

Chapter 06

소프트웨어 관리 및 컴파일



학습목표

- 리눅스의 소프트웨어 패키지에 대해 설명할 수 있다.
- 우분투 리눅스의 패키지를 관리할 수 있다.
- 소프트웨어 소스코드를 작성할 수 있다.
- Makefile과 make 명령어를 활용할 수 있다.



학습내용

- ❖ 소프트웨어 관리
 - 우분투 패키지
 - 패키지 설치 명령어
- ❖ 소프트웨어 컴파일하기
 - 패키지 설치하기
 - 프로그램 작성
 - Make 파일 활용하기





우분투 패키지

리눅스 소프트웨어 설치 방식

- 윈도우OS는 소프트웨어를 설치하기 위해 Setup.exe와 같은 설치 파일을 다운로드한 후 이 파일을 실행하여 설치한다.
 - ✓ 예) 곰플레이어를 설치한다면 곰플레이어 홈페이지에서 GOMPLAYERSETUP.exe를 다운로드한후 실행하면 C:\Program Files (x86)\GRETECH\GOMPlayer\GOM.exe라는 곰플레이어 실행파일이 생성됨
- 리눅스는 윈도우와는 달리 설치파일을 따로 다운로드 받지 않고 바로 실행파일을 설치 디렉토리에 바로 다운로드하는 방식
- 리눅스는 소프트웨어라는 용어 대신 패키지라는 용어로 많이 사용함
- 즉, 리눅스는 설치 관련 명령어를 수행하여 원격 서버에서 패키지를 가져와 설치하는 방식
 - ✓ 예) `$ apt install gcc` : 우분투 리눅스는 apt라는 명령을 통해 원격 서버에서 gcc실행파일을 가져와 바로 설치 디렉토리에 설치



우분투 패키지

리눅스 소프트웨어 설치 방식

- 리눅스 배포판마다 다른 방식의 패키지 설치 명령어와 원격 서버를 사용함.
- 우분투와 같은 데비안 계열은 apt나 dpkg와 같은 명령어를 사용
 - ✓ deb: 데비안, 우분투 계열에서 사용하는 패키지
- CentOS나 페도라 같은 레드햇 계열은 yum이나 rpm과 같은 명령어를 사용
 - ✓ RPM(Redhat Package Manager): 레드햇 계열 리눅스에서 주로 사용



우분투 패키지

우분투 패키지의 특징

- 바이너리 파일(실행파일)로 구성되어 있어 컴파일이 필요 없다.
- 패키지의 파일이 관련 디렉터리에 바로 설치된다.
- 패키지를 삭제할 때 관련된 파일을 일괄적으로 삭제할 수 있다.
- 기존에 설치한 패키지를 삭제하지 않고 바로 업그레이드할 수 있다.
- 패키지의 설치 상태를 검증할 수 있다.
- 패키지에 대한 정보를 제공한다.
- 해당 패키지와 의존성을 가지고 있는 패키지가 무엇인지 알려준다. 따라서 의존성이 있는 패키지를 미리 설치할 수도 있고, apt-get 명령을 사용하면 의존성이 있는 패키지가 자동으로 설치된다.



우분투 패키지

우분투 패키지의 카테고리

- 공식적으로 데비안 배포판에 포함된 모든 패키지는 데비안 자유 소프트웨어 지침에 따라 자유롭게 사용하고 배포할 수 있음
- 우분투도 네 개의 카테고리로 나누어 소프트웨어를 제공
 - ✓ main: 우분투에서 공식적으로 지원하는 무료 소프트웨어
 - ✓ universe: 우분투에서 지원하지 않는 무료 소프트웨어
 - ✓ restricted: 우분투에서 공식적으로 지원하는 유료 소프트웨어
 - ✓ multiverse: 우분투에서 지원하지 않는 유료 소프트웨어



우분투 패키지

우분투 패키지의 이름 구성

파일명_버전-리비전_아키텍처.deb

그림 9-2 우분투 패키지의 이름 구성

- 파일명: 첫 번째 항목은 패키지의 성격을 나타내는 파일명이다.
- 패키지 버전: 두 번째 항목은 패키지의 버전을 의미한다.
- 패키지 리비전: 리비전은 원래 소스의 버전이 업그레이드되지는 않았지만 패키지의 보안 문제나 의존성 변화, 스크립트의 변화 등이 있음을 의미한다.
- 아키텍처: 사용하는 시스템 아키텍처로 i386은 인텔을, all은 시스템과 상관없는 문서나 스크립트 등을 뜻한다.
- 확장자: 확장자는 .deb를 사용한다.



우분투 패키지

우분투 패키지의 이름 구성

```
root@server: ~/다운로드
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
root@server:~/다운로드# ls -l
합계 156
-rw-r--r-- 1 root root 159468 7월  1 14:36 calculator_2.1.4-1_amd64.deb
root@server:~/다운로드#
```

- 패키지명: calculator → 패키지(프로그램)의 이름
- 버전: 2.1.4 → 대개 세 자릿수로 구성 / 주버전, 부버전, 패치 버전 순, 숫자가 높을수록 최신
- 개정번호(revision number): 1 → 문제점을 개선할 때마다 붙이는 번호이며 높을수록 좋음
- 아키텍처: amd64 → 64비트 CPU, 즉 이 파일을 설치할 수 있는 CPU를 말함



우분투 패키지

우분투 패키지 저장소

- 우분투는 패키지와 패키지에 대한 정보를 저장하고 있는 서버인 패키지 저장소라는 개념을 사용
- 패키지 저장소에서는 패키지의 기능 추가나 보안 패치 등 지속적인 업그레이드를 집중적으로 관리
- 사용자는 저장소에 접속하여 최신 패키지를 내려받아 설치 가능
- 패키지 저장소에 대한 정보는 **/etc/apt/sources.list** 파일에 저장
 - ✓ 패키지 유형 : deb는 바이너리 패키지의 저장소를, deb-src는 패키지의 소스 저장소를 의미한다. 보통 한 저장소에 바이너리와 소스를 함께 저장
 - ✓ 저장소 주소 : http 프로토콜을 사용하는 URL 주소를 사용
 - ✓ 우분투 버전 정보 : 저장소에서 관리하는 패키지에 해당하는 우분투의 버전을 표시한다. 버전은 번호가 아니라 버전의 이름을 사용
 - ✓ 카테고리 : 저장소가 가지고 있는 소프트웨어 카테고리(main, restricted 등)를 표시



우분투 패키지

우분투 패키지 저장소

- /etc/apt/sources.list 을 열어보자.
- #으로 시작하는 줄은 주석
- 패키지 저장소의 주소는 <http://kr.archive.ubuntu.com/>

```
root@server:~# cat /etc/apt/sources.list
# deb cdrom:[Ubuntu 18.04.2 LTS _Bionic Beaver_ - Release amd64 (20190210)]/ bi
onic main restricted

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://kr.archive.ubuntu.com/ubuntu/ bionic main restricted
# deb-src http://kr.archive.ubuntu.com/ubuntu/ bionic main restricted

## Major bug fix updates produced after the final release of the
## distribution.
# deb-src http://kr.archive.ubuntu.com/ubuntu/ bionic-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://kr.archive.ubuntu.com/ubuntu/ bionic universe
# deb-src http://kr.archive.ubuntu.com/ubuntu/ bionic universe
# deb-src http://kr.archive.ubuntu.com/ubuntu/ bionic-updates universe

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://kr.archive.ubuntu.com/ubuntu/ bionic multiverse
# deb-src http://kr.archive.ubuntu.com/ubuntu/ bionic multiverse
# deb-src http://kr.archive.ubuntu.com/ubuntu/ bionic-updates multiverse
```



패키지 설치 명령어

dpkg

- 우분투 리눅스에서 패키지(프로그램)를 설치할 때 가장 많이 사용했던 명령어
- 최근에 나온 apt-get이 나오기 전에 주로 사용됨.

apt-get

- dpkg의 확장 개념
- dpkg 기능이 포함되어 있음

apt

- 최근에는 apt-get을 줄여서 apt로 많이 사용됨.



패키지 설치 명령어

dpkg

- -i 또는 --install
 - ✓ 패키지를 설치하는 옵션
- -r 또는 --remove
 - ✓ 설치되어 있는 패키지를 삭제하는 옵션
- -P 또는 --purge
 - ✓ 설치되어 있는 패키지와 설정 파일을 모두 삭제하는 옵션
- -i 또는 -L
 - ✓ 패키지 관련 정보와 파일 목록을 보여주는 옵션

```
dpkg -i 패키지파일명.deb
```

```
dpkg -r 패키지명
```

```
dpkg -P 패키지명
```

```
dpkg -l 패키지명 -- 설치된 패키지의 정보를 보여줌
```

```
dpkg -L 패키지명 -- 패키지가 설치한 파일 목록을 보여줌
```

```
root@server: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
root@server:~# dpkg -l firefox
희망상태=알수없음(U)/설치(I)/지우기(R)/깨끗이(P)/고정(H)
| 상태=아님(N)/설치(I)/설정(C)/풀림(U)/절반설치(F)/일부설치(H)/트리거대기(W)/
| / 트리거밀림(T)
|/ 오류?=(없음)/다시설치필요(R) (상태, 오류가 대문자=불량)
|/ 이름 버전 Architecture 설명
+++-----+
ii  firefox  65.0+build2- amd64      Safe and easy web browser from Mo
root@server:~# dpkg -L firefox
./
```



패키지 설치 명령어

dpkg

- --info 패키지파일명.deb
 - ✓ 아직 설치되지 않은 deb 파일을 조회하는 옵션

dpkg --info 패키지파일명.deb -- 패키지 파일의 정보를 보여줌. 어떤 기능을 설치하기 전에 deb 파일 안에 해당 기능이 포함되었는지 확인

```
root@server: ~/다운로드
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
root@server:~/다운로드# dpkg --info galculator_2.1.4-1_amd64.deb
new Debian package, version 2.0.
size 159468 bytes: control archive=2544 bytes.
    811 bytes,   17 lines   control
   4501 bytes,   58 lines  md5sums
    185 bytes,    7 lines   * postinst      #!/bin/sh
    160 bytes,    5 lines   * postrm       #!/bin/sh
Package: galculator
Version: 2.1.4-1
Architecture: amd64
```



패키지 설치 명령어

dpkg의 단점

- 의존성의 문제
 - ✓ 파이어폭스를 실행하고 싶으면 X 윈도우가 반드시 미리 설치되어야 함
 - ✓ 의존성 문제를 해결한 것이 바로, **apt-get** 명령어
 - ✓ apt-get 은 미리 설치해야 하는 패키지를 알려주거나 알아서 설치하는 방식



패키지 설치 명령어

apt-get

- apt-get 명령어는 *.deb 패키지를 설치하는 편리한 도구
- 우분투가 제공하는 deb 파일 저장소에서 자동으로 deb 파일을 다운로드하여 설치
- → 의존성 문제를 걱정하지 않아도 됨
- dpkg 명령어의 경우, *.deb 파일을 미리 다운로드한 후 설치해야 하는 번거로움이 있음



패키지 설치 명령어

apt-get의 기본 사용법

- apt-get install

- ✓ 패키지 설치 명령어, 패키지를 다운로드한 후 사용자에게 설치 여부를 묻음
- ✓ '-y' 옵션을 넣으면 사용자에게 yes/no를 묻는 부분에서 무조건 yes를 입력한 것으로 간주

`apt-get install 패키지명`

- apt-get update

- ✓ /etc/apt/sources.list 파일의 내용이 수정되면 다운로드할 패키지 목록을 업데이트

`apt-get update`

- apt-get remove

- ✓ 설치되어 있는 패키지를 삭제

`apt-get remove 패키지명`

- apt-get purge

- ✓ 설치되어 있는 패키지와 설정 파일까지 모두 삭제

`apt-get purge 패키지명`



패키지 설치 명령어

apt-get의 기본 사용법

- apt-get autoremove
 - ✓ 사용하지 않는 패키지를 모두 삭제
- apt-get clean 또는 apt-get autoclean
 - ✓ 설치할 때 다운로드한 파일과 과거의 파일을 삭제
- apt-cache show
 - ✓ 패키지의 정보를 보여줌
- apt-cache depends
 - ✓ 패키지의 의존성을 보여줌
- apt-cache rdepends
 - ✓ 패키지에 의존하는 다른 패키지의 목록을 보여줌

```
apt-get autoremove
```

```
apt-get clean 또는 apt-get autoclean
```

```
apt-cache show 패키지명
```

```
apt-cache depends 패키지명
```

```
apt-cache rdepends 패키지명
```



패키지 설치 명령어

apt-get을 사용하여 mc 패키지 설치하기

- ① 설치할 패키지의 정보를 apt-cache show mc 명령으로 확인

```
root@server: ~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
root@server:~# apt-cache show mc  
Package: mc  
Architecture: amd64  
Version: 3:4.8.19-1  
Priority: optional  
Section: universe/utils  
Maintainer: Ubuntu Maintainers <ubuntu-maint@ubuntu.com>  
Original-Maintainer: Debian MC Team <mc-devel@lists.debian.org>  
SHA1: 439660e87f3c9fe0b6f448fb91b3f3a10cc0d4  
SHA256: 494fe386a93862c523b6c6bab766f21e77717bc6601cb7af59960e67c5162a72  
Homepage: https://www.midnight-commander.org  
Description-en: Midnight Commander - a powerful file manager  
GNU Midnight Commander is a text-mode full-screen file manager. It  
uses a two panel interface and a subshell for command execution. It  
includes an internal editor with syntax highlighting and an internal  
viewer with support for binary files. Also included is Virtual  
Filesystem (VFS), that allows files on remote systems (e.g. FTP, SSH
```

- ② apt-cache depends mc 명령으로 의존성 정보도 확인

```
root@server: ~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
root@server:~# apt-cache depends mc  
mc  
의존: e2fslibs  
      libext2fs2  
의존: libc6  
의존: libglib2.0-0
```



패키지 설치 명령어

apt-get을 사용하여 mc 패키지 설치하기

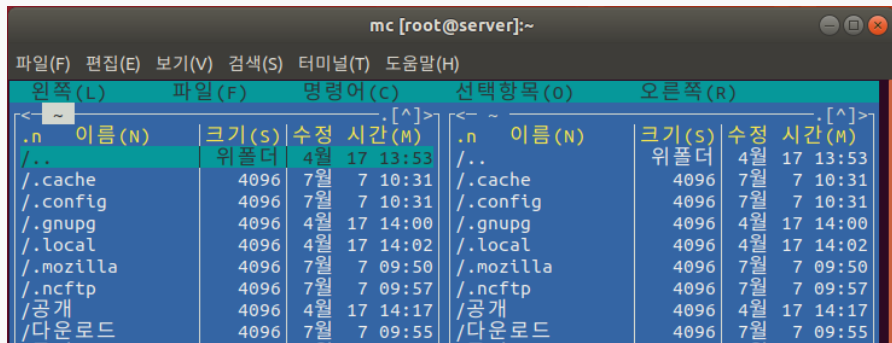
- ③ apt-get install mc 명령으로 패키지 설치 진행

'y'를 입력하면 관련 패키지가 모두 다운로드된 후 자동으로 설치됨

정상적으로 설치되면 'Unpacking, Selecting, Processing ...' 등의 메시지가 나타남

```
libssh2-1:amd64 (1.8.0-1) 설정하는 중입니다 ...  
Processing triggers for libc-bin (2.27-3ubuntu1) ...  
mc (3:4.8.19-1) 설정하는 중입니다 ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...  
Processing triggers for hicolor-icon-theme (0.17-2) ...  
root@server:~#
```

- ④ mc 명령을 입력하면 깔끔한 화면의 파일 관리자가 실행됨



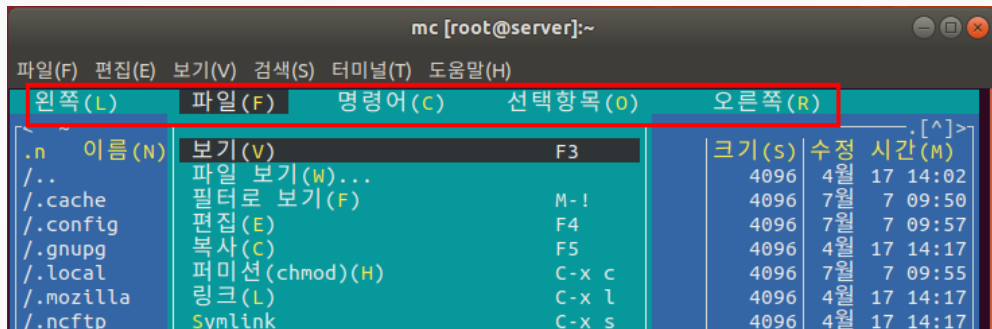
<~ 이름(N)	크기(s)	수정 시간(M)	<~ 이름(N)	크기(s)	수정 시간(M)
./	위폴더	4월 17 13:53	./	위폴더	4월 17 13:53
/.cache	4096	7월 7 10:31	/.cache	4096	7월 7 10:31
/.config	4096	7월 7 10:31	/.config	4096	7월 7 10:31
/.gnupg	4096	4월 17 14:00	/.gnupg	4096	4월 17 14:00
/.local	4096	4월 17 14:02	/.local	4096	4월 17 14:02
/.mozilla	4096	7월 7 09:50	/.mozilla	4096	7월 7 09:50
/.ncftp	4096	7월 7 09:57	/.ncftp	4096	7월 7 09:57
/공개	4096	4월 17 14:17	/공개	4096	4월 17 14:17
/다운로드	4096	7월 7 09:55	/다운로드	4096	7월 7 09:55



패키지 설치 명령어

apt-get을 사용하여 mc 패키지 설치하기

- ⑤ 상단의 메뉴를 마우스로 클릭하면 파일이나 폴더 관리 가능



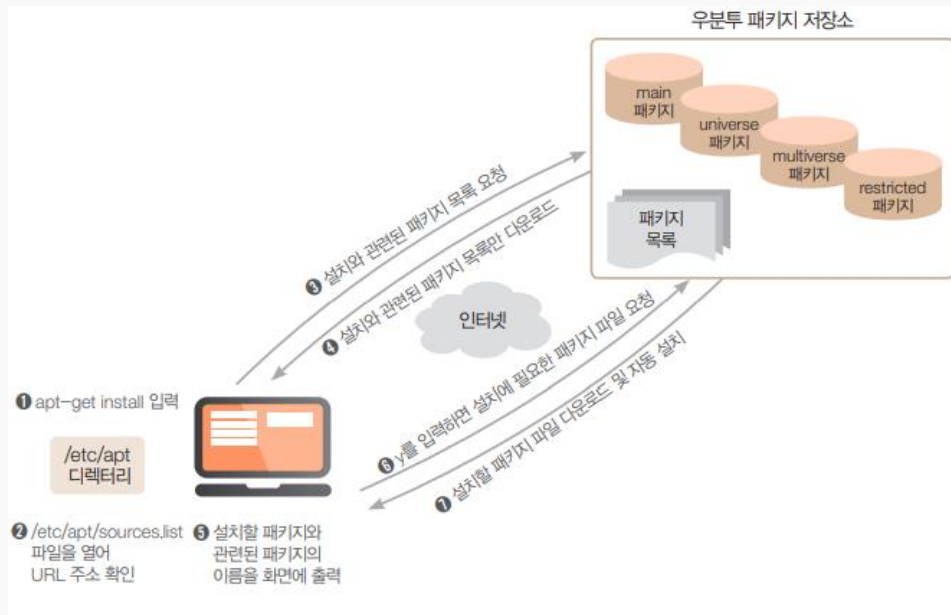
- ⑥ 메뉴의 [파일]-[끝내기] 선택 또는 exit 명령 입력, mc 프로그램 종료



패키지 설치 명령어

apt-get의 작동 방식과 설정 파일

- ① apt-get install mc 명령을 입력하면
- ② 자동으로 /etc/apt/ 디렉터리의 핵심 파일인 sources.list를 확인
- ③ 설치할 패키지와 관련된 목록 요청
- ④ 설치할 패키지와 관련된 목록만 다운로드
- ⑤ 사용자는 패키지 목록을 확인한 후 설치 의향 있으면 'y'를 입력, 실제 패키지 다운로드를 요청
- ⑥ 패키지 파일(deb 파일)이 다운로드되어 자동으로 설치됨
- ⑦ apt-get -y install mc 명령을 실행하면 ②~⑦이 한번에 이루어짐





패키지 설치 명령어

미러(mirror) 사이트

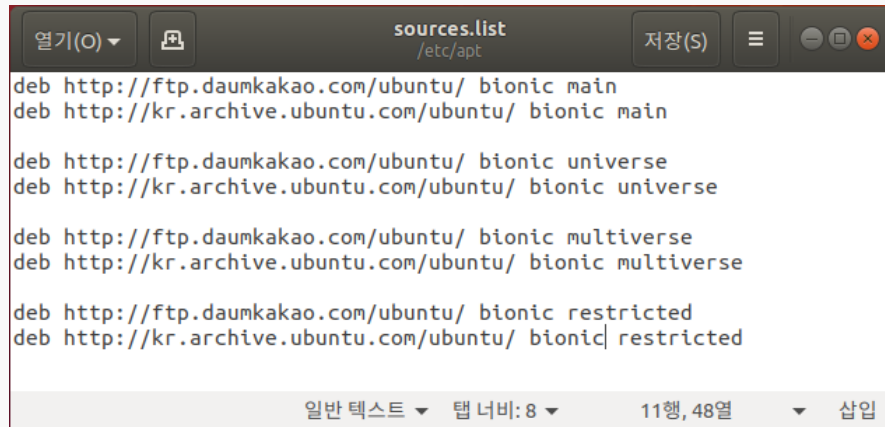
- 우분투 패키지 저장소는 우분투 사이트(<http://www.ubuntu.com>)에서 제공
- 전 세계적으로 동일한 저장소가 수백 개 존재
 - ✓ 대학, 연구소, 기업 등이 자발적으로 구축한 것, 우리나라의 기업과 대학도 참여
- 이러한 저장소를 미러(mirror) 사이트라고 함
 - ✓ <http://launchpad.net/ubuntu/+cdmirrors>
- 일반 사용자의 경우, `apt-get -y install 패키지명` 명령을 실행하면 `sources.list`에 기록된 사이트에 자동으로 접속해서 다운로드 가능



패키지 설치 명령어

/etc/apt/sources.list 파일 구성

- 각 행은 'deb 우분투패키지저장소URL 버전코드명 저장소종류'를 의미
- 첫 행을 보면 우분투 패키지 저장소 URL이 <http://ftp.daumkakao.com/ubuntu/>로, 버전 코드명은 18.04 LTS를 의미하는 bionic으로, 저장소 종류는 main으로 지정되어 있음
- 첫 번째 행과 두 번째 행은 같은 내용임 → 첫 번째 행의 사이트가 작동하지 않을 것에 대비해 두 번째 행에 추가해둔 것
- bionic은 우분투 18.04 LTS가 출시된 시점에 제공되는 패키지 버전만 설치하겠다는 의미
- 그 이후에 업그레이드된 최신 버전의 패키지를 설치하고 싶을 때는 'bionic'을 'bionic-updates'로 수정하여 바로 아래 행에 추가하면 됨
- 만약 무료 제품만 사용하고 싶다면 main과 universe가 있는 행만 남기고 나머지는 맨 앞에 #를 붙여서 주석 처리



```
deb http://ftp.daumkakao.com/ubuntu/ bionic main
deb http://kr.archive.ubuntu.com/ubuntu/ bionic main

deb http://ftp.daumkakao.com/ubuntu/ bionic universe
deb http://kr.archive.ubuntu.com/ubuntu/ bionic universe

deb http://ftp.daumkakao.com/ubuntu/ bionic multiverse
deb http://kr.archive.ubuntu.com/ubuntu/ bionic multiverse

deb http://ftp.daumkakao.com/ubuntu/ bionic restricted
deb http://kr.archive.ubuntu.com/ubuntu/ bionic restricted
```



패키지 설치 명령어

자주 사용하는 apt-get 명령어

- 패키지를 설치하거나 업데이트할 때 대개 아래 3가지 명령만 실행하면 됨.
 - ✓ `$ sudo apt-get update` : 사용할 수 있는 패키지에 대한 정보 갱신
 - ✓ `$ sudo apt-get install 패키지` : 특정 패키지 설치
 - ✓ `$ sudo apt-get upgrade` : 기존 설치된 패키지를 모두 자동으로 최신 버전으로 upgrade



패키지 설치 명령어

apt-get update

- 패키지 정보 업데이트하기 : update
- /etc/apt/sources.list에 명시한 저장소에서 패키지 정보를 읽어 동기화
- 새로운 패키지 정보를 가져와서 APT 캐시를 수정

```
root@server:~# apt-get update
기존:1 http://kr.archive.ubuntu.com/ubuntu bionic InRelease
패키지 목록을 읽는 중입니다... 완료
```



패키지 설치 명령어

apt-get upgrade

- 패키지 업그레이드하기 : upgrade
- 현재 설치되어 있는 모든 패키지 중에서 새로운 버전이 있는 패키지를 모두 업그레이드

```
root@server:~# apt-get upgrade
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
업그레이드를 계산하는 중입니다... 완료
0개 업그레이드, 0개 새로 설치, 0개 제거 및 0개 업그레이드 안 함.
```



패키지 설치 명령어

apt-get install

- 특정 패키지 설치 또는 업그레이드하기 : install
- 여러 패키지를 한 번에 설치하려면 다음과 같이 패키지 이름을 나열

```
sudo apt-get install nethogs goaccess
```

- 패키지를 설치할 때 업그레이드를 하지 않으려면 '--no-upgrade' 옵션을 사용

```
sudo apt-get install netcat --no-upgrade
```

- 새로운 패키지를 설치하지 않고 업그레이드만 할 때는 '--only-upgrade' 옵션을 사용

```
sudo apt-get install netcat --only-upgrade
```



패키지 설치하기

C언어 컴파일러 설치

- C 언어로 작성한 프로그램을 컴파일하기 위해서는 C 컴파일러가 필요
- 리눅스에서 사용하는 C 컴파일러는 GNU C 컴파일러로 패키지 이름이 gcc
- 컴파일러를 설치한 후 C언어 소스 코드를 작성
- 소스코드를 컴파일 한 후 실행

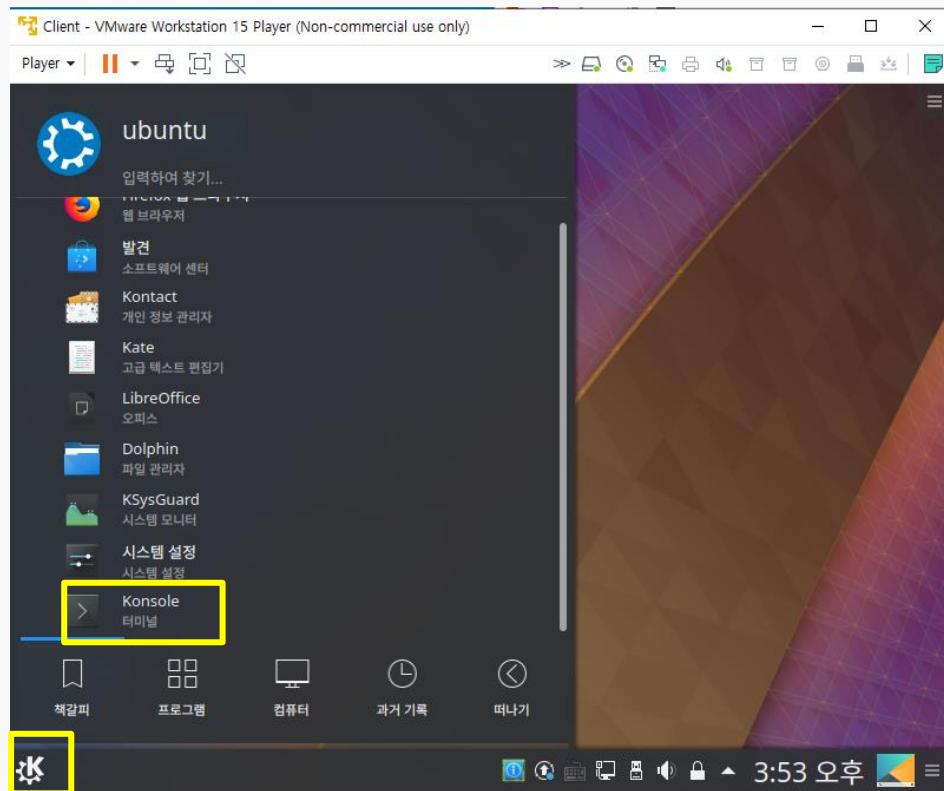
소프트웨어 컴파일하기



패키지 설치하기

C언어 컴파일러 설치

- gcc 설치
 - ① Client 가상 머신을 시작한다.
 - ② 왼쪽 아래 단에 있는 K마크를 클릭한 후 Kconsole 선택한다. 이 Kconsole은 Server머신의 터미널에 해당



소프트웨어 컴파일하기



패키지 설치하기

C언어 컴파일러 설치

- gcc 설치

③ \$ sudo apt install gcc를 입력하여 설치 시도

④ 계속 하시겠습니까? [Y/n] 이 나오면 y를 선택

⑤ 설치 작업이 끝나면 이와 같은 화면 출력

```
ubuntu@client:~$ sudo apt install gcc
[sudo] ubuntu의 비밀번호:
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
다음의 추가 패키지가 설치될 것입니다 :
binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-7 gcc-7 gcc-7-base
gcc-8-base libasan4 libatomic1 libbinutils libc-dev-bin libc6-dev libcc1-0
libcilkrts5 libgcc-7-dev libgcc1 libgomp1 libitm1 liblsan0 libmpx2 libquadmath0
libstdc++6 libtsan0 libubsan0 linux-libc-dev manpages-dev
제안하는 패키지:
binutils-doc cpp-doc gcc-7-locales gcc-multilib make autoconf automake libtool flex
bison gcc-doc gcc-7-multilib gcc-7-doc libgcc1-dbg libgomp1-dbg libitm1-dbg
libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg
libmpx2-dbg libquadmath0-dbg glibc-doc
다음 세 패키지를 설치할 것입니다:
binutils binutils-common binutils-x86-64-linux-gnu gcc gcc-7 libasan4 libatomic1
libbinutils libc-dev-bin libc6-dev libcilkrts5 libgcc-7-dev libitm1 liblsan0 libmpx2
libquadmath0 libtsan0 libubsan0 linux-libc-dev manpages-dev
다음 패키지를 업그레이드할 것입니다:
cpp cpp-7 gcc-7-base gcc-8-base libcc1-0 libgcc1 libgomp1 libstdc++6
8개 업그레이드, 20개 새로 설치, 0개 제거 및 606개 업그레이드 안 함.
26.8 M바이트 아카이브를 받아야 합니다.
이 작업 후 89.3 M바이트의 디스크 공간을 더 사용하게 됩니다.
계속 하시겠습니까? [Y/n]
```

```
cpp-7 (7.4.0-1ubuntu1~18.04.1) 설정하는 중입니다 ...
binutils-x86-64-linux-gnu (2.30-21ubuntu1~18.04.2) 설정하는 중입니다 ...
cpp (4:7.4.0-1ubuntu2.3) 설정하는 중입니다 ...
binutils (2.30-21ubuntu1~18.04.2) 설정하는 중입니다 ...
gcc-7 (7.4.0-1ubuntu1~18.04.1) 설정하는 중입니다 ...
gcc (4:7.4.0-1ubuntu2.3) 설정하는 중입니다 ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
ubuntu@client:~$
```



패키지 설치하기

C언어 컴파일러 설치

- gcc 설치

⑥ 설치 완료 확인을 위해 gcc의 버전을 확인한다. 버전을 확인하는 옵션은 -v 임.

⑦ 그림의 gcc 버전은 7.4.0

```
ubuntu@client:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.4.0-1ubuntu1~18.04.1' --with-bugurl=file:///usr/share/doc/gcc-7/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++ --prefix=/usr --with-gcc-major-version-only --program-suffix=-7 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-libmpx --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib --enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch=32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04.1)
ubuntu@client:~$
```



패키지 설치하기

문서 편집기 gedit 설치

- Gedit 설치

- ① \$ sudo apt install gedit
- ② 계속 하시겠습니까? [Y/n]에서 y 선택

```
ubuntu@client:~$ sudo apt install gedit
[sudo] ubuntu의 암호:
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
다음의 추가 패키지가 설치될 것입니다 :
  gedit-common gir1.2-gtksource-3.0 gir1.2-peas-1.0 gstreamer1.0-glib libgraphene-1.0-0
  libgspell-1-1 libgspell-1-common libgstreamer-glib1.0-0 libgstreamer-plugins-base1.0-0
  libgstreamer1.0-0 libgtksourceview-3.0-1 libgtksourceview-3.0-common
  libjavascriptcoregtk-4.0-18 libpeas-1.0-0 libpeas-common libwebkit2gtk-4.0-37
  libyelp0 python3-gi python3-gi-cairo yelp yelp-xsl zenity zenity-common
제안하는 패키지 :
  gedit-plugins libvisual-0.4-plugins gstreamer1.0-tools libwebkit2gtk-4.0-37-gtk2
다음 새 패키지를 설치할 것입니다 :
  gedit gedit-common gir1.2-gtksource-3.0 gir1.2-peas-1.0 gstreamer1.0-glib
  libgraphene-1.0-0 libgspell-1-1 libgspell-1-common libgstreamer-glib1.0-0
  libgtksourceview-3.0-1 libgtksourceview-3.0-common libjavascriptcoregtk-4.0-18
  libpeas-1.0-0 libpeas-common libwebkit2gtk-4.0-37 libyelp0 python3-gi-cairo yelp
  yelp-xsl zenity zenity-common
다음 패키지를 업그레이드할 것입니다 :
  libgstreamer-plugins-base1.0-0 libgstreamer1.0-0 python3-gi
3개 업그레이드, 21개 새로 설치, 0개 제거 및 603개 업그레이드 안 함.
20.9 M바이트 아카이브를 받아야 합니다.
이 작업 후 84.0 M바이트의 디스크 공간을 더 사용하게 됩니다.
계속 하시겠습니까? [Y/n] y
```




패키지 설치하기

Make 명령어 설치

- Make 명령어 설치

① \$ sudo apt install make

```
ubuntu@client:~$ sudo apt install make
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
제안하는 패키지:
  make-doc
다음 새 패키지를 설치할 것입니다:
  make
0개 업그레이드, 1개 새로 설치, 0개 제거 및 603개 업그레이드 안 함.
154 k바이트 아카이브를 받아야 합니다.
이 작업 후 381 k바이트의 디스크 공간을 더 사용하게 됩니다.
받기:1 http://kr.archive.ubuntu.com/ubuntu bionic/main amd64 make amd64 4.1-9.1ubuntu1 [
154 kB]
내려받기 154 k바이트, 소요시간 3초 (57.5 k바이트/초)
Selecting previously unselected package make.
(데이터베이스 읽는중 ...현재 187507개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack .../make_4.1-9.1ubuntu1_amd64.deb ...
Unpacking make (4.1-9.1ubuntu1) ...
make (4.1-9.1ubuntu1) 설정하는 중입니다 ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
ubuntu@client:~$
```

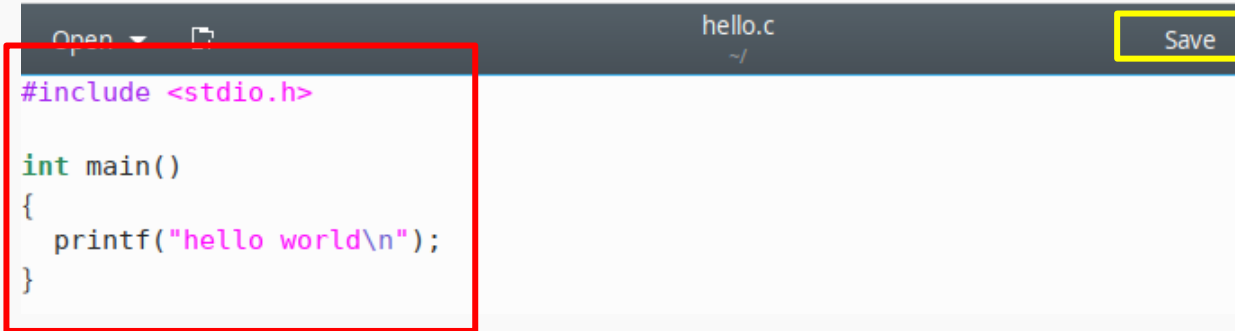


프로그램 작성

C 프로그램 작성하기

- C언어로 작성된 소스 코드를 작성한다.
- 소스 코드의 이름은 hello.c
- 컴파일 후 실행하면 화면에 “hello world”문자를 출력하는 프로그램
- 문서 편집기 gedit를 실행하여 다음과 같은 코드를 작성한 후 저장한다.

✓ \$ gedit hello.c



```
Open [icon] hello.c ~/  
[Save]  
#include <stdio.h>  
  
int main()  
{  
    printf("hello world\n");  
}
```



프로그램 작성

C 프로그램 컴파일하기

- C언어 컴파일러인 gcc를 사용하여 hello.c를 컴파일한다.
 - ✓ `$ gcc ./hello.c`
- 생성된 실행파일 이름은 a.out
- 실행하기
 - ✓ 현재 디렉토리인 . 을 이용하여 ./a.out이라고 입력하면 실행

```
ubuntu@client:~$ gcc ./hello.c
ubuntu@client:~$ ls
a.out hello.c 공개 다운로드 문서 바탕화면
ubuntu@client:~$ ./a.out
hello world
ubuntu@client:~$
```



프로그램 작성

원하는 실행파일명 만들기

- 기본 실행 파일명은 a.out이나 원하는 이름으로 지정하려면 -o 옵션을 사용한다.
- Hello라는 실행파일을 만들려면 다음과 같이 입력
 - ✓ `$ gcc -o hello ./hello.c`
- 실행하기
 - ✓ 현재 디렉토리인 . 을 이용하여 ./hello 라고 입력하면 실행

```
ubuntu@client:~$ gcc -o hello hello.c
ubuntu@client:~$ ./hello
hello world
ubuntu@client:~$
```



Make 파일 활용하기

Make명령어

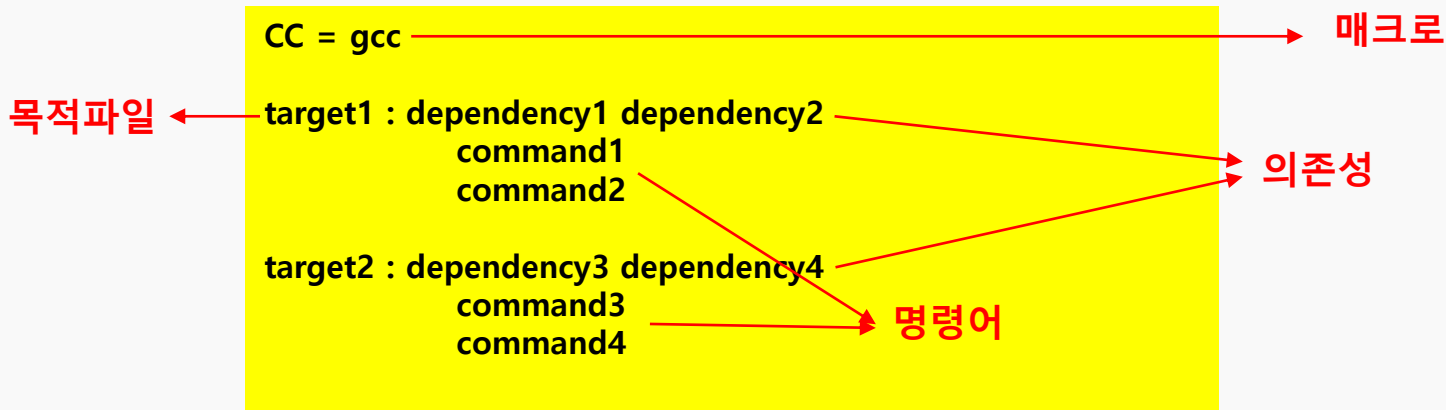
- 여러 개의 소스를 한번에 컴파일하기 위해 make라는 명령을 사용할수 있음
- make라는 명령어는 makefile, Makefile, GnuMakefile이라는 이름의 목록파일을 참조하여 여러 개의 파일 컴파일을 수행함.
- Makefile은 많은 소스 코드 파일의 의존관계를 나타내 줌.
- 의존 관계라는 것은 여러 개의 파일 중 한 파일이 수정되었다면 이 파일에 의해 다시 컴파일 되어야 하는 파일을 모두 찾아 다시 컴파일 해주는 것을 의미함.



Make 파일 활용하기

Makefile의 구성

- 목적파일(Target) : 명령어가 수행되어 나온 결과를 저장할 파일
- 의존성(Dependency) : 목적파일을 만들기 위해 필요한 재료
- 명령어(Command) : 실행 되어야 할 명령어들
- 매크로(macro) : 코드를 단순화 시키기 위한 방법





Make 파일 활용하기

Makefile의 작성 규칙

- 매크로 정의 : Makefile에 정의한 문자열로 치환한다. 앞의 예에서 CC라는 매크로는 gcc로 치환된다.
- 명령어의 시작은 반드시 tab으로 시작한다. 앞의 예에서 command1,2,3,4앞은 반드시 tab을 사용하여 들여쓰기 한다.

CC = gcc

target1 : dependency1 dependency2

 command1
command2

target2 : dependency3 dependency4

 command3
command4



Make 파일 활용하기

Makefile의 작성 규칙

- Make파일 활용 실습을 위해 2개의 C언어 소스 코드 작성
- File1.c와 file2.c를 다음과 같은 내용으로 작성하시오.
- File1.c에서 file2.c의 함수인 file2_func() 함수를 호출하는 구조

```
#include <stdio.h>

extern void file2_func();

int main()
{
    file2_func();
    printf("File1Wn");
}
```

<file1.c>

```
#include <stdio.h>

void file2_func()
{
    printf("File2Wn");
}
```

<file2.c>



Make 파일 활용하기

Makefile 작성 예

- 다음은 앞의 예인 file1.c와 file2.c를 컴파일하여 test라는 실행파일을 만드는 Makefile의 예이다.

```
test : file1.o file2.o
(tab) gcc -o test file1.o file2.o

file1.o : file1.c
(tab) gcc -c -o file1.o file1.c

file2.o : file2.c
(tab) gcc -c -o file2.o file2.c

clean :
(tab) rm *.o test
```



Make 파일 활용하기

Makefile 작성 예

- 먼저 `make clean`을 실행한다. 그러면 `.o`의 확장자를 가진 파일과 실행파일인 `test`가 지워진다

```
$ make clean
```

- 다음 컴파일을 위해 `make`를 수행한다

```
$ make
```

- 실행파일인 `./test`를 실행한다.

```
ubuntu@client:~$ make clean
rm *.o test
ubuntu@client:~$ make
gcc -c -o file1.o file1.c
gcc -c -o file2.o file2.c
gcc -o test file1.o file2.o
ubuntu@client:~$ ls
a.out  file1.o  file2.o  hello.c  test  다운로드  바탕화면  사진  뽀롱뽀
file1.c  file2.c  hello    makefile  공개  문서      비디오  음악
ubuntu@client:~$ ./test
File2
File1
```

Q & A

Thank You