

LINUX

Autumn 2025



Daejeon Univ.
Seongbok Baik
sbbaik@dju.ac.kr

Text Book

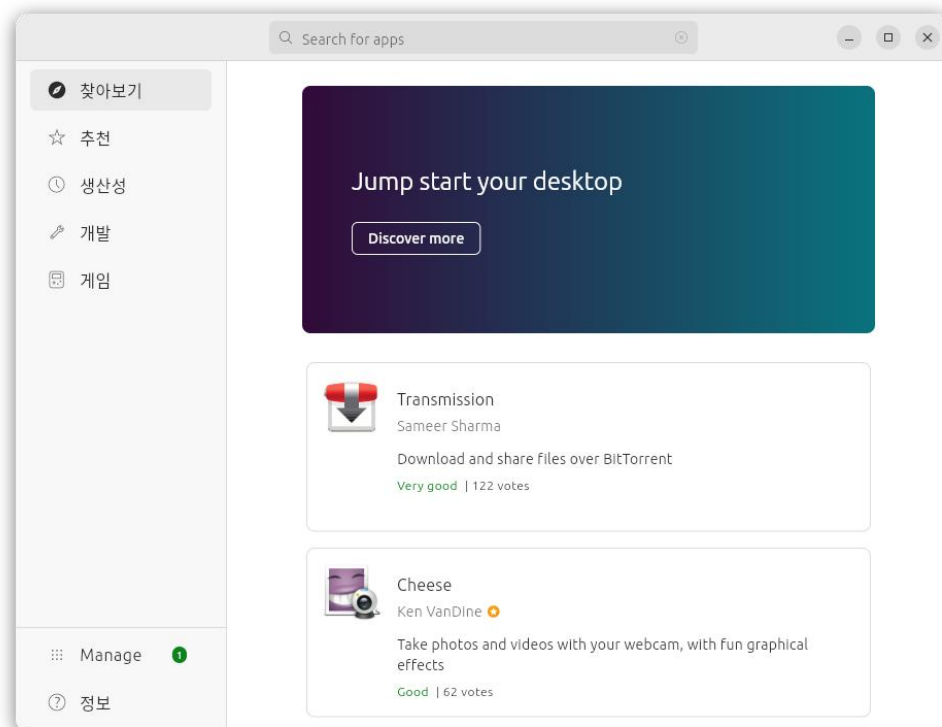
교재명	우분투 리눅스 시스템 & 서버
저자	창병모
출판사	생능출판사
발행년	2024.07.12
ISBN	979-11-92932-72-9

우분투 리눅스 시스템 & 서버

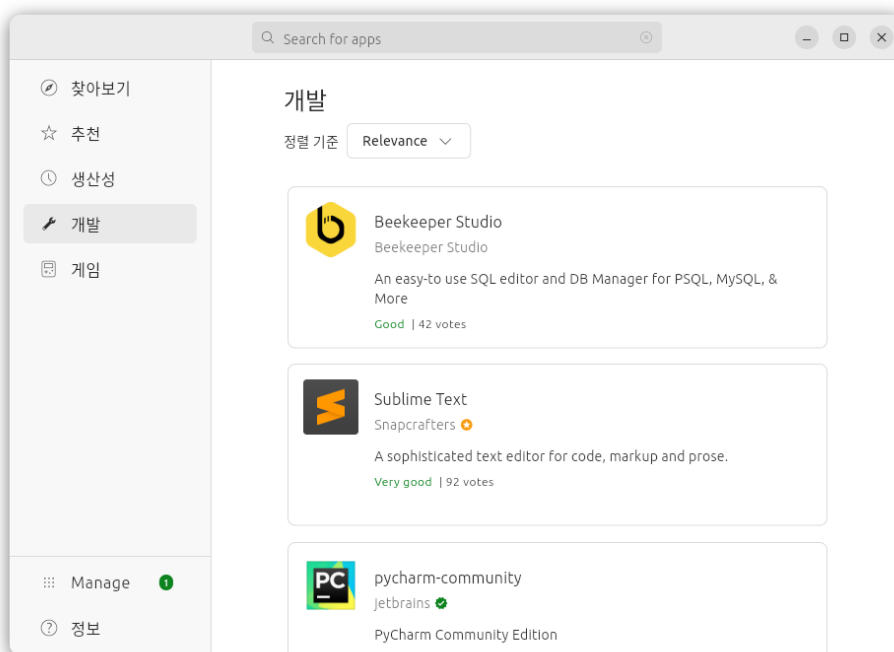


9.1 소프트웨어 센터

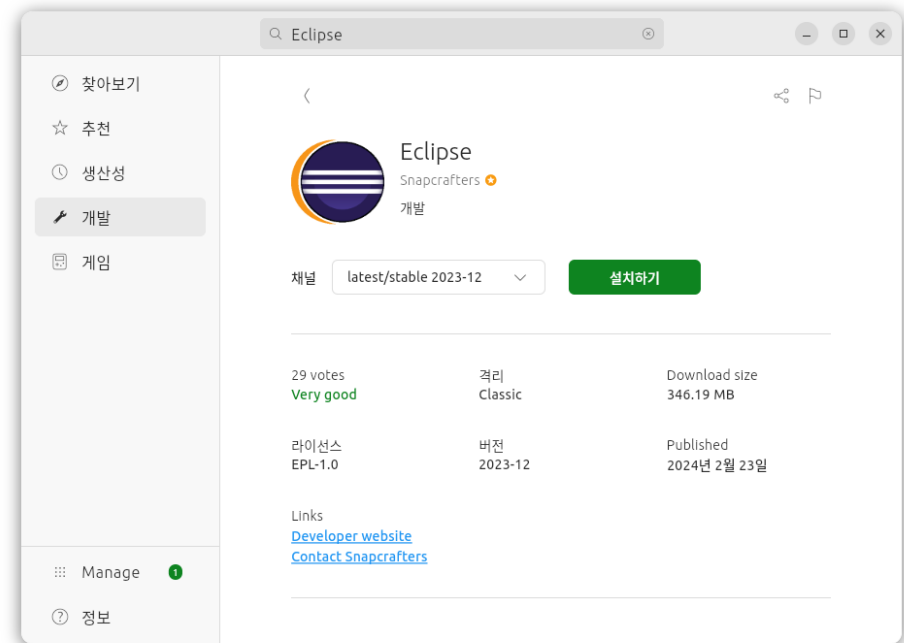
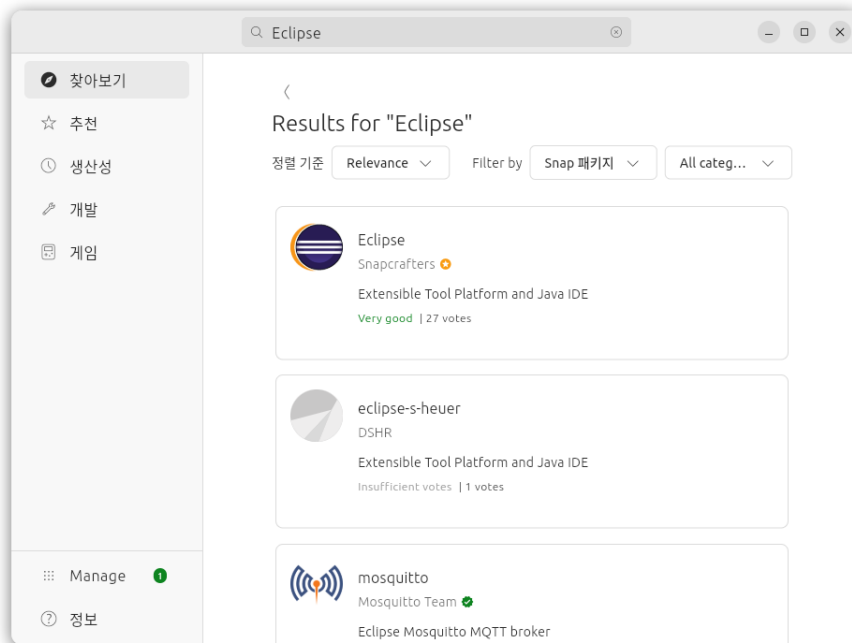
- 간단하게 소프트웨어를 설치하거나 삭제하기 위한 앱 스토어
- 여러 패키지 모음을 분류하여 관리
 - 카테고리 별 분류
 - 소프트웨어 검색
 - 프로그램 설치 및 제거
 - 업데이트 관리



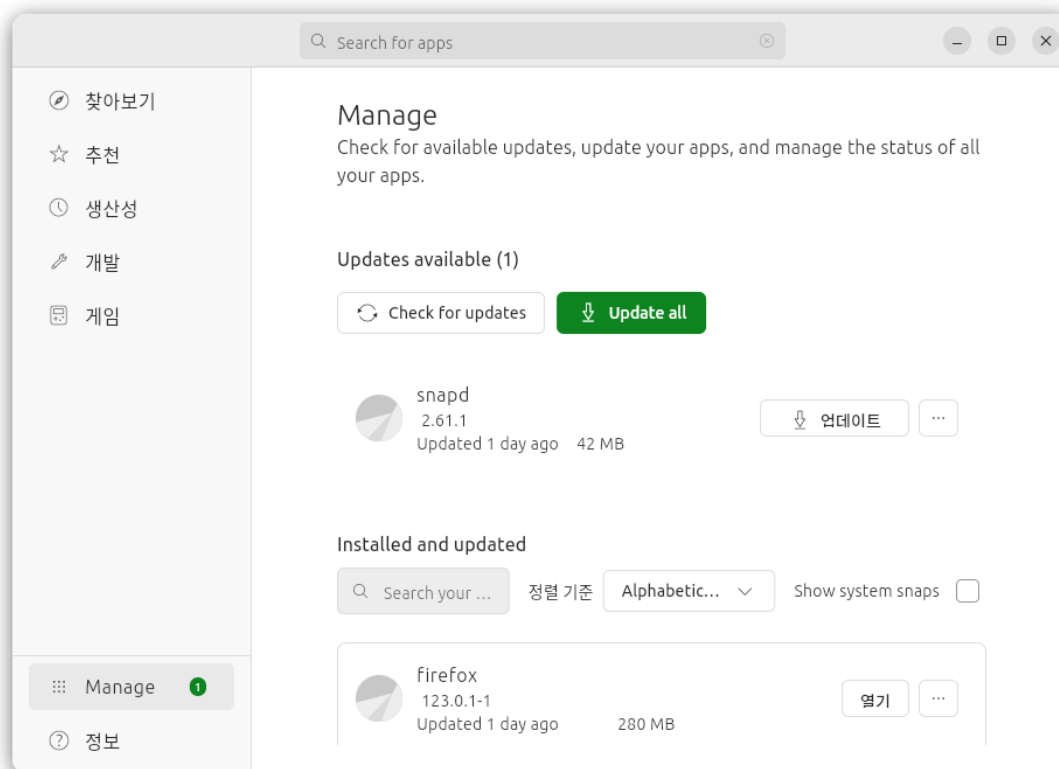
- [개발]
 - 관련 소프트웨어 패키지를 보여줌
 - 우측 상단에 체크 표시가 되어 있는 것은 이미 설치되어 있는 것
 - [설치] 혹은 [제거] 가능



- 원하는 소프트웨어 패키지 검색



- 설치함 [Manage]
 - 설치된 응용 프로그램 목록



9.2 자동 소프트웨어 설치

자동 소프트웨어 설치 도구 apt

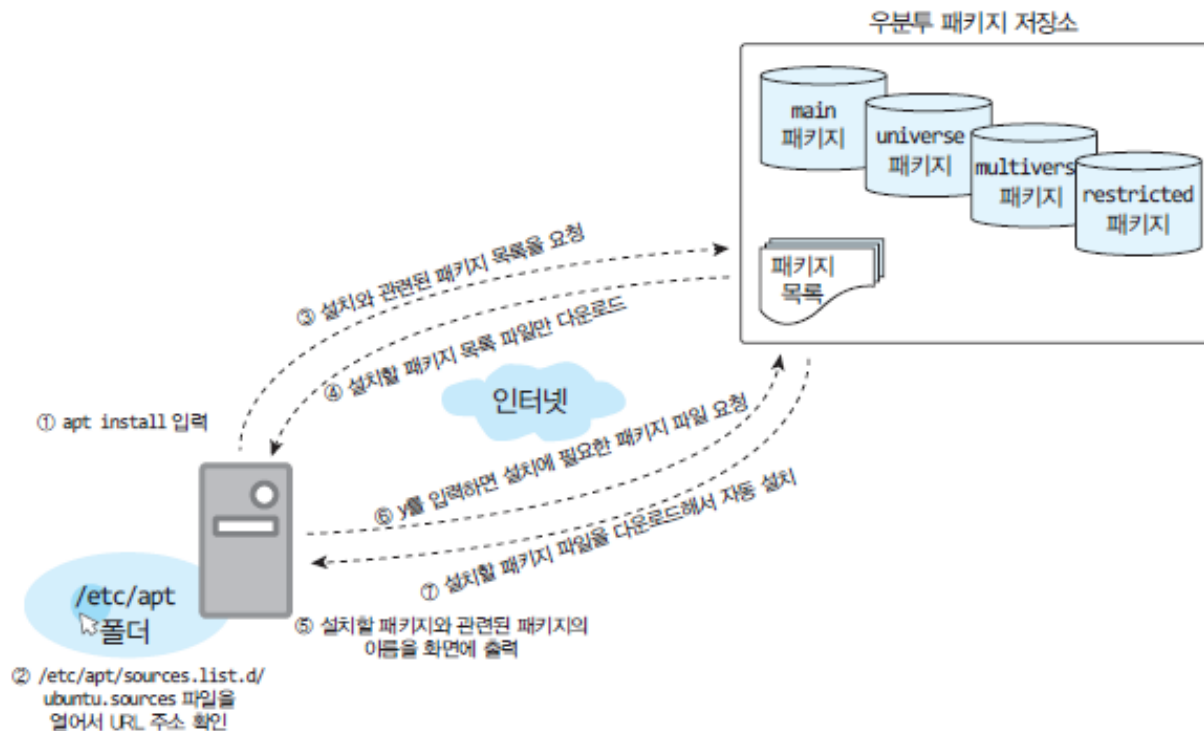


- apt(Advanced Package Tool)
 - 우분투 리눅스에서 사용하는 deb 기반의 시스템
 - 패키지 파일 다운로드부터 설치, 삭제 및 업데이트 자동



1. apt install 패키지명 입력한다.
2. 자동으로 /etc/apt/sources.list.d/ubuntu.sources 파일에서 패키지 저장소에 대한 정보를 확인한다.
3. 설치와 관련된 패키지 관련 목록을 요청한다.
4. 설치할 패키지 관련 목록을 다운로드한다.
5. 다운로드한 패키지 목록 파일을 근거로 사용자가 요청한 패키지와 의존성이 있는 패키지 목록을 화면에 출력한다.
6. y 입력해 실제 패키지 다운로드를 요청한다.
7. 해당 패키지 파일(deb 파일)을 다운로드하여 자동으로 설치한다.

apt 동작 원리



카테고리	설명
main	우분투에서 공식적으로 지원하는 무료(Free) 소프트웨어
universe	우분투에서 지원하지 않는 무료(Free) 소프트웨어
restricted	우분투에서 공식적으로 지원하는 유료(Non-Free) 소프트웨어
multiverse	우분투에서 지원하지 않는 유료(Non-Free) 소프트웨어

apt 명령어	기존 명령어	기능
apt install 패키지+	apt-get install	패키지를 설치한다.
apt remove 패키지+	apt-get remove	패키지를 삭제한다.
apt update	apt-get update	패키지 목록을 갱신한다.
apt upgrade	apt-get upgrade	패키지를 업그레이드한다.
apt autoremove	apt-get autoremove	불필요한 패키지를 제거한다.
apt search 패키지	apt-cache search	키워드로 패키지를 검색한다.
apt show 패키지	apt-cache show	패키지 정보를 출력한다.
apt list		패키지 목록을 리스트한다.

패키지 업데이트 목록 확인/검색



```
# apt list
```

이용 가능한 패키지 목록을 모두 리스팅한다.

```
# apt search 패키지
```

저장소에서 지정된 패키지를 검색하여 리스팅 한다.

```
# apt search vsftpd
```

정렬중... 완료

전체 텍스트 검색... 완료

vsftpd/focal 0.17-36.2 amd64

File Transfer Protocol (FTP) server

vsvsftpd/focal 3.0.3-12 amd64

lightweight, efficient FTP server written for security

...

```
# apt show 패키지
```

지정된 소프트웨어 패키지에 대한 정보를 출력한다.

```
# apt show vsftpd
Package: vsftpd
Version: 1.30-2
Priority: optional
Section: universe/web
Origin: Ubuntu
...
```

패키지 설치/업그레이드/삭제



```
# apt install 패키지+
```

지정된 패키지들을 자동으로 내려받아 설치하거나 업그레이드한다.

```
# apt upgrade
```

설치되어 있는 모든 패키지 중에 새로운 버전이 있는 패키지를 업그레이드한다.

```
# apt install vsftpd
```

```
# apt remove 패키지+
```

이미 설치된 소프트웨어 패키지를 삭제한다.

```
# apt remove vsftpd
```

9.3데비안 패키지 관리도구

- 데비안 패키지
 - 리눅스의 표준 패키지 중 하나로 확장자는 .deb
 - 설치할 파일들과 이름, 버전, 설명 등의 정보를 포함하는 패키지 파일
- 장점
 - 바이너리 파일로 구성되어 있어 컴파일이 필요 없다.
 - 쉽고 빠른 패키지 설치 및 삭제가 가능하다.
 - 기존에 설치한 패키지를 삭제하지 않고 바로 업그레이드 가능하다.
 - 패키지의 설치 상태를 검증할 수 있다.
 - 질의를 통하여 패키지에 대한 자세한 정보를 확인할 수 있다.

데비안 패키지 매니저 dpkg



모드	사용법	설명
패키지 목록 출력	# dpkg -l [패키지명]	설치된 패키지 목록을 출력
패키지 설치	# dpkg -i 패키지파일	패키지를 설치하거나 업그레이드
패키지 상세 정보	# dpkg -s 패키지명	패키지의 상세 정보를 출력
패키지 삭제	# dpkg -r 패키지명	패키지를 삭제
설치 파일 목록	# dpkg -L 패키지명	지정한 패키지가 설치한 파일 목록 출력
파일 검색	# dpkg -S 경로명	지정한 파일이 포함된 패키지 검색
패키지 파일 내용	# dpkg -c 패키지파일	패키지 파일의 내용 출력
패키지 파일 추출	# dpkg -x 패키지파일 디렉터리	패키지 파일의 내용을 지정한 디렉터리에 추출

```
# dpkg -i 패키지파일
```

패키지파일로부터 패키지를 새로 설치한다.

이미 설치되어 있으면 패키지를 업그레이드한다.

```
# dpkg -i google-chrome-stable_current_amd64.deb
```

Selecting previously unselected package google-chrome-stable.

(데이터베이스 읽는중 ...현재 246907개의 파일과 디렉터리가 설치되어 있다.)

Preparing to unpack google-chrome-stable_current_amd64.deb ...

Unpacking google-chrome-stable (85.0.4183.102-1) ...

google-chrome-stable (85.0.4183.102-1) 설정하는 중입니다 ...

update-alternatives: using /usr/bin/google-chrome-stable to provide /usr/bin/x-www-browser (x-www-browser) in auto mode

...

```
# dpkg -s 패키지명
```

설치되어 있는 패키지에 대한 상세 정보를 출력한다.

```
# dpkg -s google-chrome-stable
```

Package: google-chrome-stable

Status: install ok installed

Priority: optional

Section: web

Installed-Size: 230940

Maintainer: Chrome Linux Team <chromium-dev@chromium.org>

Architecture: amd64

Version: 85.0.4183.102-1

Provides: www-browser

...

Description: The web browser from Google

```
# dpkg -i [패키지명]
```

설치되어 있는 패키지 목록을 출력한다.

```
# dpkg -l google-chrome-stable
```

희망상태=알수없음(U)/설치(I)/지우기(R)/깨끗이(P)/고정(H)

| 상태=아님(N)/설치(I)/설정(C)/풀림(U)/절반설정(F)/일부설치(H)/트리거대기(W)/

| / 트리거밀림(T)

|/ 오류?=(없음)/다시설치필요(R) (상태, 오류가 대문자=불량)

||/ 이름 버전 Architecture 설명

```
+++-----  
=====
```

```
ii google-chrome-stable 85.0.4183.102-1 amd64 The web browser from Google
```

```
# dpkg -r 패키지명
```

이미 설치되어 있는 패키지를 삭제한다.

```
# dpkg -r google-chrome-stable
```

(데이터베이스 읽는중 ...현재 247020개의 파일과 디렉터리가 설치되어 있다.)

Removing google-chrome-stable (85.0.4183.102-1) ...

update-alternatives: using /usr/bin/firefox to provide /usr/bin/x-www-browser (x-www-browser) in auto mode

update-alternatives: using /usr/bin/firefox to provide /usr/bin/gnome-www-browser (gnome-www-browser) in auto mode

...

```
# dpkg -c 패키지파일
```

패키지 파일의 내용을 출력한다.

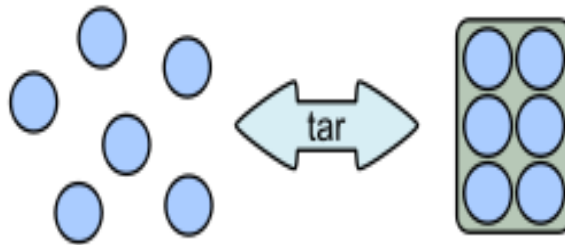
```
# dpkg -c google-chrome-stable_current_amd64.deb
drwxr-xr-x root/root 0 2020-09-04 13:42 ./
drwxr-xr-x root/root 0 2020-09-04 13:42 ./etc/
drwxr-xr-x root/root 0 2020-09-04 13:42 ./etc/cron.daily/
drwxr-xr-x root/root 0 2020-09-04 13:42 ./opt/
drwxr-xr-x root/root 0 2020-09-04 13:42 ./opt/google/
drwxr-xr-x root/root 0 2020-09-04 13:42 ./opt/google/chrome/
...
```

```
# dpkg -x 패키지파일 디렉터리
```

패키지 파일의 내용을 지정한 디렉터리에 푼다.

9.4 아카이브 및 압축

- 아카이브
 - 백업 또는 다른 장소로의 이동을 위해 여러 파일들을 하나로 묶어놓은 묶음
 - 아카이브를 만들거나 푸는데 tar(tape archive) 명령어 사용
- tar의 역학



- tar 명령어
 - 옵션: c(create), v(verbose), x(extract), t(table of contents), f(file)
 - \$ tar -cvf 타르파일 파일+
여러 파일들을 하나의 타르파일로 묶는다. 보통 확장자로 .tar 사용
 - \$ tar -xvf 타르파일
하나의 타르파일을 풀어서 원래 파일들을 복원한다.
 - \$ tar -tvf 타르파일
타르파일의 내용을 확인한다.

tar 아카이브: 사용 예



- 현재 디렉터리에 있는 모든 파일을 다른 곳으로 옮기기

```
$ tar -cvf src.tar *
```

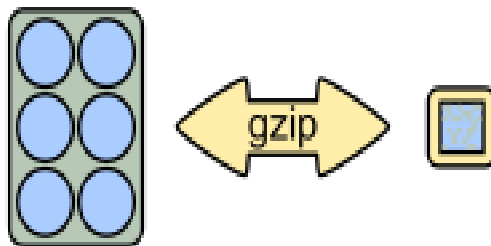
... src.tar를 다른 곳으로 이동

```
$ tar -tvf src.tar
```

```
$ tar -xvf src.tar
```

파일 압축: gzip

- gzip 명령어



\$ gzip [옵션] 파일*

파일(들)을 압축하여 .gz 파일을 만든다.

-d : 압축을 해제한다.

-l : 압축파일 안에 있는 파일 정보(압축된 크기, 압축률) 출력한다.

-r : 하위 디렉터리까지 모두 압축한다.

-v : 압축하거나 풀 때 압축률, 파일명을 출력한다.

- 사용법

```
$ gzip -d 파일.gz*
```

gzip으로 압축된 파일들을 복원한다.

```
$ gunzip 파일.gz*
```

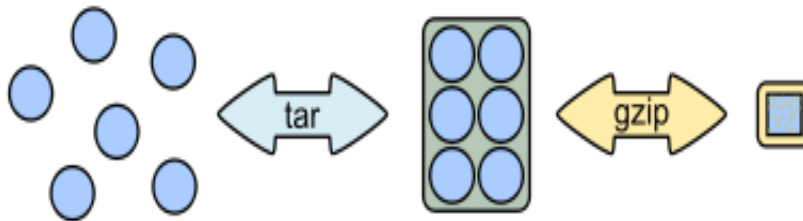
gzip으로 압축된 파일들을 복원한다.

- gzip 명령어 사용법

```
$ gzip 파일*
```

```
$ gzip -d 파일.gz*
```

- 사용 예
 - 파일들을 하나의 타르파일로 묶은 후 gzip을 사용해 압축
 - 파일 복원: 압축을 해제한 후, 타르파일을 풀어서 원래 파일들을 복원



```
$ tar -cvf src.tar *
```

```
$ gzip src.tar
```

... 이 파일을 원하는 곳으로 이동

```
$ gzip -d src.tar.gz
```

```
$ tar -xvf src.tar
```

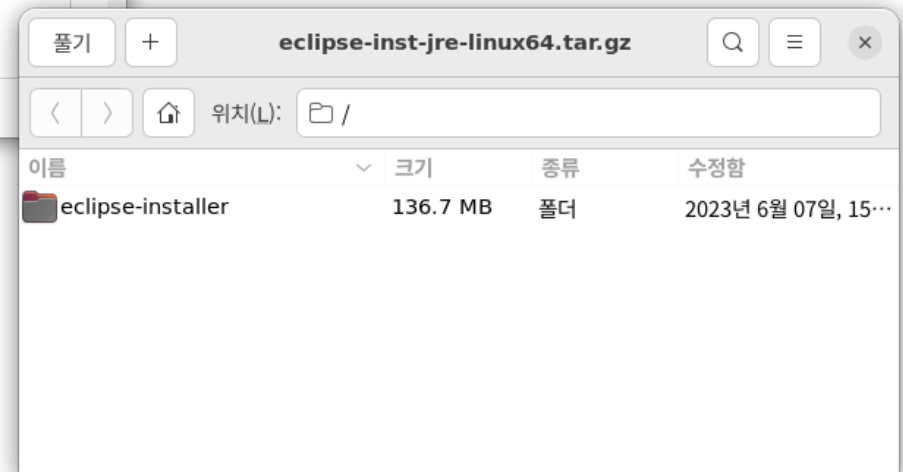
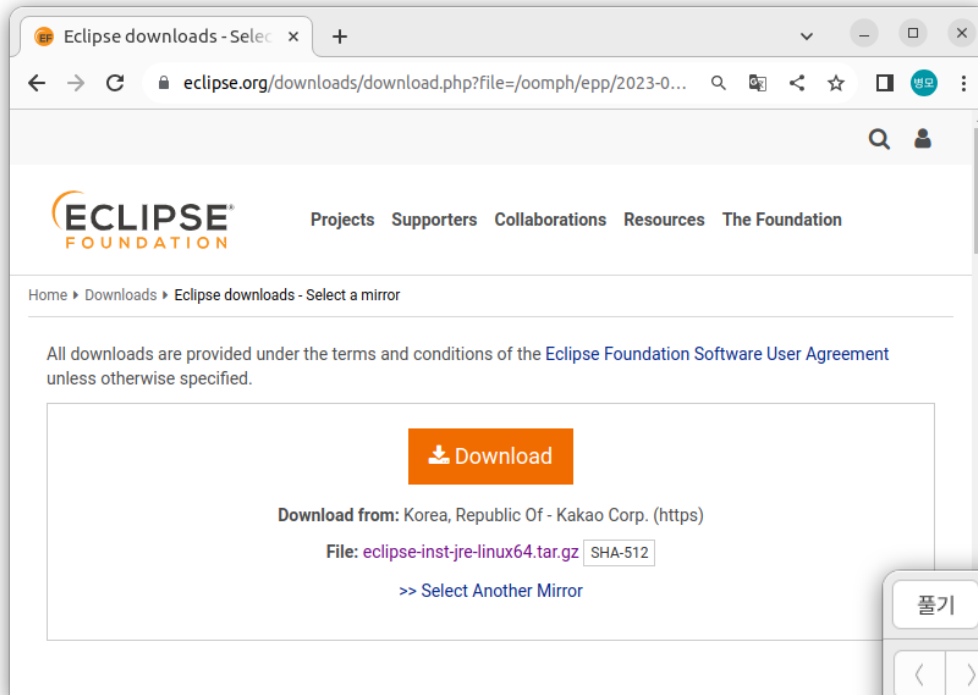
그놈 압축 관리자 Archive Manager



- 파일 압축
 - 압축할 파일을 선택하여 여러 가지 형식으로 압축할 수 있다.
 - 주로 tar.gz, tar.bz2, zip 등의 형식을 지원한다.
- 파일 압축 해제
 - 다양한 압축 형식의 파일을 선택하여 압축을 해제할 수 있다.
- 압축 형식 선택
 - 압축할 때 사용할 형식, 다양한 옵션과 설정을 조정할 수 있다.

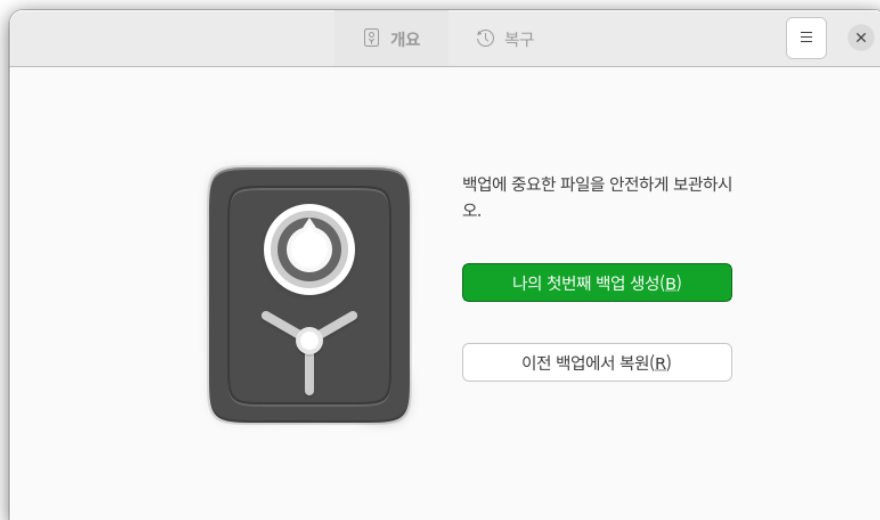


예: Eclipse IDE 설치



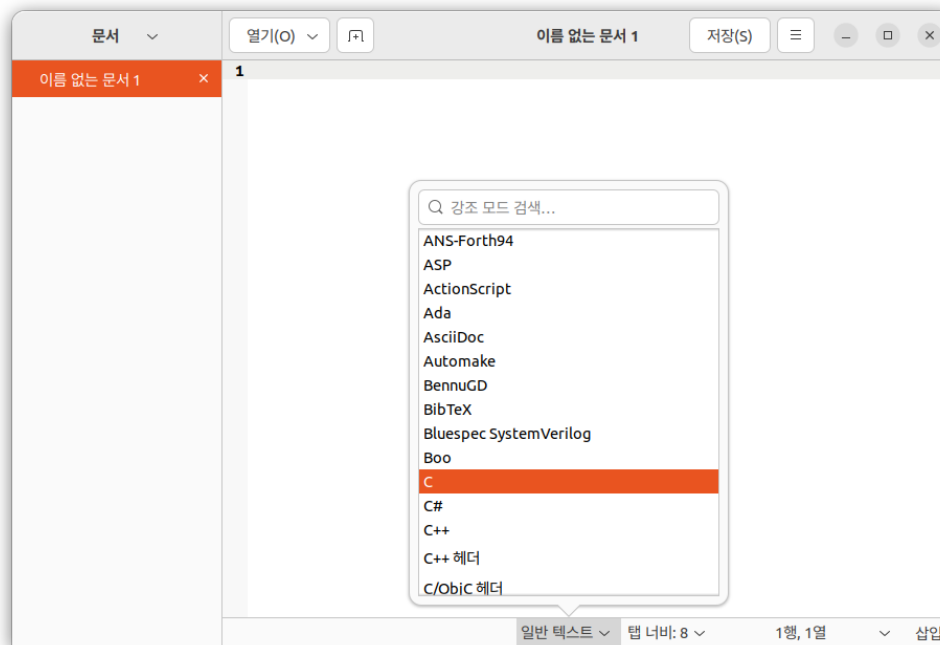
- **Deja Dup 설치**
 - Deja Dup은 일반적으로 그놈 데스크톱 환경에 기본으로 포함되어 있다.
- **Deja Dup 실행**
 - 시스템 메뉴에서 "백업" 또는 "Deja Dup"을 검색하여 실행한다.
- **백업 설정**
 - (1) 처음 실행하면 백업을 설정할 수 있는 창이 열린다.
설정을 통해 백업 주기, 저장 위치, 백업할 파일 등을 조정하여 사용
 - (2) 그림 9.15와 같이 "백업 생성" 버튼을 클릭하여 백업 설정을 시작한다.
 - (3) 백업할 디렉토리를 선택하고, 백업 타겟(로컬 디렉토리, 외부 드라이브, 네트워크 공유 등)을 선택한다.
 - (4) 백업 빈도, 보관 기간 등 추가적인 설정을 할 수 있다.
- **백업 실행**
 - 설정이 완료되면 백그라운드에서 주기적으로 백업을 실행한다.
- **복원**
 - 설정한 백업은 시스템을 복원하거나 파일을 잃었을 때 유용하게 사용됨.

그놈 백업 프로그램 Deja Dup

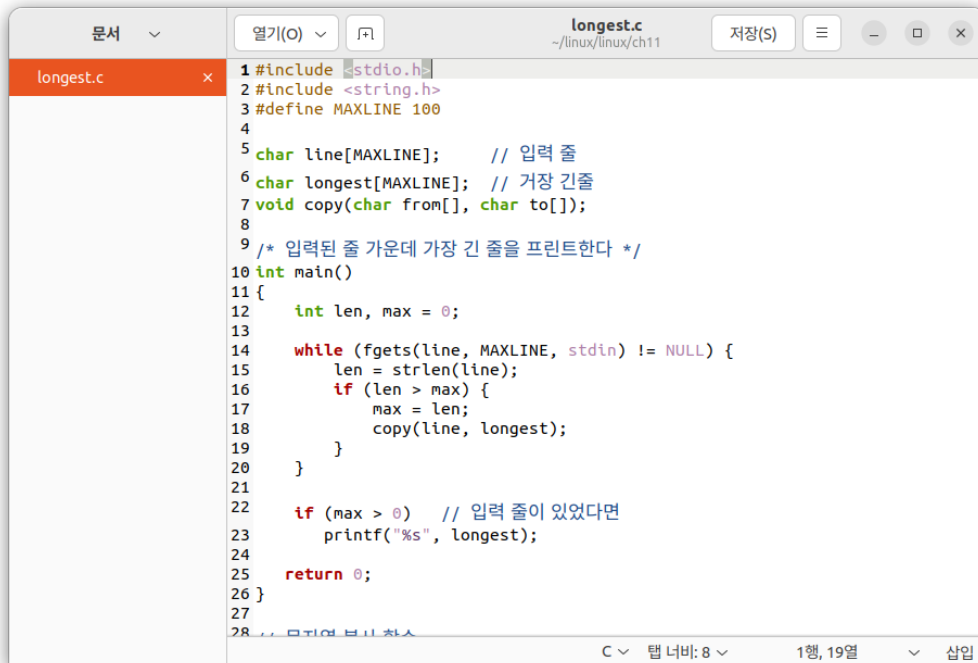


9.5 컴파일과 자동 빌드 도구

- 프로그램 작성
 - gedit 이용
- [보기] 메뉴
 - C 구문 강조 기능 설정



- 프로그램 편집하는 화면
 - #include 같은 전처리 지시자는 황색
 - 주석은 파란색
 - 자료형 이름은 초록색
 - if나 while 같은 문장 키워드는 브라운 색



```
1 #include <stdio.h>
2 #include <string.h>
3 #define MAXLINE 100
4
5 char line[MAXLINE]; // 입력 줄
6 char longest[MAXLINE]; // 저장 긴줄
7 void copy(char from[], char to[]);
8
9 /* 입력된 줄 가운데 가장 긴 줄을 프린트한다 */
10 int main()
11 {
12     int len, max = 0;
13
14     while (fgets(line, MAXLINE, stdin) != NULL) {
15         len = strlen(line);
16         if (len > max) {
17             max = len;
18             copy(line, longest);
19         }
20     }
21
22     if (max > 0) // 입력 줄이 있었다면
23         printf("%s", longest);
24
25     return 0;
26 }
27
28 // 문장 길이 비교 함수
```

단일 모듈 프로그램:longest.c



```
#include <stdio.h>
#include <string.h>
#define MAXLINE 100

char line[MAXLINE]; // 입력 줄
char longest[MAXLINE]; // 가장 긴 줄
void copy(char from[], char to[]);

/*입력 줄 가운데 가장 긴 줄 프린트 */
int main()
{
    int len, max = 0;

    while(fgets(line, MAXLINE, stdin) !=
        NULL) {
        len = strlen(line);
        if (len > max) {
            max = len;
            copy(line, longest);
        }
    }
}
```

```
if (max > 0) // 입력 줄이 있었다면
    printf("%s", longest);

return 0;
}
/* copy: from을 to에 복사; to가 충분히 크
   다고 가정*/
void copy(char from[], char to[])
{
    int i;
    i = 0;
    while ((to[i] = from[i]) != '\0')
        ++i;
}
```

- gcc(GNU cc) 컴파일러

```
$ gcc [-옵션] 파일
```

C 프로그램을 컴파일한다. 옵션을 사용하지 않으면 실행파일 a.out를 생성한다.

- 간단한 컴파일 및 실행

```
$ gcc longest.c
```

```
$ a.out // 실행
```

- -c 옵션: 목적 파일 생성

```
$ gcc -c longest.c
```

- -o 옵션: 실행 파일 생성

```
$ gcc -o longest longest.o 혹은 $ gcc -o longest longest.c
```

- 실행

```
$ longest // 실행
```

- 단일 모듈 프로그램
 - 코드의 재사용(reuse)이 어렵고,
 - 여러 사람이 참여하는 프로그래밍이 어렵다
 - 예를 들어 다른 프로그램에서 copy 함수를 재사용하기 힘들다
- 다중 모듈 프로그램
 - 여러 개의 .c 파일들로 이루어진 프로그램
 - 일반적으로 복잡하며 대단위 프로그램인 경우에 적합



- main 프로그램과 copy 함수를 분리하여 별도 파일로 작성
 - main.c
 - copy.c
 - copy.h // 함수의 프로토타입을 포함하는 헤더 파일
- 컴파일
 - \$ gcc -c main.c
 - \$ gcc -c copy.c
 - \$ gcc -o main main.o copy.o
 - 혹은
 - \$ gcc -o main main.c copy.c

```
#include <stdio.h>
#include <string.h>
#include "copy.h"
char line[MAXLINE]; // 입력 줄
char longest[MAXLINE]; // 가장 긴 줄

/*입력 줄 가운데 가장 긴 줄 프린트 */
int main()
{
    int len, max = 0;

    while (fgets(line,MAXLINE,stdin) != NULL) {
        len = strlen(line);
        if (len > max) {
            max = len;
            copy(line, longest);
        }
    }
    if (max > 0) // 입력 줄이 있었다면
        printf("%s", longest);
    return 0;
}
```

copy.c

copy.h



```
#include <stdio.h>
```

```
/* copy: from을 to에 복사; to가 충분히 크  
   다고 가정*/
```

```
void copy(char from[], char to[])
```

```
{
```

```
    int i;
```

```
    i = 0;
```

```
    while ((to[i] = from[i]) != '\0')
```

```
        ++i;
```

```
}
```

```
#define MAXLINE 100
```

```
void copy(char from[], char to[]);
```

- 다중 모듈 프로그램을 구성하는 일부 파일이 변경된 경우?
 - 변경된 파일만 컴파일하고, 파일들의 의존 관계에 따라서
 - 필요한 파일만 다시 컴파일하여 실행 파일을 만들면 좋다.
- 예
 - copy.c 소스 코드를 수정
 - 목적 파일 copy.o 생성
 - 실행파일을 생성
- make 시스템
 - 대규모 프로그램의 경우에는 헤더, 소스 파일, 목적 파일, 실행 파일의 모든 관계를 기억하고 체계적으로 관리하는 것이 필요
 - make 시스템을 이용하여 효과적으로 작업

- 메이크파일
 - 실행 파일을 만들기 위해 필요한 파일들
 - 그들 사이의 의존 관계
 - 만드는 방법을 기술
- make 시스템
 - 메이크파일을 이용하여 파일의 상호 의존 관계를 파악하여 실행 파일을 쉽게 다시 만듦
- 사용법

```
$ make [-f 메이크파일]
```

make 시스템은 메이크파일(makefile 혹은 Makefile)을 이용하여 보통 실행 파일을 빌드한다. 옵션을 사용하여 별도의 메이크파일을 지정할 수 있다.

메이크파일의 구성



- 메이크파일의 구성 형식

목표(target): 의존리스트(dependencies)

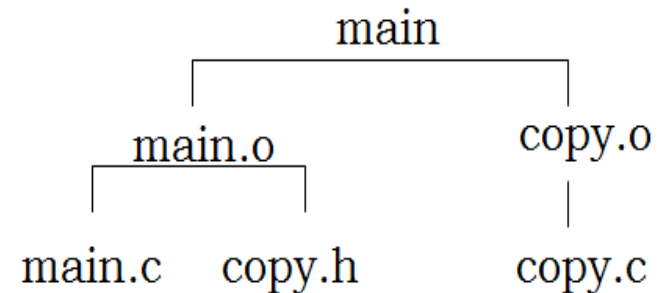
명령리스트(commands)

- 예: Makefile

```
main: main.o copy.o
    gcc -o main main.o
    copy.o
main.o: main.c copy.h
    gcc -c main.c
copy.o: copy.c
    gcc -c copy.c
```

gcc -

- 의존 관계 그래프

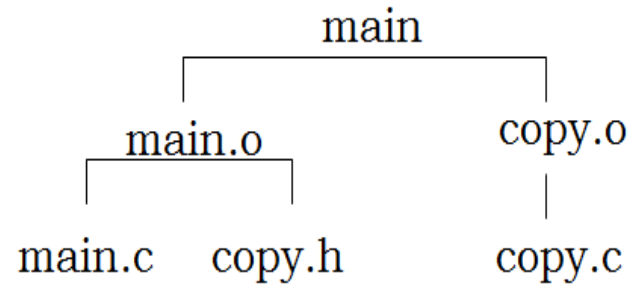


메이크파일의 구성



- make 실행

```
$ make 혹은 $ make main  
gcc -c main.c  
gcc -c copy.c  
gcc -o main main.o  
copy.o
```



- copy.c 파일이 변경된 후

```
$ make  
gcc -c copy.c  
gcc -o main main.o  
copy.o
```

- 우분투 소프트웨어 센터는 간단하게 소프트웨어를 설치하거나 삭제하기 위한 앱 스토어라고 할 수 있다.
- 자동 패키지 관리 도구 apt를 이용하면 패키지 파일 다운로드부터 패키지 설치, 삭제 및 업데이트 등을 자동으로 할 수 있다.
- dpkg 명령어는 데비안 패키지의 배포 및 관리를 위한 패키지 관리 소프트웨어로 이 명령어를 사용하여 설치된 패키지 목록을 확인하고, 패키지를 설치 또는 삭제할 수 있다.
- 리눅스에서는 tar 명령어를 사용하여 여러 파일을 하나로 묶은 후에 gzip 명령어를 사용하여 압축할 수 있다.
- 메이크파일은 make 시스템이 실행 파일을 만드는 등의 빌드 작업을 하기 위해 사용하는 일종의 명세 파일로 파일들 사이에 상호 의존 관계 및 실행 파일을 만드는 방법 등을 기술한다.