

## Chapter 08 리눅스의 부팅과 종료

# 목차

00 개요

01 리눅스 시스템의 부팅

02 system 서비스

03 리눅스 시스템의 종료

04 데몬 프로세스

05 부트 로더

# 학습목표



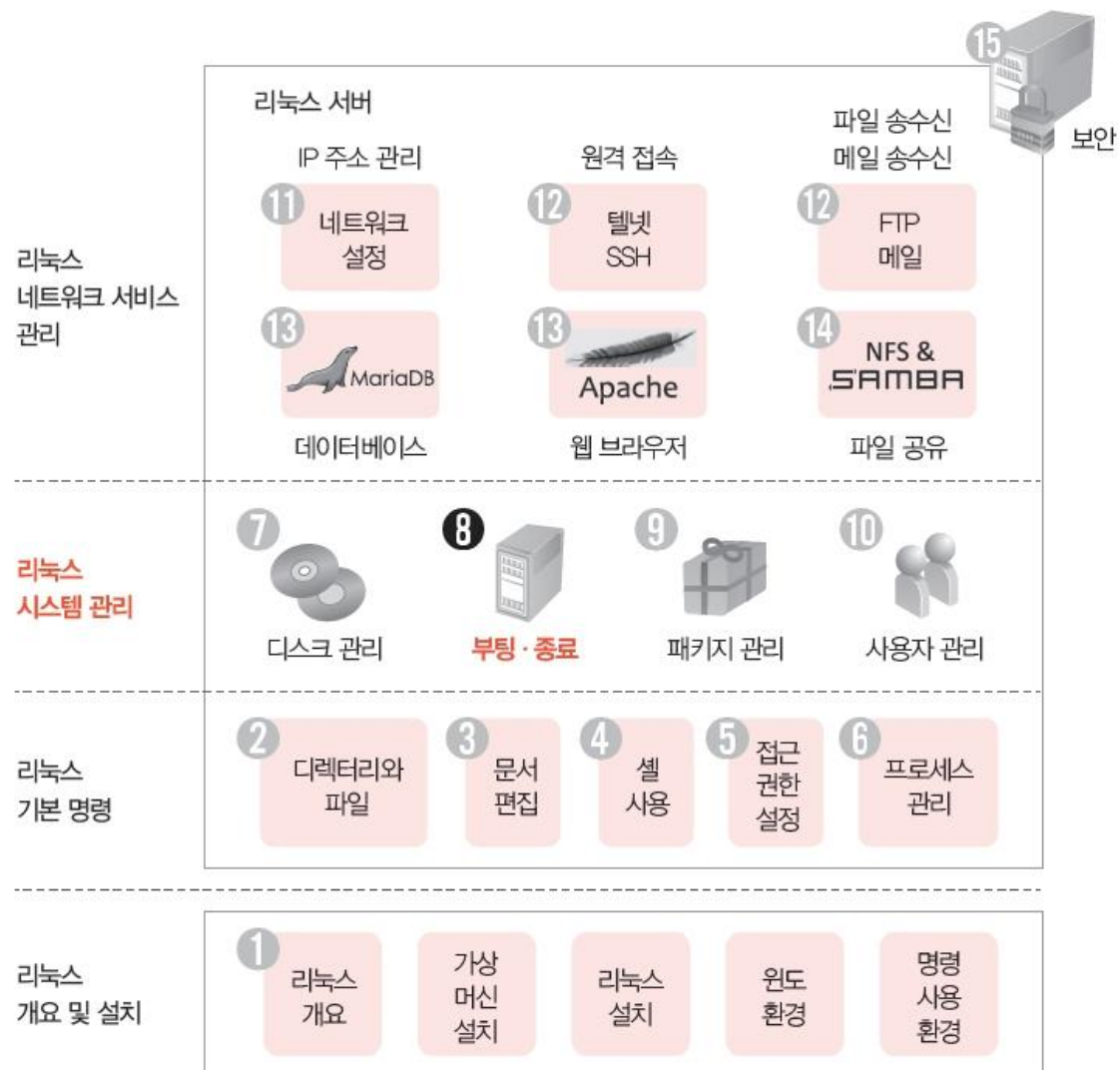
- 리눅스 시스템의 부팅 과정을 이해하고 부트 로더의 역할을 설명할 수 있다.
- systemd의 기능을 이해하고 사용법을 설명할 수 있다.
- 런레벨을 이해하고 변경할 수 있다.
- 리눅스 시스템 종료 방법을 설명할 수 있다.
- 데몬을 이해하고 슈퍼데몬의 역할을 설명할 수 있다.
- root 계정의 암호를 복구할 수 있다.

# 00 개요

# 00. 개요

## ■ 리눅스 학습 맵에서 8장의 위치

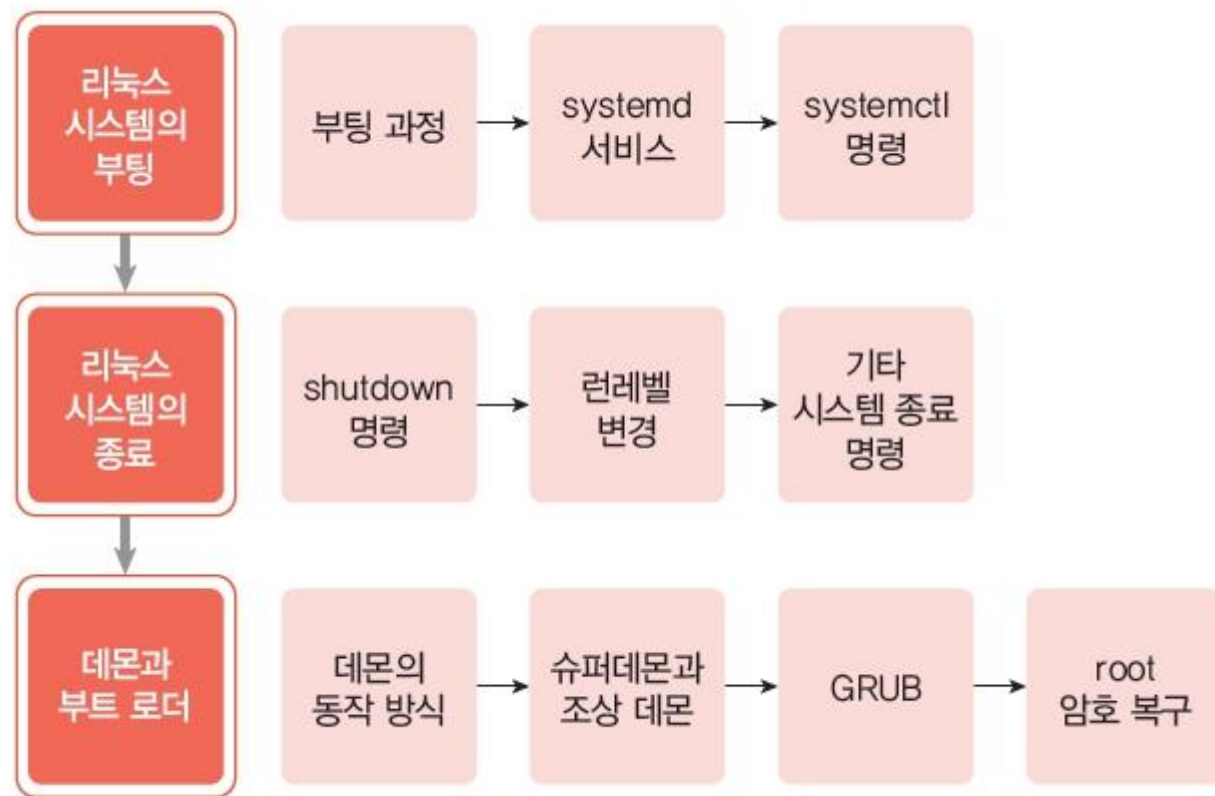
- 8장은 제3단계 중 두번째 항목으로 리눅스 시스템의 부팅과 종료를 다룬다.
- 리눅스의 부팅 과정과 시스템 종료 방법을 이해하고 다양한 서비스를 동작시키거나 종료시키는 방법을 익힌다.



# 00. 개요

## ■ 8장의 내용 구성

- 리눅스 시스템의 부팅 과정 이해하기
- systemd 서비스를 이해하고 systemctl 명령을 사용하는 방법 익히기
- 리눅스 시스템을 종료시키는 방법 익히기
- 데몬의 동작 방식을 이해하기
- 부트 로더의 역할을 이해하기
- root 암호를 복구하는 방법을 익히기



# 01 리눅스 시스템의 부팅

# 01. 리눅스 시스템의 부팅

## ■ 리눅스의 부팅 과정

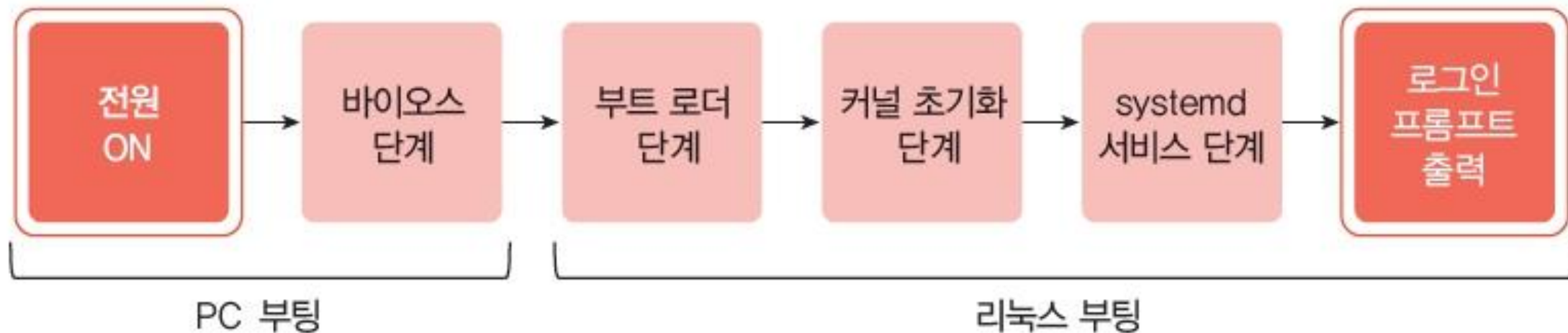


그림 8-1 리눅스의 부팅 과정



# 01. 리눅스 시스템의 부팅

## ■ 바이오스 단계

- 바이오스는 PC에 장착된 기본적인 하드웨어(키보드, 디스크 등)의 상태를 확인한 후 부팅 장치를 선택하여 부팅 디스크의 첫 섹터에서 512B를 로딩
- 512B를 마스터 부트 레코드(MBR)라고 하며 디스크의 어느 파티션에 2차 부팅 프로그램(부트 로더)이 있는지에 대한 정보가 저장되어 있음
- 메모리에 로딩된 MBR은 부트 로더를 찾아 메모리에 로딩하는 작업까지 수행

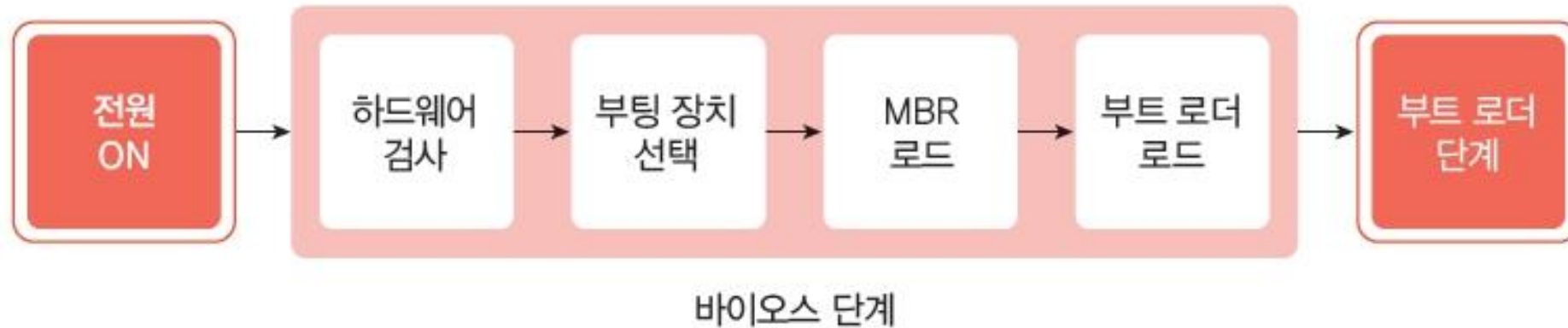


그림 8-2 바이오스 단계의 세부 동작

# 01. 리눅스 시스템의 부팅

## ■ 부트 로더 단계

- 부트 로더는 일반적으로 여러 운영체제 중에서 부팅할 운영체제를 선택할 수 있도록 메뉴를 제공
- 우분투에서는 부트 로더로 GRUB를 사용
- 부팅할 때 GRUB 메뉴를 출력하려면 /etc/default/grub 파일을 수정해야 함
  - GRUB\_TIMEOUT\_STYLE=hidden 앞에 #을 추가, GRUB\_TIMEOUT=0을 10으로 수정
  - /etc/default/grub 파일을 수정했다면 sudo update-grub를 실행하여 변경된 내용을 적용

```
user1@myubuntu:~$ sudo vi /etc/default/grub
```

```
GRUB_DEFAULT=0
```

```
#GRUB_TIMEOUT_STYLE=hidden
```

```
GRUB_TIMEOUT=10
```

→ 맨 앞에 #을 추가한다.

→ 0을 10으로 수정한다.

# 01. 리눅스 시스템의 부팅

- 우분투를 다시 시작하면 GRUB 메뉴를 출력
- 부트 로더는 리눅스 커널을 메모리에 로딩하는 역할을 수행
  - 리눅스 커널은 /boot 디렉터리 아래에 'vmlinuz-버전명'의 형태로 제공

```
user1@myubuntu:~$ ls /boot/vm*  
/boot/vmlinuz                /boot/vmlinuz-5.13.0-22-generic  
/boot/vmlinuz-5.13.0-21-generic /boot/vmlinuz.old
```

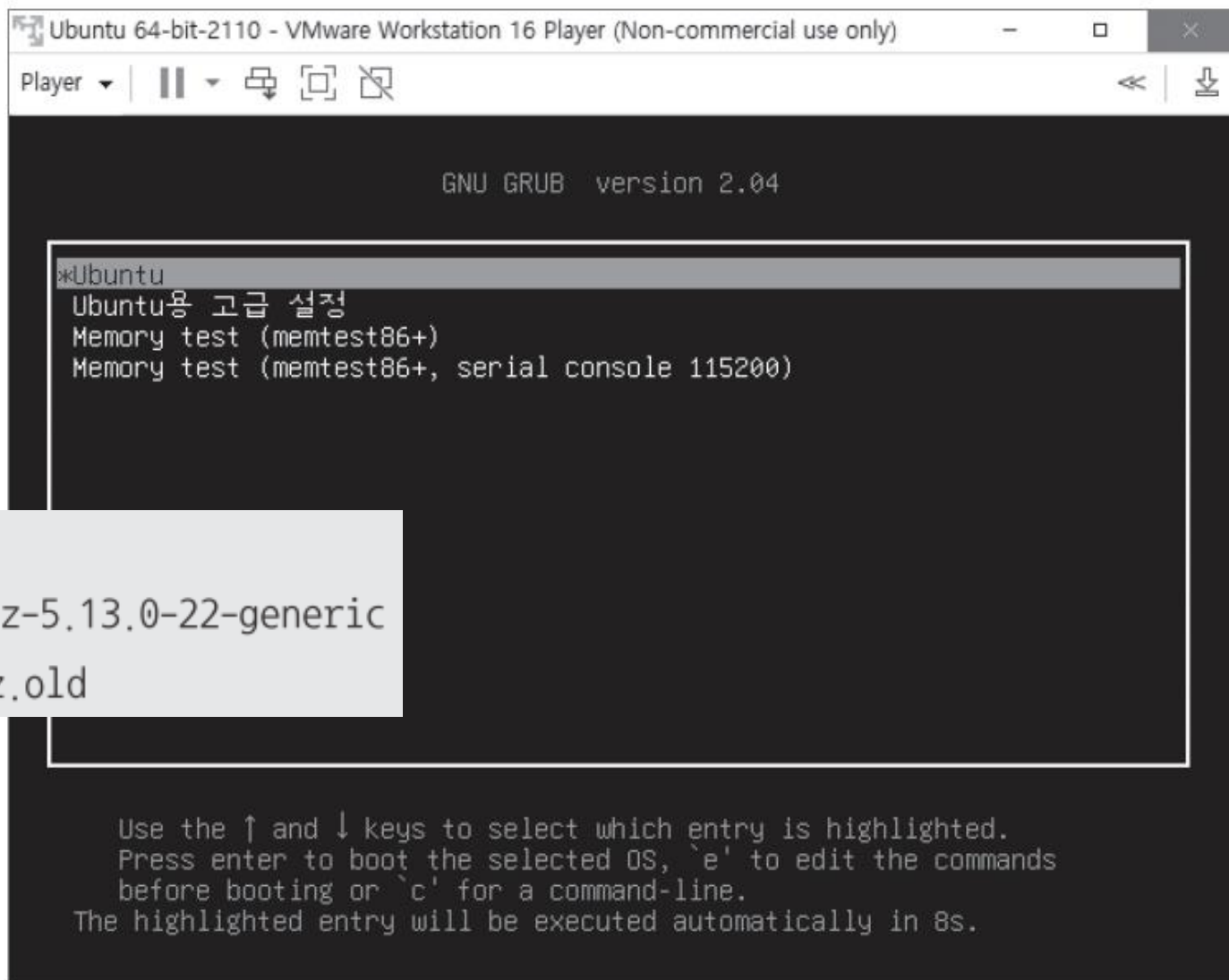


그림 8-3 부트 로더 시작 화면

# 01. 리눅스 시스템의 부팅

## ■ 커널 초기화 단계

- 부트 로더에 의해 메모리에 로딩된 커널은 가장 먼저 시스템에 연결된 메모리, 디스크, 키보드, 마우스 등의 장치를 검사
- 장치 검사 등 기본적인 초기화 과정이 끝나면 커널은 프로세스와 스레드를 생성

| UID  | PID | PPID | C | STIME | TTY | TIME     | CMD                           |
|------|-----|------|---|-------|-----|----------|-------------------------------|
| root | 1   | 0    | 0 | 09:46 | ?   | 00:00:01 | /sbin/init splash             |
| root | 2   | 0    | 0 | 09:46 | ?   | 00:00:00 | [kthreadd]                    |
| root | 3   | 2    | 0 | 09:46 | ?   | 00:00:00 | [rcu_gp]                      |
| root | 4   | 2    | 0 | 09:46 | ?   | 00:00:00 | [rcu_par_gp]                  |
| root | 6   | 2    | 0 | 09:46 | ?   | 00:00:00 | [kworker/0:0H-events_highpri] |
| root | 9   | 2    | 0 | 09:46 | ?   | 00:00:00 | [mm_percpu_wq]                |
| root | 10  | 2    | 0 | 09:46 | ?   | 00:00:00 | [rcu_tasks_rude_]             |
| root | 11  | 2    | 0 | 09:46 | ?   | 00:00:00 | [rcu_tasks_trace]             |
| root | 12  | 2    | 0 | 09:46 | ?   | 00:00:00 | [ksoftirqd/0]                 |

# 01. 리눅스 시스템의 부팅

## ■ systemd 서비스 단계: 리눅스가 본격적으로 동작

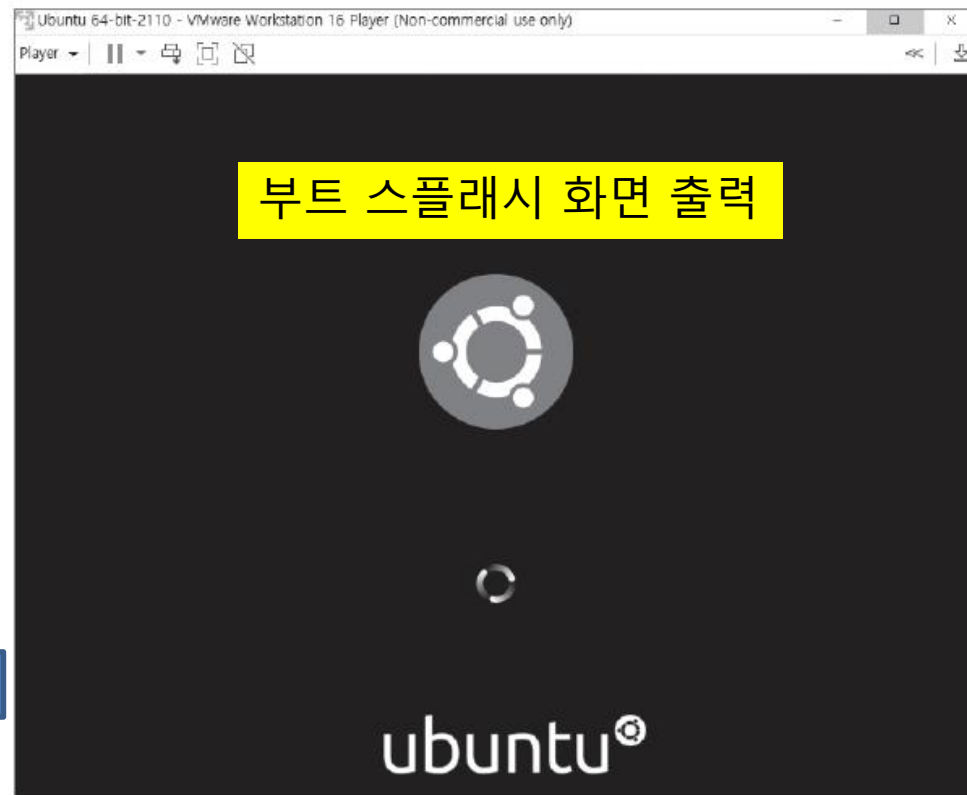


그림 8-5 부트 스플래시 화면

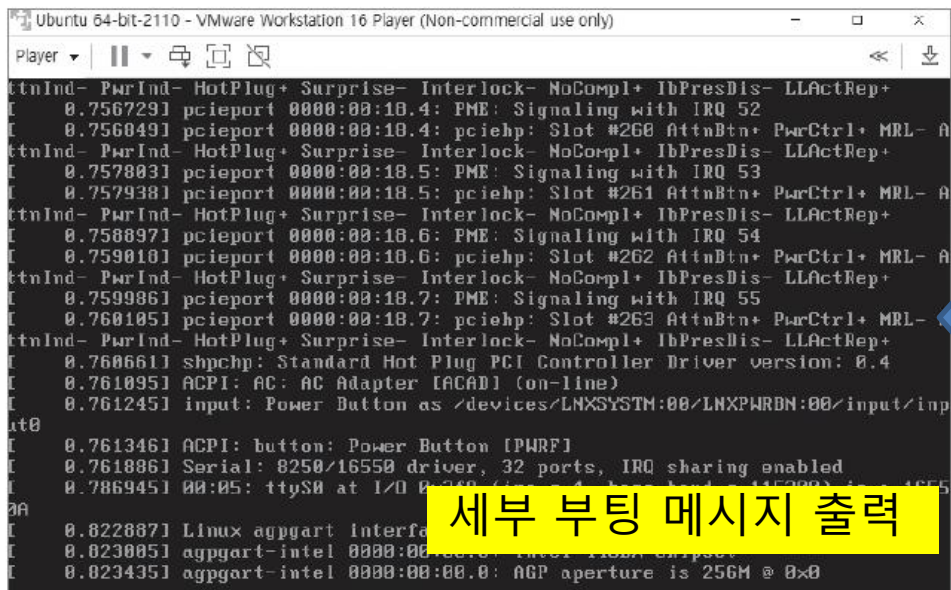
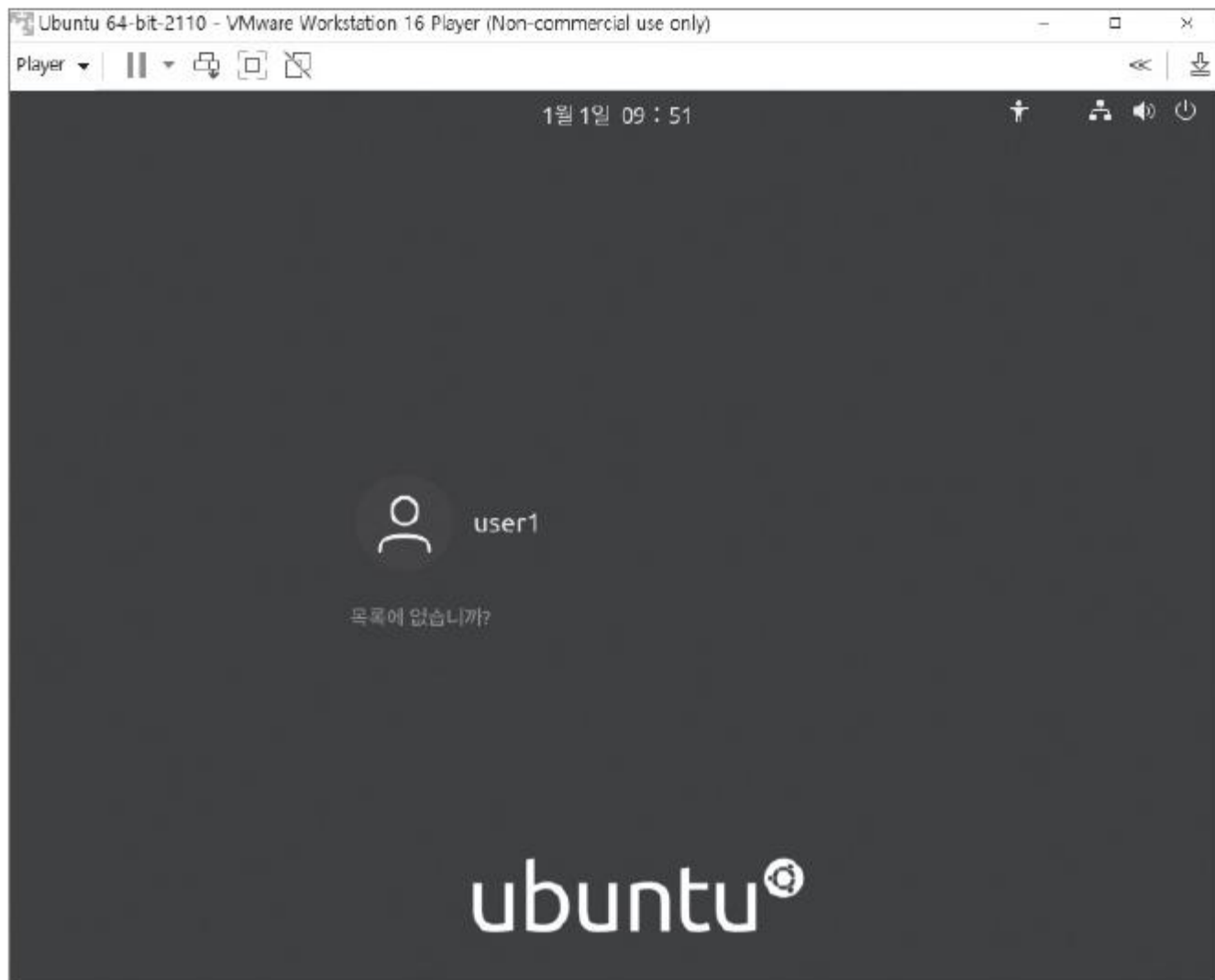


그림 8-6 세부 부팅 메시지 출력 화면

# 01. 리눅스 시스템의 부팅



부팅 완료

그림 8-7 로그인 화면

# 01. 리눅스 시스템의 부팅

## ■ 부팅 메시지 확인: dmesg 명령, more /var/log/boot.log

```
user1@myubuntu:~$ sudo dmesg | more
[    0.000000] Linux version 5.13.0-22-generic (buildd@lgw01-amd64-012) (gcc (Ubuntu 11.2.0-7ubuntu2) 11.2.0, GNU ld (GNU Binutils for Ubuntu) 2.37) #22-Ubuntu SMP Fri Nov 5 13:21:36 UTC 2021 (Ubuntu 5.13.0-22.22-generic 5.13.19)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-5.13.0-22-generic root=UUID=265c8913-ee1e-4034-885f-207969b0a23b ro splash
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Hygon HygonGenuine
[    0.000000]   Centaur CentaurHauls
[    0.000000]   zhaoxin   Shanghai
[    0.000000] Disabled fast string operations
(생략)
```

# 01. 리눅스 시스템의 부팅

## ■ 1번 프로세스

- 전통적으로 1번 프로세스는 init

```
user1@myubuntu:~$ ps -ef | more
UID          PID    PPID  C  STIME TTY          TIME CMD
root          1        0  0  12:26 ?        00:00:01 /sbin/init splash
root          2        0  0  12:26 ?        00:00:00 [kthreadd]
root          3        2  0  12:26 ?        00:00:00 [rcu_gp]
root          4        2  0  12:26 ?        00:00:00 [rcu_par_gp]
(생략)
```

- 1번 프로세스가 여전히 init인 것처럼 보이지만 사실은 systemd 파일의 심볼릭 링크

```
user1@myubuntu:~$ ls -l /sbin/init
lrwxrwxrwx 1 root root 20 11월 21 11:10 /sbin/init -> /lib/systemd/systemd
```



## 02 systemd 서비스

## 02. system 서비스

### ■ system 서비스의 역할

- 리눅스의 시스템과 서비스 관리자
- 유닉스의 init 프로세스가 하던 작업을 대신 수행
- 다양한 서비스 데몬을 시작하고, 프로세스들의 상태를 유지하며, 시스템의 상태를 관리

## 02. system 서비스

### ■ init 프로세스와 런레벨

- 현재 init 서비스는 systemd 서비스로 대체
- `man init`로 확인해보면 systemd에 대한 설명이 출력

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

NAME
    systemd, init - systemd system and service manager

SYNOPSIS
    /lib/systemd/systemd [OPTIONS...]
```

## 02. system 서비스

### ■ init 프로세스와 런레벨

- 런레벨: 시스템의 상태를 구분하는 숫자/문자

표 8-1 리눅스의 init 런레벨

| 런레벨     | 의미                  |
|---------|---------------------|
| 0       | 시스템 종료              |
| 1, S, s | 응급 복구 모드(단일 사용자 모드) |
| 2       | 다중 사용자 모드           |
| 3       |                     |
| 4       |                     |
| 5       | 그래피컬 다중 사용자 모드      |
| 6       | 재시작                 |

## 02. system 서비스

### ■ systemd의 장점

- 소켓 기반으로 동작하여 inetd와 호환성을 유지한다.
- 셸과 독립적으로 부팅이 가능하다.
- 마운트 제어가 가능하다.
- fsck 제어가 가능하다.
- 시스템 상태에 대한 스냅샷을 유지한다.
- 서비스에 시그널을 전달할 수 있다.
- 쉼다운 전에 사용자 세션의 안전한 종료가 가능하다.

## 02. system 서비스

### ■ systemd 유닛

- systemd는 전체 시스템을 시작하고 관리하는데 유닛이라 부르는 구성 요소를 사용
- systemd는 관리 대상의 이름을 '서비스명.유닛종류'의 형태로 관리

표 8-2 systemd 유닛의 종류

| 유닛        | 기능  | 예                                 |
|-----------|---|-----------------------------------|
| service   | 시스템 서비스 유닛으로 데몬을 시작 · 종료 · 재시작 · 로드한다.                              | atd.service                       |
| target    | 유닛을 그룹핑한다.<br>(예 multi-user.target → 런레벨 5에 해당하는 유닛)                | basic.target                      |
| automount | 디렉터리 계층 구조에서 자동 마운트 포인트를 관리한다.                                      | proc-sys-fs-binfmt_misc.automount |
| device    | 리눅스 장치 트리에 있는 장치를 관리한다.   | sys-module-fuse.device            |
| mount     | 디렉터리 계층 구조의 마운트 포인트를 관리한다.  | boot.mount                        |
| path      | 파일 시스템의 파일이나 디렉터리 등 경로를 관리한다.                                       | cups.path                         |
| scope     | 외부에서 생성된 프로세스를 관리한다.  | init.scope                        |
| slice     | 시스템의 프로세스를 계층적으로 관리한다.  | system-getty.slice                |
| socket    | 소켓을 관리하는 유닛으로 AF_INET, AF_INET6, AF_UNIX 소켓 스트림과 데이터그램, FIFO를 지원한다. | dbus.socket                       |
| swap      | 스왑 장치를 관리한다.  | dev-mapper-fedoraWx2dswap.swap    |
| timer     | 타이머와 관련된 기능을 관리한다.  | dnf-makecache.timer               |

## 02. system 서비스

### ■ systemd 관련 명령

#### ■ systemctl

#### systemctl

- **기능** systemd 서비스를 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 시작되도록 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. system 서비스

- 동작 중인 유닛 출력하기:  
systemctl

```
user1@myubuntu:~$ systemctl
```

| UNIT                  | LOAD   | ACTIVE | SUB    | DESCRIPTION              |
|-----------------------|--------|--------|--------|--------------------------|
| (생략)                  |        |        |        |                          |
| basic.target          | loaded | active | active | Basic System             |
| bluetooth.target      | loaded | active | active | Bluetooth                |
| cryptsetup.target     | loaded | active | active | Local Encrypted Volumes  |
| getty.target          | loaded | active | active | Login Prompts            |
| graphical.target      | loaded | active | active | Graphical Interface      |
| local-fs-pre.target   | loaded | active | active | Local File Systems (Pre) |
| local-fs.target       | loaded | active | active | Local File Systems       |
| multi-user.target     | loaded | active | active | Multi-User System        |
| network-online.target | loaded | active | active | Network is Online        |
| network.target        | loaded | active | active | Network                  |

(생략)

LOAD = Reflects whether the unit definition was properly loaded.

ACTIVE = The high-level unit activation state, i.e. generalization of SUB.

SUB = The low-level unit activation state, values depend on unit type.

229 loaded units listed. Pass --all to see loaded but inactive units, too.

To show all installed unit files use 'systemctl list-unit-files'.



## 02. system 서비스

- 전체 유닛 출력하기: systemctl -a

```
user1@myubuntu:~$ systemctl -a
```

| UNIT                          | LOAD      | ACTIVE   | SUB     | DESCRIPTION                     |
|-------------------------------|-----------|----------|---------|---------------------------------|
| (생략)                          |           |          |         |                                 |
| dev-hugepages.mount           | loaded    | active   | mounted | Huge Pages File System          |
| dev-mqueue.mount              | loaded    | active   | mounted | POSIX Message Queue File        |
| System                        |           |          |         |                                 |
| ● home.mount                  | not-found | inactive | dead    | home.mount                      |
| proc-sys-fs-binfmt_misc.mount | loaded    | inactive | dead    | Arbitrary Executable            |
| File Formats File System      |           |          |         |                                 |
| run-vmblock\x2dfuse.mount     |           | loaded   | active  | mounted VMware vmblock fuse     |
| mount                         |           |          |         |                                 |
| snap-bare-5.mount             | loaded    | active   | mounted | Mount unit for bare, revision 5 |
| snap-core-11743.mount         | loaded    | active   | mounted | Mount unit for core, revision   |
| 11743                         |           |          |         |                                 |
| (생략)                          |           |          |         |                                 |

## 02. system 서비스

- 특정 유닛 출력하기: systemctl -t

```
user1@myubuntu:~$ systemctl -t service
```

| UNIT                    | LOAD   | ACTIVE | SUB     | DESCRIPTION                   |
|-------------------------|--------|--------|---------|-------------------------------|
| accounts-daemon.service | loaded | active | running | Accounts Service              |
| acpid.service           | loaded | active | running | ACPI event daemon             |
| alsa-restore.service    | loaded | active | exited  | Save/Restore Sound Card State |
| apparmor.service        | loaded | active | exited  | Load AppArmor profiles        |
| apport.service          | loaded | active | exited  | LSB: automatic crash report   |
| generation              |        |        |         |                               |
| atd.service             | loaded | active | running | Deferred execution scheduler  |
| avahi-daemon.service    | loaded | active | running | Avahi mDNS/DNS-SD Stack       |

(생략)

## 02. system 서비스

- 유닛 서비스 시작하기: start

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

cron 서비스 시작

## 02. system 서비스

### ■ 유닛의 상태 확인하기: status

```
user1@myubuntu:~$ systemctl status cron.service
```

```
● cron.service - Regular background program processing daemon
```

```
Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
```

```
Active: active (running) since Sat 2022-01-01 10:01:26 KST; 3h 21min ago
```

```
Docs: man:cron(8)
```

```
Main PID: 833 (cron)
```

```
Tasks: 1 (limit: 4608)
```

```
Memory: 452.0K
```

```
CPU: 73ms
```

```
CGroup: /system.slice/cron.service
```

```
└─833 /usr/sbin/cron -f -P
```

```
1월 01 11:30:01 myubuntu CRON[2622]: pam_unix(cron:session): session opened for user root>
```

```
1월 01 11:30:01 myubuntu CRON[2622]: pam_unix(cron:session): session closed for user root
```

```
1월 01 12:17:01 myubuntu CRON[2699]: pam_unix(cron:session): session opened for user root>
```

```
1월 01 12:17:01 myubuntu CRON[2700]: (root) CMD ( cd / && run-parts --report /etc/cron>
```

```
1월 01 12:17:01 myubuntu CRON[2699]: pam_unix(cron:session): session closed for user root
```

cron 서비스  
상태 확인하기

## 02. system 서비스

- 유닛 서비스 정지하기: stop

cron 서비스 정지

```
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: enabled)
```

```
Active: inactive (dead) since Sat 2022-01-01 13:28:29 KST; 6s ago
```

```
Docs: man:cron(8)
```

```
Process: 833 ExecStart=/usr/sbin/cron -f -P $EXTRA_OPTS (code=killed, signal=TERM)
```

```
Main PID: 833 (code=killed, signal=TERM)
```

```
CPU: 73ms
```

```
1월 01 12:17:01 myubuntu CRON[2700]: (root) CMD ( cd / && run-parts --report /etc/cron>
```

```
1월 01 12:17:01 myubuntu CRON[2699]: pam_unix(cron:session): session closed for user root
```

## 02. system 서비스

### ■ systemd와 런레벨

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     |

## 02. system 서비스

- 현재 target과 런레벨 확인하기

```
user1@myubuntu:~$ systemctl get-default  
graphical.target
```

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

런레벨 5로 부팅

- 기본 target 지정하기

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

target을 multi-user로 바꾸기

```
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 1월 1 13:31 /etc/systemd/system/default.target -> /lib/systemd/system/multi-user.target
```

## 02. system 서비스

- 기본 target을 다시 graphical.target으로 변경

```
user1@myubuntu:~$ sudo systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target → /lib/systemd/system/graphical.target.
```

- target 변경하기

```
systemctl isolate multi-user
```

```
systemctl isolate graphical
```

```
systemctl isolate runlevel3
```

```
systemctl isolate runlevel5
```

multi-user.target(런레벨 3)으로  
변경

graphical.target(런레벨 5)으로  
변경



## 02. system 서비스

- 런레벨 변경하기: telinit, init

```
user1@myubuntu:~$ init --help
```

```
init [OPTIONS...] COMMAND
```

```
Send control commands to the init daemon.
```

```
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            |

```
Options:
```

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

```
See the telinit(8) man page for details.
```

```
user1@myubuntu:~$ ls -l /sbin/telinit
```

```
lrwxrwxrwx 1 root root 14 11월 21 11:10 /sbin/telinit -> /bin/systemctl
```

## 02. system 서비스

- 단일 사용자 모드로 전환하기: rescue.target(런레벨 1)
  - 시스템에 문제가 있을 경우 시스템을 rescue.target 유닛(런레벨 1, 런레벨 S)으로 변경하여 점검

```
systemctl isolate rescue
```

```
systemctl isolate runlevel1
```

```
init 1
```

```
telinit S
```

- 단일 사용자 모드에서 다중 사용자 모드로 전환하려면 reboot 명령이나 systemctl default 명령 사용

## 03 리눅스 시스템의 종료

## 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.

## 03. 리눅스 시스템의 종료

### ■ shutdown 명령

- 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: 종료한 후 재시작한다.
  - h: 종료하고 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

## 03. 리눅스 시스템의 종료

- 시스템 즉시 종료하기

```
sudo shutdown -h now
```

- 셧다운한다는 메시지 보내고 종료하기

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

- 시스템 재시작하기

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

- 명령 취소하기

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

- 메시지만 보내기

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

## 03. 리눅스 시스템의 종료

### ■ 런레벨 변경하기

- 런레벨을 0으로 바꾸면 시스템 종료

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

- 시스템 재시작은 런레벨 6

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

- systemd로 종료하기

```
user1@myubuntu:~$ sudo systemctl isolate poweroff.target
```

```
user1@myubuntu:~$ sudo systemctl isolate runlevel0.target
```

- systemd로 재시작

```
user1@myubuntu:~$ sudo systemctl isolate reboot.target
```

```
user1@myubuntu:~$ sudo systemctl isolate runlevel6.target
```

## 03. 리눅스 시스템의 종료

### ■ 기타 시스템 종료 명령: halt, poweroff, reboot

- 모두 systemctl 명령의 심볼릭 링크

```
user1@myubuntu:~$ ls -l /sbin/halt
```

```
lrwxrwxrwx 1 root root 14 11월 21 11:10 /sbin/halt -> /bin/systemctl
```

```
user1@myubuntu:~$ ls -l /sbin/poweroff
```

```
lrwxrwxrwx 1 root root 14 11월 21 11:10 /sbin/poweroff -> /bin/systemctl
```

```
user1@myubuntu:~$ ls -l /sbin/reboot
```

```
lrwxrwxrwx 1 root root 14 11월 21 11:10 /sbin/reboot -> /bin/systemctl
```



## 04 데몬 프로세스

## 04. 데몬 프로세스

### ■ 데몬

- 리눅스의 백그라운드에서 동작하며 특정한 서비스를 제공하는 프로세스

### ■ 데몬의 동작방식

- 독자형 데몬
- 슈퍼데몬을 통한 동작

### ■ 슈퍼 데몬

- 데몬들을 관리하는 데몬
- 사용자가 네트워크 서비스를 요청하면 슈퍼데몬이 이를 받아서 해당하는 서비스 데몬을 동작시킴
- 유닉스의 슈퍼데몬은 inetd였으나 우분투에서는 보안 기능이 포함된 xinetd를 사용

## 04. 데몬 프로세스

### ■ 데몬의 조상

#### ■ systemd 데몬

- 대부분의 조상 프로세스
- 시스템의 상태를 종합적으로 관리하는 역할을 수행

#### ■ 커널 스레드 데몬

- 커널 기능의 일부분을 프로세스처럼 관리하는 데몬
- ps 명령으로 확인했을 때 대괄호([ ])에 들어 있는 프로세스들임
- 커널 데몬은 대부분 입출력이나 메모리 관리, 디스크 동기화 등을 수행

```
user1@myubuntu:~$ pstree
systemd───ModemManager───2*[{ModemManager}]
          │NetworkManager──2*[{NetworkManager}]
          │VGAuthService
          │accounts-daemon──2*[{accounts-daemon}]
          │acpid
          │at-spi-bus-laun─┬─dbus-daemon
          │                └─3*[{at-spi-bus-laun}]
          │at-spi2-registr──2*[{at-spi2-registr}]
          │atd
          │avahi-daemon───avahi-daemon
          │bluetoothd
          │colord───2*[{colord}]
          │cron
          │cups-browsed───2*[{cups-browsed}]
          │cupsd
          └─dbus-daemon
```

(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). | popd       | 기본 편지함 서비스를 제공한다.        |
| cron     | 주기적으로 실행하도록 예약한 명령을 실행한다.              | routed     | 자동 IP 라우터 테이블 서비스를 제공한다. |
| dhcpcd   | 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.        |
| smtpd    | 메일 전송 데몬이다.                            |            |                          |

## 05 부트 로더

# 05. 부트 로더

## ■ 부트 로더

- 커널을 메모리에 로딩하는 역할을 수행
- 우분투에서는 GRUB를 기본으로 지원

## ■ GRUB의 개요

- 이전 부트 로더인 LILO는 리눅스에서만 사용할 수 있지만 GRUB는 윈도우에서도 사용 가능
- LILO에 비해 설정과 사용이 편리
- 부팅할 때 명령을 사용하여 수정 가능
- 멀티 부팅 기능을 지원
- GRUB의 최신 버전은 GRUB2
  - GRUB에 비해 GRUB2는 이식성과 모듈화가 더 좋아져서 동적으로 모듈을 로딩
  - 우분투는 GRUB2를 기본 부트 로더로 사용

# 05. 부트 로더

## ■ GRUB2 관련 디렉터리와 파일

- /boot/grub/grub.cfg 파일
  - GRUB2의 기본 설정 파일로 사용자가 직접 수정 불가
  - /etc/default/grub 파일과 /etc/grub.d 디렉터리 아래에 있는 스크립트를 읽어서 자동 생성

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

## 05. 부트 로더

- /etc/grub.d 디렉터리
  - GRUB 스크립트를 가지고 있는 디렉터리
  - 이 스크립트들은 GRUB의 명령이 실행될 때 순서대로 실행되어 grub.cfg 파일을 생성

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



## 05. 부트 로더

- /etc/default/grub 파일
  - GRUB 메뉴 설정 내용이 저장되어 있음
  - GRUB 스크립트가 이 파일을 읽어서 grub.cfg에 기록

```
user1@myubuntu:~$ cat /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_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="splash"
#GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
(생략)
```

## 05. 부트 로더

### ■ 암호 복구하기

#### ① 시스템 재시작하기

#### ② GRUB 편집 모드로 전환하기

- GRUB Boot Menu가 출력될 때 재빨리 E를 눌러 편집 모드로 전환

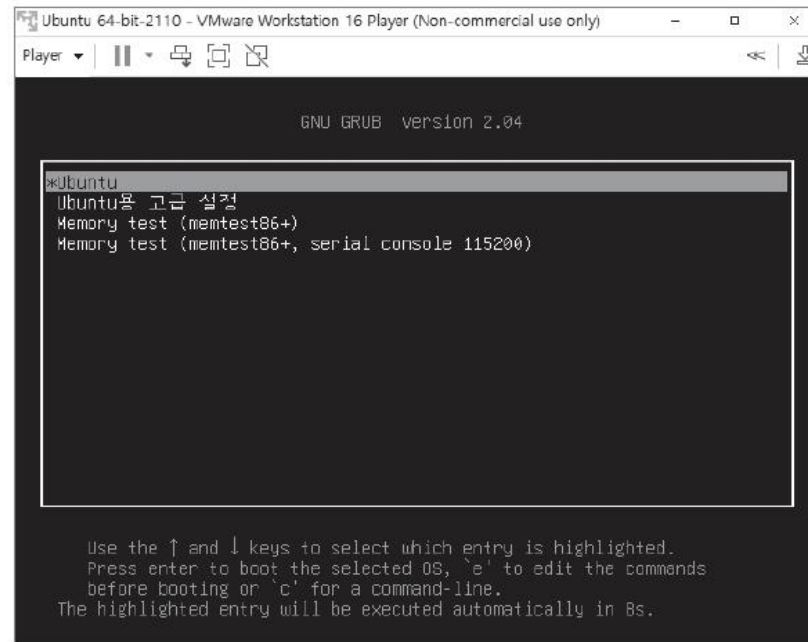


그림 8-8 GRUB 메뉴 초기 화면

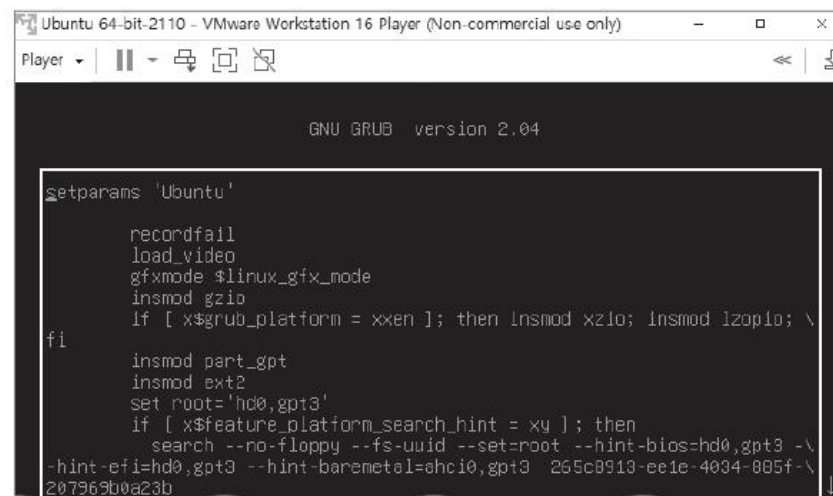
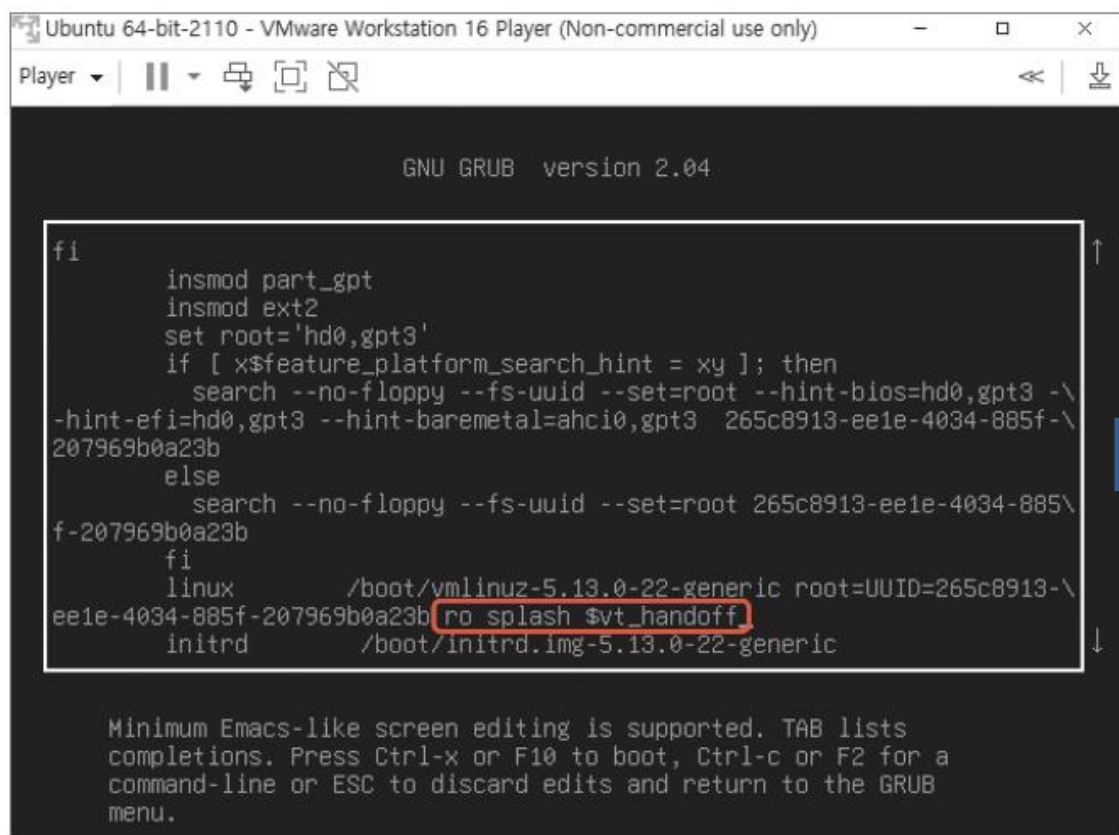


그림 8-9 GRUB 편집 화면

## 05. 부트 로더

### ③ 단일 사용자 모드로 수정하기

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

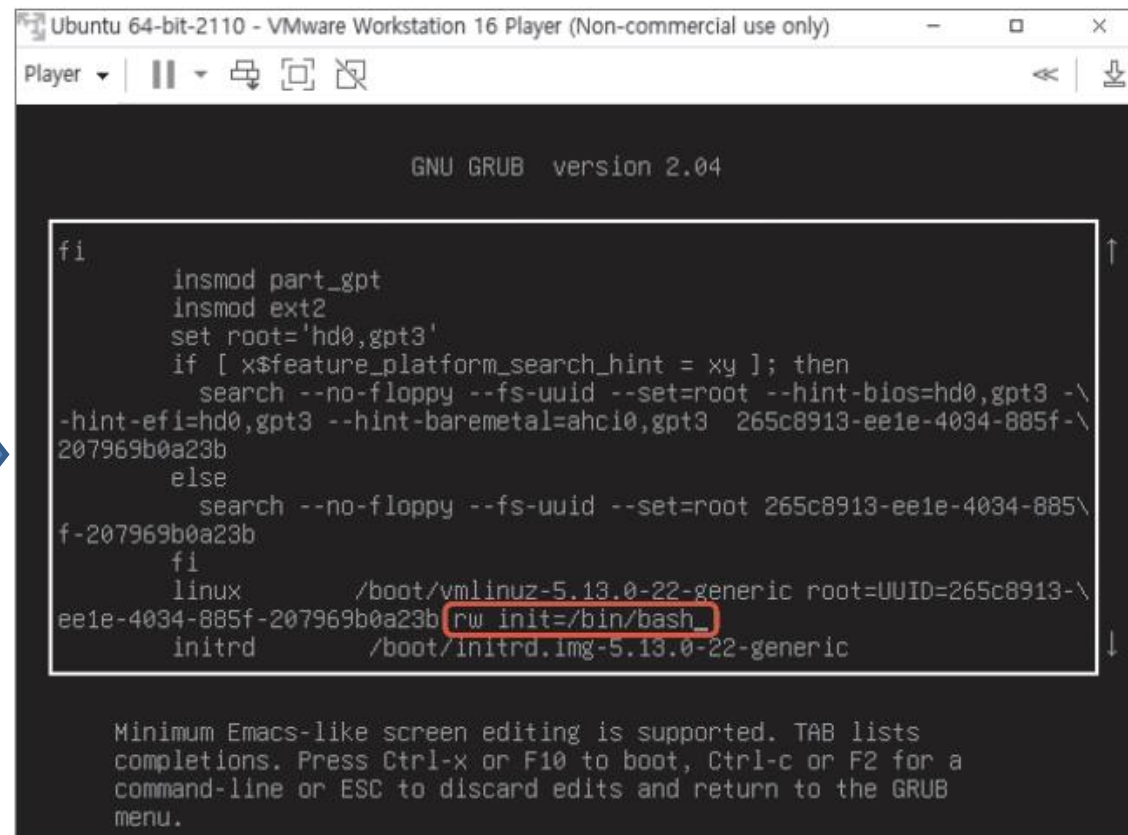


```
GNU GRUB version 2.04

f1
  insmod part_gpt
  insmod ext2
  set root='hd0,gpt3'
  if [ x$feature_platform_search_hint = xy ]; then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,gpt3 -\
-hint-efi=hd0,gpt3 --hint-baremetal=ahci0,gpt3 265c8913-ee1e-4034-885f-\
207969b0a23b
  else
    search --no-floppy --fs-uuid --set=root 265c8913-ee1e-4034-885\
f-207969b0a23b
  fi
  linux /boot/vmlinuz-5.13.0-22-generic root=UUID=265c8913-\
ee1e-4034-885f-207969b0a23b ro splash $vt_handoff
  initrd /boot/initrd.img-5.13.0-22-generic

Minimum Emacs-like screen editing is supported. TAB lists
completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for a
command-line or ESC to discard edits and return to the GRUB
menu.
```

(a) 수정하기 전 화면



```
GNU GRUB version 2.04

f1
  insmod part_gpt
  insmod ext2
  set root='hd0,gpt3'
  if [ x$feature_platform_search_hint = xy ]; then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,gpt3 -\
-hint-efi=hd0,gpt3 --hint-baremetal=ahci0,gpt3 265c8913-ee1e-4034-885f-\
207969b0a23b
  else
    search --no-floppy --fs-uuid --set=root 265c8913-ee1e-4034-885\
f-207969b0a23b
  fi
  linux /boot/vmlinuz-5.13.0-22-generic root=UUID=265c8913-\
ee1e-4034-885f-207969b0a23b rw init=/bin/bash
  initrd /boot/initrd.img-5.13.0-22-generic

Minimum Emacs-like screen editing is supported. TAB lists
completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for a
command-line or ESC to discard edits and return to the GRUB
menu.
```

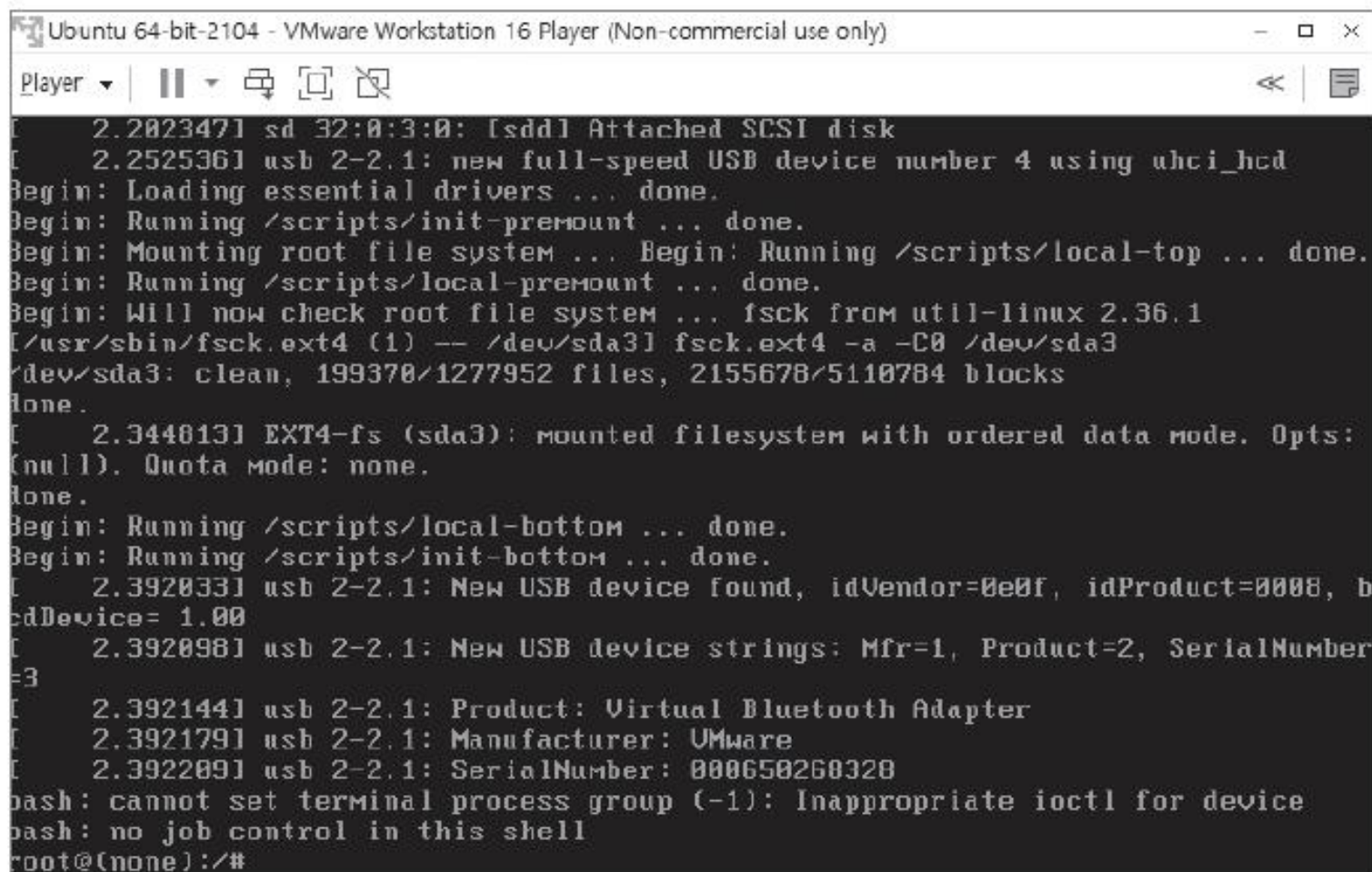
(b) 수정한 화면

그림 8-10 단일 사용자 모드로 부팅하기 위해 리눅스 커널 항목 수정

## 05. 부트 로더

④ 재시작하기: root 계정으로 동작

⑤ 작업 완료 후 재부팅  
: reboot -f



```
Ubuntu 64-bit-2104 - VMware Workstation 16 Player (Non-commercial use only)
Player | || | |
[ 2.202347] sd 32:0:3:0: [sdd] Attached SCSI disk
[ 2.252536] usb 2-2.1: new full-speed USB device number 4 using uhci_hcd
Begin: Loading essential drivers ... done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/local-top ... done.
Begin: Running /scripts/local-premount ... done.
Begin: Will now check root file system ... fsck from util-linux 2.36.1
[/usr/sbin/fsck.ext4 (1) -- /dev/sda3] fsck.ext4 -a -C0 /dev/sda3
/dev/sda3: clean, 199370/1277952 files, 2155678/5110784 blocks
done.
[ 2.344013] EXT4-fs (sda3): mounted filesystem with ordered data mode. Opts:
(null). Quota mode: none.
done.
Begin: Running /scripts/local-bottom ... done.
Begin: Running /scripts/init-bottom ... done.
[ 2.392033] usb 2-2.1: New USB device found, idVendor=0e0f, idProduct=0008, b
cdDevice= 1.00
[ 2.392098] usb 2-2.1: New USB device strings: Mfr=1, Product=2, SerialNumber
=3
[ 2.392144] usb 2-2.1: Product: Virtual Bluetooth Adapter
[ 2.392179] usb 2-2.1: Manufacturer: VMware
[ 2.392209] usb 2-2.1: SerialNumber: 000650260320
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
root@none):/#
```

그림 8-11 root 계정 화면

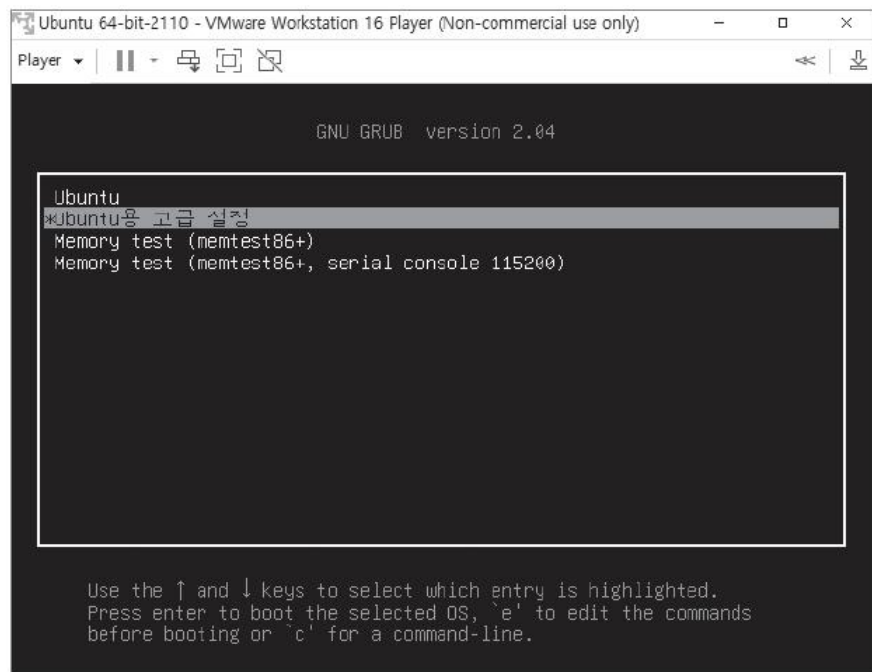
# 05. 부트 로더

## ■ 복구 모드로 부팅하기

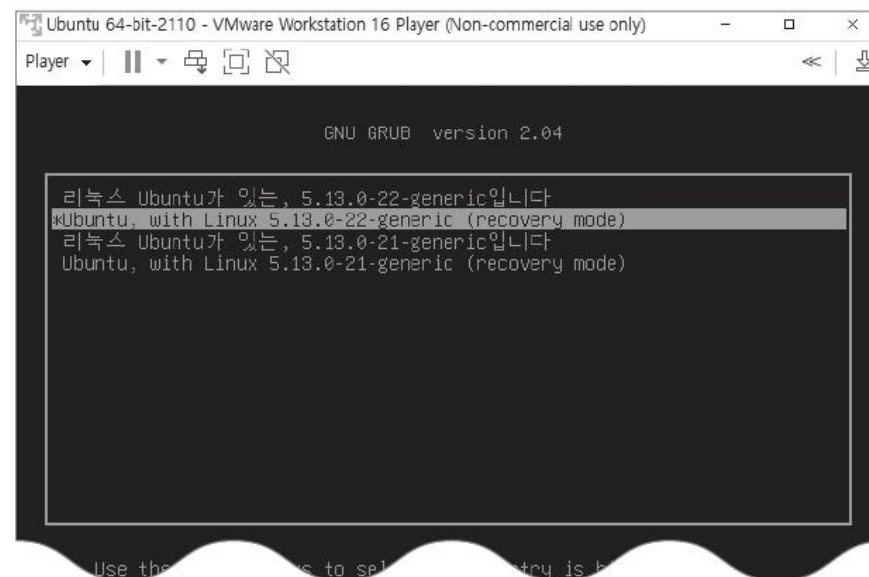
- 복구 모드는 가장 기본적인 서비스만 제공하며 명령 모드로 작업 가능

### ① 복구 모드 선택하기

- ① GRUB 메뉴 초기 화면에서 'Ubuntu용 고급 설정'을 선택
- ② 메뉴가 출력되면 recovery mode를 선택



(a) Ubuntu용 고급 설정 선택



(b) recovery mode 선택 화면

그림 8-12 recovery mode 선택

## 05. 부트 로더

### ② 복구 메뉴 항목에서 root 항목 선택하기

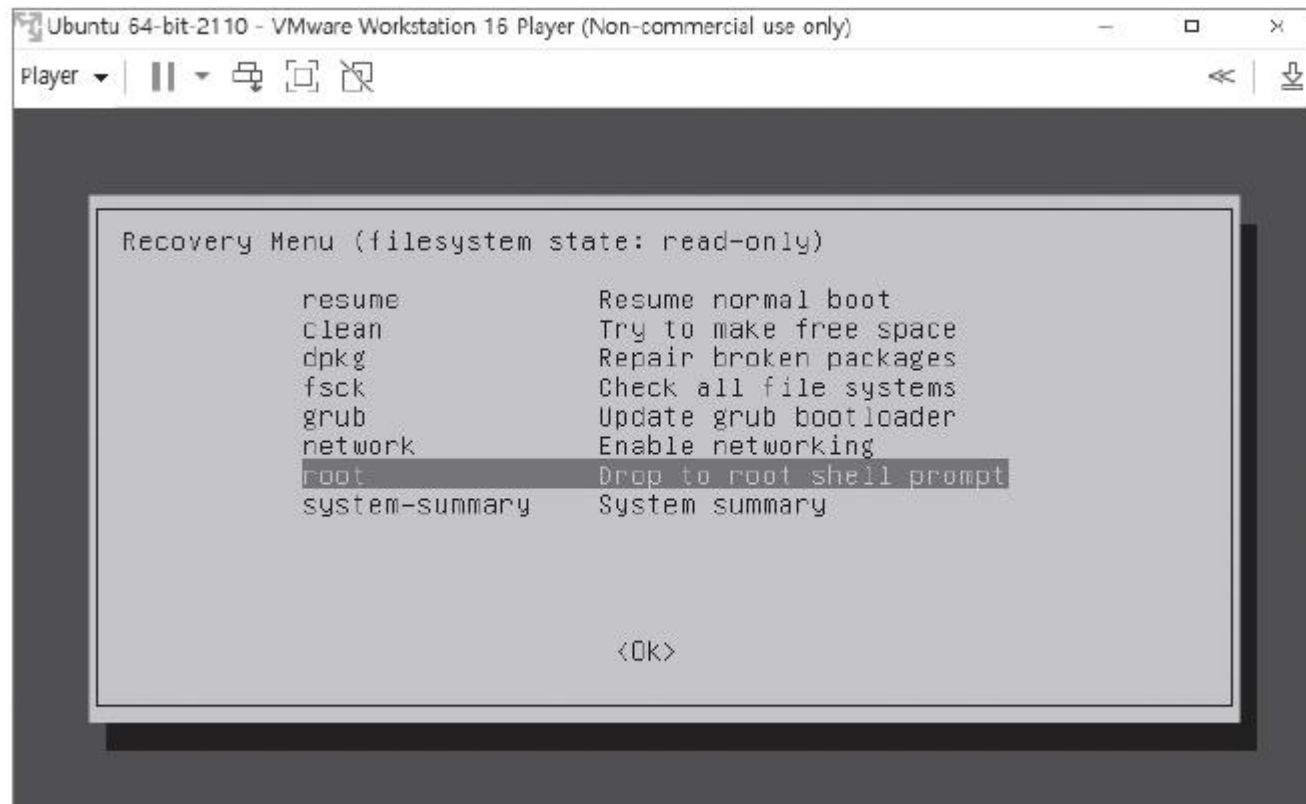


그림 8-13 복구 메뉴 선택 화면

## 05. 부트 로더

- ③ root 항목을 선택하면 바로 root 프롬프트가 출력됨
- ④ 읽기/쓰기 모드로 다시 마운트하기

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

- ⑤ 복구 작업 수행 후 재시작하기  
: reboot -f

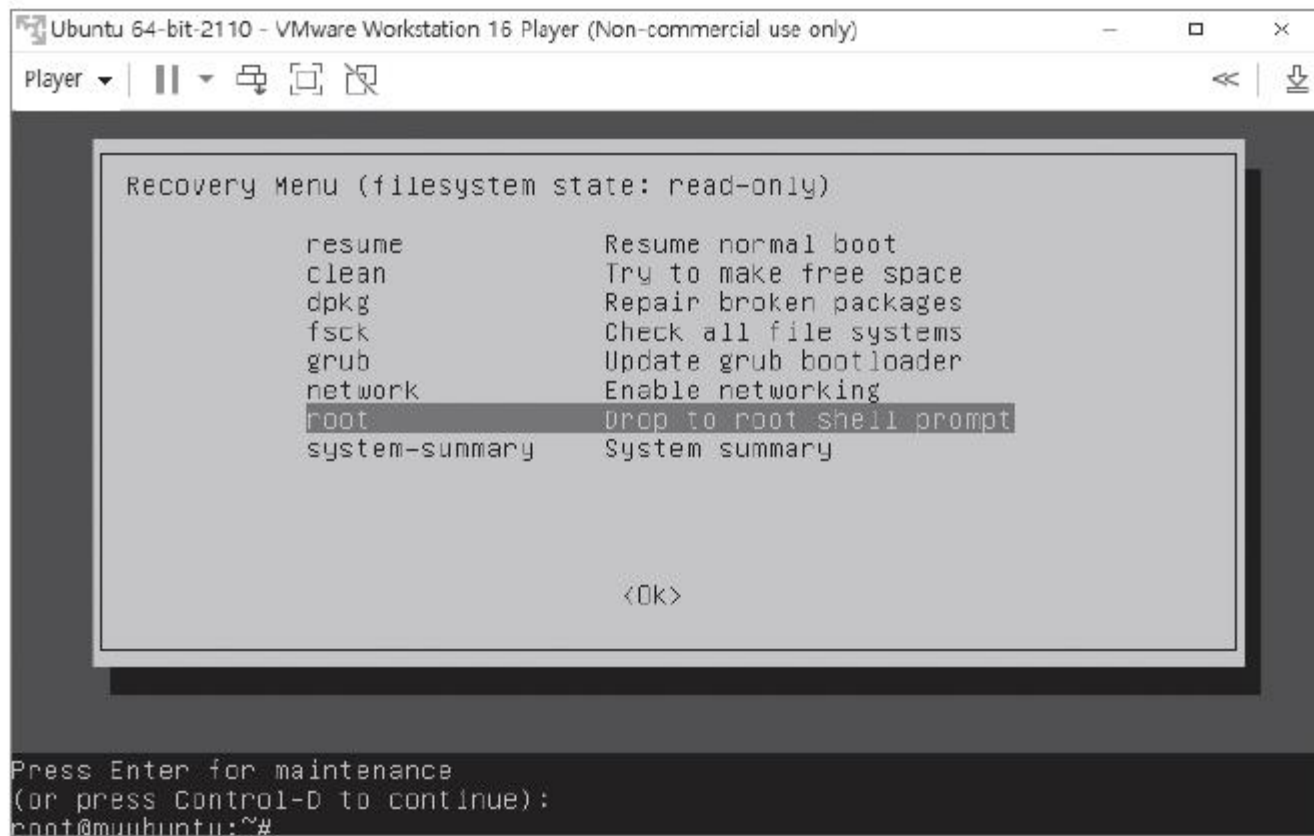


그림 8-14 root 프롬프트 출력 화면

# Thank You !