

ABOUT LINUX

PART H

Linux 개요 및 환경 구축

1

2

- Linux 살펴보기

- Ubuntu Linux 설치 및 설정

CH01 – Linux 살펴보기

▶ 리눅스

- ▶ 1989년 핀란드 헬싱키대학에 재학 중이던 리누스 토르발스(Linus Torvalds)가 유닉스를 기반으로 개발한 공개용 오퍼레이팅시스템(OS)
- ▶ 1991년 11월 버전 0.02이 일반에 공개되면서 확대 보급되기 시작
- ▶ 소스 코드를 완전 무료로 공개하여 전 세계적으로 약 5백만 명이 넘는 프로그램 개발자 그룹을 형성

▶ 리눅스

- ▶ 파일구성이나 시스템기능의 일부는 유닉스 기반으로 하면서, 핵심 **커널 부분은 유닉스와 다르게 작성**
- ▶ 인터넷 프로토콜인 TCP/IP를 강력하게 지원하는 등 네트워킹에 특히 강점
- ▶ 각종 주변기기에 따라 혹은 사용하는 시스템 특성에 맞게 소스를 변경할 수 있어 **다양한 변종** 출현

▶ 리눅스

▶ GNU(Gnu is Not Unix) 프로젝트

- ▶ 유닉스(Unix)의 상업적 확산에 반발해 1984년부터 진행된 프로젝트
- ▶ '모두가 공유할 수 있는 소프트웨어'를 만드는 것
- ▶ 1985년 리처드 스톤먼은 FSF재단 설립

Linux 살펴보기

▶ 리눅스

▶ GNU(Gnu is Not Unix) 프로젝트

▶ <http://www.gnu.org>



GNU 운영 체제

Sponsored by the [Free Software Foundation](#)

[GNU에 대하여](#) [철학](#) [라이선스](#) [교육](#) [SOFTWARE](#) [문서](#) [GNU 둡기](#)

[이 글은 영어로 작성된 원문을 한국어로 번역한 것입니다.](#)

GNU란 무엇인가?

GNU는 유닉스 형태의 운영체제로 [자유 소프트웨어](#)이며 여러분의 자유를 존중합니다. GNU의 개발은 여러분의 자유를 억압할 수 있는 소프트웨어를 사용하지 않도록 하였습니다.

자유 소프트웨어로만 만들어진 [설치 가능한 GNU 버전](#)을 추천합니다. (좀 더 정확히 말하면, GNU/Linux 배포판) [GNU의 상세 내용](#).

지금 GNU/Linux! 내려받기

▶ 리눅스

▶ FSF(Free Software Foundation) 비영리 단체

- ▶ GNU 프로젝트에서 제작한 소프트웨어 지원
- ▶ 프로그램 복제, 변경, 소스 코드 사용 제한 철폐
- ▶ GPL(GNU General Public License) 라이선스

'누구나 자유롭게 소프트웨어와 소프트웨어 소스를 사용 및 변경 할 수 있다.
단, 변경된 소스 또는 같은 라이선스 하에 공개되어야 한다

첫째, 프로그램을 복제(copying)하고 이를 공유할 수 있는 자유

둘째, 소스 코드를 원용해서 이를 개작(modification)할 수 있는 자유

셋째, 개작된 프로그램을 배포(distribution)할 수 있는 자유

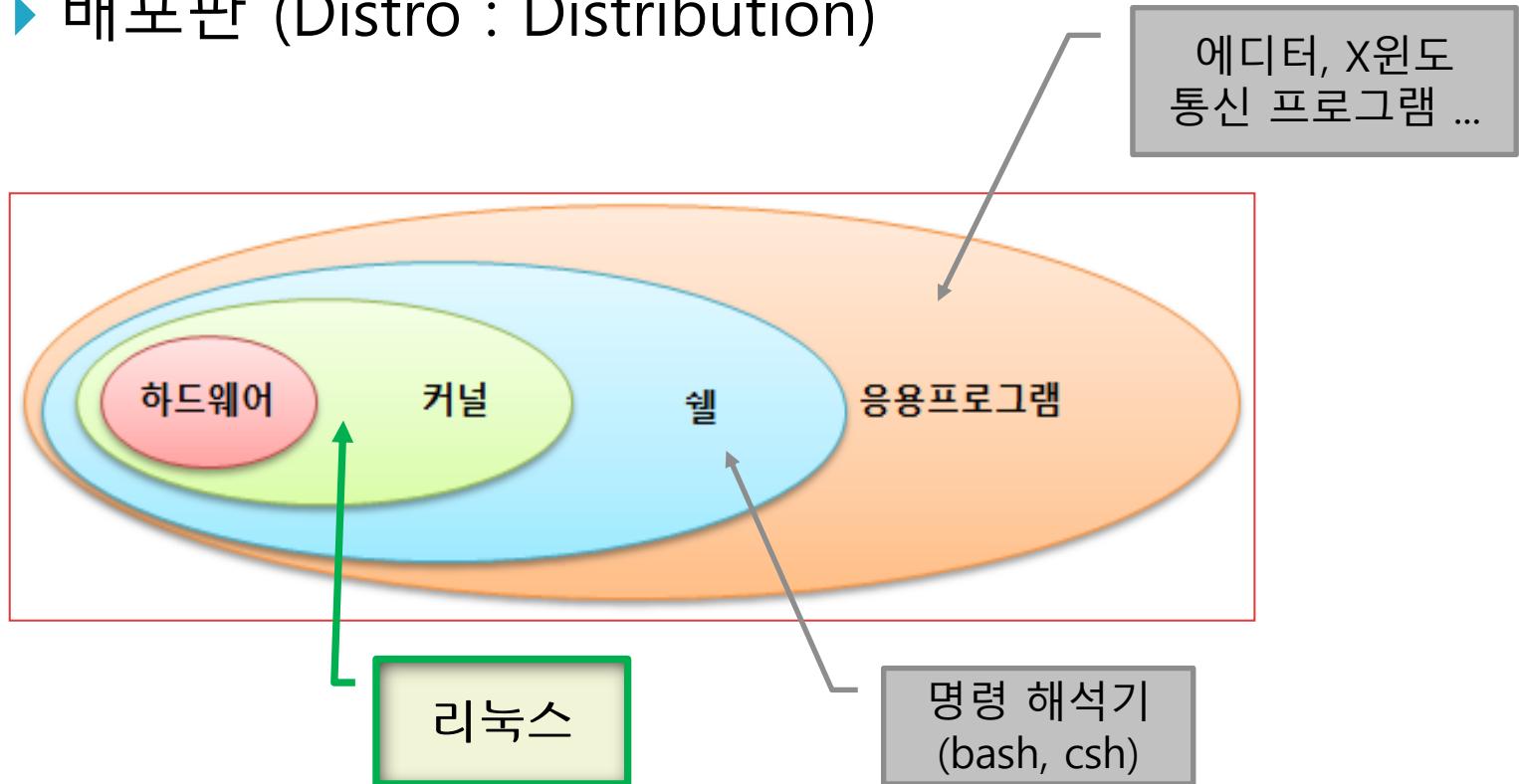
▶ 리눅스

- ▶ 배포판 (Distro : Distribution)
 - ▶ 리눅스 운영체제의 커널에 각종 응용 프로그램을 묶어서 사용하기 편하게 만든 것
 - ▶ 커널은 동일하지만 구성 응용 프로그램이 조금씩 다름
 - ▶ 레드햇(RedHat), 수세(SuSe), 데비안(Debian), 젠투(Gentoo), 슬렉웨어(Slackware) 등등

Linux 살펴보기

▶ 리눅스

▶ 배포판 (Distro : Distribution)



▶ 리눅스

▶ 커널 (Kernel)

- ▶ 현재 지원하는 H/W 장치 지원 여부, H/W 성능, 제어 코드
- ▶ 리눅스 토르발스는 커널을 지금도 계속 업그레이드 중
- ▶ www.kernel.org에서 최신 커널 다운로드 가능
- ▶ 이미 사용 중인 리눅스에 커널을 최신으로 업그레이드 가능
 - ▶ '커널 업그레이드' 또는 '커널 컴파일'

Linux 살펴보기

▶ 리눅스

▶ 커널 (Kernel)

The Linux Kernel Archives



About Contact us FAQ Releases Signatures Site news

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel:

 **4.16.10**

mainline:	4.17-rc6	2018-05-20	[tarball]	[patch]	[inc. patch]	[view diff]	[browse]		
stable:	4.16.10	2018-05-19	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.14.42	2018-05-19	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.9.101	2018-05-19	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.4.132	2018-05-16	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.1.51	2018-03-27	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.18.109 [EOL]	2018-05-16	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.16.56	2018-03-19	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.2.101	2018-03-19	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20180517	2018-05-17							[browse]

Linux 살펴보기

▶ 리눅스

▶ 커널 (Kernel)

커널 버전	0.01	1.0	2.0	2.2	2.4	2.8	3.0	3.8	3.19	4.0	4.8	4.16
발표 연도	1991	1994	1996	1999	2001	2003	2011	2013	2015	2015	2016	2018

안정 버전 (Stable Version)

→ 이미 검증된 개발 완료 코드로 구성

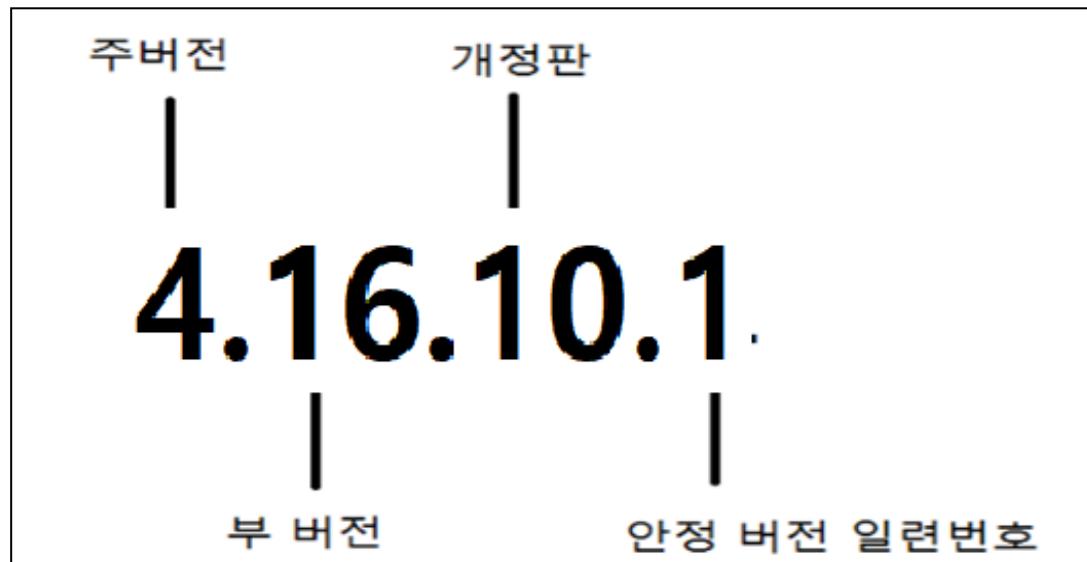
개발 버전 (Developmental Version)

→ 현재 개발 중인 버전

Linux 살펴보기

▶ 리눅스

▶ 커널 (Kernel)



부 버전 → 짹수 : 안정 버전 → 홀수 : 개발 버전

▶ 리눅스 특징

▶ 멀티 유저 시스템

- ▶ 리눅스 시스템을 사용하기 위해서는 사용자 계정을 부여 받아서 적절한 인증 과정을 거쳐야 함

▶ 관리자 계정 **root**

- ✓ 시스템에 대한 모든 권한 가짐
- ✓ 다른 계정의 권한 수정, 계정 추가, 삭제 가능

▶ 일반 사용자 계정

- ✓ root가 정해준 권한 범위 안에서 시스템 사용
- ✓ 다른 사용자 계정의 파일, 디렉토리 접근 불가

▶ 리눅스 특징

▶ 쉘(shell)



- ▶ 리눅스 커널과 사용자를 연결 시켜주는 인터페이스 역할 수행
- ▶ 텍스트 기반의 사용자 명령어를 입력 받을 수 있는 환경 제공
- ▶ 리눅스 시스템에 명령을 내리고 결과 확인 및 시스템 상태 확인

▶ 리눅스 특징

▶ 쉘(shell)

- ▶ 다양한 쉘 프로그램 존재
- ▶ /etc/shells 파일에서 현재 시스템에서 사용 가능한 쉘 확인
- ▶ 가장 많이 사용되는 것이 bash(Bourne Again shell) 쉘로 GNU 프로젝트에서 만들어서 배포

```
pi@raspberrypi: /tmp/shk
```

```
pi@raspberrypi:/tmp/shk $ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
```

▶ 리눅스 특징

▶ 파일시스템

- ▶ 데이터를 찾기 쉽도록 정렬하고 분류하는 방식
- ▶ 저장 매체의 모든 파일과 디렉터리를 포함하기 위한 **루트 디렉터리**가 존재
- ▶ **다양한 저장 매체**가 존재함으로써 **다양한 파일 시스템과 다양한 루트 디렉터리**가 존재

Linux 살펴보기

▶ 리눅스 특징

▶ 파일시스템

종 류	특징 및 용도
EFI 시스템 파티션	UEFI 시스템에서 GPT (GUID partition table)로 장치 부팅에 필요한 작은 파티션
BIOS 부트	BIOS 시스템에서 GPT (GUID partition table)로 장치 부팅에 필요한 작은 파티션
swap	가상 메모리를 지원하는데 사용 시스템이 처리하는 데이터를 저장할 RAM이 충분하지 않을 때 데이터가 기록 보통 RAM의 2배 정도 지정
VFAT	FAT 파일 시스템의 Microsoft Windows 긴 파일명과 호환 가능
EXT2	일반 파일, 디렉토리, 심볼릭 링크 등을 포함하여 표준 Unix 파일 유형 지원 255 자까지 허용하는 긴 파일 이름을 부여 할 수 있는 기능 제공 FSCK(File System Check) 라는 파일 시스템 검사 및 복구 기능

Linux 살펴보기

▶ 리눅스 특징

▶ 파일시스템

종 류	특징 및 용도
EXT3	<p>ext2 파일 시스템을 기반으로 저널링 기술 도입하여 빠르고 안정적 복구 가능</p> <ul style="list-style-type: none">• 저널링(Journaling) 기술 → 주 파일 시스템에 변경사항을 반영(commit)하기 전에, 저널(주로 파일 시스템의 지정된 영역 안의 원형 로그)안에 생성되는 변경사항 추적• 시스템 충돌이나 전원 문제가 발생하면, 빠르게 online 상태로 돌아옴
EXT4	<p>ext3 파일 시스템 기반, 여러가지 기능 개선</p> <p>대용량 파일 시스템 및 대용량 파일 지원, 디스크 공간의 빠르고 효과적인 할당, 디렉토리에 있는 하위 디렉토리 수 제한 없으며 빠른 파일 시스템 확인 및 보다 강력한 저널링 기능 포함</p>
XFS	<p>크기 변경이 자유롭고, 고성능의 파일 시스템이며 16엑사바이트까지 지원</p> <p>메타데이터 저널링을 지원하고 빠른 복구를 제공하며 xfs 파일 시스템은 마운트 중에도 단편화, 크기 변경이 가능하다. 기본적으로 선택을 추천한다. 파티션 최대 지원크기는 500TB</p>

▶ 리눅스 특징

▶ 파일시스템

▶ 루트(Root) 디렉터리

- ▶ 메인 디렉터리 역할
- ▶ 다른 모든 파일과 디렉터리를 담는 곳
- ▶ 이 디렉터리를 중심으로 파일 및 디렉터리가 뻗어 나감

▶ 리눅스 특징

▶ 파일시스템

▶ 분리형 루트(Root)

- ▶ MS Window 운영체제가 채택하고 있는 방식
- ▶ 장치마다 파일 시스템과 루트 디렉터리 따로 사용
- ▶ 각 루트 디렉터리에 **문자로 이름을 할당**
 - ▶ 시스템 디스크는 C: 드라이브
 - ▶ 플로피 디스크는 A: B: 드라이브
 - ▶ CD-ROM은 D: 드라이브
- ▶ 각 장치 별 독립성 보장
- ▶ 문자 26개로 장치의 개수가 26개로 제한

▶ 리눅스 특징

▶ 파일시스템

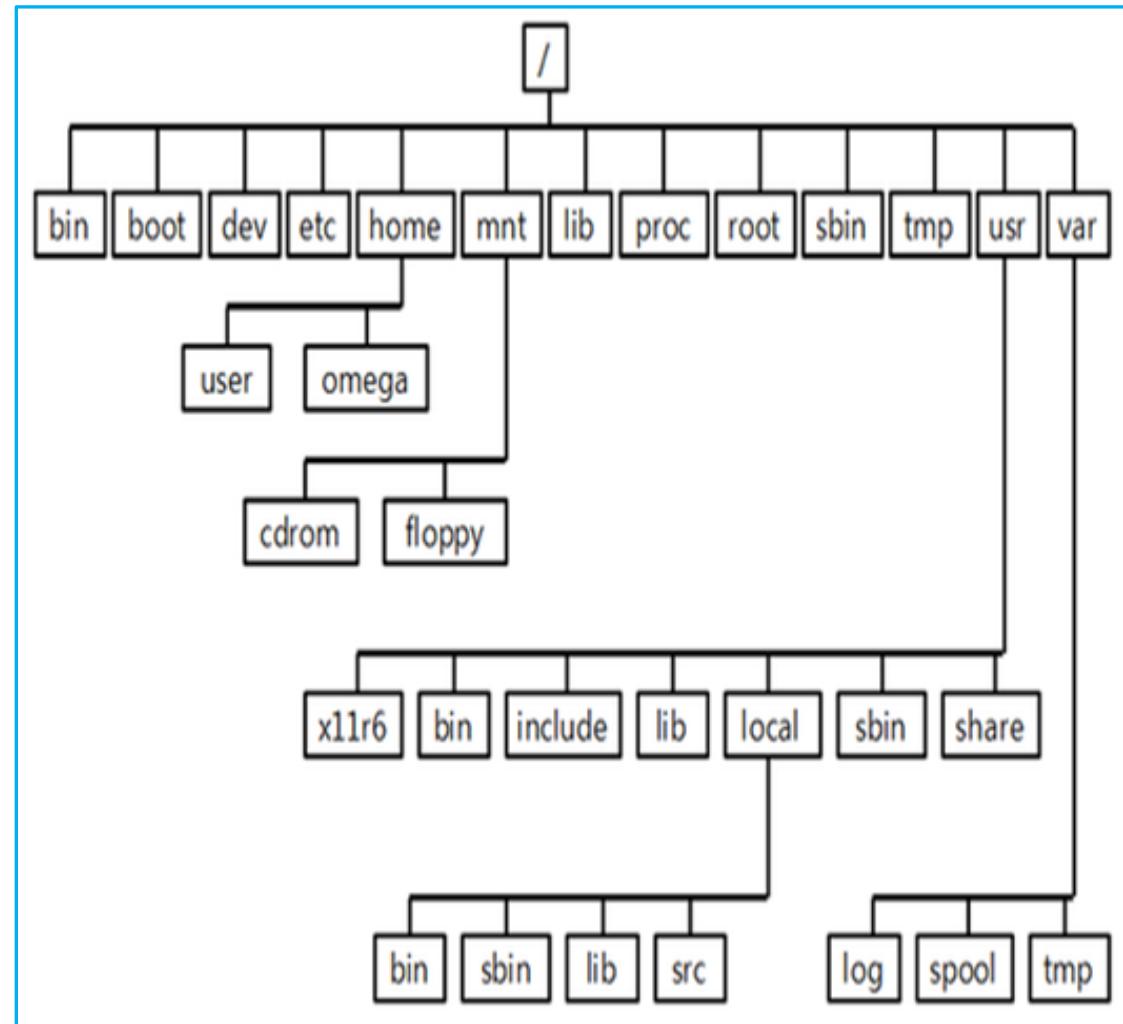
▶ 통합형 루트(Root)

- ▶ 유닉스와 리눅스에서 사용하는 방식
- ▶ 루트 디렉터리가 하나만 존재
- ▶ 새로운 장치의 루트 디렉터리를 추가하는 방식
- ▶ 장치의 개수 제한 없이 확장
- ▶ 하드디스크에서 USB 접근 시 장치 전환 인지 불가
- ▶ 장치별 독립성 없음

Linux 살펴보기

▶ 리눅스 특징

▶ 파일시스템



▶ 리눅스 특징

▶ / 디렉터리

- ▶ 리눅스 파일 시스템의 **최상위 디렉터리**
- ▶ 시스템의 근간이 되며 **절대경로 기준**
- ▶ **파티션 설정 시에 반드시 존재해야 함**
- ▶ 비교 : Window 운영체제의 C 드라이브

▶ 리눅스 특징

▶ /bin 디렉터리

- ▶ binary 약자
- ▶ 기본 실행 파일들 즉 사용자 명령어 및 프로그램 존재
- ▶ 리눅스 기본 명령어 즉, 시스템 운영 명령어 존재
- ▶ **부트에 필요한 명령어가 존재**
- ▶ 부팅 후 시스템 사용자 사용 명령어 존재

▶ 리눅스 특징

▶ /sbin 디렉터리

- ▶ system binary 약자
- ▶ 시스템 관리 실행 명령어 및 관리용 프로그램 존재
- ▶ 시스템 점검 및 복구 명령
- ▶ 시스템 초기 및 종료 명령 등등

▶ 리눅스 특징

▶ /lib 디렉터리

- ▶ 프로그램에서 사용되는 라이브러리 패키지 존재
- ▶ **/lib/modules**에는 커널 모듈 파일 존재
- ▶ 대부분의 라이브러리는 링크로 연결

▶ 리눅스 특징

▶ /proc 디렉터리

- ▶ 각종 프로세서, 프로그램 정보, 하드웨어 정보 저장
- ▶ 가상 파일시스템
- ▶ 커널에 의해서 **메모리에 저장**
- ▶ 대부분 읽기 전용이고 일부 쓰기 가능 파일 존재
- ▶ cat 명령어로 시스템 정보 확인 가능

▶ 리눅스 특징

▶ /etc 디렉터리

- ▶ 시스템 환경 설정 파일 존재
- ▶ 네트워크 관련 설정 파일
- ▶ 사용자 정보 및 암호정보
- ▶ 파일 시스템 정보
- ▶ 보안 파일
- ▶ 시스템 초기화 파일 등 중요 설정 파일

Linux 살펴보기

▶ 리눅스 특징

▶ /dev 디렉터리

- ▶ device 약자
- ▶ 하드디스크, CD-ROM, USB 등등 입출력 장치 파일

autofs	mmcblk0p1	spidev0.1	tty35	tty9
block	mmcblk0p2	stderr	tty36	ttyAMA0
btrfs-control	mqueue	stdin	tty37	ttyprintk
bus	net	stdout	tty38	ttyS0
cachefiles	network_latency	tty	tty39	uhid
char	network_throughput	tty0	tty4	uinput
console	null	tty1	tty40	urandom
cpu_dma_latency	ppp	tty10	tty41	vc-cma
cuse	ptmx	tty11	tty42	vchiq

▶ 리눅스 특징

▶ */var* 디렉터리

- ▶ 시스템에서 사용되는 동적이 파일 존재
- ▶ 각종 시스템 로그파일
- ▶ 사용자 로그인에 대한 보안기록
- ▶ 메일서버 운영 시 전송되는 메일 임시 저장

▶ 리눅스 특징

▶ /usr 디렉터리

- ▶ 사용자들을 위한 대부분의 프로그램 및 라이브러리
- ▶ 가장 사이즈가 큰 디렉터리

▶ 리눅스 특징

▶ /mnt 디렉터리

- ▶ 네트워크 공유, 외장 하드디스크, CD-ROM 등 마운트 할 때 사용하는 디렉터리

▶ /media 디렉터리

- ▶ USB, 카메라, DVD 등 이동식 미디어 마운트 되는 디렉터리

▶ 리눅스 특징

▶ **/home 디렉터리**

- ▶ 일반 사용자의 홈 디렉터리가 만들어 지는 디렉터리
- ▶ 계정과 같은 이름으로 새로운 사용자 디렉터리 생성

▶ /boot 디렉터리

- ▶ 커널 이미지와 부팅 정보 파일
- ▶ /etc/lilo.conf에서 지정한 커널 부팅 이미지 파일 존재

▶ 리눅스 특징

▶ /root 디렉터리

- ▶ 슈퍼유저(root) 사용자의 홈 디렉터리
- ▶ /와 /root 디렉터리는 이름은 같지만 서로 다름

▶ 리눅스 특징

▶ /tmp 디렉터리

- ▶ 프로세스 진행 시 임시 파일이 저장하는 디렉터리
- ▶ 모든 사용자가 쓰기, 읽기 가능

▶ /lost+found 디렉터리

- ▶ 파일 시스템이 추적하다가 놓친 파일들 존재
- ▶ 결함 있는 파일들에 대한 정보 보관
- ▶ /home 하위 사용자 디렉터리에도 존재할 수 있음

CH02 – Ubuntu Linux 설치

▶ Debian Linux

- ▶ Debian 프로젝트에서 제작한 Debian Linux
- ▶ 이안 머독이 1993년 창시
- ▶ 토이 스토리 캐릭터 이름으로 코드명 작성
 - ▶ 1996년 1.1 버전 (코드명 Buzz)
 - ▶ 2015년 8.0 버전 (코드명 Jessie)
- ▶ 패키지의 설치 및 업그레이드가 단순
- ▶ apt 프로그램 통해 SW 설치 및 업데이트 자동

Ubuntu Linux 설치

▶ Ubuntu Linux

- ▶ Debian Linux + 유니티 데스크톱 환경 사용 배포판
- ▶ 2004년 10월 4.10버전 시작
- ▶ 캐노니컬 사가 개발 시작 -> 현재 Ubuntu 재단 개발
- ▶ 2004년 이후 꾸준히 업그레이드
- ▶ 현재 가장 인기있는 Linux 배포판



Ubuntu Linux 설치

▶ Ubuntu Linux

▶ 배포판

▶ Ubuntu Desktop

- ▶ X 윈도 환경 지원
- ▶ 리브레오피스, 파이어폭스, 김프 편집기 등 다양한 GUI 제공

▶ Ubuntu Server

- ▶ TUI(Text User Interface) 환경 인터페이스 지원

▶ Ubuntu Flavours

- ▶ 쿠분투, 우분투 그놈, 우분투 기린, 루분투 등등

Ubuntu Linux 설치

▶ Ubuntu Linux

▶ 버전 & 배포

	일반 버전	LST 버전 (Long Term Support)
지원 기간	9개월	장기 5년
배포 간격	6개월 마다	2년 마다

▶ 버전 이름 → 발표 연도.월

(예) Ubuntu 15.10 → 2015년 10월 발표한 일반 버전

Ubuntu 16.04 LTS → 2016년 4월 발표한 LTS 버전

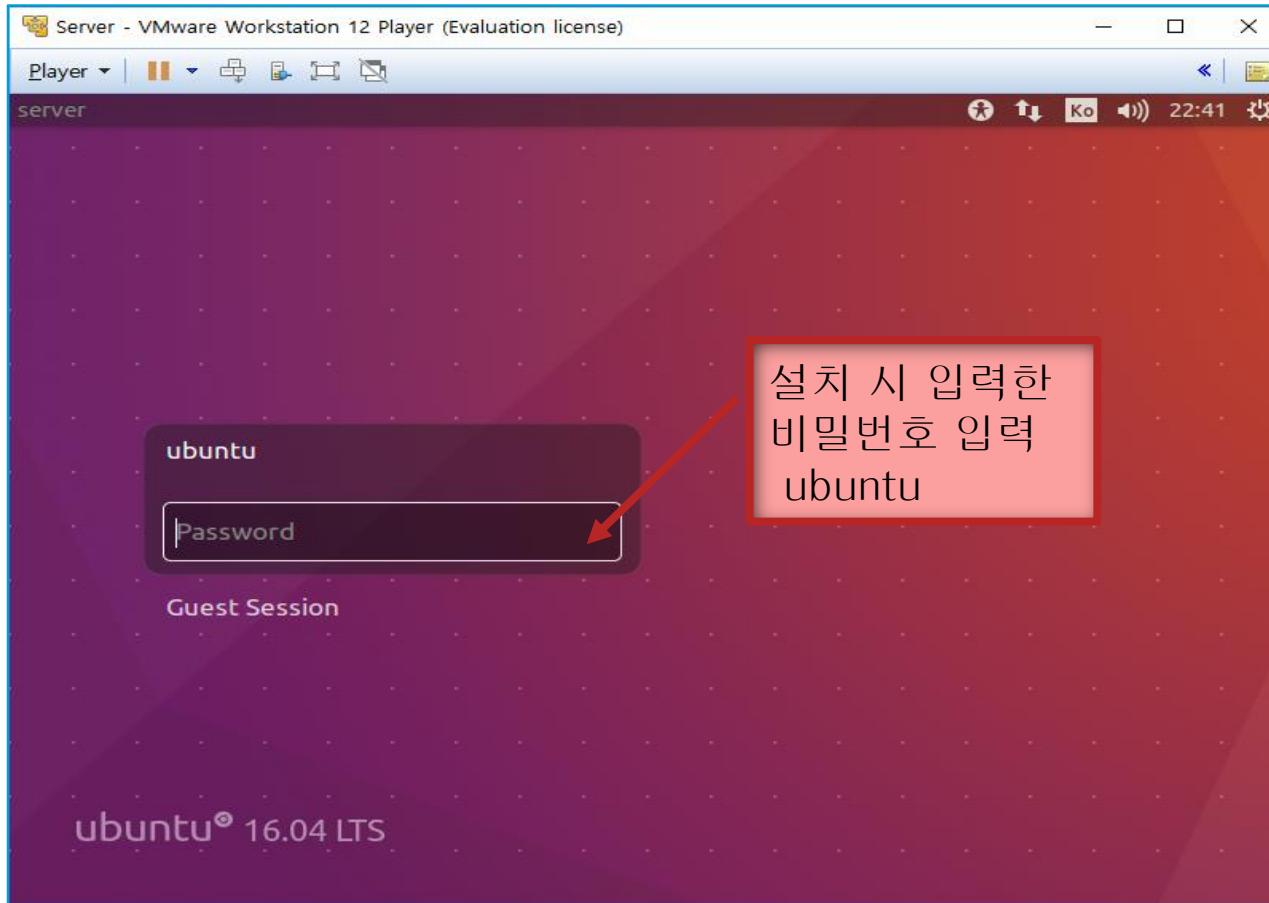
▶ Ubuntu Linux

▶ ubuntu 16.04 LTS H/W 요구사항

- ▶ CPU : 700 MHz보다 빠른 프로세서
- ▶ HDD 여유공간 : 5GB이상의 여유 공간
- ▶ 메모리 : 최소 512MB, 권장 1024MB
- ▶ 그래픽 카드 : 1024 X 768 이상의 해상도 지원

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

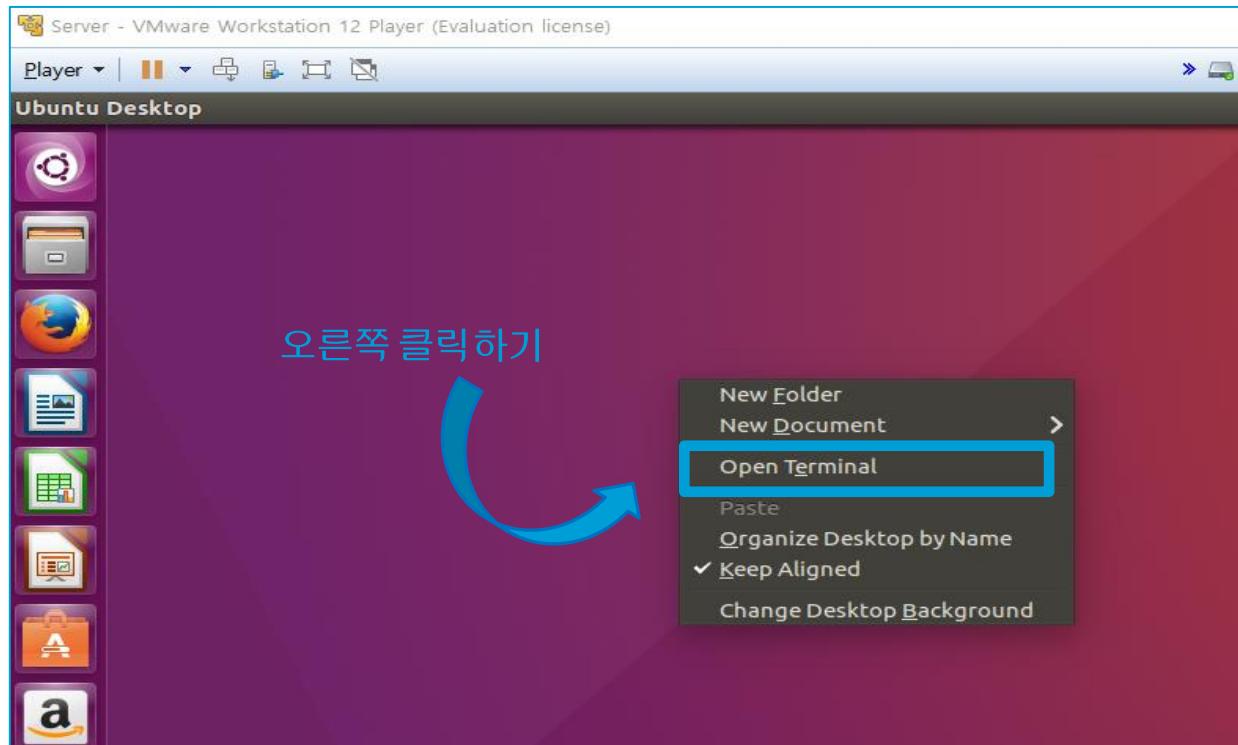


Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 관리자 root 계정 활성화

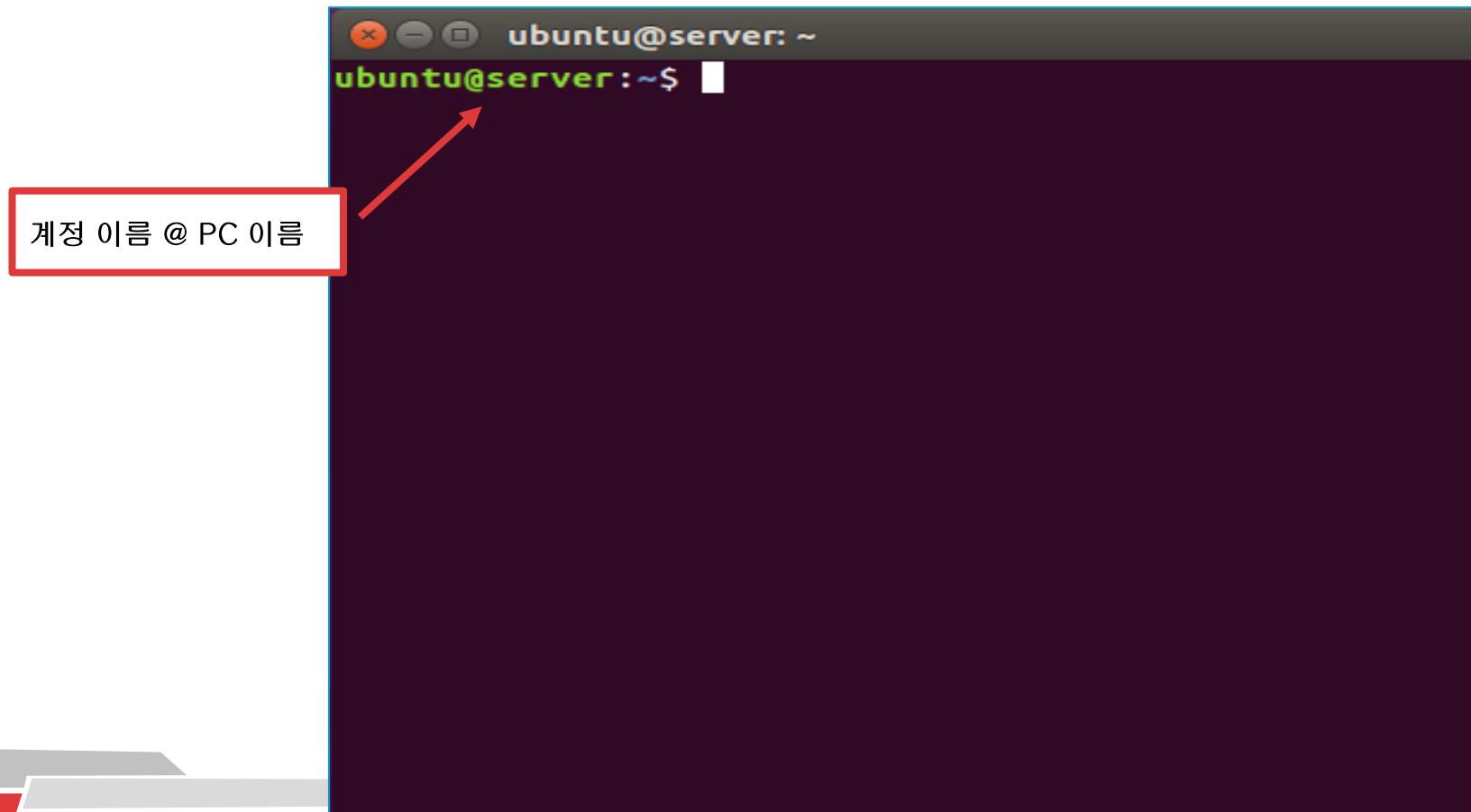
- ▶ 바탕화면 R.Click → 터미널 프로그램 실행



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 관리자 root 계정 활성화



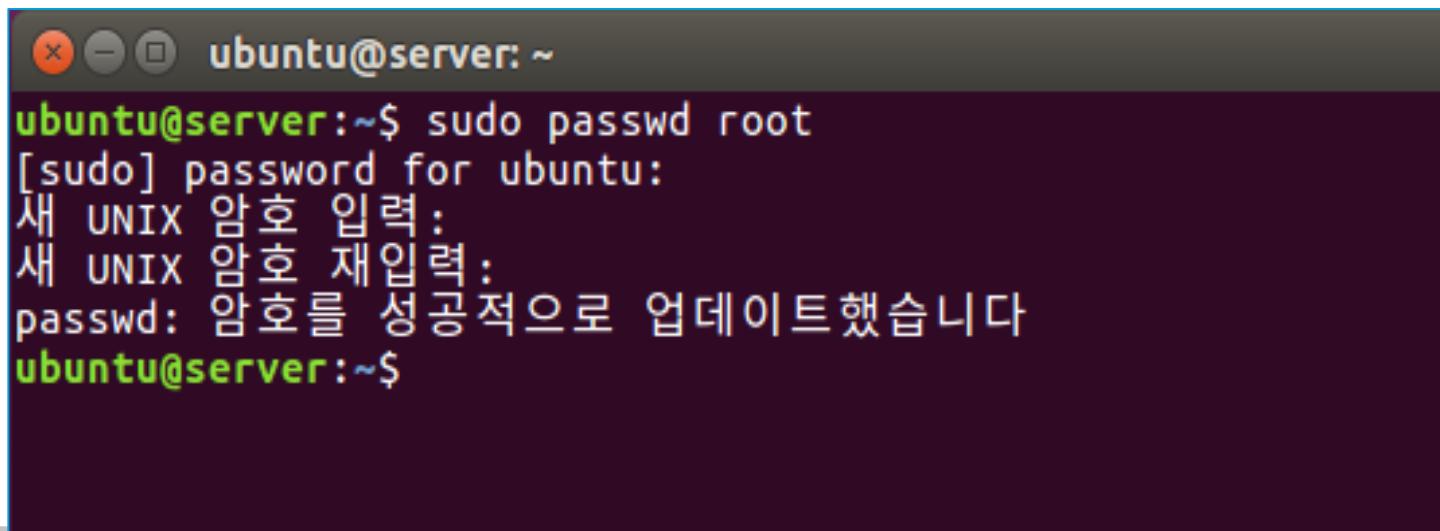
Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 관리자 root 계정 활성화

[sudo 명령어] 명령어를 관리자 권한으로 실행

[passwd 계정] 계정의 비밀번호 설정



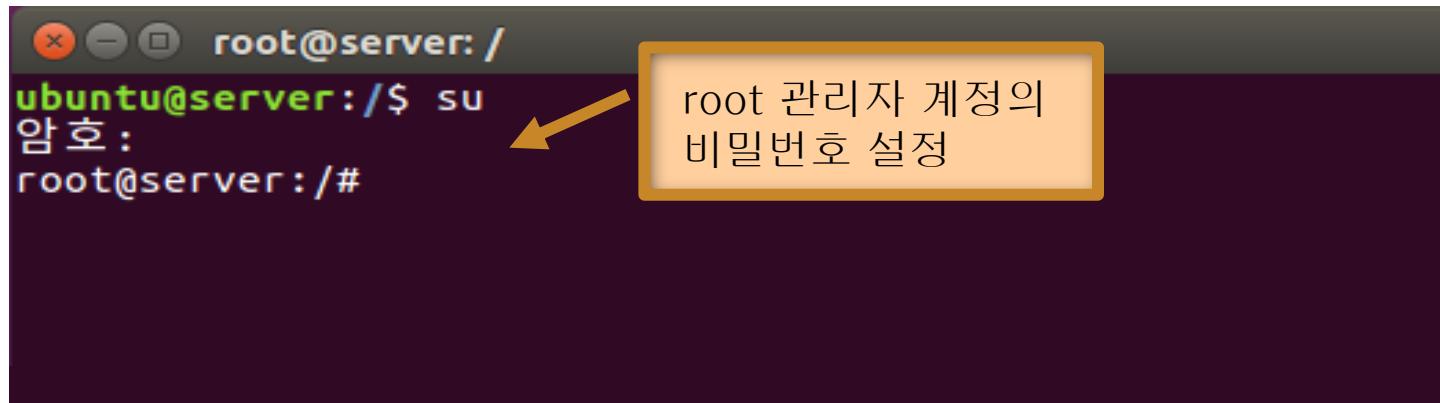
```
ubuntu@server: ~
ubuntu@server:~$ sudo passwd root
[sudo] password for ubuntu:
새 UNIX 암호 입력:
새 UNIX 암호 재입력:
passwd: 암호를 성공적으로 업데이트했습니다
ubuntu@server:~$
```

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 관리자 root 계정 활성화

[su 명령어] substitute 약자, 계정 변경



A screenshot of a terminal window titled "root@server: /". The command "ubuntu@server:/\$ su" is entered, followed by a password prompt "암호:". A yellow callout box with an arrow points to the password input field, containing the Korean text "root 관리자 계정의 비밀번호 설정".

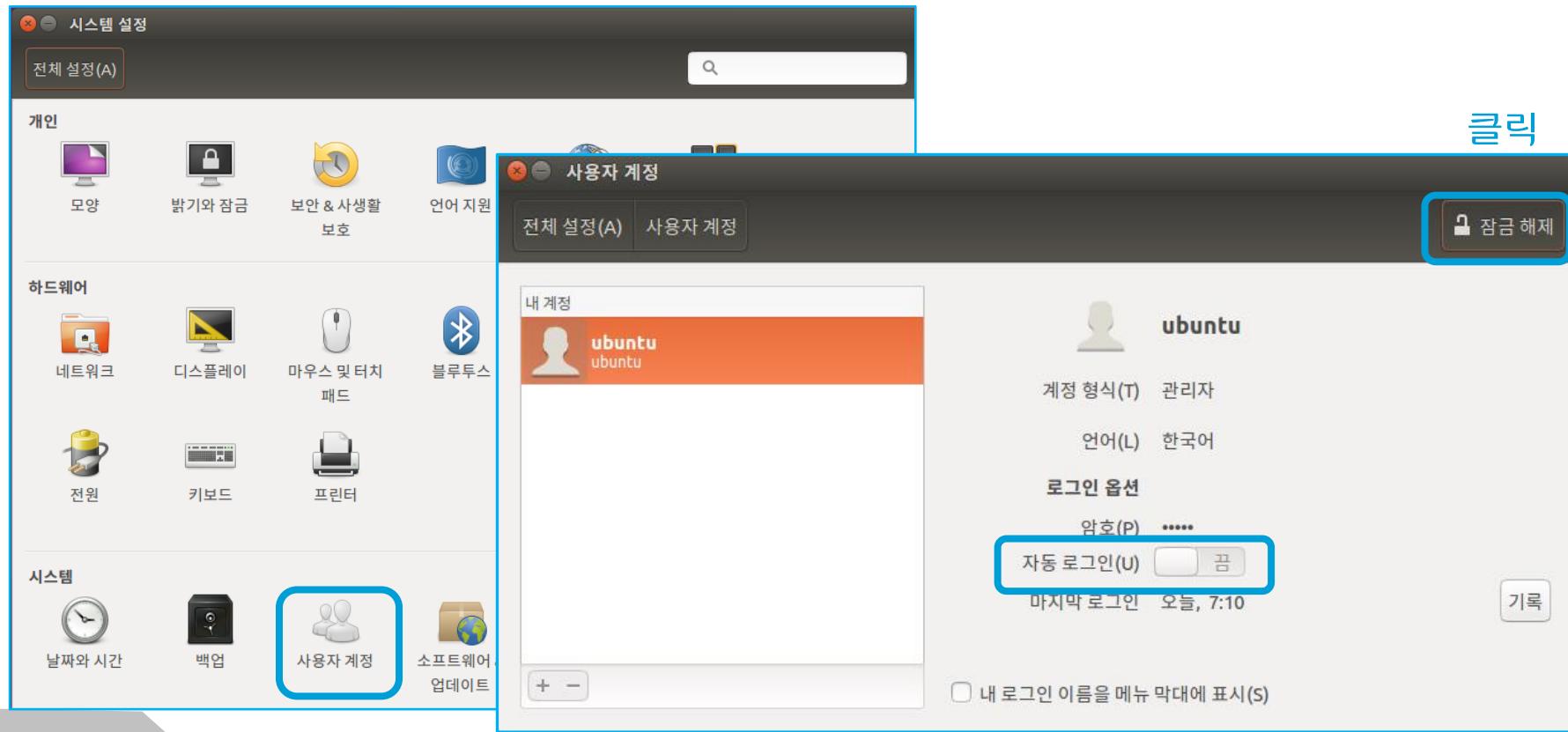
```
root@server: /  
ubuntu@server:/$ su  
암호:  
root@server: #
```

root 관리자 계정의
비밀번호 설정

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

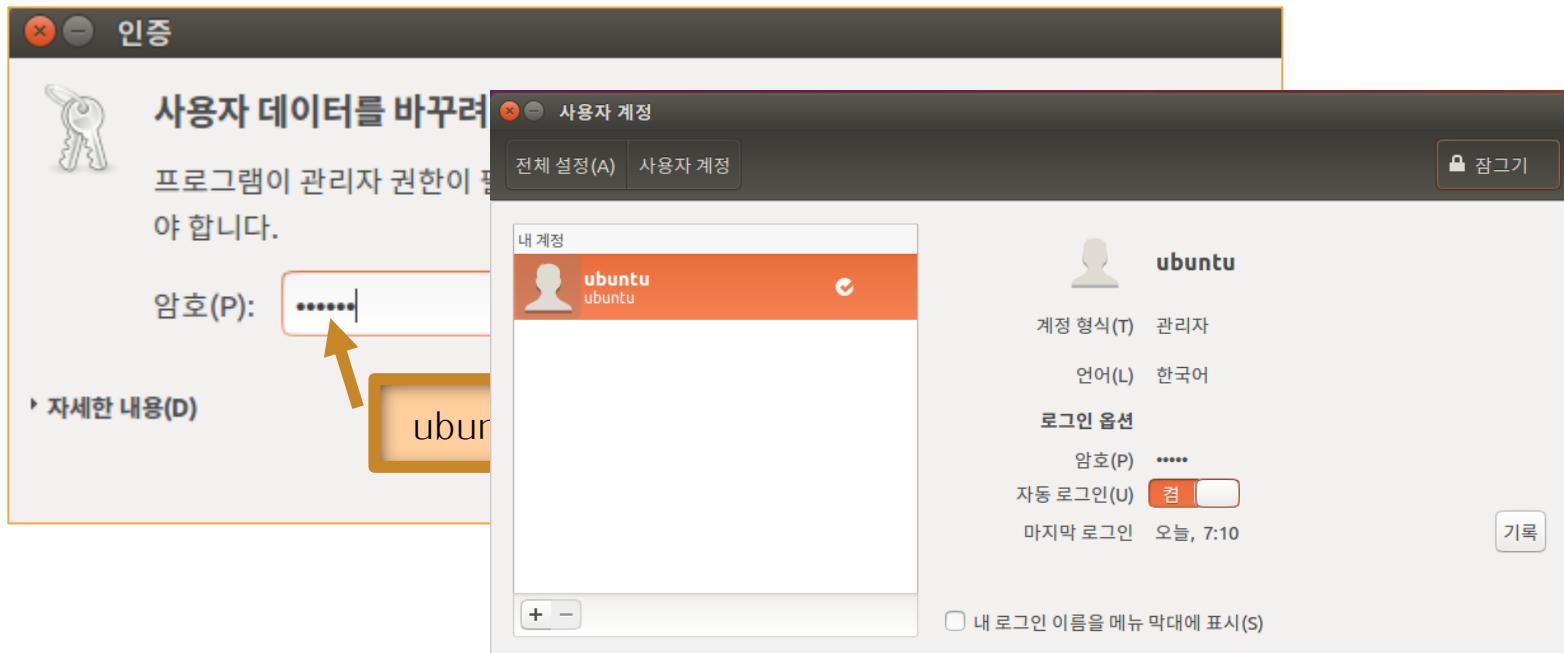
▶ 자동 로그인 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

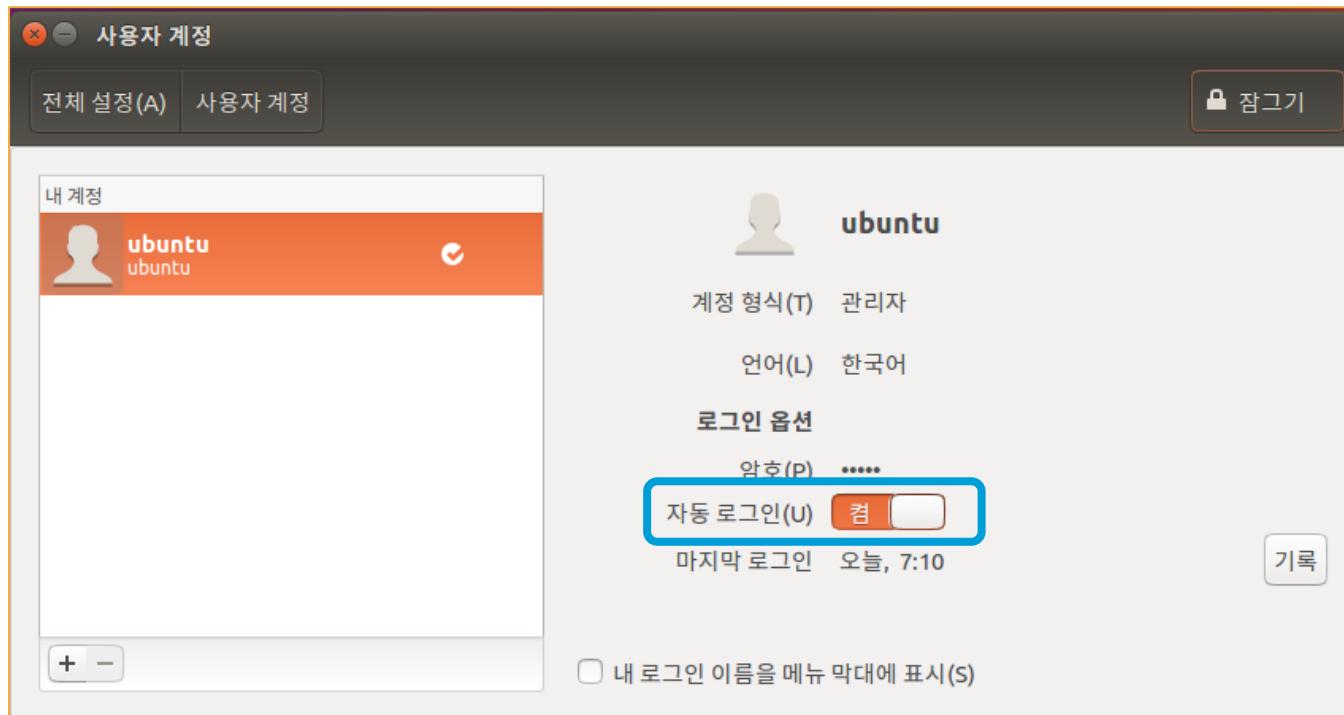
▶ 자동 로그인 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 자동 로그인 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 관리자 root 설정 -> root 자동 로그인 설정

The screenshot shows a terminal window with the following command history:

```
root@server: ~  
ubuntu@server:~$ su - root  
암호:  
root@server:~# gedit /etc/lightdm/lightdm.conf
```

A yellow callout box on the right side provides information about LightDM:

LightDM(The Light Display Manager)

- 로그인 화면 관리 프로그램
- Ubuntu11.10부터 사용

A blue arrow points from the terminal window to the first instance of the configuration file in gedit.

The first instance of the configuration file shows the line:

```
[Seat:*]  
autologin-user=ubuntu
```

This line is circled in pink.

A blue arrow points from the first gedit window to the second instance of the configuration file.

The second instance of the configuration file shows the line:

```
[Seat:*]  
autologin-user=root|
```

This line is also circled in pink.

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 관리자 root 설정 ->root 프로파일 수정

```
root@server:~  
root@server:~# gedit /root/.profile
```

The image shows a terminal window at the top with the command `root@server:~# gedit /root/.profile`. Below it are two screenshots of the Gedit text editor. A blue arrow points from the terminal to the first Gedit window. A pink oval highlights the line `#mesg n || true`. A blue arrow points from the first Gedit window to the second Gedit window, which shows the same line with a pink oval highlighting it.

```
# ~/.profile: executed by Bourne-compatible login shells.  
  
if [ "$BASH" ]; then  
  if [ -f ~/.bashrc ]; then  
    . ~/.bashrc  
  fi  
fi  
  
#mesg n || true
```

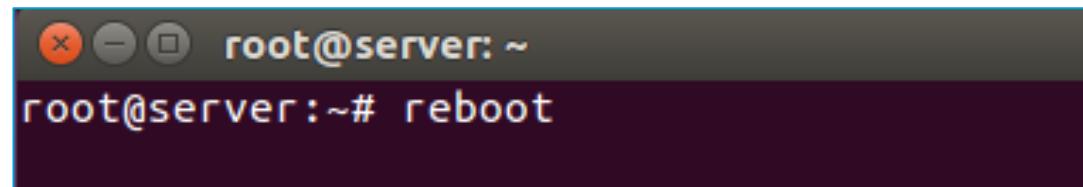
```
# ~/.profile: executed by Bourne-compatible login shells.  
  
if [ "$BASH" ]; then  
  if [ -f ~/.bashrc ]; then  
    . ~/.bashrc  
  fi  
fi  
  
#mesg n || true
```

Ubuntu Linux 설치

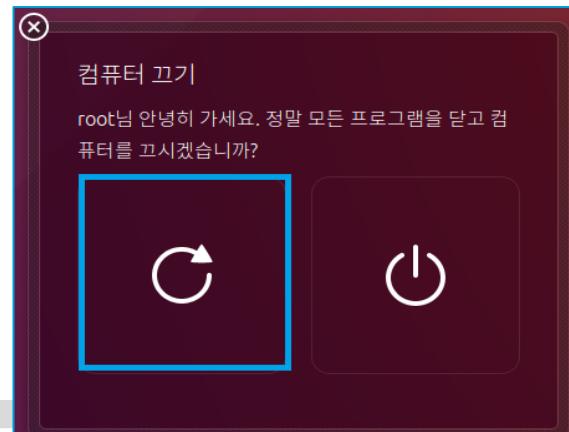
▶ Ubuntu Linux 설정

▶ 관리자 root 설정 적용

- ▶ 터미널 프로그램 → reboot 입력



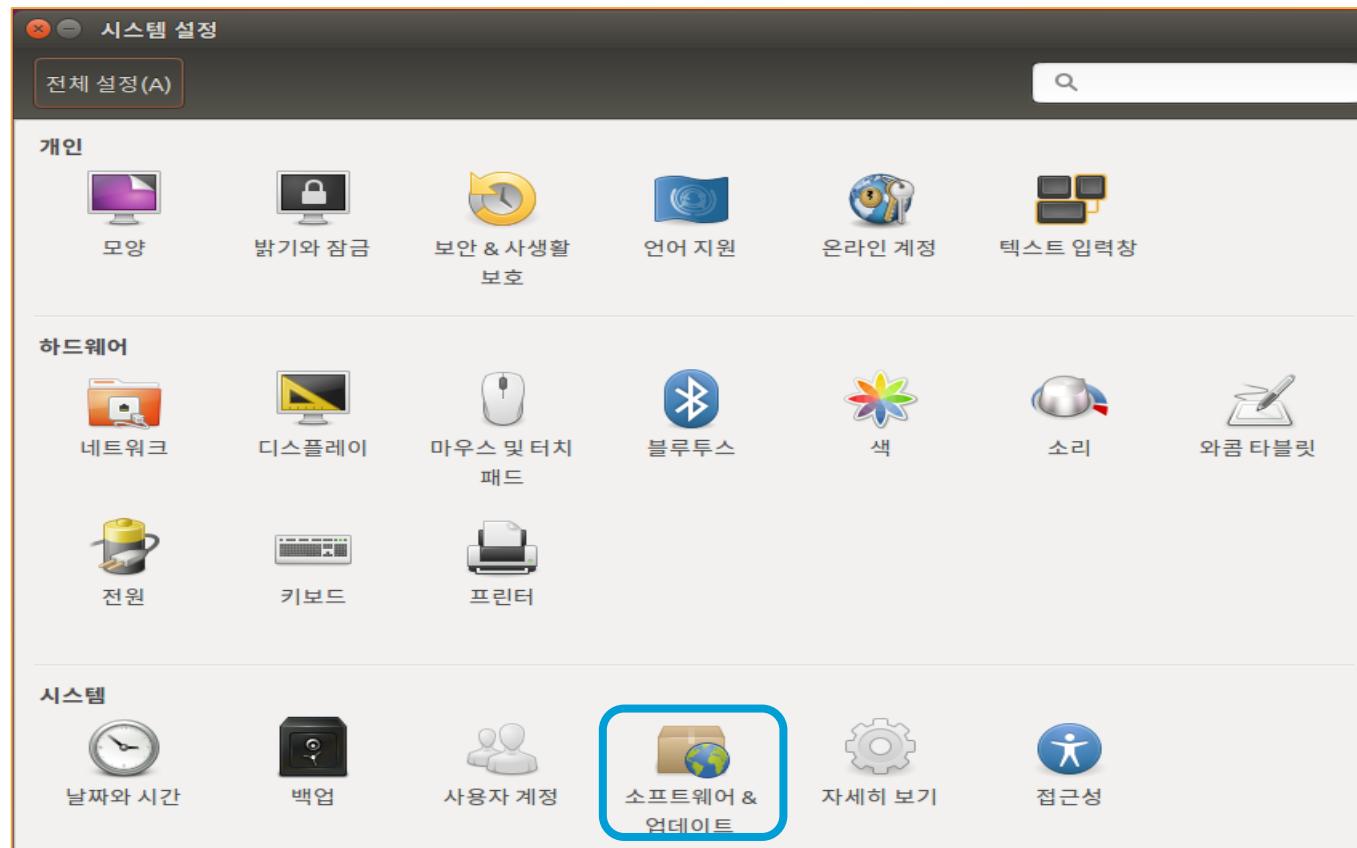
- ▶ 클릭 → 컴퓨터 크기 → 다시 시작



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

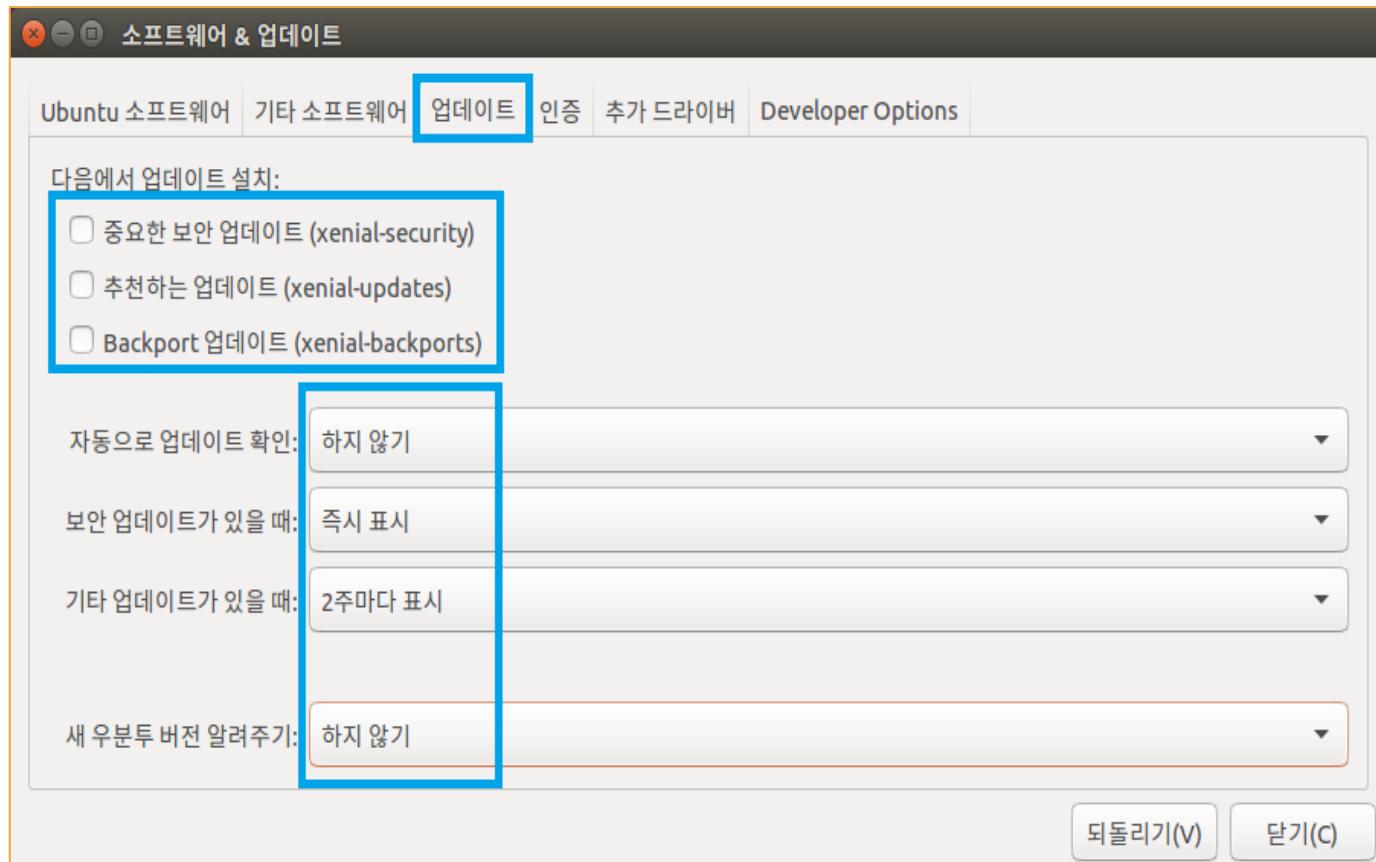
▶ 소프트웨어 & 업데이트 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 소프트웨어 & 업데이트 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 소프트웨어 & 업데이트 설정

▶ sources.list

▶ apt-get 명령어로 mirror site에서 파일 복사해올 주소 리스트

```
deb      http://사이트 distribution section1 section2 section3  
deb-src  http://사이트 distribution section1 section2 section3
```

* deb : 바이너리나 컴파일 된 패키지 존재

* deb-src : 패키지 소스 코드 존재

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 소프트웨어 & 업데이트 설정

▶ sources.list 파일 백업

```
root@server:/etc/apt
root@server:~# cd /etc/apt
root@server:/etc/apt# ls
apt.conf.d      sources.list      sources.list.save  trusted.gpg.d
preferences.d   sources.list.d   trusted.gpg
root@server:/etc/apt# ls -a
.    apt.conf.d      sources.list      sources.list.save  trusted.gpg.d
..   preferences.d   sources.list.d   trusted.gpg
root@server:/etc/apt# mv sources.list sources.list.back
root@server:/etc/apt# ls
apt.conf.d      sources.list.back  sources.list.save  trusted.gpg.d
preferences.d   sources.list.d   trusted.gpg
```

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 소프트웨어 & 업데이트 설정

- ▶ sources.list 파일을 /etc/apt 아래로 복사

```
# wget http://download.hanbit.co.kr/ubuntu/16.04/sources.list
```

- ▶ 변경된 sources.list 적용을 위해 업데이트

```
# apt-get update
```

```
# exit
```

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

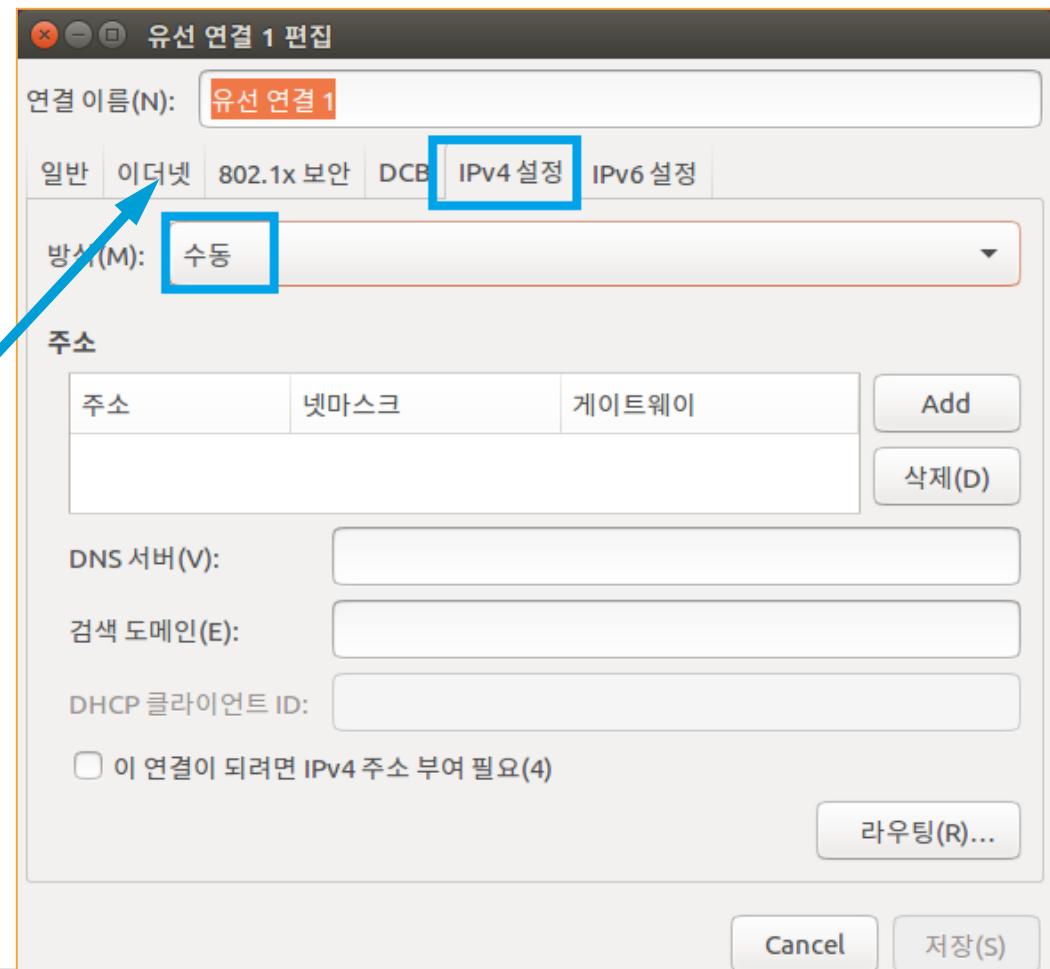
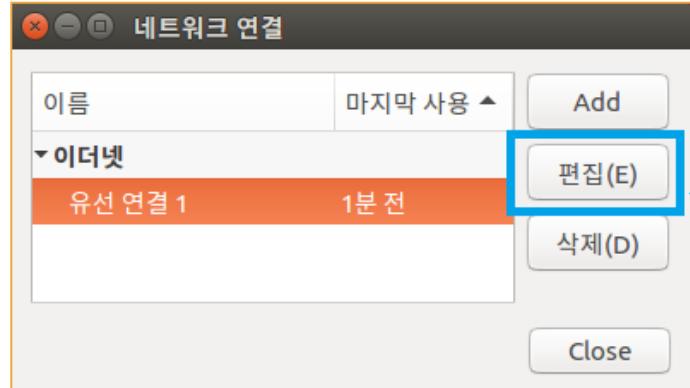
▶ Network 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

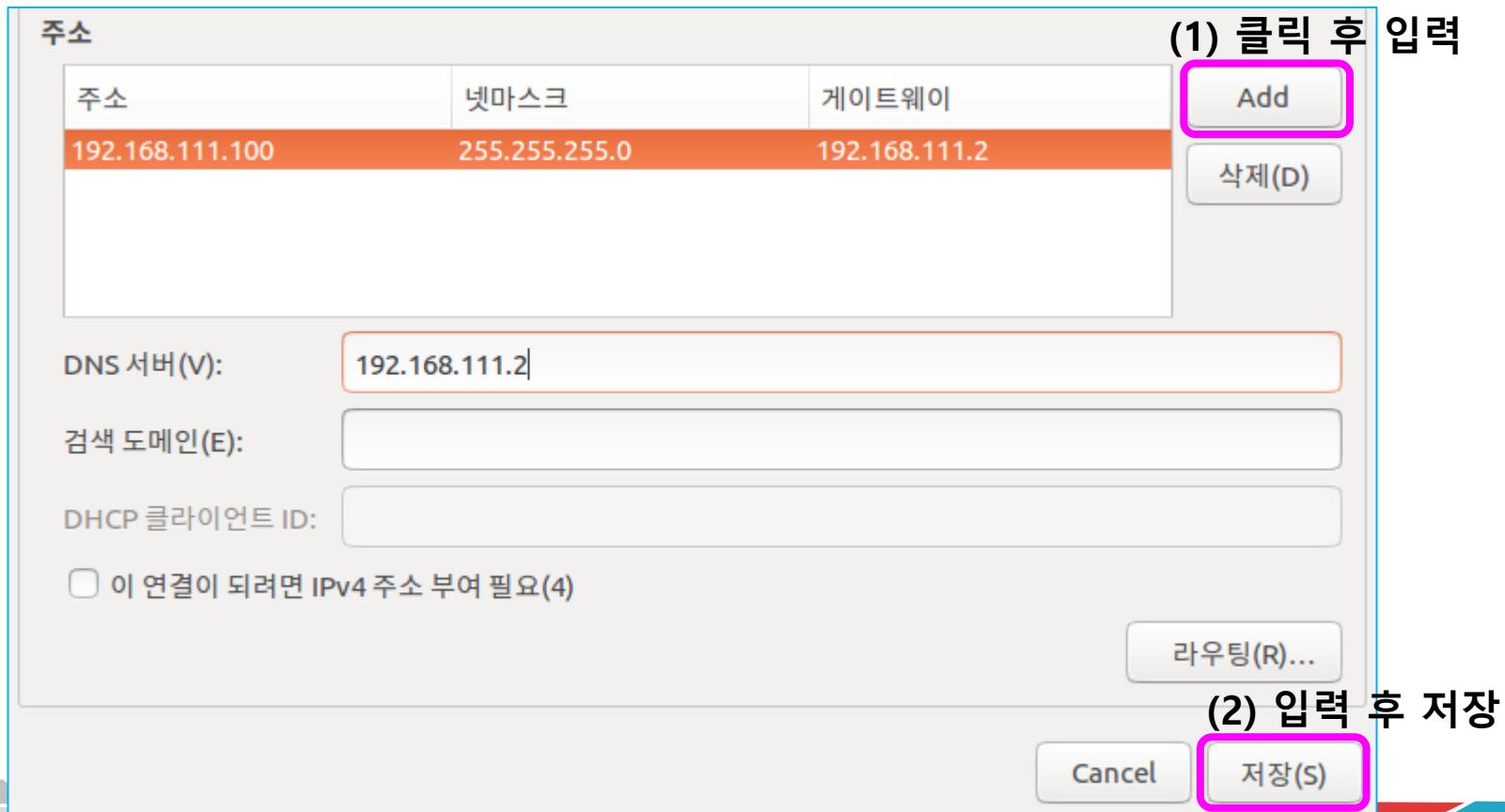
▶ Network 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

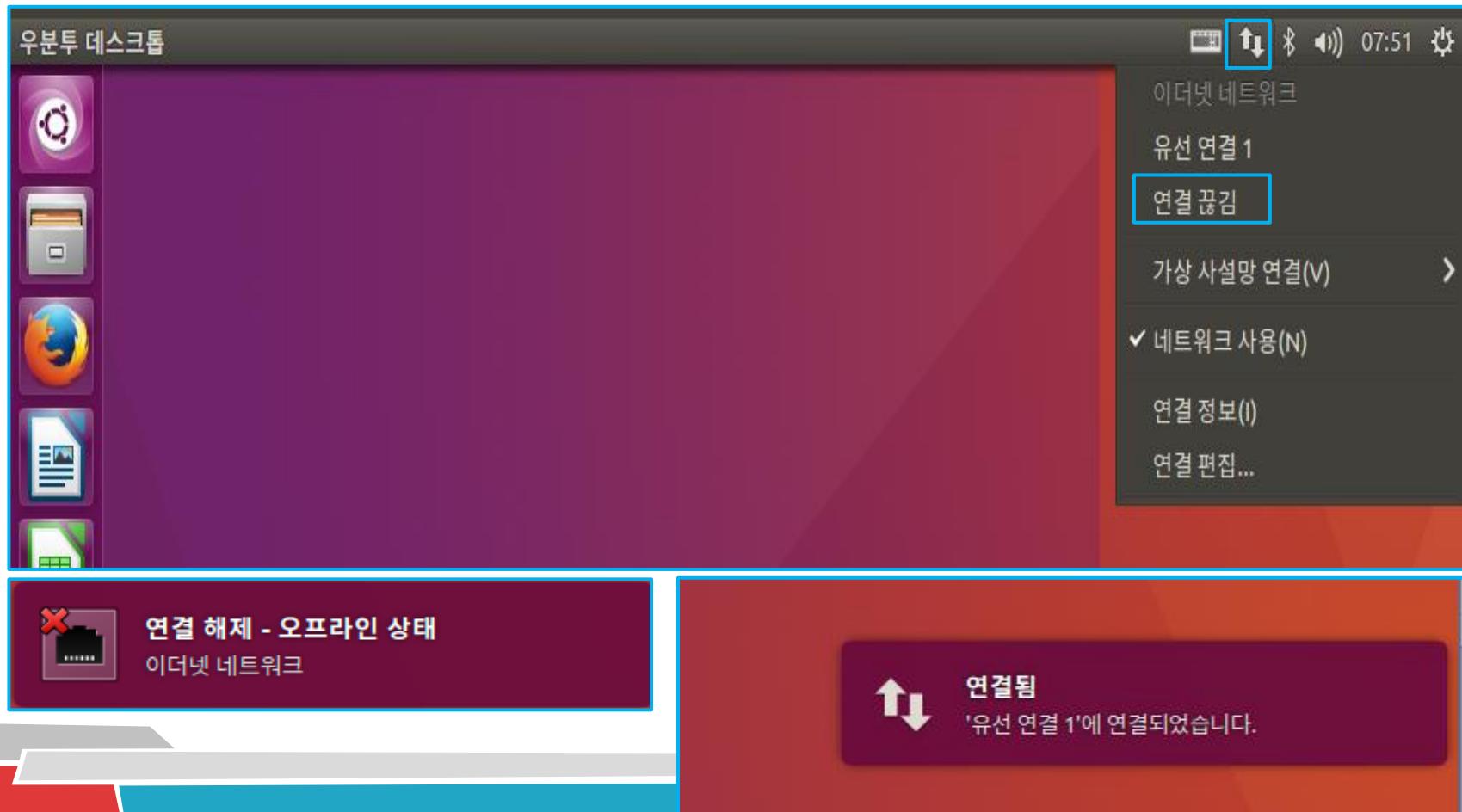
▶ Network 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

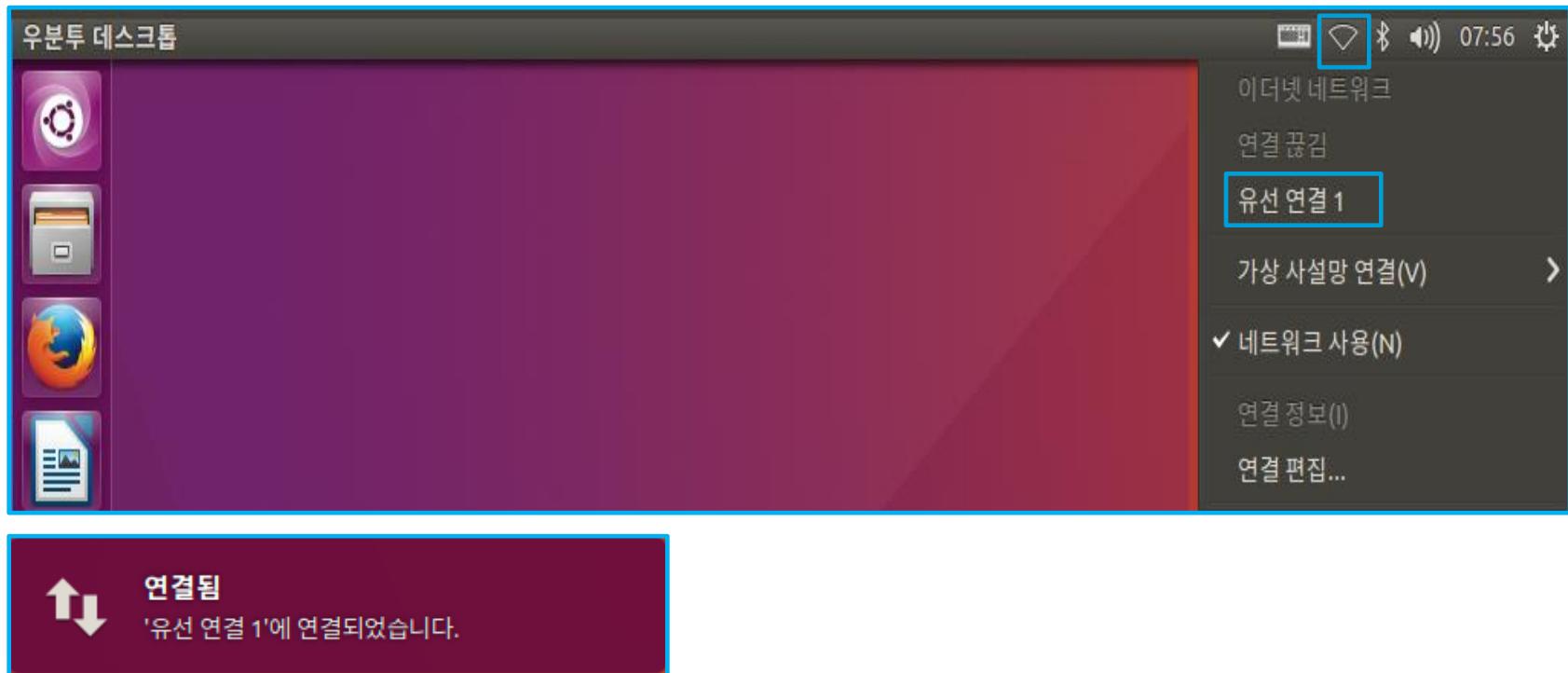
▶ Network 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ Network 설정

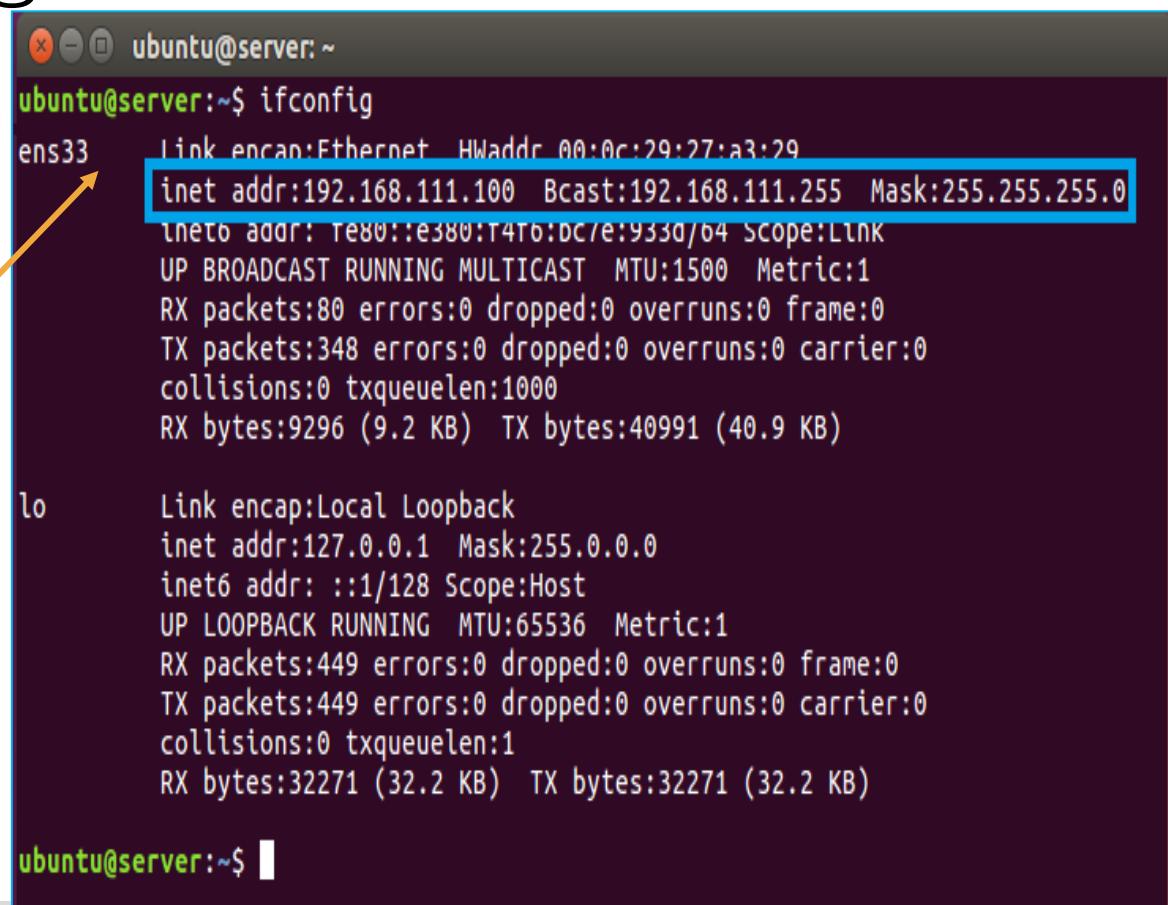


Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ Network 설정

ifconfig 명령어 입력



```
ubuntu@server:~$ ifconfig
ens33      Link encap:Ethernet HWaddr 00:0c:29:27:a3:29
            inet addr:192.168.111.100 Bcast:192.168.111.255 Mask:255.255.255.0
              inet6 addr: fe80::e380:t4ff:fe93:3a3/64 Scope:Link
                UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                RX packets:80 errors:0 dropped:0 overruns:0 frame:0
                TX packets:348 errors:0 dropped:0 overruns:0 carrier:0
                collisions:0 txqueuelen:1000
                RX bytes:9296 (9.2 KB) TX bytes:40991 (40.9 KB)

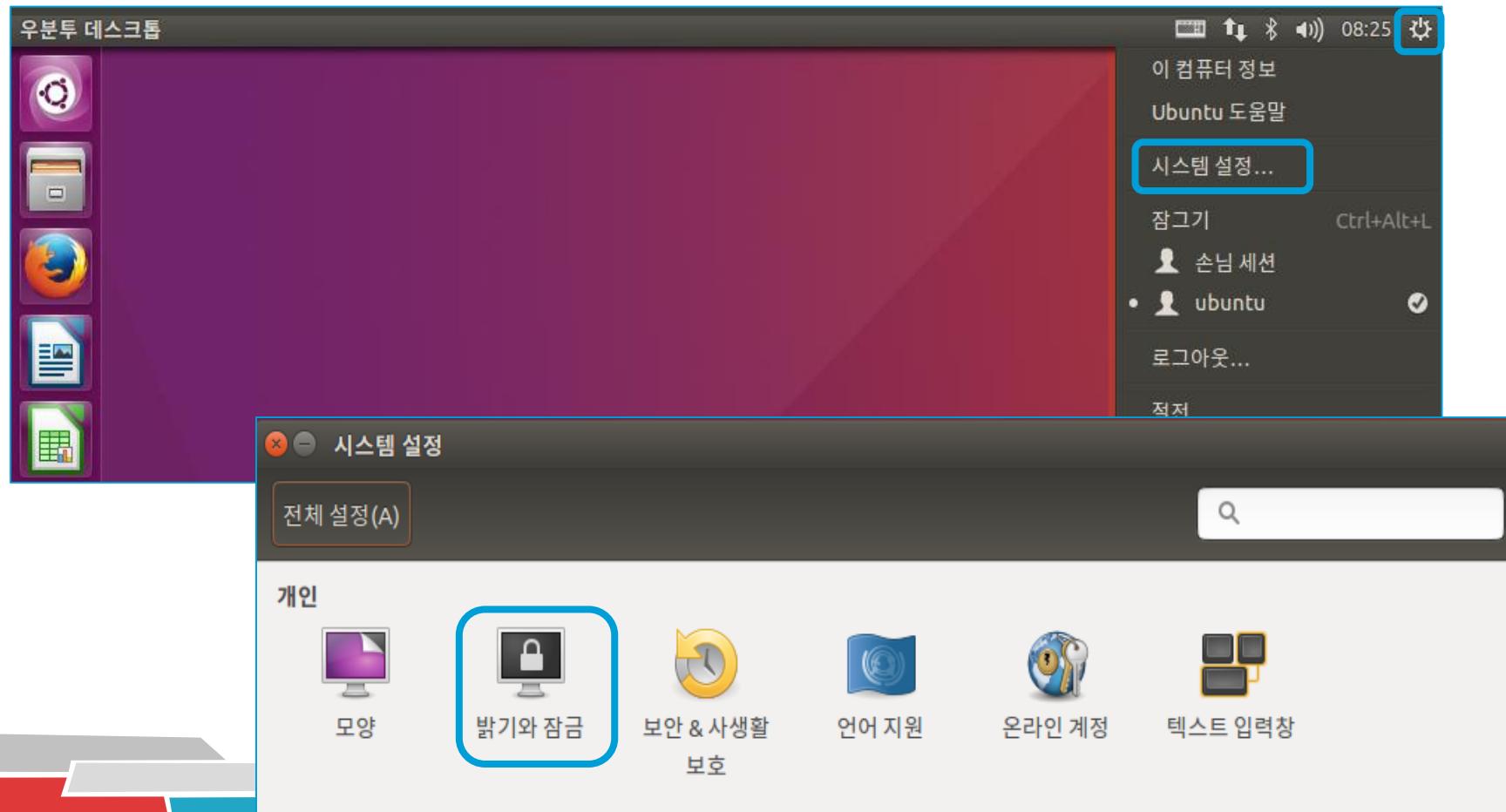
lo         Link encap:Local Loopback
            inet addr:127.0.0.1 Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING MTU:65536 Metric:1
              RX packets:449 errors:0 dropped:0 overruns:0 frame:0
              TX packets:449 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1
              RX bytes:32271 (32.2 KB) TX bytes:32271 (32.2 KB)

ubuntu@server:~$
```

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 화면 보호기 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

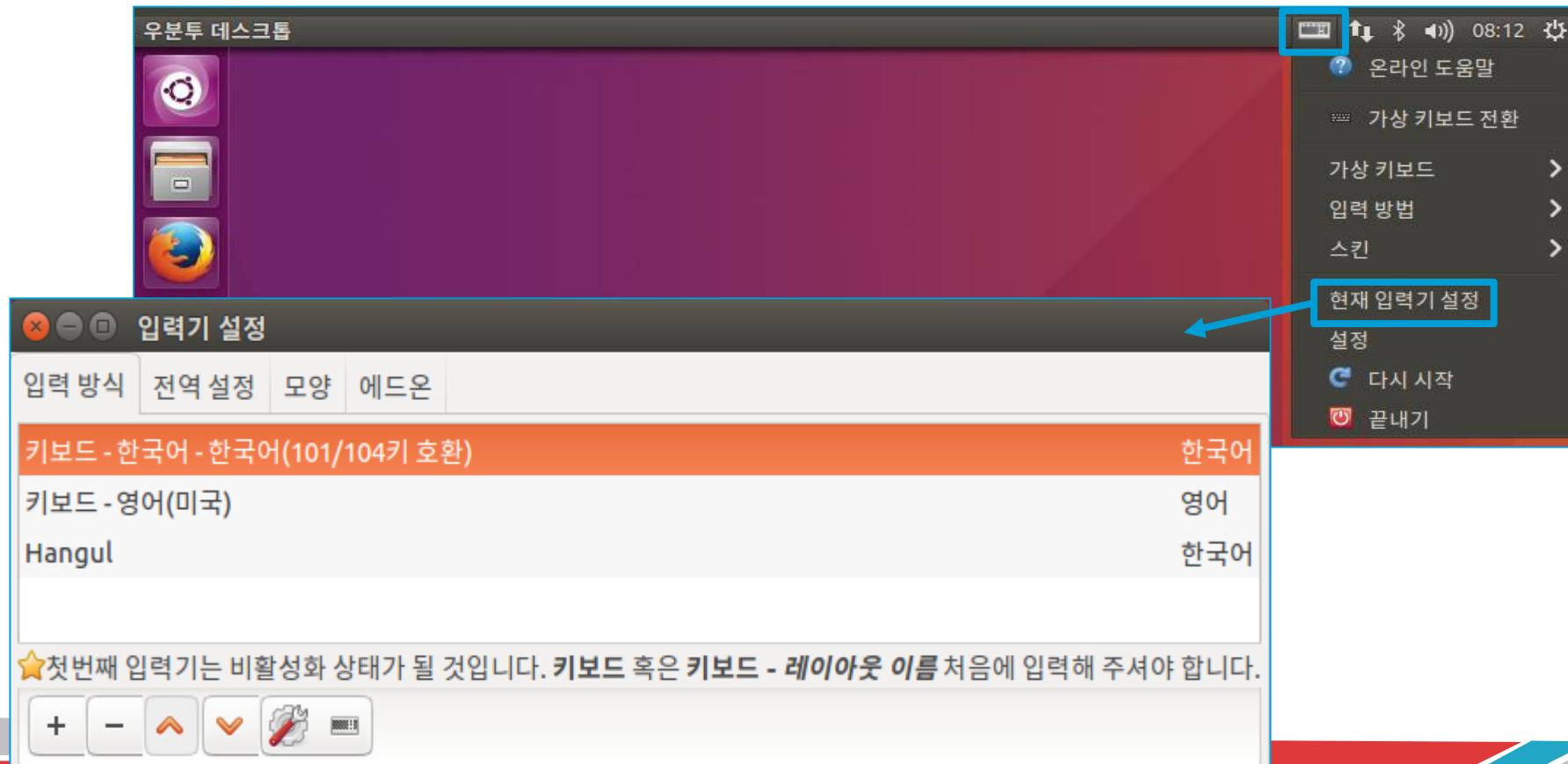
▶ 화면 보호기 설정



Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 한글 키보드 설정

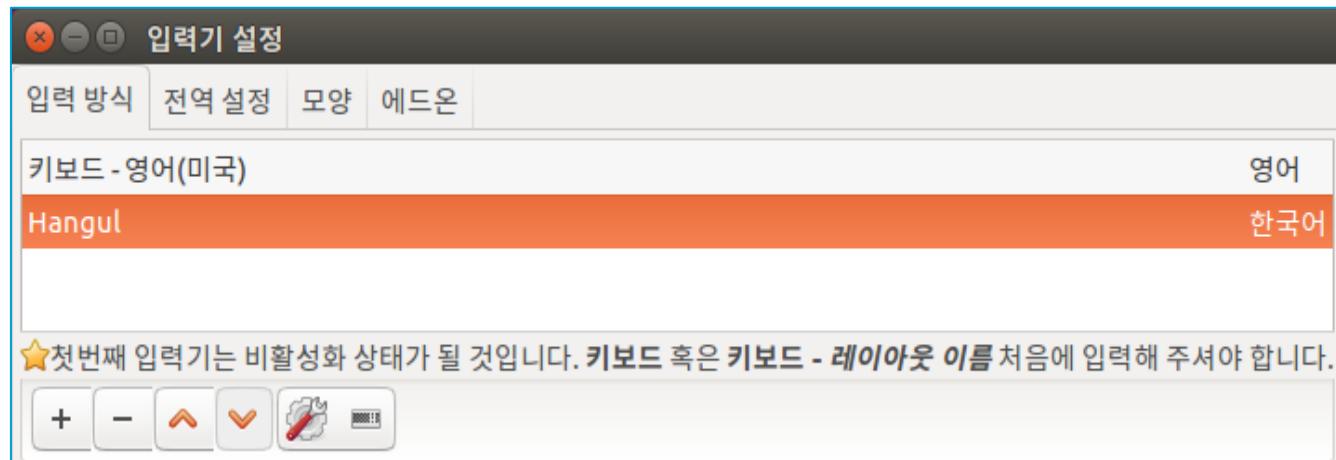


Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 한글 키보드 설정

- ▶ 두개의 한글 입력기 중 하나 제거
 - ▶ 키보드- 한국어-한국어(101/104키 호환) 제거 하기



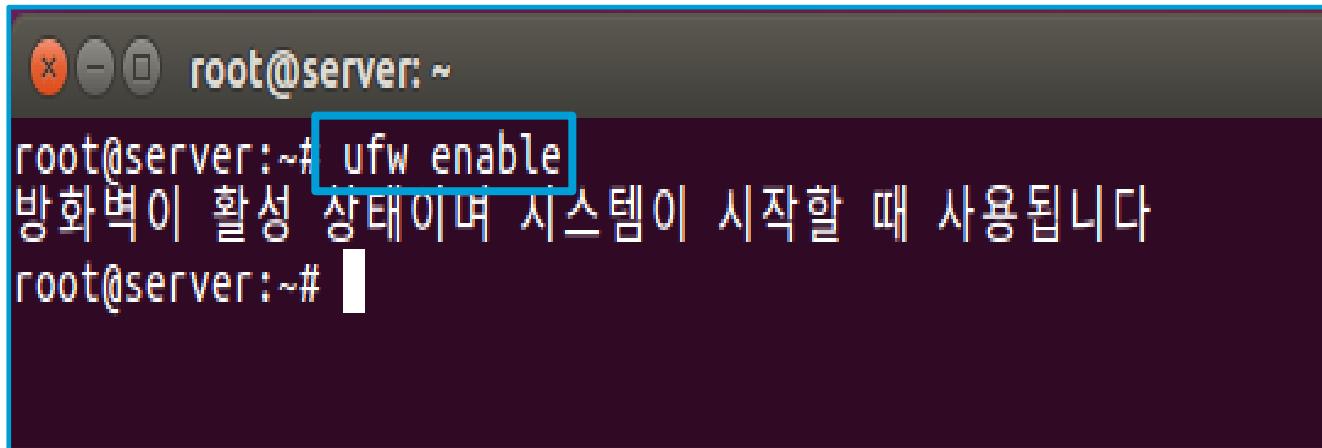
- ▶ 한/영 전환 → **window** + **Space**

▶ Ubuntu Linux 설정

▶ 방화벽 설정

▶ UFW(The Uncomplicated Firewall)

- ▶ Ubuntu Linux 배포판에 기본 설치 → 미 사용 설정
- ▶ root 계정으로만 제어 가능



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says 'root@server: ~'. Below that, a command is being typed: 'root@server:~# ufw enable'. A blue rectangular box highlights the command 'ufw enable'. After the command, there is a message in Korean: '방화벽이 활성 상태이며 시스템이 시작할 때 사용됩니다'. At the bottom, it says 'root@server:~# []' where the cursor is located.

▶ Ubuntu Linux 설정

▶ 에러 해결

- ▶ Ubuntu 14.04 이후 버전부터 부팅 시 발생하는 메시지
- ▶ DHCP가 없는 환경에서도 수동 설정 없이 자동으로 네트워킹을 할 수 있는 환경을 만들어주는 서비스



네트워크 서비스 탐색 사용하지 않기

현재 네트워크는 .local 도메인을 가지고 있어
Avahi 네트워크 서비스 탐색에 문제를 일으킬 수
있습니다. 서비스를 사용하지 않습니다.

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 에러 해결

```
root@Server64: ~
root@Server64:~# nano /usr/lib/avahi/avahi-daemon-check-dns.sh
root@Server64:~#
```

AVAHI_DAEMON_DETECT_LOCAL=0

```
root@Server64: ~
GNU nano 2.5.3    파일 : /usr/lib/avahi/avahi-daemon-check-dns.sh
PATH=/bin:/usr/bin:/sbin:/usr/sbin

RUNDIR="/var/run/avahi-daemon/"
DISABLE_TAG="$RUNDIR/disabled-for-unicast-local"
NS_CACHE="$RUNDIR/checked_nameservers"

AVAHI_DAEMON_DETECT_LOCAL=1 →
test -f /etc/default/avahi-daemon && . /etc/default/avahi-daemon

if [ "$AVAHI_DAEMON_DETECT_LOCAL" != "1" ]; then
    exit 0
fi
```

▶ Ubuntu Linux 설정

▶ 에러 해결

```
root@Server64: ~
root@Server64:~# nano /etc/default/avahi-daemon
root@Server64:~#
```

```
root@Server64: ~
GNU nano 2.5.3          파일: /etc/default/avahi-daemon          변경됨..
# 1 = Try to detect unicast dns servers that serve .local and disable avahi in
# that case, 0 = Don't try to detect .local unicast dns servers, can cause
# troubles on misconfigured networks
AVAHIDAEON_DETECT_LOCAL=0
```

Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 에러 해결하기

- ▶ 시스템 에러 해결
- ▶ Ubuntu 14.04 이후 버전부터 부팅 시 발생하는 메시지

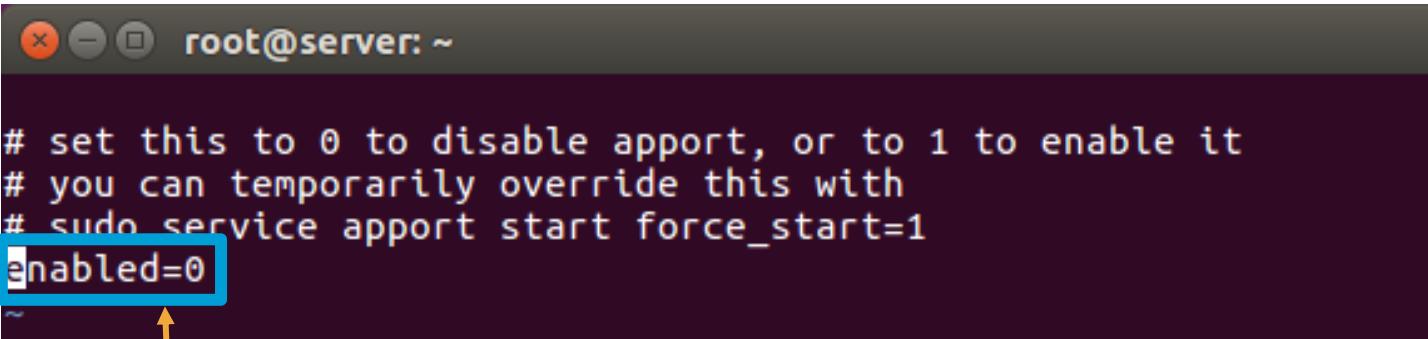


Ubuntu Linux 설치

▶ Ubuntu Linux 설정

▶ 에러 해결하기

```
~$ vi /etc/default/apport
```



```
root@server: ~

# set this to 0 to disable apport, or to 1 to enable it
# you can temporarily override this with
# sudo service apport start force_start=1
enabled=0
```

1을 0으로 값 설정

```
~$ reboot
```

PART II

Linux 명령어 및 구조 이해

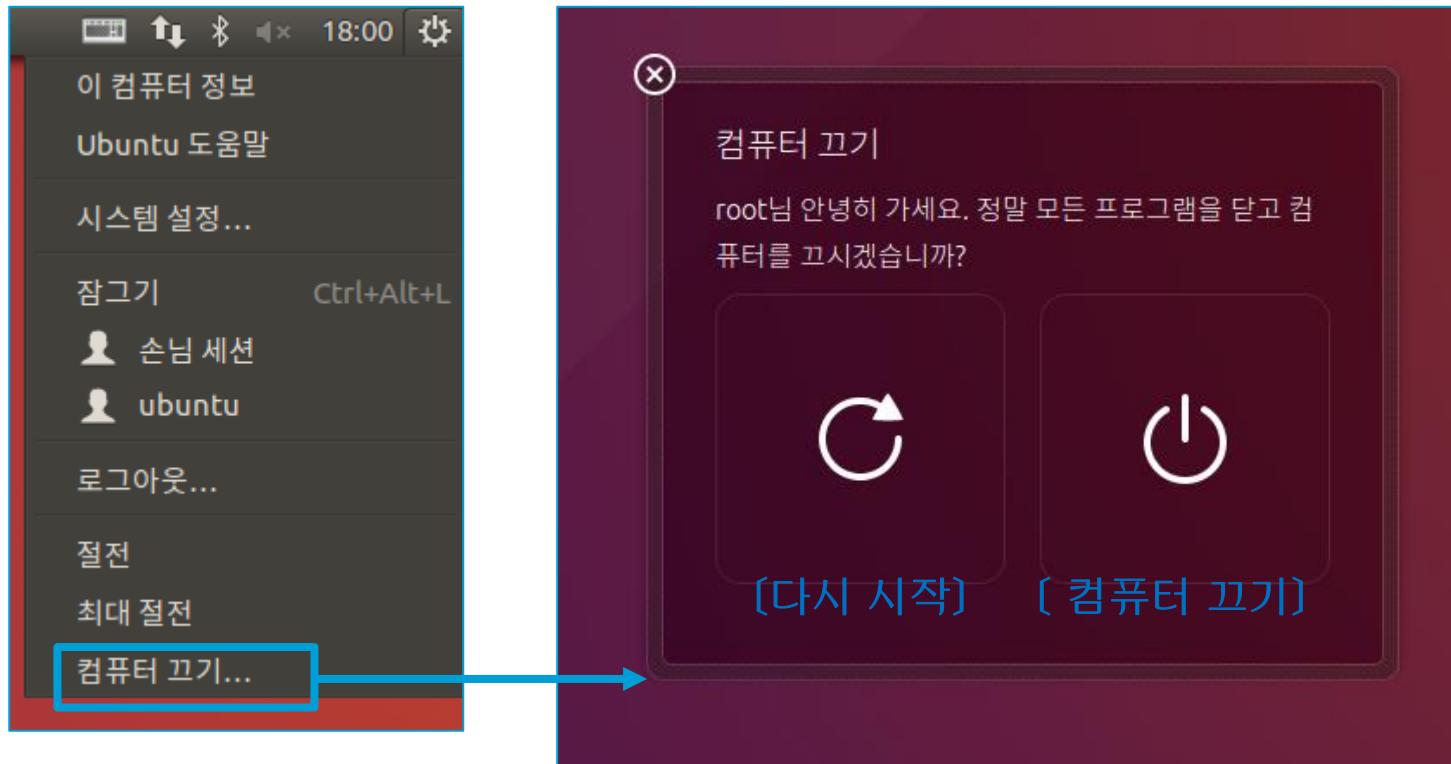
- 1 • Linux 사용 기본 사용 정보
- 2 • 기본 명령어
- 3 • 사용자 및 파일 속성 명령어
- 4 • 관리자 명령어
- 5 • 프로세스 & 데몬
- 6 • X-Window
- 7 • 쉘 스크립트 프로그래밍

CH01 – Linux 기본 사용 정보

Linux 기본 사용 정보

▶ 시작 & 종료

▶ 오른쪽 상단 아이콘 선택



▶ 시작 & 종료

▶ 터미널 실행 후 명령어 입력

- root@server :~# poweroff
- root@server :~# shutdown -P now
- root@server :~# halt -p
- root@server :~# init 0
- * P : poweroff 약자

shutdown -P +10 : 10분 후 종료
shutdown -r 22:00 : 10시에 재부팅
shutdown -c : 예약된 shutdown 취소
shutdown -k +15 : 현재 접속 사용자에게
15분 후 종료 메시지 전송,
종료는 되지 않음

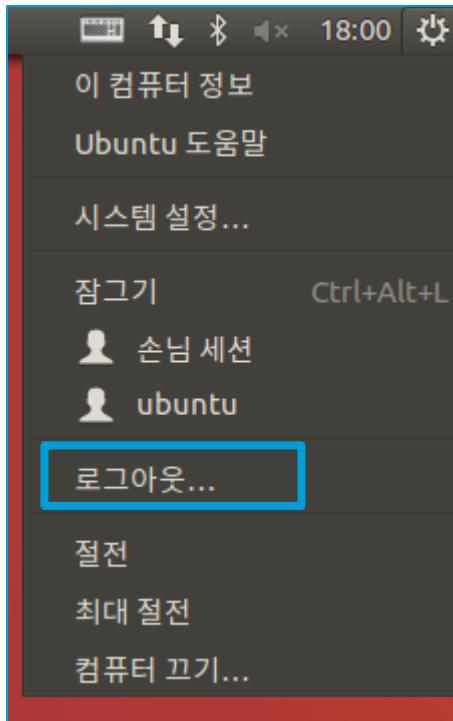
▶ 시스템 재부팅

▶ 터미널 실행 후 명령어 입력

- root@server :~# reboot
- root@server :~# shutdown -r now
- root@server :~# init 6

▶ 로그아웃

- ▶ 리눅스의 멀티 유저 환경에서 현재 사용자의 시스템 접속 종료를 의미



- root@server :~# logout
- root@server :~# exit

▶ 가상 콘솔

- ▶ 가상의 모니터와 같은 개념
- ▶ Ubuntu에는 7개의 가상 콘솔 존재
- ▶ 7개 콘솔 전환 키

 +  +  ~ 

- ▶ X-Window → F7

Linux 기본 사용 정보

▶ 런 레벨 (Run Level)

▶ 리눅스의 시스템 가동 방법

런레벨	모드	설명
0	Power Off	종료 모드
1	Rescuer	시스템 복구 모드, 단일 사용자 모드
2	Multi-User	사용하지 않음. 호환성을 위해 3과 동일한 것으로 취급
3	Multi-User	텍스트 모드의 다중 사용자 모드
4	Multi-User	사용하지 않음. 호환성을 위해 3과 동일한 것으로 취급
5	Graphical	그래픽 모드의 다중 사용자 모드
6	Reboot	

- init 명령어에서 사용

▶ 런레벨(RunLevel)

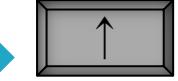
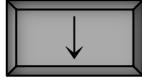
▶ 현재 시스템의 런레벨 확인

```
root@server:~# cd
root@server:~# pwd
/root
root@server:~# ls -l /lib/systemd/system/default.target
lrwxrwxrwx 1 root root 16  9월 29  2016 /lib/systemd/system/default.target -> graphical.target
root@server:~#
```

X-window로 부팅 → graphical mode
런레벨 5

▶ 자동 완성 기능

▶ 터미널 프로그램 지원 기능

- ▶ 파일 이름의 일부만 입력하고  키 눌러 나머지 파일 이름, 폴더 이름 자동 완성되는 기능
- ▶  ,  키를 사용해서 이전에 입력된 명령어 출력
- ▶ History 명령으로 기존 사용 명령어 제어
 - ▶ History : 기존 사용된 명령어 모두 출력
 - ▶ History -c : 기존 사용된 명령어 모두 삭제

▶ 마운트 (Mount)

- ▶ 하드디스크 파티션, CD/DVD, USB 메모리 등 **물리적인 장치를 특정한 위치 폴더에 연결시켜주는 과정**

명령어	설명
mount	현재 마운트 된 장치 확인 mount -t [파일시스템타입] [장치 파일][마운트 포인트]
umount	기존 마운트 해제 umount [장치파일]

Linux 기본 사용 정보

▶ 마운트 (Mount)

```
root@server:~# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=479236k,nr_inodes=119809,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=92776k,mode=755)
/dev/sda2 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
```

▶ 마운트 (Mount)

▶ CD/DVD 마운트 하기

```
root@server:~# umount /dev/cdrom
umount: /dev/cdrom: not mounted
root@server:~#
```

▶ 마운트 (Mount)

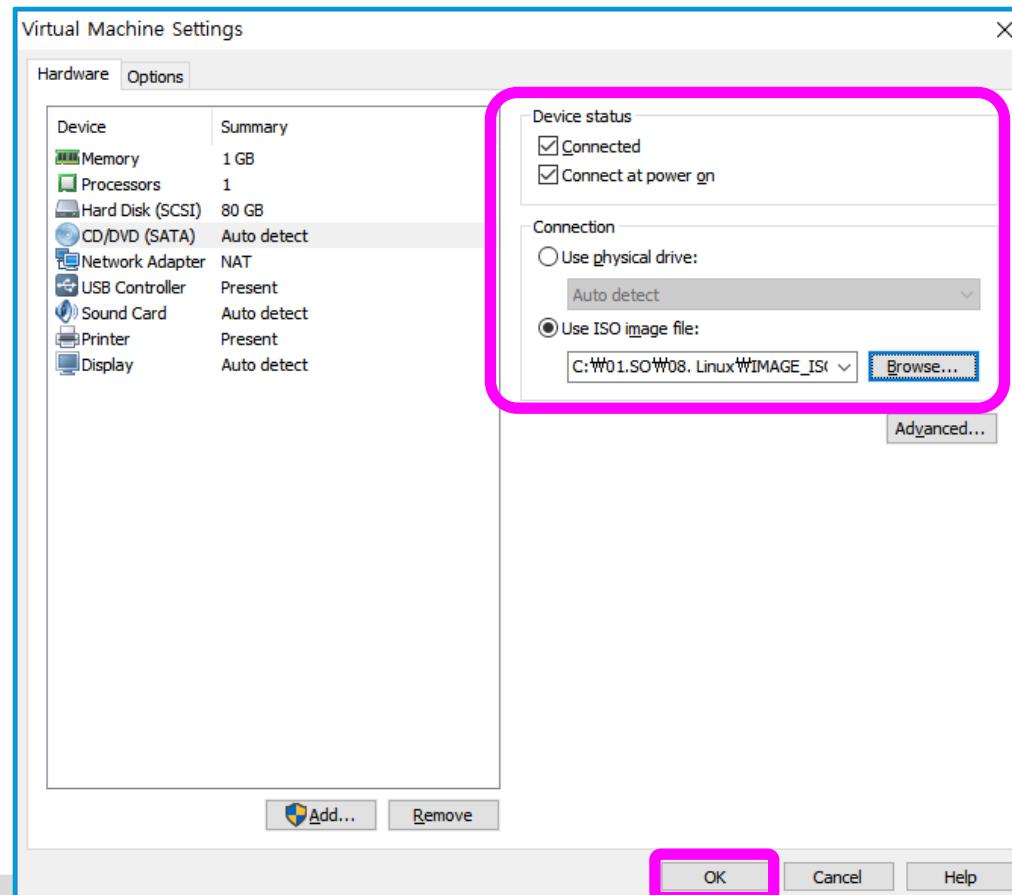
▶ CD/DVD 마운트 하기



Linux 기본 사용 정보

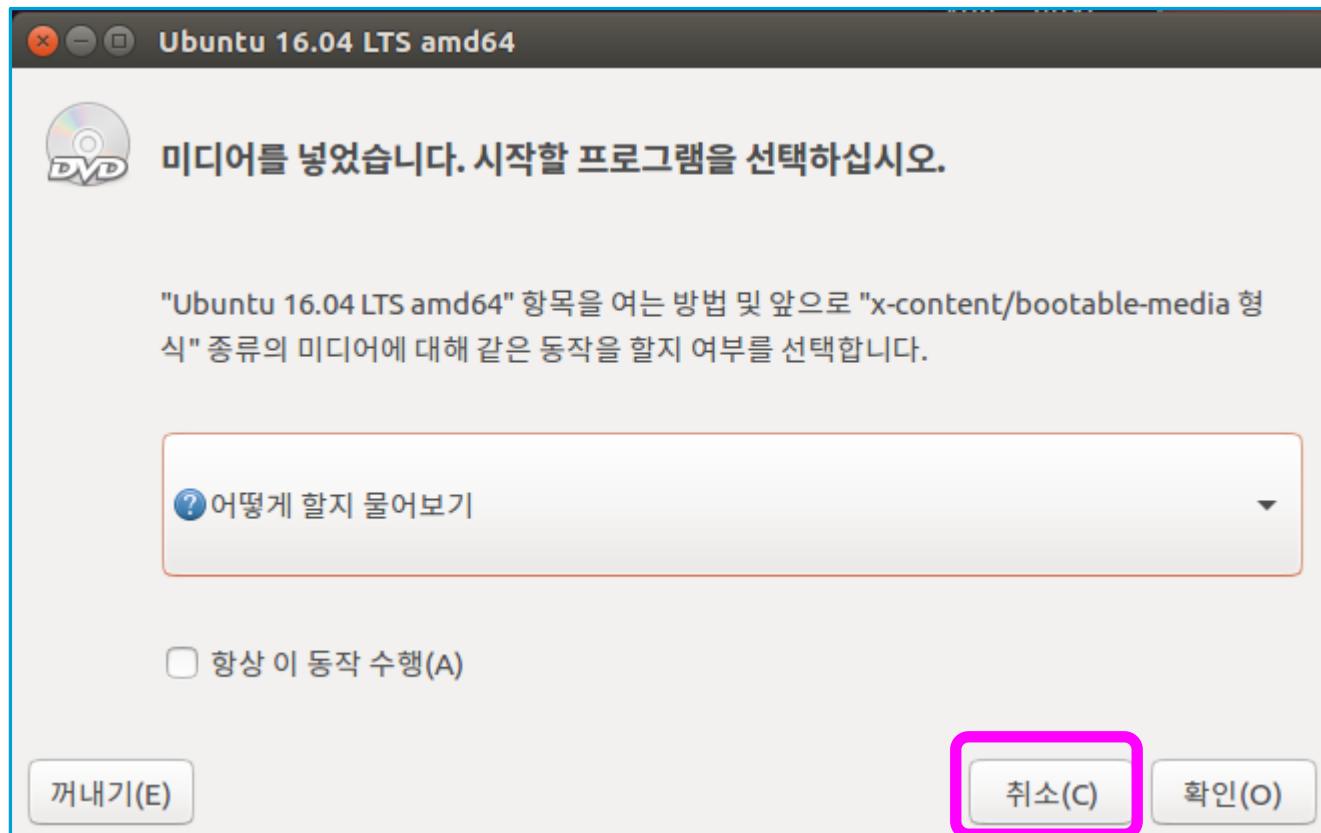
▶ 마운트 (Mount)

▶ CD/DVD 마운트 하기



▶ 마운트 (Mount)

▶ CD/DVD 마운트 하기



▶ 마운트 (Mount)

▶ CD/DVD 마운트 하기

```
root@server:~# umount /dev/cdrom
umount: /dev/cdrom: not mounted
root@server:~# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=479236k,nr_inodes=119809,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=99776k,mode=755)
/dev/sda2 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=99776k,mode=700)
avfsd-fuse on /run/user/0/avfs type fuse_avfsd-fuse (rw,nosuid,nodev,relatime,user_id=0,group_id=0)
/dev/sr0 on /media/root/Ubuntu 16.04 LTS amd64 type iso9660 (ro,nosuid,nodev,relatime,uid=0,gid=0,iocharset=utf8,
mode=0400,dmode=0500,uhelper=udisks2)
root@server:~#
```

자동으로 마운트 된 CD/DVD

▶ 마운트 (Mount)

▶ CD/DVD 마운트 하기

마운트 된 CD/DVD 디렉토리 이동

```
root@server:/media/root/Ubuntu 16.04 LTS amd64/casper
root@server:~# cd /media/root/Ubuntu\ 16.04\ LTS\ amd64/
root@server:/media/root/Ubuntu 16.04 LTS amd64# ls
EFI README.diskdefines boot casper dists install isolinux md5sum.txt pics pool preseed ubuntu
root@server:/media/root/Ubuntu 16.04 LTS amd64# cd casper/
root@server:/media/root/Ubuntu 16.04 LTS amd64/casper# ls
filesystem.manifest      filesystem.size      filesystem.squashfs.gpg  vmlinuz.efi
filesystem.manifest-remove filesystem.squashfs  initrd.lz
root@server:/media/root/Ubuntu 16.04 LTS amd64/casper#
```

우분투 설치 전체 파일

▶ 마운트 (Mount)

▶ CD/DVD 마운트 해제

```
root@server:~# umount /dev/cdrom
umount: /dev/cdrom: not mounted
root@server:~# █
```

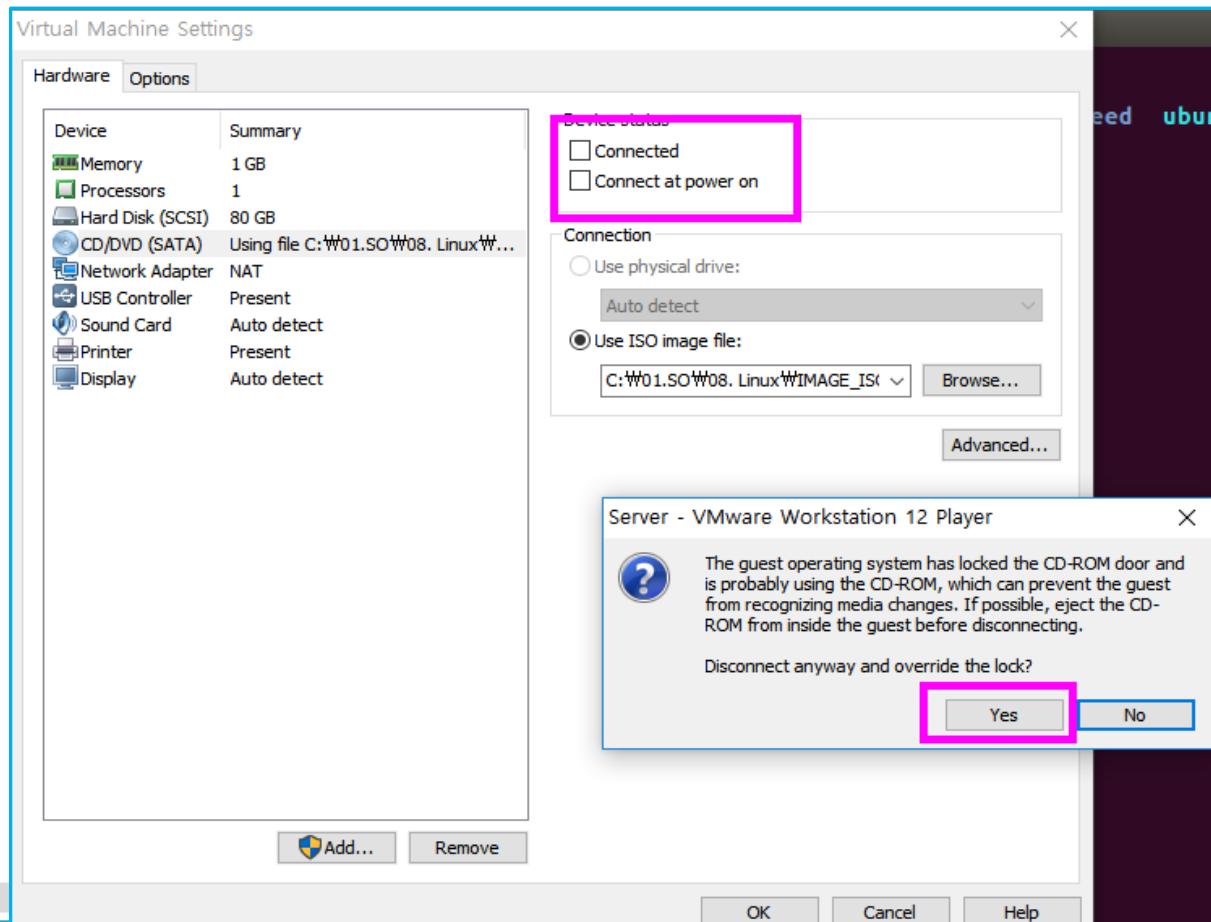
▶ 마운트 (Mount)

▶ CD/DVD 마운트 해제



▶ 마운트 (Mount)

▶ CD/DVD 마운트 해제



CH02 - Linux 기본 명령어

▶ ls 명령어

- ▶ list directory contents 약자
- ▶ 해당 디렉토리에 있는 파일 목록 나열

- ls
- ls /etc/systemd
- ls -a
- ls -l
- ls *.conf
- ls -l /etc/systemd/b*

- 현재 디렉터리 파일 목록 출력
- /etc/systemd 디렉터리 목록 출력
- 현재 디렉터리 숨김 파일 포함 목록 출력
- 현재 디렉터리 목록을 자세히 출력
- 확장자 conf인 목록 출력
- /etc/systemd 디렉터리에서 앞 글자가 b인 목록

Linux 기본 명령어

▶ ls 명령어

```
root@server: ~
root@server:~# ls -al
합계 136
drwxr-xr-x 2 root root 4096 9월  2 08:51 템플릿
root@server:~# ls -al
.
..
.rw-r--r-- 1 root root 2226 9월  2 19:41 .ICEauthority
.rw-r--r-- 1 root root   51 9월  2 19:41 .Xauthority
.rw-r--r-- 1 root root 1492 9월  3 11:26 .bash_history
.rw-r--r-- 1 root root 3106 10월 23 2015 .bashrc
drwxr-xr-x 12 root root 4096 9월  2 18:31 .cache
drwxr-xr-x 15 root root 4096 9월  2 08:52 .config
drwxr-xr-x  3 root root 4096 9월  2 08:39 .dbus
drwxr-xr-x  2 root root 4096 9월  2 17:30 .gconf
drwxr-xr-x  3 root root 4096 9월  2 19:41 .gnupg
drwxr-xr-x  3 root root 4096 9월  2 08:39 .local
drwxr-xr-x  4 root root 4096 9월  2 17:30 .mozilla
drwxr-xr-x  2 root root 4096 9월  2 07:03 .nano
drwxr-xr-x  2 root root 4096 9월  2 08:51 .presage
.rw-r--r--  1 root root   149 9월  2 08:49 .profile
.rw-r--r--  1 root root 12288 9월  3 07:47 .swp
.rw-r--r--  1 root root    179 9월  3 12:16 .xsession-errors
.rw-r--r--  1 root root   1272 9월  2 19:38 .xsession-errors.old
drwxr-xr-x  2 root root 4096 9월  3 10:03 Test
.rw-r--r--  1 root root     58 9월  3 08:52 myTest.txt
.rw-r--r--  1 root root     18 9월  3 11:31 myTest_02.txt
.rw-r--r--  1 root root     96 9월  3 11:23 test-01.txt
.rw-r--r--  1 root root     15 9월  3 07:33 test.txt
.rw-r--r--  1 root root      0 9월  3 09:20 test_00.txt
drwxr-xr-x  2 root root 4096 9월  2 08:51 공개
```

▶ cd 명령어

- ▶ change directory 약자
- ▶ 디렉토리로 이동

- cd
- cd ~ubuntu
- cd ..
- cd /etc/systemd
- cd ../etc/systemd

- 현재 사용자의 홈 디렉터리 이동
- Ubuntu 사용자의 홈 디렉터리로 이동
- 바로 상위 디렉터리로 이동
- /etc/systemd 디렉터리로 이동
- 현재 상위 디렉터리로 이동 후 /etc/systemd 디렉터리로 이동

▶ **pwd** 명령어

- ▶ Print Working Directory 약자
- ▶ 현재 디렉토리의 전체 경로 출력

- **pwd**

- 현재 작업 디렉터리 경로 출력

[상대 경로]

- . -> 현재 디렉터리
- .. -> 현재 디렉터리의 바로 상위 디렉터리
- / -> 최상위 디렉터리

▶ rm 명령어

- ▶ Remove 약자
- ▶ 파일이나 디렉터리 삭제, 삭제 권한 필요!!
- ▶ 삭제 후 복구 불가함으로 주의!!!

- rm abc.txt
- rm -i abc.txt
- rm -f abc.txt
- rm -r abc

- 해당 파일 삭제, 내부적으로 rm -f 연결
- 삭제 확인 메시지 출력
- 해당 파일 바로 삭제
- abc 디렉터리와 하위 디렉터리까지 모두 삭제

▶ cp 명령어

- ▶ Copy 약자
- ▶ 파일이나 디렉터리 복사
- ▶ 파일에 대한 읽기 권한 필요

- cp abc.txt bbb.txt
- cp -r abc bbb

- abc.txt 파일을 bbb.txt로 복사
- abc 디렉터리를 bbb 디렉터리로 복사

▶ touch 명령어

- ▶ 크기가 0인 파일 생성
- ▶ 존재하는 파일일 경우 파일 최종 수정 시간 변경

- touch abc.txt

- 빈 abc.txt 파일 생성 또는 최종 수정 시간 변경

▶ mv 명령어

- ▶ Move 약자
- ▶ 파일이나 디렉터리 이름 변경 또는 다른 디렉터리 이동

- mv abc.txt /etc/systemd/
- mv aaa bbb ccc ddd
- mv abc.txt www.txt

- abc.txt를 /etc/systemd/ 디렉터리로 이동
- aaa, bbb, ccc 파일을 /ddd 디렉터리로 이동
- abc.txt를 [www.txt](#)로 변경

▶ mkdir 명령어

- ▶ Make Directory 약자
- ▶ 새로운 디렉터리 생성
- ▶ 디렉터리 생성한 사용자의 소유가 됨

- `mkdir abc`
- `mkdir -p ../SO/aaa`

- 현재 디렉터리에 abc 디렉터리 생성
- aaa 디렉터리의 부모인 SO 디렉터리가 없다면 부모 디렉터리 SO도 생성

▶ rmdir 명령어

- ▶ Remove Directory 약자
- ▶ 디렉터리 삭제, 디렉터리에 대한 삭제 권한 필요

- `rmdir abc`
- `rmdir -r abc`

- abc 디렉터리 삭제
- abc 디렉터리의 하위 디렉터리까지 모두 삭제

▶ cat 명령어

- ▶ concatenate 약자
- ▶ 파일의 내용을 화면에 보여줌

- cat a.txt b.txt

- a.txt와 b.txt 파일 내용을 화면에 연결해서
보여 줌

▶ head, tail 명령어

- ▶ 텍스트 형식 파일의 앞 10행, 마지막 10행 출력

```
head /etc/systemd/bootchart.conf  
head -3 /etc/systemd/bootchart.conf  
tail -5 /etc/systemd/bootchart.conf
```

- bootchart.conf 파일 처음 10행 출력
- bootchart.conf 파일 처음 3행 출력
- bootchart.conf 파일 마지막 5행 출력

▶ more 명령어

- ▶ 텍스트 파일을 페이지 단위로 화면 출력
- ▶ **Space Bar** 누르면 다음 페이지로 이동
- ▶ **B** 누르면 앞 페이지 이동, **Q** 종료

- more /etc/systemd/system.conf
- more +10 /etc/systemd/system.conf

- system.conf 파일 출력
- system.conf 파일 10행부터 출력

▶ less 명령어

- ▶ 텍스트 파일을 페이지 단위로 화면 출력
- ▶ more 명령어 보다 강력
- ▶ `Page Up` , `Page Down` 키 추가 사용 가능

- `less /etc/passwd`
- `less +10 /etc/passwd`

- `passwd` 파일 출력
- `passwd` 파일 10행부터 출력

▶ file 명령어

▶ 해당 파일이 어떤 파일인지 정보 표시

- file /etc/passwd
- file /bin/gzip

- 텍스트 파일로 아스키 파일로 표시
- 실행 파일로 ‘ELF 64-bit LSB executable’ 표시

▶ clear 명령어

▶ 현재 터미널 화면 지우기

- clear

- 화면 지우기

CH03 – 사용자 및 파일 속성 명령어

▶ 사용자와 그룹

▶ 다중 사용자 시스템

- ▶ 1대 리눅스 시스템에 여러 명 동시 접속 가능
- ▶ 설치 완료 후 기본으로 root 슈퍼 유저 존재

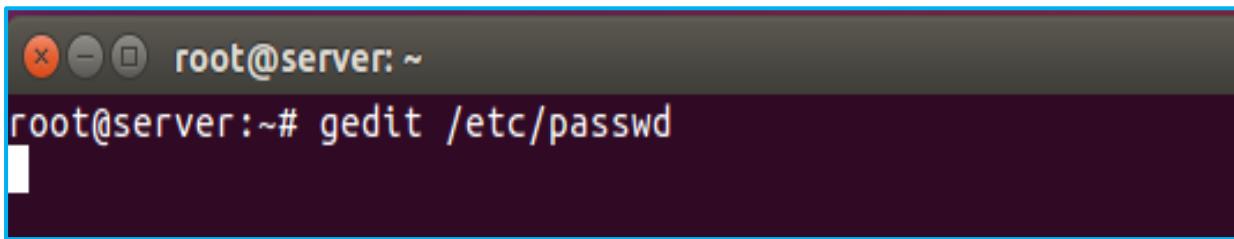
▶ root 슈퍼 유저

- ▶ 시스템의 모든 작업 수행 권한
- ▶ 사용자 생성 권한

▶ 사용자와 그룹

▶ 다중 사용자 시스템

▶ 존재 사용자 확인



A screenshot of a terminal window with a dark background and light-colored text. The window title bar shows 'root@server: ~'. The command 'root@server:~# gedit /etc/passwd' is typed into the terminal. The window has a standard Linux-style title bar with close, minimize, and maximize buttons.

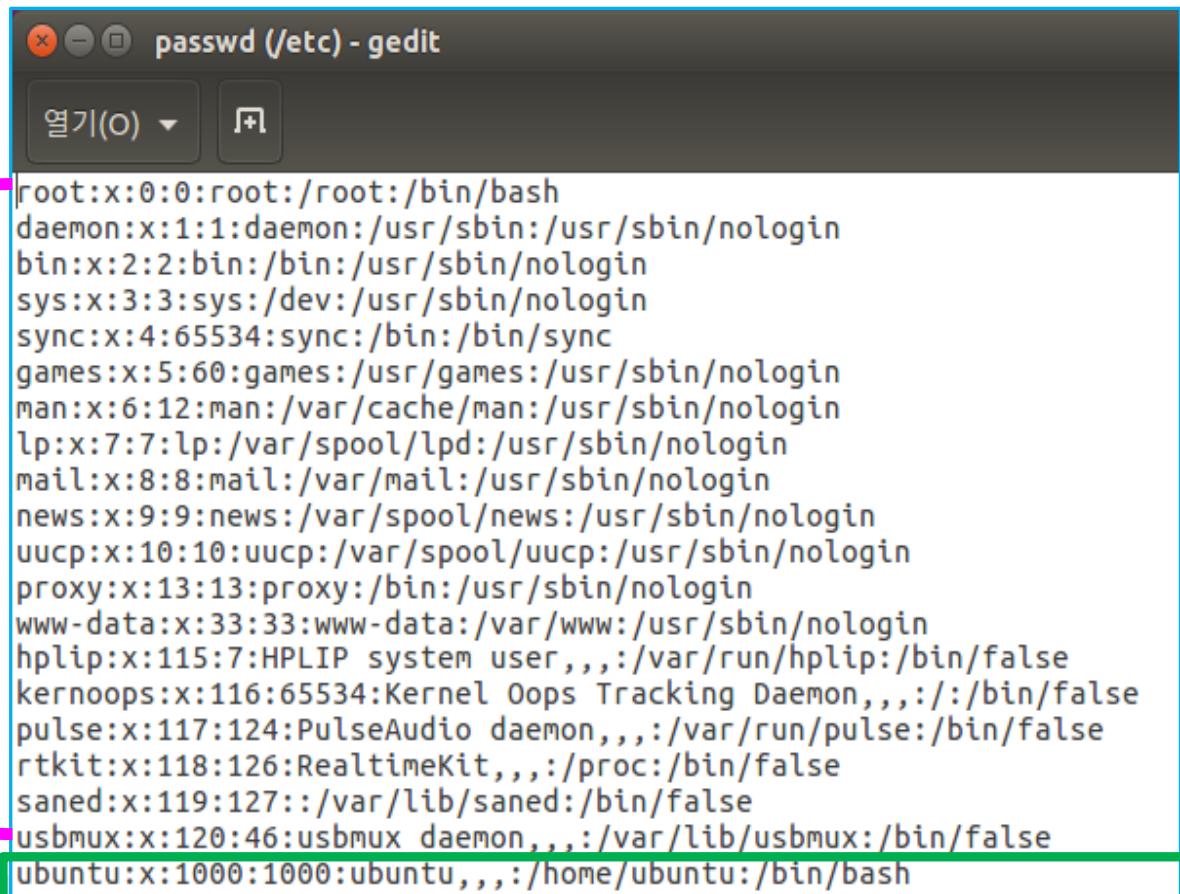
사용자 및 파일 속성 명령어

▶ 사용자와 그룹

▶ 다중 사용자 시스템

리눅스
기본존재
표준 사용자

추가한
사용자



```
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/bin/false
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
ubuntu:x:1000:1000:ubuntu,,,:/home/ubuntu:/bin/bash
```

사용자 및 파일 속성 명령어

▶ 사용자와 그룹

▶ 다중 사용자 시스템

▶ 사용자 정보 항목

사용자 이름 : 암호:사용자 ID:사용자 소속 그룹 ID : 추가 정보 : 홈 디렉터리 : 기본 쉘

```
root:x:0:0:root:/root:/bin/bash
```

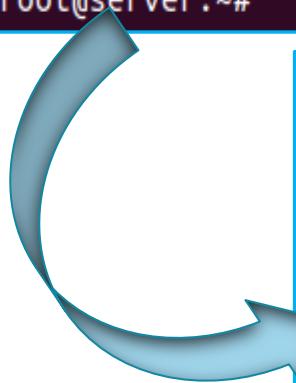
```
ubuntu:x:1000:1000:ubuntu,,,:/home/ubuntu:/bin/bash
```

사용자 및 파일 속성 명령어

▶ 사용자와 그룹

▶ 다중 사용자 비밀 번호 확인

```
root@server:~# gedit /etc/passwd  
root@server:~# gedit /etc/shadow
```



```
shadow (/etc) - gedit  
열기(O) ▾   
  
root:$6$NdQksiiF$JKg/MtldZqhfa]Edz/uDM1Pj4AMk0056E.6uFkWkTD8N02Pi8/  
vTIgfqr6Cv1SKI.G.KBTqudR2.YMKa2t/9511:17410:0:99999:7:::  
daemon:*:16911:0:99999:7:::  
bin:*:16911:0:99999:7:::  
sys:*:16911:0:99999:7:::  
sync:*:16911:0:99999:7:::  
games:*:16911:0:99999:7:::  
man:*:16911:0:99999:7:::  
lp:*:16911:0:99999:7:::  
mail:*:16911:0:99999:7:::  
news:*:16911:0:99999:7:::  
saned:*:16911:0:99999:7:::  
usbmux:*:16911:0:99999:7:::  
ubuntu:$6$.WP9hX/B$0zQa02xZmwlr4JvChl/E72CGMPpkWZTLH80ZE75cINxmQ1r8o/BiByLlbU3iv9r0PXii/  
gBulgx747u.4qUu:17409:0:99999:7:::
```

사용자 및 파일 속성 명령어

▶ 사용자와 그룹

▶ 다중 사용자 시스템

▶ 그룹 ID 확인

```
root@server: ~
root@server:~# gedit /etc/passwd
root@server:~# gedit /etc/group
root@server:~#
```

그룹 이름 : 암호:그룹 ID:보조 그룹 사용자

```
group (/etc) - gedit
열기(O) ▾  [+]

root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,ubuntu
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
colord:x:123:
pulse:x:124:
pulse-access:x:125:
rtkit:x:126:
saned:x:127:
ubuntu:x:1000:
sambashare:x:128:ubuntu
```

▶ adduser 명령어

- ▶ 새로운 사용자 추가
- ▶ 비밀번호, 사용자 정보, 쉘 입력한 계정 생성
- ▶ 홈 디렉터리 생성
- ▶ /etc/passwd, /etc/shadow, /etc/group 행 추가

```
- adduser user1  
- adduser --uid 1111 user2  
- adduser --gid 1000 user3  
- adduser --home /USER4 user4  
- adduser -shell /bin/csh user5
```

-user1 이름의 사용자 생성
-사용자 ID 1111 user2 사용자 생성
-그룹ID 1000 user3 사용자 생성
-/USER4 홈 디렉터리의 user4 사용자 생성
-/bin/csh 기본 셸의 user5 사용자 생성

사용자 및 파일 속성 명령어

▶ adduser 명령어

```
root@server:~# adduser user1
'user1' 사용자를 추가 중...
새 그룹 'user1' (1001) 추가 ...
새 사용자 'user1' (1001) 을(를) 그룹 'user1' (으)로 추가 ...
'/home/user1' 홈 디렉터리를 생성하는 중...
'/etc/skel'에서 파일들을 복사하는 중...
새 UNIX 암호 입력:
새 UNIX 암호 재입력:
passwd: 암호를 성공적으로 업데이트했습니다
user1의 사용자의 정보를 바꿉니다
새로운 값을 넣거나, 기본값을 원하시면 엔터를 치세요
이름 []: KSH
방 번호 []:
직장 전화번호 []:
집 전화번호 []:
기타 []:
정보가 올바릅니까? [Y/n] Y
```

사용자 및 파일 속성 명령어

▶ passwd 명령어

- ▶ 새로 생성한 사용자의 경우 비밀번호 지정
- ▶ 비밀번호가 있는 경우 변경

- passwd user1

-user1 사용자의 비밀번호 지정 또는 변경

▶ usermod 명령어

▶ 사용자 속성 변경

▶ /etc/passwd, /etc/shadow, /etc/group 파일에 변경

- **usermod -s** /bin/csh user1
- **usermod -g** ubuntu user1
- **usermod -d** /home2 -m user1

-user1 사용자 기본 쉘 변경
-user1 사용자 보조그룹에 ubuntu 추가
-user1 홈 디렉터리 변경

사용자 및 파일 속성 명령어

▶ userdel 명령어

- ▶ 사용자 삭제
- ▶ /etc/passwd, /etc/shadow, /etc/group 파일에 변경

- **userdel** user1
- **userdel** -r user2

-user1 사용자 삭제, 단 홈 디렉터리 삭제 안됨
-user2 사용자, 홈 디렉터리까지 삭제 됨

▶ chage 명령어

▶ 사용자 암호를 주기적으로 변경되도록 설정

- chage -l user1
- chage -m 2 user1
- chage -M 30 user1
- chage -E 2023/12/12 user1
- chage -W 10 user1

- user1 사용자 설정 정보 확인
- user1의 최소 2일 암호 사용 설정
- user1의 최대 30일 암호 사용 일자 설정
- user1의 암호 만료 일자 설정
- user1 암호 만료 10일 전 경고 메시지 표시 (기본 7일)

사용자 및 파일 속성 명령어

▶ groups 명령어

▶ 사용자 소속된 그룹 정보

- groups
- groups user1

- 현재 사용자가 소속된 그룹 정보 출력
- user1 소속된 그룹 정보 출력

```
root@server:/home# groups root
root : root
root@server:/home# groups user1
user1 : user1 ubuntu
root@server:/home# groups ubuntu
ubuntu : ubuntu adm cdrom sudo dip plugdev lpadmin sambashare
```

사용자 및 파일 속성 명령어

▶ groupadd 명령어

▶ 새로운 사용자 그룹 생성

```
-groupadd newgroup1
```

```
-groupadd -gid 2222 newgroup2
```

-newgroup1 그룹 생성

-그룹 ID 2222로 지정된 newgroup2 그룹생성

▶ groupmod 명령어

▶ 그룹 속성 변경

```
-groupmod -new-name shk1 newgroup1
```

-newgroup1 그룹을 shk1로 이름 변경

사용자 및 파일 속성 명령어

▶ groupdel 명령어

▶ 그룹 삭제

```
-groupdel newgroup1
```

-newgroup1 그룹 삭제
단, 주요 그룹으로 지정된 사용자가 없어야 함

▶ gpasswd 명령어

▶ 그룹 암호 설정 또는 그룹 관리

```
-gpasswd newgroup1  
-gpasswd -A user1 newgroup1  
-gpasswd -a user1 newgroup1  
-gpasswd -d user1 newgroup1
```

-newgroup1 그룹 암호 지정
-user1을 newgroup1 그룹 관리자 지정
-user1을 newgroup1 그룹 사용자로 추가
-user1을 newgroup1 그룹 사용자에서 제거

사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 소유와 허가권

- ▶ 각 파일과 디렉터리마다 소유권과 허가권 속성 존재
- ▶ ls –l 명령어로 확인 가능

```
-rw-r--r-- 1 root root 15 9월 3 07:33 test.txt
-rw-r--r-- 1 root root 0 9월 3 09:20 test_00.txt
drwxr-xr-x 2 root root 4096 9월 2 08:51 공개
drwxr-xr-x 2 root root 4096 9월 2 08:51 다운로드
drwxr-xr-x 2 root root 4096 9월 2 08:51 문서
crw----- 1 root root 10, 175 9월 2 19:41 agpgart
crw----- 1 root root 10, 235 9월 2 19:41 autofs
drwxr-xr-x 2 root root 280 9월 2 19:41 block
drwxr-xr-x 2 root root 80 9월 2 19:41 bsg
crw----- 1 root root 10, 234 9월 2 19:41 btrfs-control
drwxr-xr-x 3 root root 60 9월 2 19:41 bus
lrwxrwxrwx 1 root root 3 9월 3 12:31 cdrom -> sr0
lrwxrwxrwx 1 root root 3 9월 3 12:31 cdrw -> sr0
drwxr-xr-x 2 root root 3640 9월 2 19:41 char
crw----- 1 root root 5, 1 9월 2 19:41 console
lrwxrwxrwx 1 root root 11 9월 2 19:41 core -> /proc/kcore
drwxr-xr-x 2 root root 60 9월 2 19:41 cpu
crw----- 1 root root 10, 59 9월 2 19:41 cpu_dma_latency
crw----- 1 root root 10, 203 9월 2 19:41 cuse
drwxr-xr-x 5 root root 100 9월 3 12:31 disk
```

사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 소유와 허가권

파일유형 : d(디렉터리), -(일반파일), b(블록 디바이스), c(문자 디바이스), l(링크)

```
-rwxr-xr-x 1 root root 0 9월 3 09:20 test_00.txt  
drwxr-xr-x 2 root root 4096 9월 2 08:51 공개
```

파일접근허가권(permission) : 소유자 허가권 + 그룹 허가권 + 그 외 사용자 허가권
r(read), w(write), x(execute)

소유자(User)			그룹(Group)			그 외 사용자(Other)		
r	w	e	r	w	e	r	w	e
4	2	1	4	2	1	4	2	1

▶ 디렉터리 ➔ 해당 디렉터리 이동을 위한 x 허가권 필요

사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 소유와 허가권

▶ 파일 허가권 chmod 명령어

▶ 파일이나 디렉터리에 부여된 권한 변경

[숫자 표기법]

-chmod 777 a.txt
-chmod 755 abc

-a.txt에 소유자, 그룹, 외 사용자 읽기/쓰기/실행 권한 부여
-abc 디렉터리에 소유자는 읽기/쓰기/실행, 그룹 및 외 사용자는 읽기/실행 권한만 부여

[기호문자열 표기법]

-chmod u+x a.txt
-chmod u-x a.txt
-chmod u=rx, g=r, o=r a.txt

-a.txt 소유자에게 파일 실행 권한 추가
-a.txt 소유자에게 파일 실행 권한 제거
-a.txt 소유자에게는 읽기/쓰기, 그룹과 외 사용자에게는 읽기 권한 부여

사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 소유와 허가권

▶ 파일 소유권 chown 명령어

```
-rw-r--r-- 1 root root 0 9월 3 09:20 test_00.txt  
drwxr-xr-x 2 root root 4096 9월 2 08:51 공개
```

파일 소유권(ownership) : 파일 소유 사용자 파일 속한 그룹

```
-chown ubuntu a.txt  
-chown ubuntu/ubuntu a.txt  
-chgrp ubuntu a.txt
```

```
-a.txt 소유자를 ubuntu로 변경  
-a.txt 소유자 및 그룹을 ubuntu로 변경  
-a.txt 그룹만 ubuntu로 변경
```

사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 링크

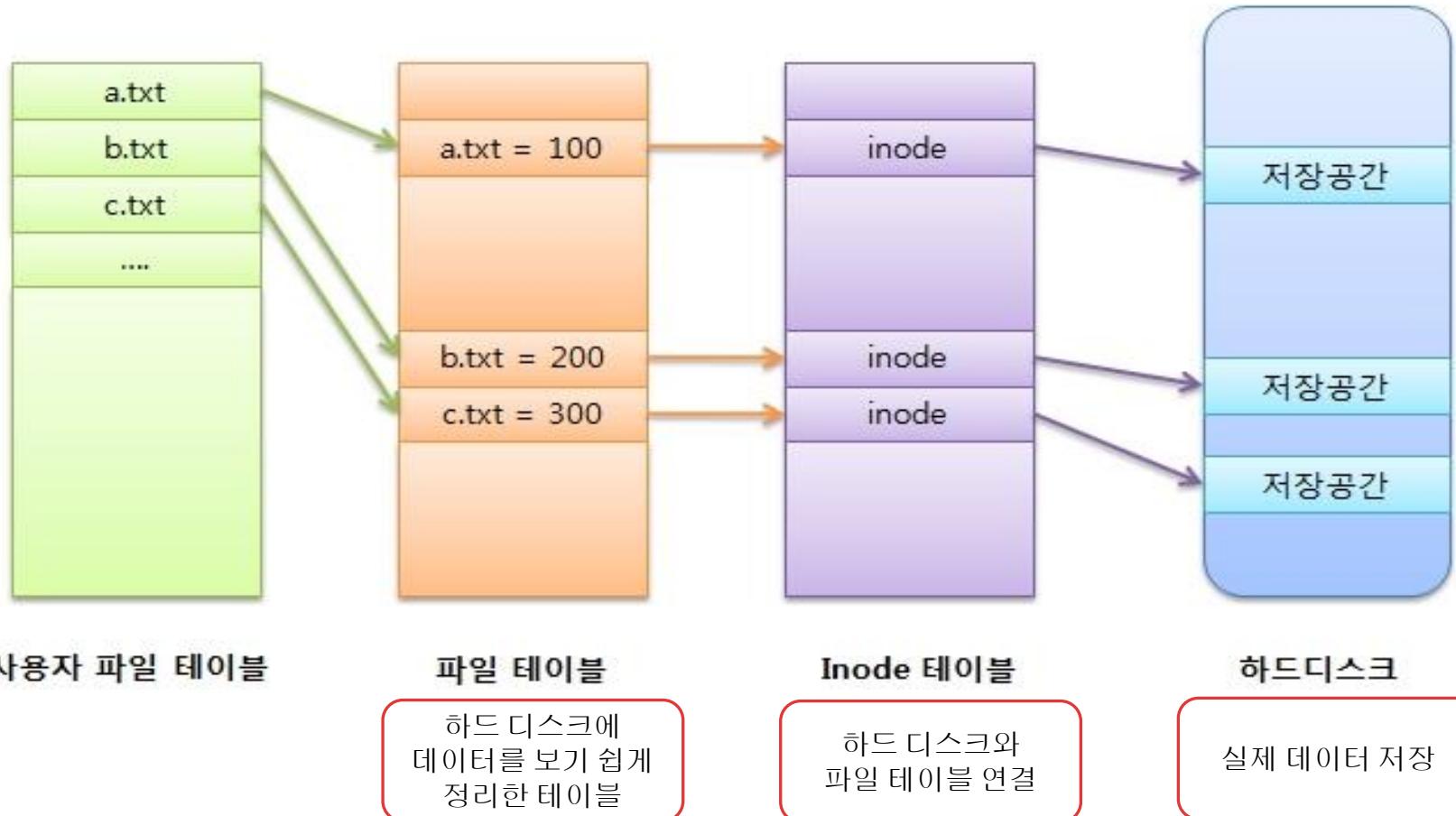
- ▶ window의 바로가기 아이콘과 동일한 기능
- ▶ 하드 링크(Hard Link) & 심볼릭 링크(Symbolic Link)

링크 수 : 해당 파일 또는 디렉터리의 하드 링크 개수

```
-rwx-r--r-- 1 root root 0 9월 3 09:20 test_00.txt  
drwxr-xr-x 2 root root 4096 9월 2 08:51 공개
```

사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 링크



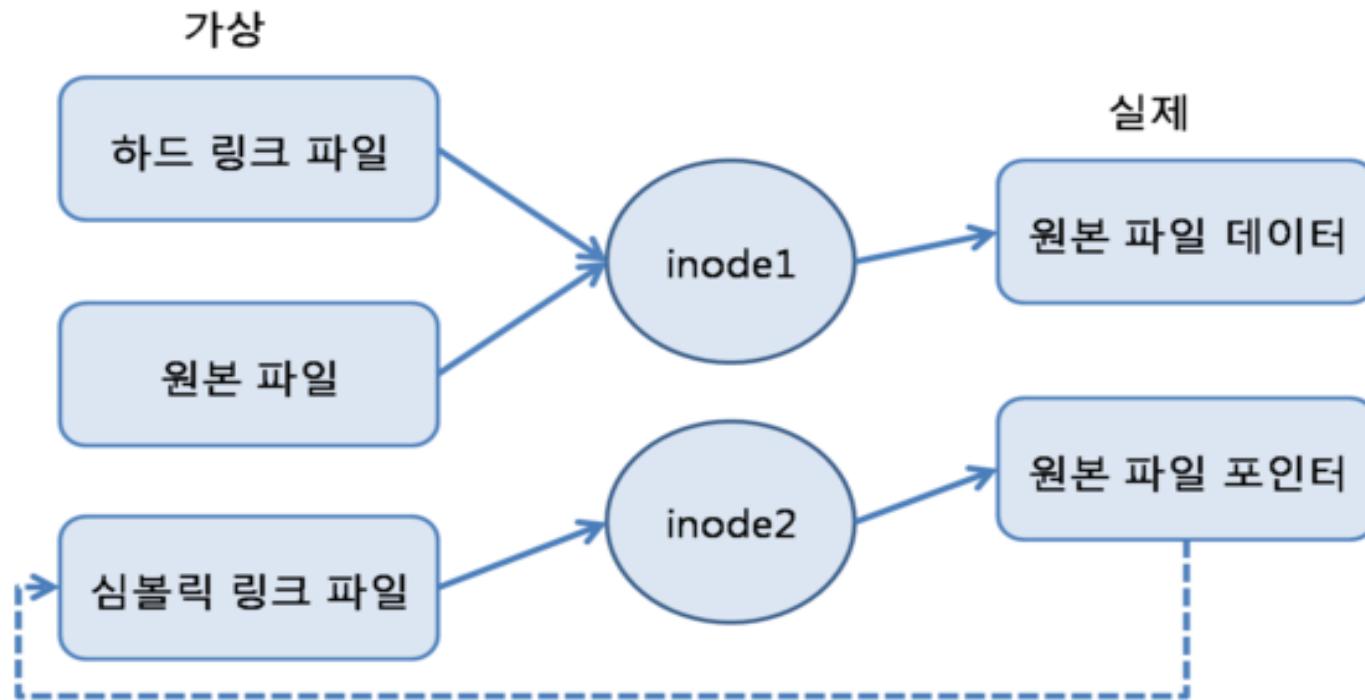
▶ 파일과 디렉터리 링크

▶ inode 란?

- ▶ 파일 시스템에서 사용하는 자료구조
- ▶ 소유권, 허가권, 파일 종류, 실제 데이터 위치 등 정보 저장
- ▶ inode가 모인 inode 블록은 전체 디스크의 1% 차지
- ▶ 파일과 디렉터리는 1개씩 inode를 가짐

사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 링크



사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 링크

하드 링크 (Hard Link)	심볼릭 링크 (Symbolic Link)
원본 파일과 동일 inode 사용	새로운 inode 생성(공간 차지)
원본 수정 시 하드링크도 수정	원본 파일의 포인터 역할
원본 파일 삭제, 이동 시 존재	

사용자 및 파일 속성 명령어

▶ 파일과 디렉터리 링크

▶ 심볼릭 링크 ln 명령어

ln basefile hardlink
ln -s basefile softlink
ls -il

-basefile의 하드링크 파일 생성
-basefile의 심볼릭 링크 파일 생성

```
root@server:~# ln item.txt hitem
root@server:~# ln -s item.txt sitem
root@server:~# ls -ll
합계 40
1573093 -rw-r--r-- 2 root root 8 9월 12 00:37 hitem
1573093 -rw-r--r-- 2 root root 8 9월 12 00:37 item.txt
1573092 lrwxrwxrwx 1 root root 8 9월 12 00:37 sitem -> item.txt
1573091 lrwxrwxrwx 1 root root 7 9월 12 00:33 stlink -> link.txt
1573694 lrwxrwxrwx 1 root root 12 9월 12 00:34 syLn -> basefile.txt
1572895 drwxr-xr-x 2 root root 4096 9월 2 08:51 공개
```

CH04 – 관리자 명령어

▶ 패키지란?

- ▶ 데비안 리눅스에서 Window의 setup.exe 설치 실행파일과 비슷한 **확장자가 deb인 설치 파일** 제작
- ▶ 이것을 패키지(Package)라 함
- ▶ Ubuntu 전체 패키지 파일 위치
 - ▶ <http://kr.archive.ubuntu.com/ubuntu/pool>

▶ 패키지 파일 이름

패키지이름_버전-개정번호_아키텍처.deb

패키지 이름 - 프로그램 이름, 버전 바로 앞 까지가 이름

버전 - 3자리 수로 구성, 주 버전 + 부 버전

개정 번호 - 문제점 개선 시마다 부여

아키텍처 - 파일 설치 가능한 CPU

▶ 예) **calculator_2.1.4.1-1_amd64.deb**

firefox-gnome-support_22.0+build2-0ubuntu0.12.04.2_amd64.deb

▶ 패키지 명령어 **dpkg**

- ▶ 초기 ubuntu에서 사용되던 명령어
- ▶ 의존성 문제로 설치 불편

-dpkg -i 패키지이름.deb
-dpkg -r 패키지이름
-dpkg -P 패키지이름
-dpkg -l 패키지이름
-dpkg -L 패키지이름
-dpkg --info 패키지이름

- install, 패키지이름.deb 설치
- remove, 기존 설치된 패키지 삭제
- purge, 기존 설치 패키지 및 관련 설정파일까지 삭제
- 설치된 패키지에 대한 정보 출력
- 패키지가 설치된 파일 목록 출력
- 설치되지 않은 패키지 파일 정보 출력

▶ 패키지 명령어 apt-get

- ▶ dpkg 명령어의 단점인 의존성 부분 보완
- ▶ 특정 패키지 설치 시 관련 패키지 자동 먼저 설치
- ▶ 미리 다운로드 받을 필요 없이 deb파일 저장소에서 의존성 있는 deb파일까지 모두 다운로드 후 자동 설치
- ▶ 반드시 인터넷 연결 필요
- ▶ 저장소 URL -> /etc/apt/sources.list

▶ 패키지 명령어 apt-get

- apt-get -y install 패키지이름
- apt-get update
- apt-get remove 패키지이름
- apt-get purge 패키지이름
- apt-get autoremove
- apt-get clean
- apt-get autoclean
- apt-get upgrade

- 패키지 설치, 설치 시 질문에는 모두 Yes
- /etc/apt/sources.list 내용 업데이트
- 기존 설치 패키지 삭제
- 기존 설치 패키지 및 관련 설정파일까지 삭제
- 사용하지 않는 패키지 모두 삭제
- 설치 시 내려 받은 파일 및 과거 파일 삭제
- 설치 시 내려 받은 파일 및 과거 파일 삭제
- 전체 시스템 업그레이드

▶ 패키지 명령어 apt-cache

▶ 패키지 설치 전 패키지 정보 및 의존성 문제 확인

```
-apt-cache show 패키지이름  
-apt-cache depends 패키지이름  
-apt-cache remove 패키지이름
```

- 패키지 정보 출력
- 패키지에 대한 의존성 정보 출력
- 패키지에 의존하는 다른 패키지 목록 출력

▶ 파일 압축 명령어 xz

▶ 확장명 xz로 압축 또는 압축해제

- xz 파일이름
- xz -d 파일이름.xz
- xz -l 파일이름.xz
- xz -k 파일이름

- 파일 압축하여 파일이름.xz생성, 파일 삭제
- 압축해제
- 압축 파일에 포함된 파일 목록과 압축률 출력
- 압축 후에 기존 파일 유지

▶ 파일 압축 명령어 bzip2

▶ 확장명 bz2로 압축 또는 압축해제

- bzip2 파일이름
- bzip2 -d 파일이름.bz2
- bzip2 -k 파일이름

- 파일 압축하여 파일이름.xz생성, 파일 삭제
- 압축해제
- 압축 후에 기존 파일 유지

▶ 파일 압축 명령어 gzip

▶ 확장명 gz로 압축 또는 압축해제

-gzip2 파일이름

-gzip2 -d 파일이름.gz

- 파일 압축하여 파일이름.gz생성, 파일 삭제
- 압축해제

▶ 파일 압축 명령어 zip / unzip

▶ 확장명zip로 압축 또는 압축해제

-zip 새로생성파일.zip 압축파일이름
-unzip 압축파일이름.zip

- 파일 압축하여 파일이름.zip생성
- 압축해제

▶ 파일 묶기 명령어 tar

▶ 확장명 tar로 묶음 파일 생성 또는 풀기

-tar [동작][옵션]

[동작] c – 새로운 묶음 생성
x – 묶음 파일 풀기
t – 묶음 풀기 전 경로 출력
C – 지정된 경로에 압축 해제

[옵션] f – 필수, 묶음 파일 이름 지정
v – 파일 묶기, 풀기 과정 출력
J – tar + xz
z – tar + gzip
j – tar + bzip2

▶ 파일 위치 검색 명령어 **find**

▶ 조건에 맞는 파일 찾기

-find 경로 [옵션] [조건] 동작

[옵션] -name : 파일 이름
-user : 파일 소유자
-perm : 허가권
-size : 크기

[동작] – print : 검색 정보 출력
– exec : 외부 명령 실행

▶ 파일 위치 검색 명령어 **find**

```
-find /etc -name “*.conf”
-find /home -user ubuntu
-find -perm 644
```

- /etc 디렉터리 하위에 확장명이 *.conf 파일 검색
- /home 디렉터리 하위에 소유자가 ubuntu인 파일
- 현재 사용자 홈 디렉터리 하위에 허가권이 644인 파일 검색

▶ 파일 위치 검색 명령어 **which**

- ▶ PATH에 설정된 디렉터리만 검색
- ▶ 항상 절대경로 포함한 위치 검색

-which 실행파일이름

which gzip

```
root@server:~# which gzip  
/bin/gzip
```

▶ 파일 위치 검색 명령어 whereis

▶ 실행 파일 및 소스까지 검색

-whereis 실행파일이름

whereis gzip

```
root@server:~# which gzip  
/bin/gzip  
root@server:~# whereis gzip  
gzip: /bin/gzip /usr/share/man/man1/gzip.1.gz /usr/share/info/gzip.info.gz
```

▶ 파일 위치 검색 명령어 locate

- ▶ 파일 목록 데이터베이스에서 검색
- ▶ 매우 빠른 속도
- ▶ updatedb 명령어 실행이 필요

-locate 파일 이름

locate gzip

```
root@server:~# locate gzip
/bin/gzip
/usr/lib/apt/methods/gzip
/usr/lib/cups/filter/gzipoany
/usr/lib/klibc/bin/gzip
/usr/lib/python2.7/gzip.py
/usr/lib/python2.7/gzip.pyc
/usr/lib/python3.5/gzip.py
/usr/lib/python3.5/__pycache__/gzip.cpython-35.pyc
```

▶ 방화벽 설정 명령어 ufw

- ▶ The Uncomplicated Firewall 약자로 우분투 기본 제공 방화벽 SW
- ▶ root계정에서만 제어가능
- ▶ 처음 설치 시 OFF -> 보안을 위해서 ON
- ▶ 외부 접속 모든 포트 Close -> 필요한 포트만 Open

-ufw status
-ufw enable
-ufw disable
-ufw allow 포트번호
-ufw allow 포트명
-ufw allow from 0.0.0.0/24
-ufw deny from 0.0.0.0/24

- 현재 시스템의 방화벽 상태 출력
- 방화벽 ON
- 방화벽 OFF
- 포트번호 방화벽 해제
- 포트명 방화벽 해제
- 특정 IP 주소 허가
- 특정 IP 주소 불허

▶ 시스템 데몬 설정 명령어 rcconf

- ▶ 데몬(서비스)의 시작, 중지, 재시작 및 사용여부 설정
- ▶ rcconf 패키지 설치 필요
- ▶ runlevel configuration tool 약자

```
# apt-get -y install rcconf
```

▶ 시스템 데몬 설정 명령어 rcconf

rcconf

```
rcconf - Debian Runlevel Configuration tool

[*] acpid           Start the Advanced Configuration
[*] alsa-utils       Restore and store ALSA driver se
[*] anacron          Run anacron jobs
[*] apparmor         AppArmor initialization
[*] apport           automatic crash report generatio
[*] avahi-daemon     Avahi mDNS/DNS-SD Daemon
[*] bluetooth        Start bluetooth daemons
[*] brltty           Braille terminal driver
[*] checkroot-bootclean.sh
[*] console-setup    Set console keymap
[*] cron             Regular background program proce
[*] cups              CUPS Printing spooler and server
[*] cups-browsed     cups-browsed - Make remote CUPS
[*] dbus              D-Bus systemwide message bus
[*] grub-common      Record successful boot for GRUB

<확인>           <취소>
```

▶ 시스템 작업 예약 명령어 cron

- ▶ 주기적 반복되는 일 자동 실행하도록 예약
- ▶ 관련 데몬(서비스)은 crond
- ▶ 관련 파일 형식 /etc/crontab → [분] [시] [일] [월] [요일] [작업내용]
- ▶ root 권한 필요

▶ 일회성 작업 예약 명령어 at

▶ 한 번만 실행되고 소멸

-at 3:00am tomorrow

-at 11:00pm January 30

- 내일 새벽 3시

- 1월 30일 오후 11시

▶ 파이프(Pipe)

- ▶ 2개의 프로그램을 연결해 주는 연결 통로
- ▶ 앞에 실행한 명령어의 출력 결과를 뒤에 실행하는 명령어의 입력 값으로 넣어 줌
- ▶ ' | ' 사용 → 형식 : CMD | CMD

```
#ls -l /etc | less
```

/etc에 대한 ls -l 결과를 less로 넘겨서 처리

▶ 필터(Filter)

- ▶ 필요한 것만 걸러주는 명령어
- ▶ grep, tail, wc, sort, awk, sed 명령어 해당
- ▶ 파이프와 함께 사용

```
#ls -l /etc | grep conf
```

/etc에 대한 ls -l 결과에서 conf 글자 포함된 것 출력

▶ 리디렉션(Redirection)

- ▶ 표준 입출력 방향 변경
- ▶ 표준 입력 -> 키보드, 표준 출력-> 모니터

```
#ls -l > list.txt  
#ls -l >> list.txt  
#sort < list.txt  
#sort < list.txt > out.txt
```

- ls -l 명령 결과를 list.txt에 덮어쓰기로 저장
- ls -l 명령 결과를 list.txt에 이어쓰기로 저장
- list.txt 내용 정렬해서 화면 출력
- list.txt 파일 정렬해서 out.txt 파일에 출력

▶ 리디렉션(Redirection)

- ▶ 표준 입출력 방향 변경
- ▶ 표준 입력 -> 키보드, 표준 출력-> 모니터

```
#ls -l > list.txt  
#ls -l >> list.txt  
#sort < list.txt  
#sort < list.txt > out.txt
```

- ls -l 명령 결과를 list.txt에 덮어쓰기로 저장
- ls -l 명령 결과를 list.txt에 이어쓰기로 저장
- list.txt 내용 정렬해서 화면 출력
- list.txt 파일 정렬해서 out.txt 파일에 출력

CH05 – 프로세스 & 데몬

▶ 프로세스(Process)

- ▶ 프로그램 실행 코드가 메모리에 로딩되어 활성화된 것
- ▶ 현재 실행 중인 프로그램으로 태스크(Task)라고도 함
- ▶ CPU Core는 한 순간에 하나의 프로세스만 실행
- ▶ 시분할 스케줄링 알고리즘으로 프로세스 제어

▶ 프로세스(Process)

종류	특징
포그라운드 프로세스	사용자에게 보여지는 프로세스 사용자와 상호작용하여 동작하는 프로세스
배치 프로세스	실행될 프로그램 리스트에 따라 순차적으로 진행되는 프로세스
백그라운드 프로세스	눈에 보이지는 않지만 뒤에서 실행되고 있는 것 백신, 방화벽 같은 프로세스

▶ 프로세스(Process)

▶ 프로세스 번호 (PID)

- ▶ 프로세스를 구분하기 위해 고유번호 할당
- ▶ 메모리에 활성화된 프로세스를 관리

▶ 작업 번호

- ▶ 현재 실행되고 있는 백그라운드 프로세스의 순차 번호

▶ 프로세스(Process)

▶ 부모 프로세스 / 자식 프로세스

- ▶ 모든 프로세스는 혼자 독립되어 실행되지 않음
- ▶ 부모 프로세스 하위에 종속되어 실행

- ▶ 예) X-윈도우 (부모) --- Firefox (자식)
X-윈도우 강제 종료 시 Firefox도 강제 종료됨

▶ 프로세스(Process) 명령어

▶ ps 명령어 – processes 약자

ps -u
ps -f
ps -l
ps -e
ps -a
ps -x
ps -r
ps -j

-프로세스와 사용자 이름과 시작 출력
프로세스의 정보를 한 줄로 자세히 출력 (full)
프로세스의 정보를 한 줄로 보다 자세히 길게 출력 (long)
환경에 대한 정보 출력 (environment)
다른 사용자들의 프로세스들을 모두 표시
로그인 상태에서 완료되지 않은 프로세스 표시
현재 실행 중인 프로세스들을 표시
작업 중심 형태로 출력 (job)

```
# ps -ef | grep 프로세스이름
```

▶ 프로세스 강제 종료 명령어 kill

- ▶ -9 옵션과 함께 사용
- ▶ 무조건 프로세스 종료

```
# kill -9 프로세스번호
```

▶ 프로세스 관계 확인 명령어 pstree

- ▶ 프로세스의 부모 & 자식 프로세스 관계를 트리 형태 출력

```
pstree  
pstree -h  
pstree -p  
pstree -l
```

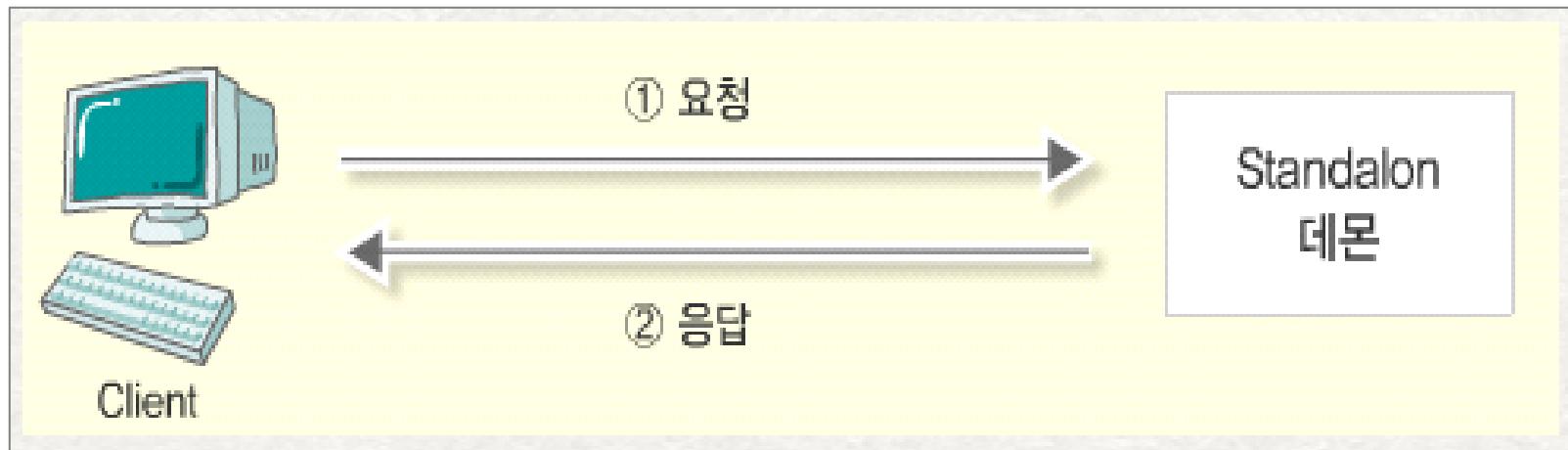
현재 실행 프로세스의 관계 출력
현재 프로세스와 부모 프로세스 강조 출력
실행된 모든 프로세스의 PID 출력
부모와 자식 프로세스의 관계 모두 출력

▶ 데몬(Daemon)

- ▶ 눈에 보이지 않고 현재 시스템에서 동작 중인 프로세스
- ▶ 백그라운드 프로세스의 일종
- ▶ 서비스라고도 함
- ▶ 다양한 **서비스를 제공하는 서버 프로그램**

▶ Standalone 데몬

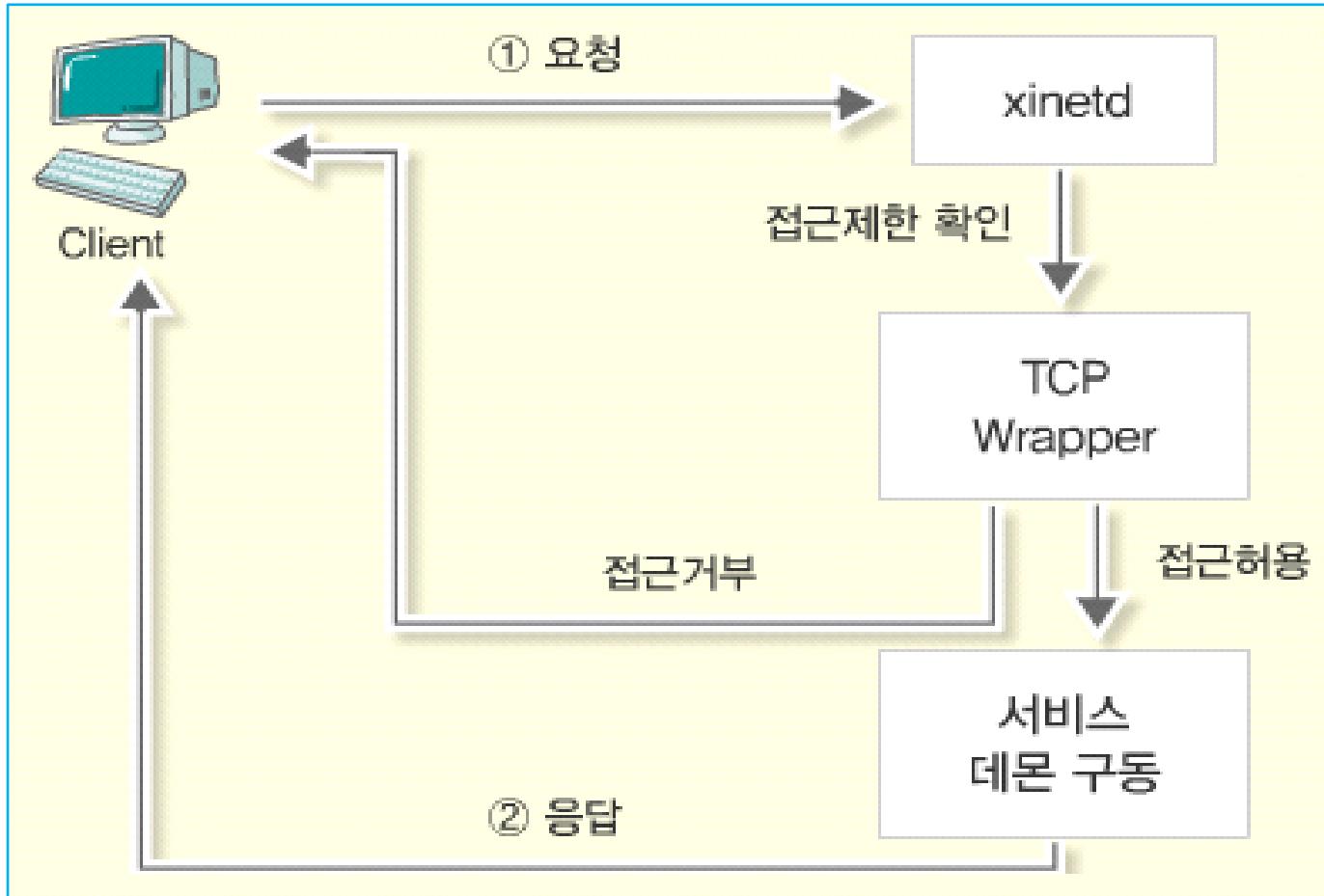
- ▶ 데몬 구동과 동시에 메모리 상주
- ▶ 클라이언트 요청 대기, 빠른 응답 제공
- ▶ 자원 낭비 초래



▶ xinetd 데몬

- ▶ extended internet daemon 약자
- ▶ 여러가지 서비스를 한꺼번에 관리하는 슈퍼 데몬
- ▶ 인터넷 기반 연결 관리 데몬
- ▶ 요청이 들어오면 Xinetd에 종속된 프로그램 실행
- ▶ 처리가 끝나면 다시 휴면상태

▶ xinetd 데몬



프로세스 & 데몬

▶ 중요 데몬 역할

데 몬	역 할	데 몬	역 할
crond	cron 서비스 데몬	syslog	시스템 로그 관리 데몬
atd	at 서비스 데몬	vsftpd	FTP 서버 데몬
dhcpd	DHCP 데몬	xinetd	xinetd 방식 데몬 관리하는 슈퍼 데몬
httpd	Apache 웹 서버 데몬	sshd	ssh 서버 데몬
iptables	방화벽 데몬	sedmail	메일서버 데몬
mysqld	MySQL 데이터베이스 데몬	nfs	네트워크 파일 시스템 데몬
named	DNS 네임서버 데몬	netfs	원격지 파일 시스템 마운트 데몬
network	네트워크 서비스 데몬		

▶ 데몬 제어 명령어 systemctl

▶ 데몬 즉 서비스의 동작 및 상태 관련 제어

systemctl start	서비스이름
systemctl stop	서비스이름
systemctl restart	서비스이름
systemctl reload	서비스이름
systemctl status	서비스이름
systemctl enable	서비스이름
systemctl disable	서비스이름
systemctl is-enabled	서비스이름
systemctl is-active	서비스이름

데몬 즉 서비스 시작
데몬 즉 서비스 종료
데몬 즉 서비스 재시작
데몬 즉 서비스 리로딩
데몬 즉 서비스 상태 확인
데몬 즉 서비스 부팅 시 사용
데몬 즉 서비스 부팅 시 사용 불가
데몬 즉 서비스 부팅 시 동작여부 확인
데몬 즉 서비스 현재 동작상태여부

▶ 데몬 제어 명령어

[시작 / 정지 / 재시작]

```
#systemctl start mariadb.service  
#systemctl enable mariadb.service
```

[시스템 데몬 리스트]

```
#ls /usr/lib/systemd/system/*.service
```

[실행 스크립트 파일]

```
/lib/systemd/system/ 서비스이름.service
```

[부팅과 동시에 자동 실행 여부 설정]

방법1) # systemctl list-unit-files // 현재 able /enable 확인

방법2) #rcconf

▶ 소켓(Socket)

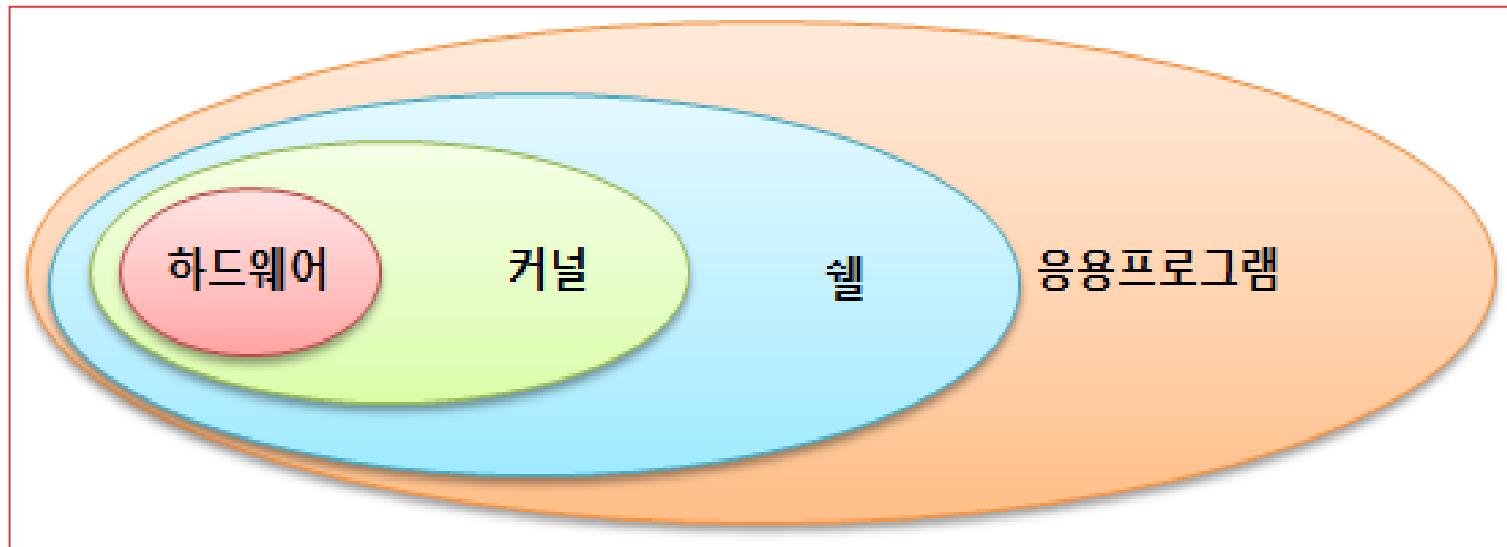
- ▶ 외부에서 특정 서비스 요청 시 systemd 구동
- ▶ 요청 끝나면 소켓 종료
- ▶ 서비스보다 시동 시간 더 소요
- ▶ xinetd 대시에 사용
- ▶ 스크립트 파일 : /lib/systemd/system/소켓이름.socket

```
# ls /lib/systemd/system/*.socket
```

CH06 – 웰 스크립트

▶ Shell이란?

- ▶ 리눅스 커널과 사용자를 연결 시켜주는 인터페이스 역할 수행
- ▶ 텍스트 기반의 사용자 명령어를 입력 받을 수 있는 환경
- ▶ 리눅스 시스템에 명령을 내리고 결과 확인 및 시스템 상태 확인



Shell Script

▶ Shell 프로그램

- ▶ /etc/shells 파일에 현재 시스템에서 사용 가능한 쉘 프로그램
- ▶ 가장 많이 사용되는 것이 bash(Bourne Again shell) 쉘로 GNU 프로젝트에서 만들어서 배포

 pi@raspberrypi: /tmp/shk

```
pi@raspberrypi:/tmp/shk $ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
```

▶ bash Shell

- ▶ ubuntu의 기본 셀
- ▶ 특징
 - ▶ Alias 기능
 - ▶ History 기능
 - ▶ 연산 기능
 - ▶ Job Control 기능
 - ▶ 자동 이름 완성 기능
 - ▶ 프롬프트 제어 기능
 - ▶ 명령 편집 기능

Shell Script

▶ bash Shell

▶ 환경 변수

- ▶ 확인 명령 : echo \$환경변수이름
- ▶ 변경 명령 : export 환경변수 = 값

환경변수	설명	환경변수	설명
HOME	현재 사용자 홈 디렉토리	PATH	실행 파일을 찾는 디렉토리 경로
LANG	기본 지원 언어	PWD	사용자의 현재 작업 디렉토리
TERM	로그인 터미널 타입	SHELL	로그인에서 사용하는 셸
USER	현재 사용자 이름	DISPLAY	X 디스플레이 이름
COLUMNS	현재 터미널의 컬럼 수	LINES	현재 터미널 라인의 수
PS1	1차 명령 프롬프트	PS2	2차 명령 프롬프트
BASH	bash 셸 경로	BASH_VERSION	bash 버전
HISTFILE	히스토리 파일 경로	HISTSIZE	히스토리 파일에 저장되는 개수
HOSTNAME	호스트 이름	USERNAME	현재 사용자 이름
LOGNAME	로그인 이름	LS_COLORS	ls 명령의 확장자 색상 옵션
MAIL	메일 보관 경로	OSTYPE	운영체제 타입

▶ bash Shell

▶ 명령문 처리 방법

프롬프트 명령 [옵션...] [인자...]

```
# ls -l  
# rm -rf /mydir  
# find . / -name "*.conf"
```

▶ 셸 스크립트 프로그래밍

- ▶ C언어와 유사한 프로그래밍
- ▶ 일반적인 프로그래밍과 유사
 - ▶ 변수, 반복문, 제어문 등등 사용
- ▶ 컴파일하지 않음
- ▶ 텍스트 파일 형태로 셸에서 바로 실행

Shell Script

▶ 셸 스크립트 프로그래밍

▶ gedit 실행 & 셸 스크립트 작성

파일 명 : name.sh

```
#!/bin/sh  
echo “사용자 이름 : “ $USER  
echo “홈 디렉터리 : “ $HOME  
exit 0
```

- #! 인터프리터
- echo : 화면 출력
- exit 0 : 정상 종료

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 실행 명령어 sh

```
# sh 셸_스크립트_파일명
```

▶ '실행 가능 속성' 실행

```
# chmod -x 셸 스크립트 파일명
```

```
# ./셀_스크립트_파일명
```

▶ 셸 스크립트 프로그래밍

▶ 변수

- ▶ 사용 전 선언하지 않으면 변수 값 할당 시 자동 변수 생성
- ▶ **값은 전부 문자열로 취급**
- ▶ 변수 이름 **대소문자 구분**
- ▶ 대입 '=' 좌우에 **공백 불가**

▶ 입력과 출력

입력	echo 출력 데이터(변수, 수식 ...)
출력	read 변수명

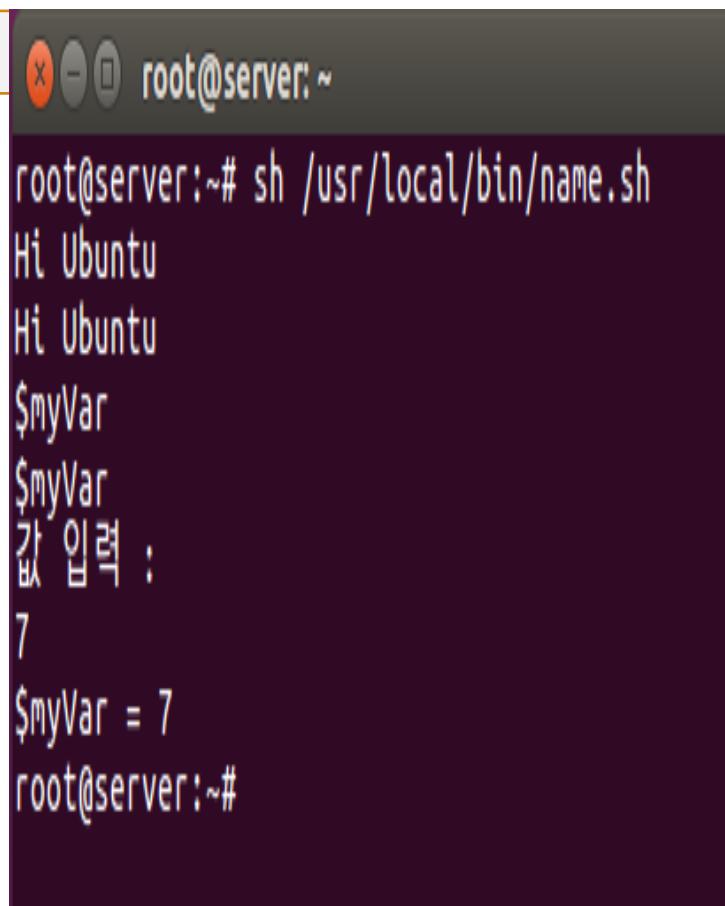
Shell Script

▶ 셸 스크립트 프로그래밍

▶ 변수

파일 명 : name.sh

```
#!/bin/sh
myVar="Hi Ubuntu" ← 변수 선언 & 대입
echo $myVar          변수 값 출력
echo "$myVar"
echo '$myVar'         변수명 출력
echo $$myVar
echo 값 입력 :
read myVar           ← 값 입력
echo '$myVar' = $myVar
exit 0
```



A terminal window titled 'root@server:~'. The command 'sh /usr/local/bin/name.sh' is run, followed by the output of the script's execution. The output shows the variable assignment, the value of the variable, and the user input prompt.

```
root@server:~# sh /usr/local/bin/name.sh
Hi Ubuntu
Hi Ubuntu
$myVar
$myVar
값 입력 :
7
$myVar = 7
root@server:~#
```

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 연산

파일 명 : name.sh

```
#! /bin/sh
```

```
num1=100  
num2=$num1+200  
echo $num2
```

```
num3= `expr $num1 +200`  
echo $num3
```

```
num4= `expr \$num1 + 200 \* /10 \* 2`  
echo $num4
```

```
exit 0
```

expr 키워드

- 문자열 값을 숫자 값으로 변환
- 키보드 숫자1 왼쪽의 역 따옴표()

역 슬래시(\) + 이스케이프문자 & 기호

- 4칙 연산자 중 곱하기(*)만 *
- (.), * 등의 특수 문자

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 파라미터 변수

▶ 실행하는 명령의 부분 부분을 변수로 지정하는 변수

명령어	apt-get	-y	install	gedit
파라미터	\$0	\$1	\$2	\$3

▶ 셸 스크립트 프로그래밍

▶ 파라미터 변수

파일 명 : param.sh

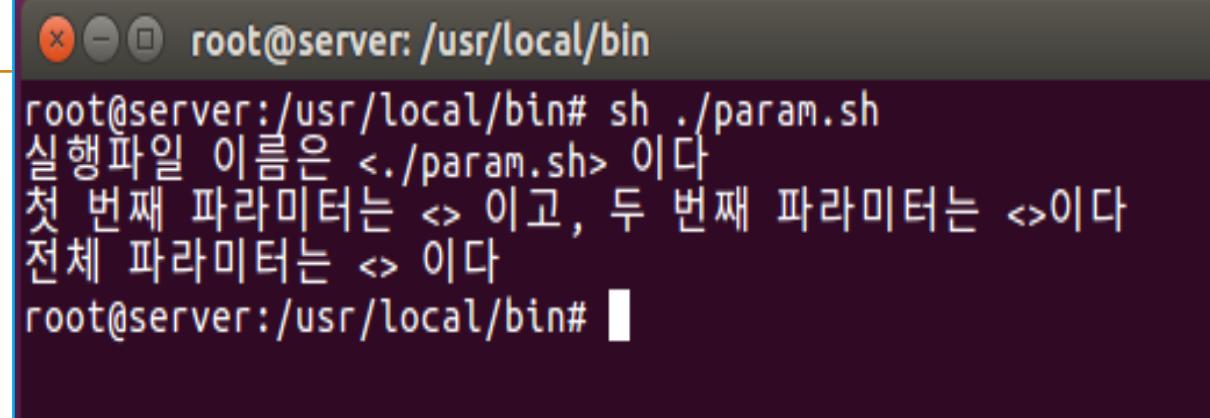
```
#!/bin/sh
```

```
echo “실행파일 이름은 <$0>이다”
```

```
echo “첫번째 파라미터는 <$1>이고, 두 번째 파라미터는 <$2>다  
“
```

```
echo “전체 파라미터는 <$*>이다”
```

```
exit 0
```



A terminal window titled "root@server: /usr/local/bin" is shown. The command "sh ./param.sh" is entered, followed by three lines of output from the script: "실행파일 이름은 <./param.sh>이다", "첫 번째 파라미터는 <>이고, 두 번째 파라미터는 <>이다", and "전체 파라미터는 <>이다". The terminal prompt "root@server:/usr/local/bin#" is visible at the bottom.

```
root@server: /usr/local/bin
root@server:/usr/local/bin# sh ./param.sh
실행파일 이름은 <./param.sh>이다
첫 번째 파라미터는 <>이고, 두 번째 파라미터는 <>이다
전체 파라미터는 <>이다
root@server:/usr/local/bin#
```

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 조건문 – if ~fi

파일 명 : ex_if.sh

```
#!/bin/sh

if [ "woo" = "woo" ]
then
    echo 참입니다.
fi

exit 0
```

[조건]

– 사이의 각 단어에 모두 공백 있어야 함

```
#!/bin/sh

if [ "woo" = "woo" ]
then
    echo 참입니다.
else
    echo 거짓입니다.
fi

exit 0
```

▶ 셸 스크립트 프로그래밍

▶ 조건문 – if ~fi

▶ 비교 연산자

문자열 비교	설 명
“문자열1” = “문자열2”	두 문자열이 같으면 참
“문자열1” != “문자열2”	두 문자열이 같지 않으면 참
-n “문자열”	문자열이 NULL이 아니면 참
-z “문자열”	문자열이 NULL이면 참

▶ 셸 스크립트 프로그래밍

▶ 조건문 – if ~fi

▶ 비교 연산자

산술 비교	설 명
수식1 -eq 수식2	두 수식 또는 변수가 같으면 참
수식1 -ne 수식2	두 수식 또는 변수가 같지 않으면 참
수식1 -gt 수식2	수식1이 크다면 참 (수식1 > 수식2)
수식1 -ge 수식2	수식1이 크거나 같다면 참 (수식1 >= 수식2)
수식1 -lt 수식2	수식1이 작으면 참 (수식1 < 수식2)
수식1 -le 수식2	수식1이 작거나 같다면 참 (수식1 <= 수식2)
!수식	수식이 거짓이면 참

▶ 셸 스크립트 프로그래밍

▶ 조건문 – if ~fi

파일 명 : **ex_compare.sh**

```
#!/bin/sh

if [ 100 -eq 200 ]
then
    echo "100과 200은 같다."
else
    echo "100과 200은 서로 다르다."
fi

exit 0
```

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 조건문 – if ~fi

▶ 파일관련 조건

파일 조건	설 명
-d 파일이름	파일이 디렉토리면 참
-e 파일이름	파일이 존재하면 참
-f 파일이름	파일이 일반 파일이면 참
-g 파일이름	파일이 그룹ID가 설정되면 참 (set-group-id)
-r 파일이름	파일이 읽기 가능이면 참

파일 조건	설 명
-s 파일이름	파일이 디렉토리면 참
-u 파일이름	파일이 존재하면 참
-w 파일이름	파일이 쓰기 가능 상태이면 참
-x 파일이름	파일이 실행 가능 상태이면 참

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 조건문 – if ~fi

파일 명 : ex_file.sh

```
#!/bin/sh

fname=/lib/systemd/system/cron.service

if [ -f $fname ]
then
    head -5 $fname
else
    echo "cron 서버가 설치되지 않았습니다."
fi

exit 0
```

[조건]

- 해경 경로의 파일이 일반 파일(-f) 여부

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 조건문 – if ~fi

파일 명 : ex_file_02.sh

```
#!/bin/sh
```

```
echo "찾고 싶은 파일명을 입력하세요."
read filename
```

```
if [ -f $filename ] && [ -s $filename ];
then
    head -5 $filename
else
    echo "$filename 파일이 없거나, 크기가 0입니다."
fi

exit 0
```

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 조건문 – case ~ esac

▶ 다중 조건에 대한 처리

[첫 번째 파라미터(\$1)에 따라서 처리]

- \$1이 start 일 경우
- \$1이 stop일 경우
- \$1이 restart 일 경우
- \$1이 그 외 일 경우

[실행] sh ./ex_case.sh stop

파일 명 : ex_case.sh

```
#!/bin/sh
```

```
case "$1" in
  start)
    echo "시작 ~" ;;
  stop)
    echo "중지!" ;;
  restart)
    echo "다시 시작 ~" ;;
  *)
    echo "알 수 없음!" ;;
esac
```

```
exit 0
```

▶ 셸 스크립트 프로그래밍

▶ 조건문

▶ case ~ esac

파일 명 : ex_case02.sh

```
#!/bin/sh
```

```
echo "가을이 좋은 가요?(yes/no)"  
read answer
```

```
case $answer in  
    yes | y | Y | Yes | YES)  
        echo "너무 좋아요~"  
        echo "최고의 계절이죠" ;;  
    [nN]*)  
        echo "싫어요~ㅜㅜ" ;;  
    *)  
        echo "무슨 말인지..."  
        echo "yes 아니면 no만 입력하세요"  
        exit 1;;  
esac
```

```
exit 0
```

▶ 셸 스크립트 프로그래밍

▶ 반복문 – for ~ in

파일 명 : ex_for_in.sh

```
#!/bin/sh
```

```
hap=0
```

```
for i in 1 2 3 4 5 6 7 8 9 10
```

```
do
```

```
    hap=`expr $hap + $i`
```

```
done
```

```
echo “1부터 10까지의 합 : “ $hap
```

```
exit 0
```

▶ 셸 스크립트 프로그래밍

▶ 반복문 – while

[1부터 ~ 10까지 반복문]

- le : less or equal
- ge : great or equal
- lt : less than
- gt : great than



파일 명 : ex_while_02.sh

```
#!/bin/sh

num=1

while [ $num -le 10 ]

do
    echo '$num = '$num
    num=`expr $num + 1`
done

echo 10번 반복 끝~

exit 0
```

▶ 셸 스크립트 프로그래밍

▶ break

- ▶ 반복문 종료

▶ continue

- ▶ 반복문을 조건식으로 이동

▶ exit

- ▶ 해당 프로그램 완전 종료

▶ return

- ▶ 함수 내에서 사용 가능. 함수 호출한 곳으로 돌아가기

▶ 셸 스크립트 프로그래밍

파일 명 : ex_loop.sh

```
#!/bin/sh

echo "무한반복 입력 시작 (b: break, c: continue, e: exit)"

while [ 1 ] ; do
    read input
    case $input in
        b|B)
            break ;;
        c|C)
            echo "continue 누르면 while 조건으로 이동"
            continue ;;
        e|E)
            echo "exit 누르면 프로그램(함수) 완전 종료"
            exit 1 ;;
    esac;
done
echo "break를 누르면 while 빠져나와서 지금 문장 출력"
exit 0
```

▶ 셸 스크립트 프로그래밍

▶ 사용자 정의 함수

▶ 함수 정의

```
함수이름 ( ) {
```

```
내용들 ...
```

```
}
```

```
함수이름
```

▶ 함수 호출

▶ 셸 스크립트 프로그래밍

▶ 사용자 정의 함수

▶ 함수 정의

```
함수이름 ( ) {
```

```
$1, $2
```

```
}
```

▶ 함수 호출

```
함수이름  파라미터1  파라미터2
```

Shell Script

▶ 셸 스크립트 프로그래밍

▶ 사용자 정의 함수

파일 명 : ex_userFunc.sh

```
#!/bin/sh

myFunc () {
    echo "나의 함수"
    return
}

echo "프로그램 시작"

myFunc

echo "프로그램 종료"

exit 0
```

파일 명 : ex_userFunc02.sh

```
#!/bin/sh

hap () {
    echo `expr $1 + $2`
    return
}

echo "10 + 20 = "

hap 10 20

echo "프로그램 종료"

exit 0
```

Shell Script

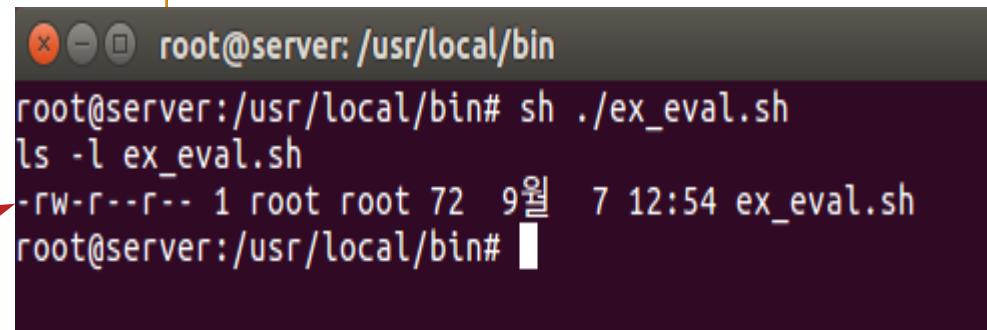
▶ 셸 스크립트 프로그래밍

▶ eval 키워드

▶ 문자열을 명령문으로 인식 & 실행

파일 명 : **ex_eval.sh**

```
#!/bin/sh  
  
str="ls -l ex_eval.sh"  
  
echo $str  
  
eval $str  
  
exit 0
```



```
root@server:/usr/local/bin# sh ./ex_eval.sh  
ls -l ex_eval.sh  
-rw-r--r-- 1 root root 72 9월 7 12:54 ex_eval.sh  
root@server:/usr/local/bin#
```

Shell Script

▶ 셸 스크립트 프로그래밍

▶ export 키워드

▶ 외부 변수로 선언

파일 명 : **ex_export1.sh**

```
#!/bin/sh
```

```
echo $var1
```

```
echo $var2
```

```
exit 0
```

파일 명 : **ex_export2.sh**

```
#!/bin/sh
```

```
var1="지역 변수"
```

```
export var2="외부 변수"
```

```
sh ./ex_export1.sh
```

```
exit 0
```

▶ 셸 스크립트 프로그래밍

▶ printf 함수

▶ C언어의 printf 함수와 동일한 방식

파일 명 : ex_printf.sh

```
#!/bin/sh

var1=100.5
var2="재미있는 세상..."

printf "%5.2f %n%s %n" $var1 $var2

exit 0
```

▶ 셸 스크립트 프로그래밍

▶ \$(명령)과 set 명령

- ▶ \$(명령) - 명령 결과 사용 할 경우
- ▶ set - 결과를 파라미터로 사용

파일 명 : ex_set.sh

```
#!/bin/sh  
  
echo "오늘 날짜는 $(date)입니다."  
  
set $(date)  
  
echo "오늘은 $4 요일 입니다."  
  
exit 0
```

[결과 → 파라미터로]

- \$(date)의 결과 값을 \$1 \$2 \$3... 파라미터 변수

▶ 셸 스크립트 프로그래밍

▶ shift 명령

- ▶ 파라미터 변수를 왼쪽으로 이동 시켜주는 명령어

파일 명 : ex_shift.sh

```
#!/bin/sh

myFunc() {
    str=""
    while [ "$1" != "" ]
    do
        str="$str $1"
        shift
    done
    echo $str
}
```

myFunc AAA BBB CCC DDD EEE GGG

exit 0