system is going down for poweroff at Sat 2017-11-18 22:09:27 KST!
```

03 리눅스 시스템의 종료

런레벨 변경하여 종료하기

- 런레벨을 0으로 바꾸면 시스템이 종료

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

- 재시작하려면 런레벨을 6으로 변경

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

systemd로 종료하기

- systemd에서 target 유닛을 변경하면 시스템을 종료하거나 재시작 가능

```
systemctl isolate poweroff.target
```

```
systemctl isolate runlevel0.target
```

- 시스템 재시작하기

```
systemctl isolate reboot.target
```

```
systemctl isolate runlevel6.target
```

03 리눅스 시스템의 종료

기타 시스템 종료 명령

- 시스템을 종료하거나 재시작하기 위해 사용할 수 있는 명령: reboot , halt, poweroff
- 이러현 명령은 모두 systemctl의 심벌릭 링크

```
user1@myubuntu:~$ ls -l /sbin/reboot
lrwxrwxrwx 1 root root 14 10월 26 20:56 /sbin/reboot -> /bin/systemctl
user1@myubuntu:~$ ls -l /sbin/halt
lrwxrwxrwx 1 root root 14 10월 26 20:56 /sbin/halt -> /bin/systemctl
user1@myubuntu:~$ ls -l /sbin/poweroff
lrwxrwxrwx 1 root root 14 10월 26 20:56 /sbin/poweroff -> /bin/systemctl
user1@myubuntu:~$
```

- reboot, halt, poweroff 명령은 /var/log/wtmp 파일에 시스템 종료 기록을 남기고 시스템을 종료하거나 재시작
- 사용할 수 있는 옵션
  - -n: 재시작이나 종료 전에 sync를 호출하지 않는다.
  - -w: 실제로 재시작하거나 종료하지는 않지만 wtmp 파일에 기록을 남긴다.
  - -d: wtmp 파일에 기록을 남기지 않는다. -n 옵션은 -d 옵션을 포함한다.
  - -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
- 슈퍼데몬에 의한 동작 방식
  - 평소에는 슈퍼데몬만 동작하다가 서비스 요청이 오면 슈퍼데몬이 해당 데몬을 동작 시킴
  - 독자형보다는 서비스에 응답하는 데 시간이 약간 더 걸릴 수 있지만 자원을 효율적으로 사용한다는 장점

슈퍼데몬

- 유닉스에서 슈퍼데몬의 이름은 inetd
- 우분투에서는 보안 기능이 포함된 xinetd를 사용

04 데몬 프로세스

- **데몬의 조상 : systemd와 커널 스레드 데몬**
- **systemd 데몬**
  - 대부분의 프로세스의 조상 프로세스
  - pstree 명령으로 확인

```
user1@myubuntu:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─dhclient
        │                └─2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─atd
        ├─avahi-daemon───avahi-daemon
        ├─bluetoothd
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd───3*[dbus]
        ├─dbus-daemon
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b─┬─
gnome-shel
(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

```
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 21:56 ?        00:00:01 /sbin/init splash
root           2       0  0 21:56 ?        00:00:00 [kthreadd]
root           3       2  0 21:56 ?        00:00:00 [kworker/0:0]
root           4       2  0 21:56 ?        00:00:00 [kworker/0:0H]
root           6       2  0 21:56 ?        00:00:00 [mm_percpu_wq]
root           7       2  0 21:56 ?        00:00:00 [ksoftirqd/0]
root           8       2  0 21:56 ?        00:00:00 [rcu_sched]
root           9       2  0 21:56 ?        00:00:00 [rcu_bh]
root          10       2  0 21:56 ?        00:00:00 [migration/0]
root          11       2  0 21:56 ?        00:00:00 [watchdog/0]
root          12       2  0 21:56 ?        00:00:00 [cpuhp/0]
(생략)
```

04 데몬 프로세스

■ 주요 데몬

표 8-4 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.
dhcpd	동적으로 IP 주소를 부여하는 서비스를 제공한다.	smb	삼바 서비스를 제공한다.
httpd	웹 서비스를 제공한다.	syslogd	로그 기록 서비스를 제공한다.
lpd	프린트 서비스를 제공한다.	sshd	원격 보안 접속 서비스를 제공한다.
nfs	네트워크 파일 시스템 서비스를 제공한다.	in.telnetd	원격 접속 서비스를 제공한다.
named	DNS 서비스를 제공한다.	ftpd	파일 송수신 서비스를 제공한다.
sendmail	이메일 서비스를 제공한다.	ntpd	시간 동기화 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 윈도에서도 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 사용

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 $prefix/grubenv ]; then
  set have_grubenv=true
  load_env
fi
if [ "${next_entry}" ] ; then
   set default="${next_entry}"
   set next_entry=
   save_env next_entry
   set boot_once=true
else
   set default="0"
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

■ 암호 복구하기

① 시스템 재시작하기: 부팅할 때 GRUB 메뉴 초기 화면이 출력

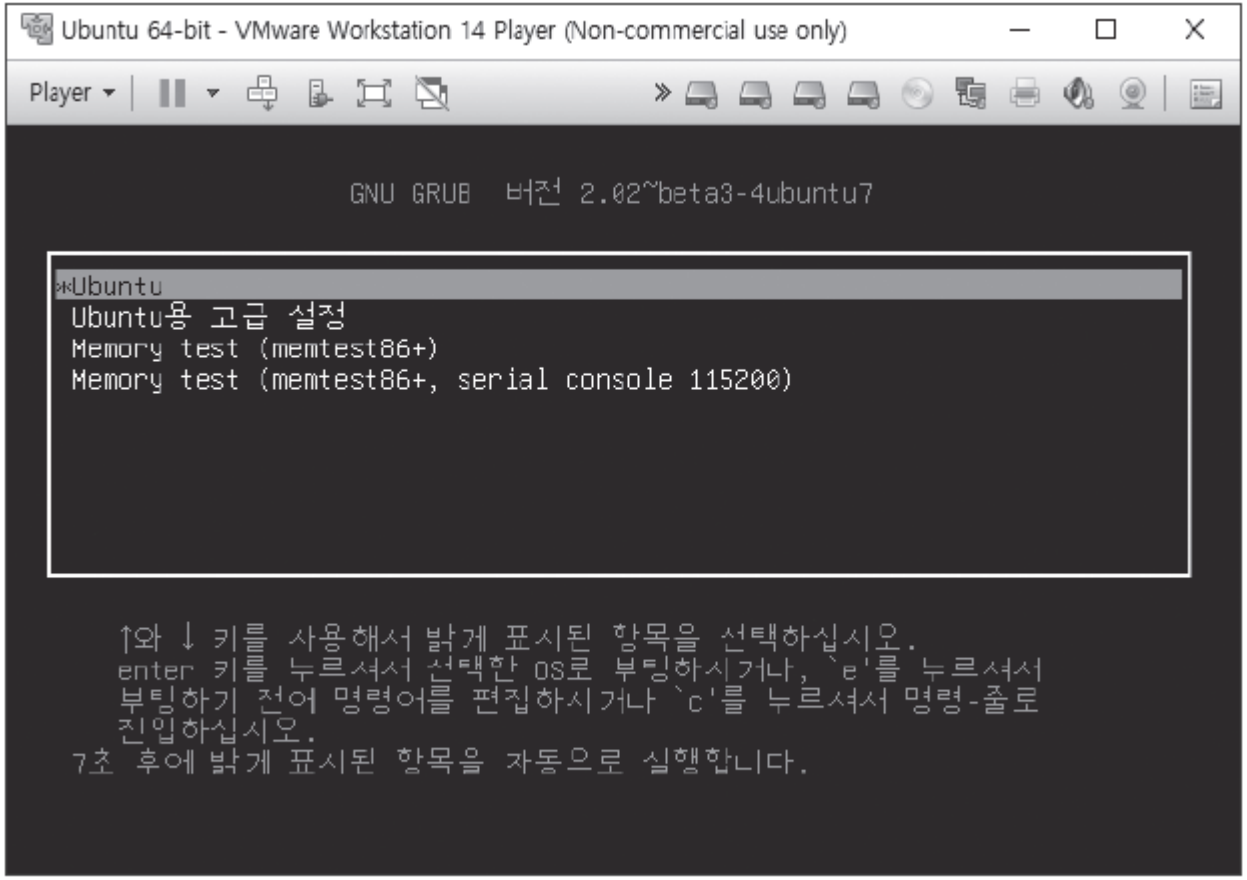


그림 8-10 GRUB 메뉴 초기 화면

05 부트 로더

■ 암호 복구하기

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

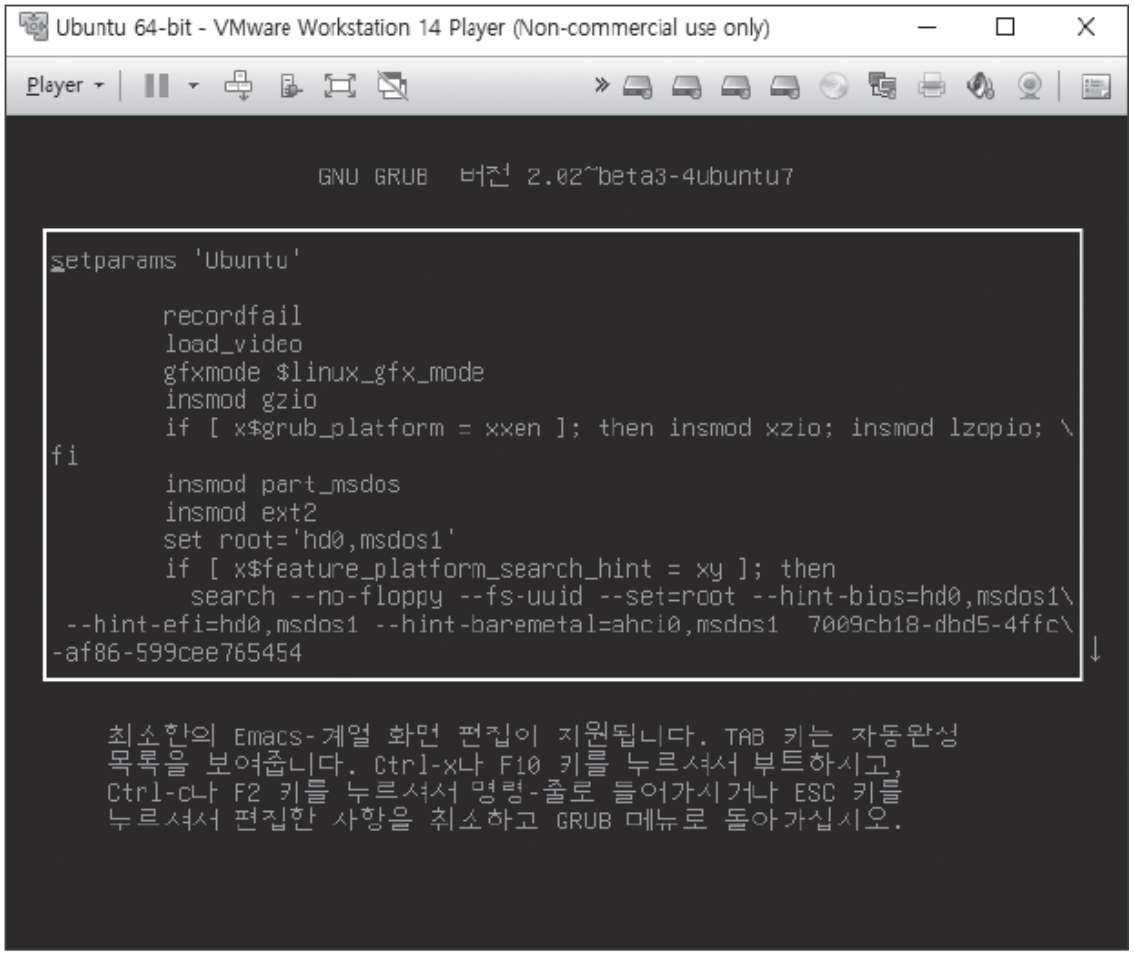


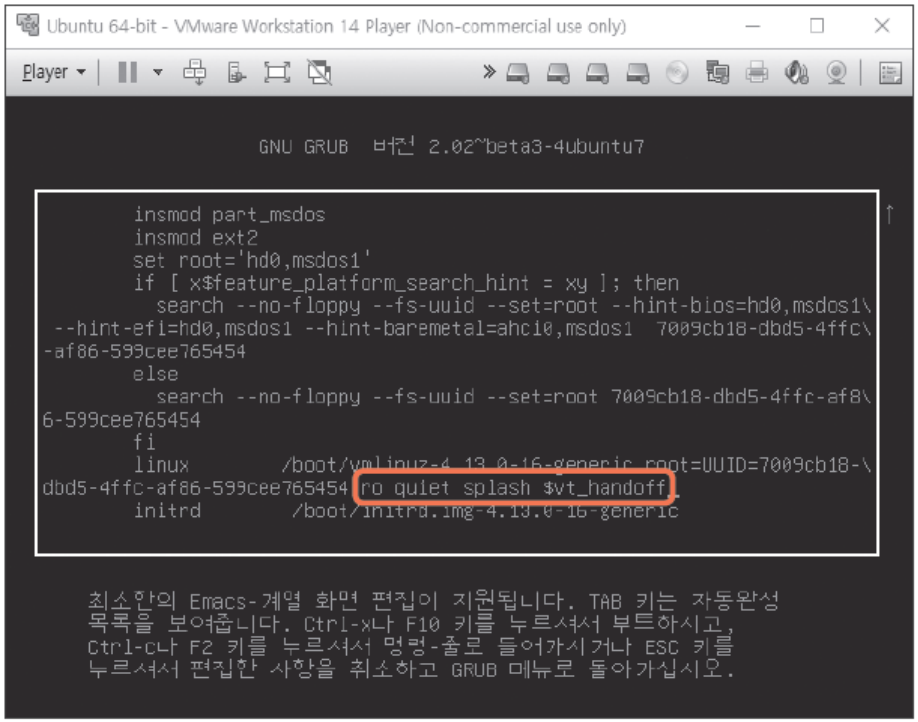
그림 8-11 GRUB 편집 화면

05 부트 로더

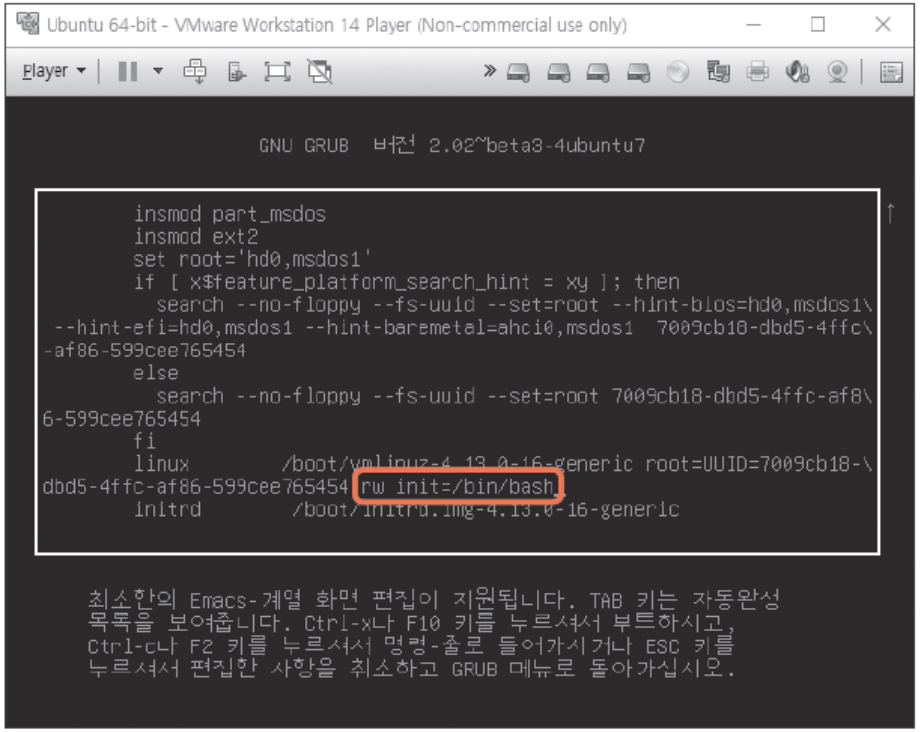
■ 암호 복구하기

③ 단일 사용자 모드로 수정하기

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



(a) 수정하기 전 화면



(b) 수정한 화면

그림 8-12 단일 사용자 모드로 부팅하기 위해 리눅스 커널 항목 수정

05 부트 로더

■ 암호 복구하기

④ F10키를 눌러 재시작하면 root 계정으로 동작

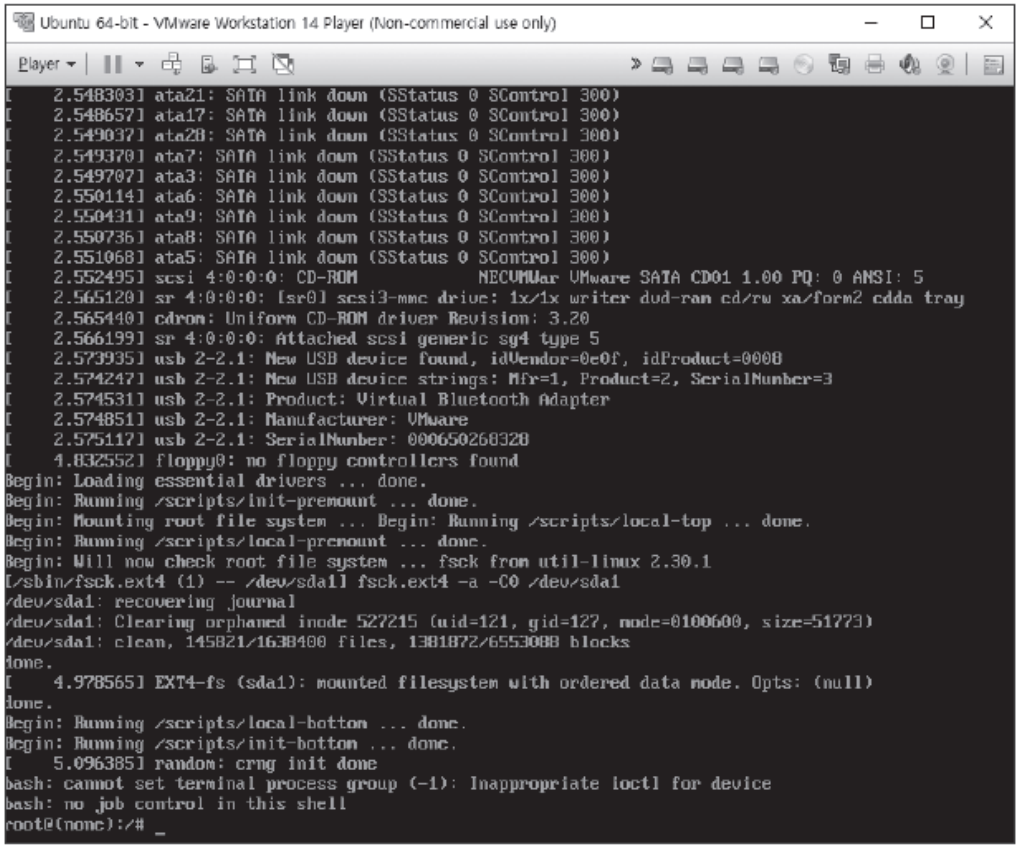


그림 8-13 root 계정 화면

⑤ 재부팅하기: reboot -f 명령으로 시스템을 재시작했을 때 GRUB 화면 출력. 이 상태에서 Enter 키를 눌러 우분투 부팅

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 복구 모드 선택하기: 시스템 재시작-GRUB 메뉴 초기 화면-'Ubuntu용 고급 설정'-recovery mode

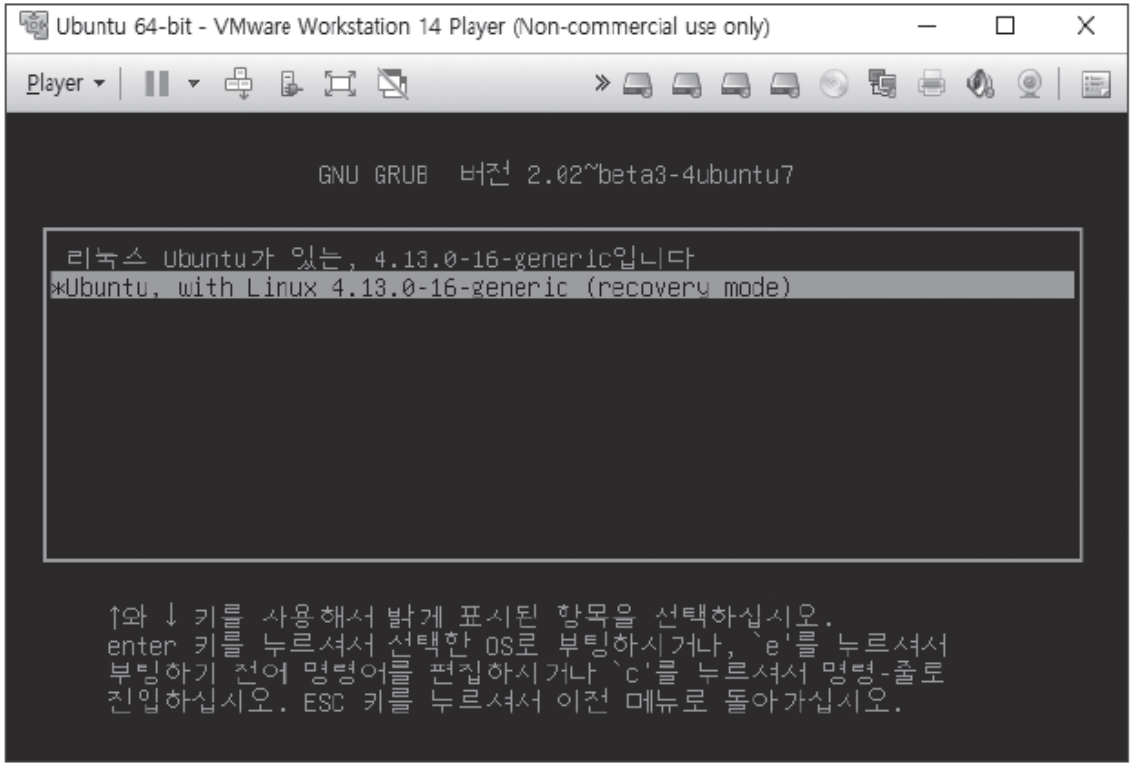


그림 8-14 recovery mode 선택 화면

05 부트 로더

■ 복구 모드로 부팅하기

② root 항목 선택하기

- root 항목을 선택

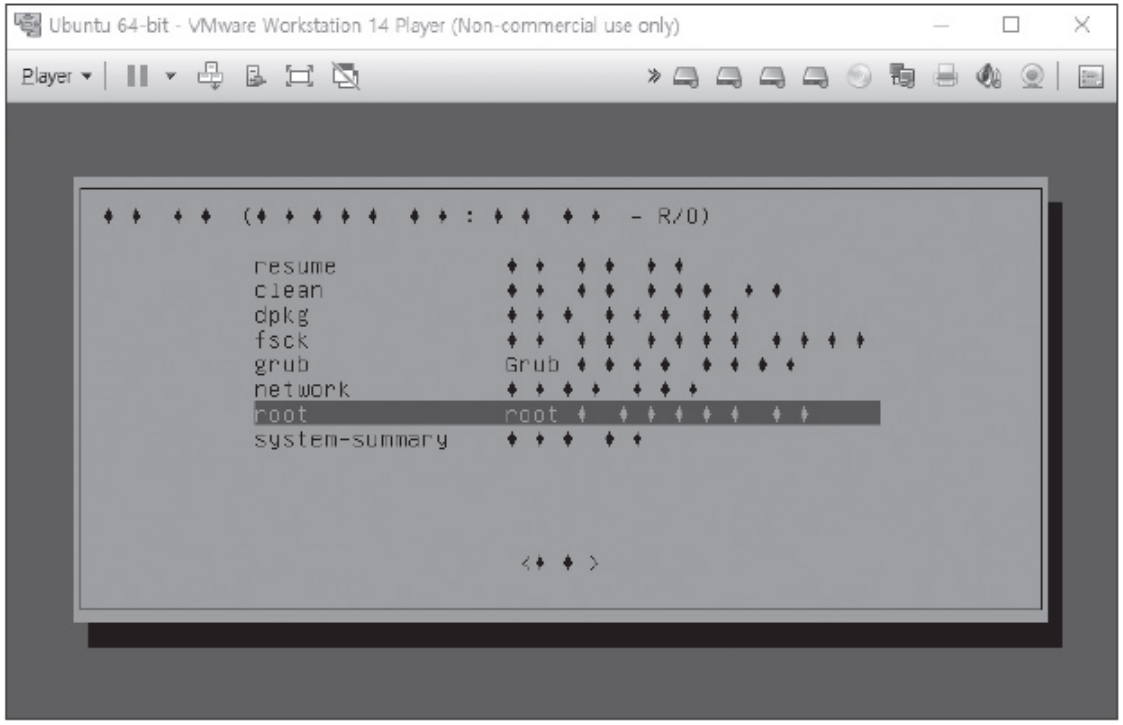


그림 8-15 복구 메뉴 선택 화면

05 부트 로더

■ 복구 모드로 부팅하기

③ root로 로그인하기

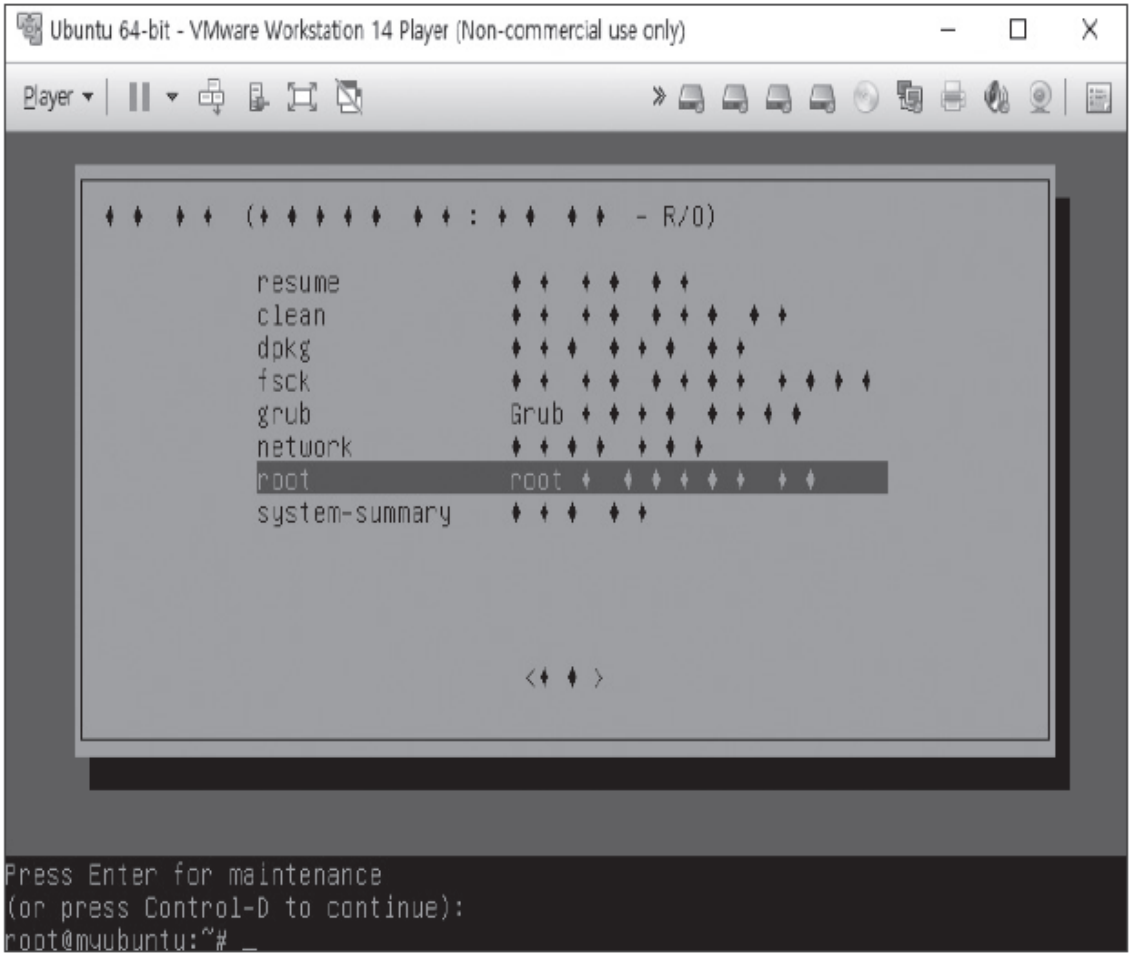


그림 8-16 root 프롬프트 출력 화면

05 부트 로더

복구 모드로 부팅하기

④ 다시 마운트하기

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

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

⑤ 재시작하기

- 작업이 완료되면 reboot -f 명령으로 리눅스를 재시작