

Open Source Software

# 강의 개요와 소프트웨어에 대한 이해

About..

컴퓨터소프트웨어공학과

김 원 일



# 강의 개요 - 1



- 강의 내용

- 공개 소프트웨어와 관련된 소프트웨어 활용 방법과 실습
- 작성된 **프로그램을 아름답게 관리**하기 위한 방법
- 다양한 공개 소프트웨어 학습과 활용 방법

- 수업 구성

- 이론 : 30% - 이론은 많지 않으나 실습보다 중요하고 어려움.
- 실습 : 70% - 이론을 이해한 상태에서 진행해야 문제 없이 진행.
- 과제 : 간단한 과제로 **수업에 충실하면 1시간 이내로 해결** 가능
- 개인 저장소와 팀 저장소를 이용한 협업 실습이 주 목표



## 강의 개요 - 2



### • 성적 및 출석

- **중간고사** : 30% - 개인 서술형 시험
  - 공개 소프트웨어에 대한 이해와 이론적인 내용 이해에 대한 평가
- **기말고사** : 30% - 팀 단위 프로젝트 시험 (예정)
  - 공개 소프트웨어를 이용한 협업 프로젝트 결과물 평가
  - 공개 소프트웨어에 대한 심화, 핵심 내용에 대한 서술형 시험
- **과제** : 20% - 반드시 필요한 경우
  - 수업 내용에서 반드시 숙지가 필요한 내용에 대한 과제
  - 수업 시간을 이용하여 과제를 수행하도록 구성할 수도 있음
  - A에서 C로 학점이 순식간에 떨어질 수 있음
- **출석** : 20% - 가장 중요한 성적
  - 출석에 따라 학점 변동이 매우 큼
  - A를 받은 학생도 출석에 따라 F학점을 받을 수 있음



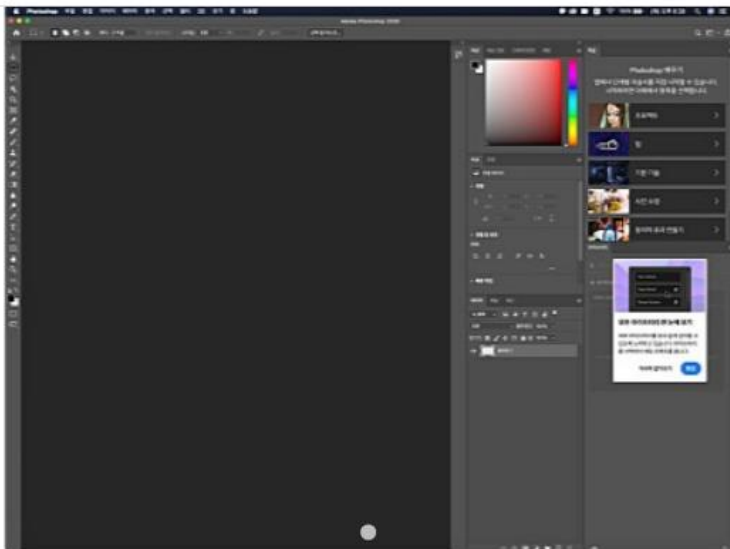
# About Software – 1



## • 소프트웨어의 종류

### – 데모(Demo)

- 소프트웨어 기능 확인 및 광고 등의 목적으로 배포하는 무료 체험판
- 주요 기능 사용 금지, 사용 기간의 제한 등의 제한적인 소프트웨어 사용
- 기능 또는 제품 광고를 위해 배포하는 형태
- 게임의 경우 챕터 1까지만 진행 가능한 형태로 배포



분류	이미지 편집
버전	v14.0
용량	NaNMB
사용범위	데모 - 개인, 국내/국외
지원 OS	Windows
금주 다운 수	0
누적 다운 수	0
개발사	Adobe




# About Software – 2

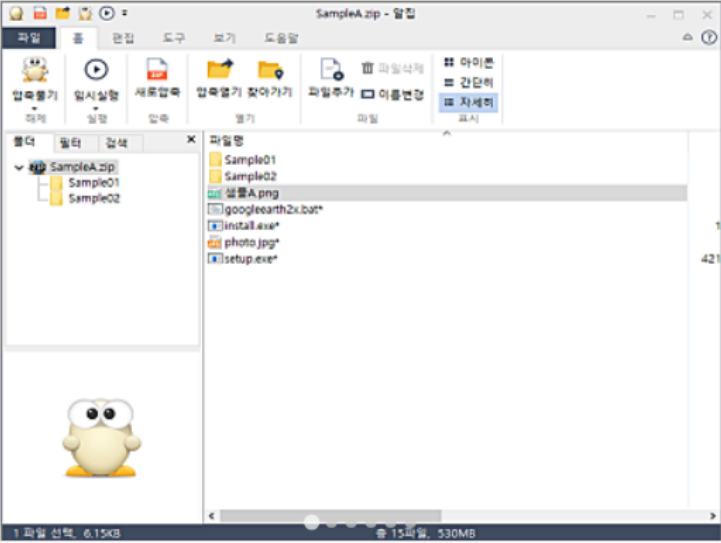


## • 소프트웨어의 종류

### – 프리웨어(Freeware)

- 개인/회사가 무료로 배포하며 무료로 대부분의 기능 사용 가능
- 수정이나 별도 배포는 불가능하며, 데모보다 제한 없이 사용 가능
- 주로 개인이 사용할 때는 문제 없으나 회사에서 사용할 때는 라이선스 문제 발생
- 무료이기 때문에 버그 수정이나 업데이트가 느리거나 없는 경우도 존재

 **알집** ALZip



분류 파일 압축

버전 v12.1

용량 20.98MB

**사용범위 프리 - 개인, 국내**

지원 OS Windows

금주 다운 수 62

누적 다운 수 13,613

개발사 이스트소프트




# About Software – 3

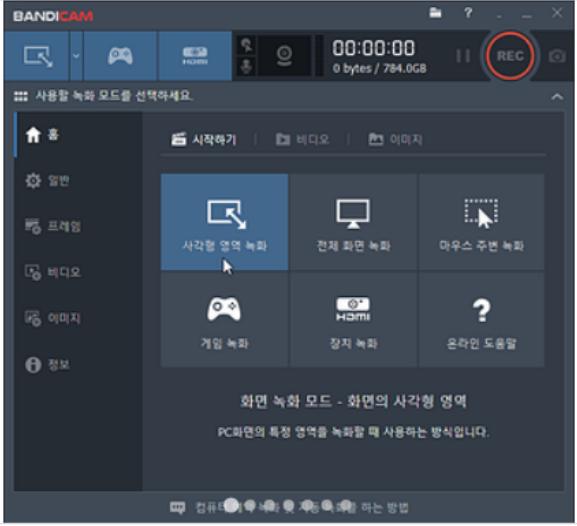


## • 소프트웨어의 종류

### – 쉐어웨어(Shareware)

- 체험판 데모와 비슷한 형태로 배포되며 대부분의 기능을 사용할 수 있음
- 데모는 광고가 목적이지만, 쉐어웨어는 판매가 목적
- 대부분은 사용 기간의 제한으로 소프트웨어를 사용할 수 있음
- 소프트웨어 키를 구매하여 쉐어웨어를 등록하여 정품으로 사용 가능
- 일반적으로 개인 사용은 문제 없으나 기업에서 사용할 때는 문제 발생 가능

 반디캠 Bandicam



분류	동영상 캡처/녹화
버전	5.2.1.1860
용량	20.42MB
사용범위	셰어 - 개인/기업
지원 OS	Windows
금주 다운 수	5
누적 다운 수	2,792
개발사	반디캠컴퍼니



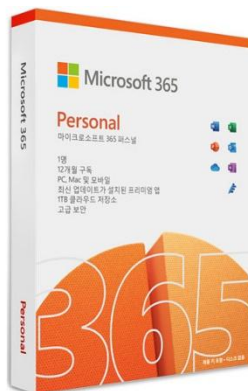
# About Software – 4



## • 소프트웨어의 종류

### - 상용 소프트웨어

- 사용자가 비용을 지불하고 사용권을 구매하는 형태의 정식 소프트웨어
- 매우 전문적인 기능 구현으로 소프트웨어의 **판매가 목적**인 소프트웨어
- **사용권**을 구매하는 형태로 개인/기업 모두 반드시 구매하여 사용해야 함
- 기업 솔루션은 특히 정품이 없을 경우 강력한 처벌이 발생할 수 있음
- 단순히 설치 파일을 가지고 있는 경우에도 라이선스 위반으로 보고 있음
- 개인 사용자는 큰 제약이 없으나 회사에서는 큰 문제가 발생할 수 있음
- 대체할 수 있는 무료 소프트웨어를 사용하여 라이선스 문제 회피가 가능
- 사용이 어렵거나 인터페이스가 약하더라도 대체할 수 있는 방법은 많이 있음



# OSS(Open Source Software) – 1



## • 공개 소프트웨어

- **소스 코드가 공개**되어 누구나 접근이 가능한 소프트웨어
- 소스 코드 확인/추가/수정/삭제, 기능 추가/삭제 후 사용 가능
- 소스 코드를 다운로드 받아 빌드하여 사용도 가능
- 소스 코드를 공개할 수 있도록 다양한 방법이 사용됨
  - 홈 페이지/웹 사이트를 통해 소스 공개
  - ftp(File Transfer Protocol)를 이용하여 소스 공개
  - 최근 공식적인 공개 방법은 **github** 사이트를 통해 소스 공개
- 소스 코드 또는 소프트웨어 사용에 제약이 거의 없음
- 반대는 Closed Source 또는 Proprietary Software
- 일반 회사에서도 많이 사용하기 때문에 개념은 알고 있어야 함
- 별도의 공부가 필요하며, 많이 알수록 관련 문제 방지가 가능



# ❖ OSS(Open Source Software) – 2



## • 관련 용어

- FSF(Free Software Foundation)
  - 미국에서 만들어진 자유 소프트웨어 재단은 자유 소프트웨어 생산과 보급 장려가 목적
  - 1985년 10월 4일 설립되었으며, 현재 **리처드 스톨만**이 이사장
  - **Copyright**의 반대인 **Copyleft** 개념을 만들었음
  - “모든 프로그램이나 정보 및 미디어는 소수에게 독점되어서는 안되며, 자유롭게 공유되어야 한다.”가 주요 사상
- GNU(GNU is Not Unix) 프로젝트
  - FSF에서 진행하며 유지하는 프로젝트로 **자유 소프트웨어 생태계 구축**이 목적
  - 운영체제부터 다양한 소프트웨어를 제공하여 개발 또는 사용 환경을 제공
- GPL(General Public License) : **일반 공중 사용 허가서**
  - OSS 제품들의 기본 라이선스 계약서로 다양한 버전이 존재
  - 프로그램을 어떠한 목적으로도 사용 가능. **소스 코드의 변경 및 사용** 가능
  - 변경된 소스 코드를 배포하고 동일 라이선스를 사용할 것
  - 법적 책임을 지지 않고, 보증하지 않음



# 이번 학기의 공개 소프트웨어



- 소프트웨어 개발 도구

- **git** : <https://git-scm.com/>
  - <https://git-scm.com/book/ko/v2>
- **Visual Studio Code** : <https://code.visualstudio.com/>
- **Doxygen** : <https://www.doxygen.nl/>
- **Docker** : <https://www.docker.com/>

- 웹 사이트

- **github** : <https://github.com/>

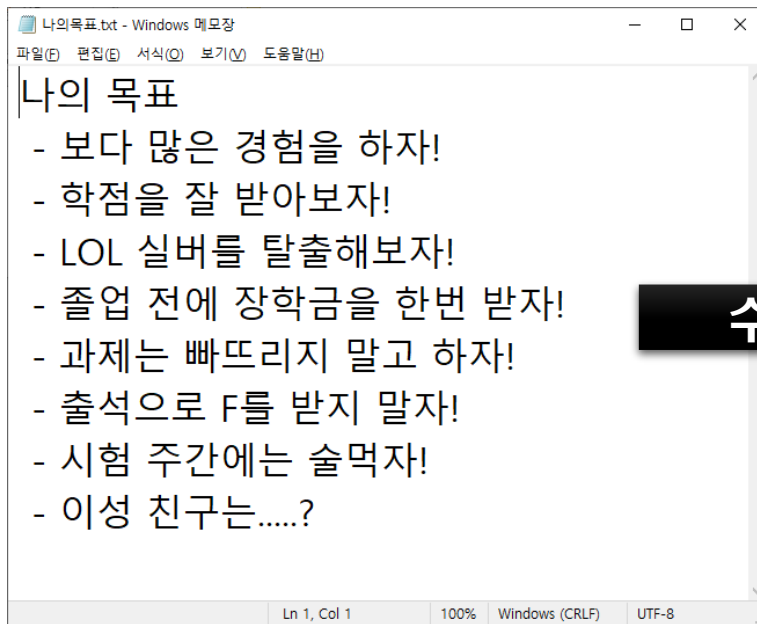


# 파일 관리 - 1

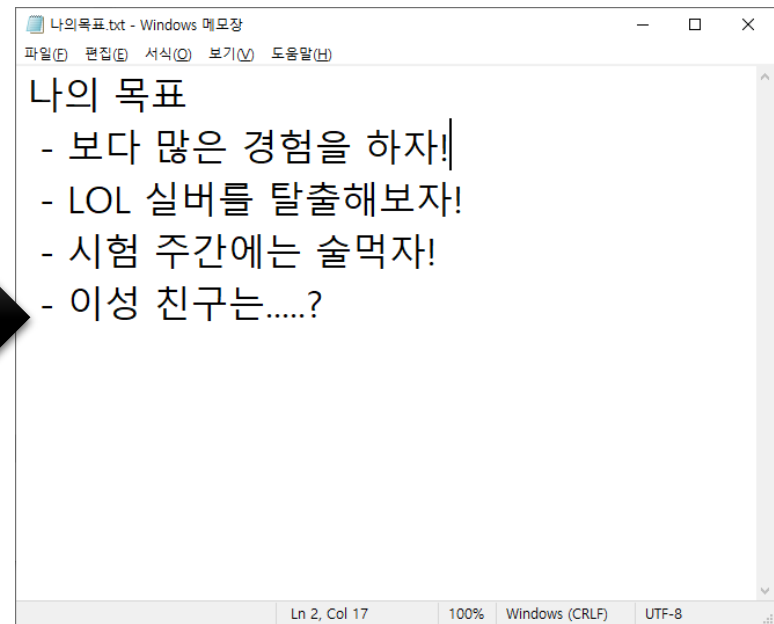


## • 가장 쉬운 파일 관리 방법

- 하나의 파일에 변경 내용을 적용하는 형식
- 하나만 존재하기 때문에 변경 이전의 내용 확인 불가
- 이전 내용에 대한 정보를 본인이 기억해야 함



수정



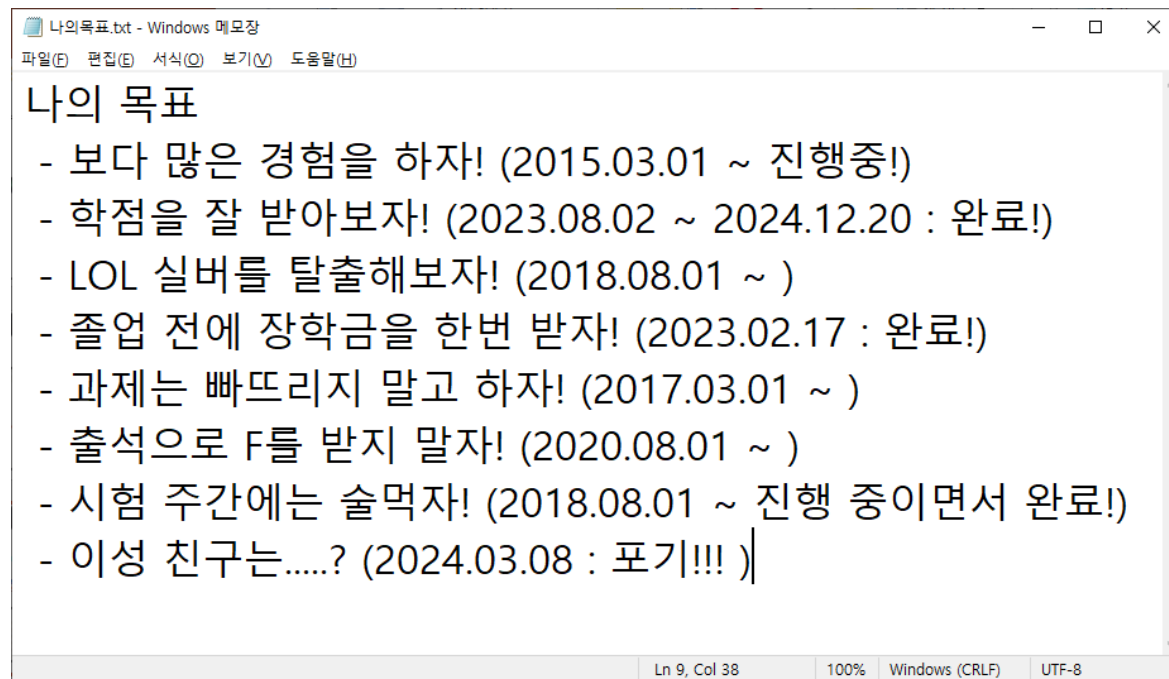


## 파일 관리 - 2



- 두 번째로 쉬운 파일 관리

- 파일 내용 변경이 발생할 때마다 기록을 남김
- 시작 일자와 변경된 내용을 지속적으로 기록해야 함
- 관련 내용에 대해 꾸준한 관심과 확인이 필요



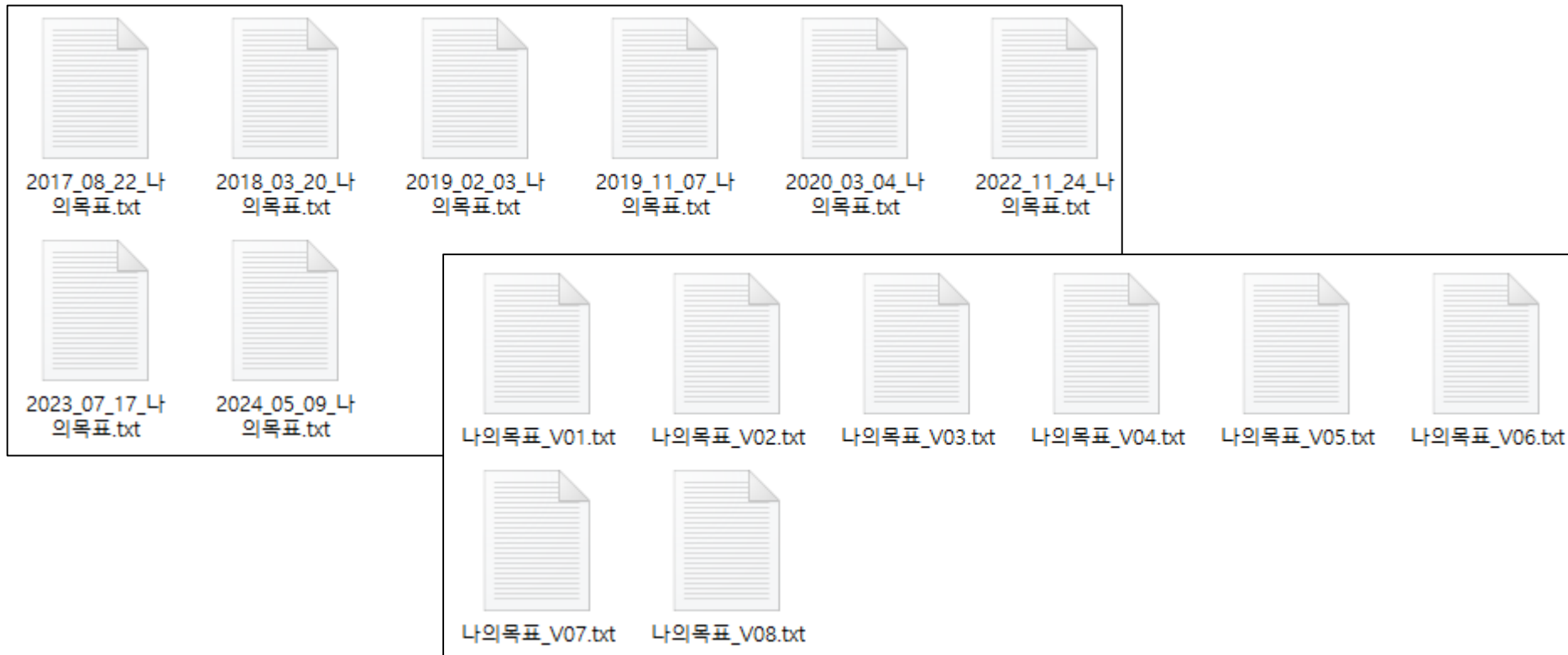


## 파일 관리 - 3



- 가장 많이 사용하는 파일 관리 방법

- 한 내용을 다수 파일로 생성하고, 변경할 때마다 지속적인 생성
- 대부분 날짜 또는 버전 형식으로 관리
- 변경 사항 확인을 위해서는 이전 파일 내용을 확인해야 함





## 파일 관리 - 4



- 관리 문제

- 생각보다 복잡하고 부지런해야 함
- 한번 관리하지 못하면 관리를 위해 시간 투자가 필요함
- 여러 번에 걸친 변경 사항을 적용하지 않으면 귀찮음
- 새로운 관리를 시작하여 중간 과정을 생략하면 뭔가 찝찝함

- 다수 파일 관리는...?

- 하나의 파일 관리보다 더 복잡하고 더 부지런해야 함
- 한번 관리하지 못하면 정말 많은 시간 투자가 필요함
- 여러 차례 관리를 놓치면 손을 대기 싫을 정도
- 새롭게 시작하면 이전 정보 확인이 매우 어려움
- 파일 내용이 서로 관계가 있다면 복잡도와 어려움은 말도 하기 어려움



# 소스 파일 관리 - 1



## • 소스 파일

- 프로그램 개발을 위한 소스 코드는 파일에 기록
- 기록된 파일은 컴파일러와 링커를 통해 실행 가능한 프로그램으로 구성
- 프로그램 개발 과정에서 소스 코드 파일이 가장 중요
  - 요구 사항 확인 및 테스트 등으로 지속적인 변경 발생
  - 소스 파일이 변경되면 이후의 모든 과정에 변경이 발생
  - 수정이 발생할 때마다 이전 내용이 삭제되는 현상이 발생

```
#include "framework.h"
#include "TestManager.h"
#include "ProcessManager.h"
#include "ControlManager.h"
#include <IHelp32.h>

#define MAX_LOADSTRING 100

// 전역 변수:
HINSTANCE hInst;                // 현재 인스턴스입니다.
WCHAR szTitle[MAX_LOADSTRING]; // 제목 표시줄 텍스트입니다.
WCHAR szWindowClass[MAX_LOADSTRING]; // 기본 창 클래스 이름입니다.

// 주요 인스턴스 생성
ControlManager *g_controlManager;
ProcessManager *g_processManager;

// 이 코드 모듈에 포함된 함수의 선언을 전달합니다:
ATOM MyRegisterClass(HINSTANCE hInstance);
BOOL InitInstance(HINSTANCE, int);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
                     _In_opt_ HINSTANCE hPrevInstance,
```

전처리기

컴파일러  
(번역기)

링커

실행파일  
(.exe)



## 소스 파일 관리 - 2



### • 소스 파일 관리

- 하나의 파일만 존재한다면 부지런히 관리하면 큰 문제 없음
- 주석을 이용하여 이전 정보를 유지하는 것도 한계가 있음
- 주석이 많아지면 실제 코드보다 주석이 더 많아짐
- 굳이 이러한 정보 유지가 필요한지에 대한 의문
- 변경이 발생할 때마다 정보 입력을 해주어야 함
- 코드 작성보다 주석 작성으로 의욕 하락이 발생할 수 있음

```
// 2023.08.08. 코드 추가 : 마우스 이동에 따른 카운트 초기화
case WM_MOUSEMOVE:
    // 2023.09.02. 코드 삭제 : 변수명 변경으로 코드 삭제
    // g_buttonCount = 0;
    // 2023.09.02. 코드 수정 : 변수명 변경으로 코드 수정
    // g_buttonCount ==> g_lbcnt
    g_lbcnt = 0;
    // 2023.09.03. 코드 추가 : 마우스 오른쪽 버튼 추가
    g_rbcnt = 0;
    break;
```



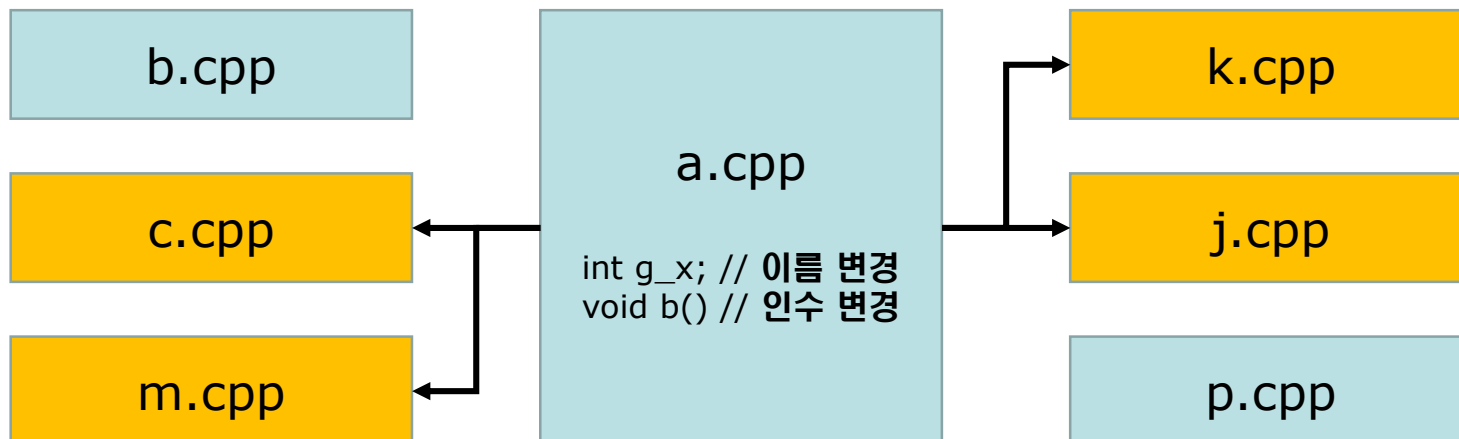


# 소스 파일 관리 - 3



## • 다수의 소스 파일 관리

- 여러 파일로 소스를 구성하면 파일 관리가 매우 어려움
- 함수 변경에 따른 정보 남기기가 어려움
  - 함수의 이름, 인수, 반환 또는 인수의 자료형, 인수의 순서 등
- 변수 이름 변경 사항이 빈번한 경우 관련 정보 관리가 어려움
- 함수나 (전역) 변수 등의 이름 변경에도 문제가 발생
  - 함수를 호출하는 모든 파일을 수정해야 정상적으로
- 정상적으로 동작하는 소스를 별도로 관리하는 경우도 있음





# 수동 소스 파일 관리 - 1



## • 일반적인 로컬 소스 파일 버전 관리

- 디렉터리에 **파일을 복사하여 관리**하는 방법을 사용
- 디렉터리명에 **날짜와 시간**을 포함하여 관리하는 것이 일반적
- 수정이나 변경 사항 발생에 대해 수동으로 적용해야 함
- 해당 날짜에 변경된 사항이나 수정된 사항 관리가 어려움
- 이전 버전의 파일을 가져오면 처음부터 작성하는 것이 편할 수 있음
- 이전 버전 유지 여부를 결정하기 어렵고, 단순히 쌓이기만 할 수 있음

이름	수정한 날짜	유형
 2021_04_11	2021-04-26 오후 5:12	파일 폴더
 2021_04_12	2021-04-26 오후 5:12	파일 폴더
 2021_04_12_1	2021-04-26 오후 5:12	파일 폴더
 2021_04_12_2	2021-04-26 오후 5:12	파일 폴더
 2021_04_13	2021-04-26 오후 5:12	파일 폴더
 2021_04_14	2021-04-26 오후 5:12	파일 폴더



## 수동 소스 파일 관리 - 2



- 일반 적인 문제점

- 디렉터리 삭제나 잘못된 파일 수정 및 복사 등으로 관리가 어려움
- 해당 디렉터리 생성 시점 등 필요한 **정보를 별도로 관리**해야 함
  - 날짜와 시간만으로는 해당 시점에 관련된 정보를 모두 확인하기 어려움

- 소스 코드를 병합하는 과정에서의 문제점

- 소스 코드들 간의 중복이 발생할 수 있음
  - 코드 중복은 프로그램의 크기 증가와 속도 저하 발생
  - 소프트웨어 QC(Quality Control), QA(Quality Assurance)에서 지양
- 다수의 비슷한 코드들이 산재하여 혼동 발생
  - 이름은 다른데 다른 기능을 수행하는 코드/함수



# QA? QC?



- QC(Quality Control)
  - 소프트웨어 품질관리
  - 소프트웨어 개발 중 발생할 수 있는 각종 버그와 문제점에 대한 관리
  - 소프트웨어의 기능의 정상적인 동작과 처리에 대한 품질에 중점
  - 주요 업무 : 테스트(Testing), 리뷰(Review), 검사(Inspection)
- QA(Quality Assurance)
  - 소프트웨어 품질보증
  - QC보다 상위 개념으로 QC는 QA에 포함
  - 소프트웨어 전 과정의 모니터링과 관리가 목적
  - 프로젝트 표준화, 이슈 해결, 릴리즈, 테스트 및 보고서 작성 관리



# 파일 버전 관리 - 1



- **VCS** (Version Control System)
  - **파일 관리 문제점을 해결**하기 위해 고안된 관리 시스템
  - 파일의 추가/수정/삭제 사항을 관리할 수 있는 시스템을 통칭
  - 단일 파일 뿐만 아니라 다수의 파일과 디렉터리까지 관리
  - 프로젝트와 IDE 및 다양한 개발 환경을 지원하고 관리
  - 전체적인 파일 관리의 문제점을 해결하기 위해 등장
  - 1990년대부터 다양한 제품이 개발되고 사용되어 왔음
  - 대표적으로 Subversion, BitKeeper, CVS 등이 있음



## Concurrent Versions System - Summary

Group

[Main](#) [Homepage](#) [Download](#) [Mailing lists](#) [Source code](#) [Bugs](#) [Tasks](#) [Patches](#) [News](#)



## 파일 버전 관리 - 2



### • 버전 관리의 특징

- **파일 역사**(History) 관리 시스템
  - 다양한 형태의 파일의 변화를 저장하고 관리 가능
  - 단위 파일 뿐만 아니라 프로젝트 전체를 관리할 수 있음
- 다양한 형태로 프로젝트 변화를 확인할 수 있음
  - **파일의 이전 상태** 확인
  - **프로젝트**의 이전 상태 확인
  - **수정 내용 비교**
  - 문제를 발생시킨 **사용자에 대한 정보** 확인
  - 파일의 **생성과 변화 전체를** 확인 가능
  - 이슈 발생 시점과 관련 정보 확인 가능
- 프로그램 인수 인계와 지속적인 관리에 편리
- 프로그램 변화 과정 확인으로 코드 작성 효율의 증가
- 기존 작성자의 의도와 코드 변경 사유 등에 대해서도 확인 가능



# 중앙집중식 버전 관리



- **CVCS**(Central VCS)

- 서버를 이용하여 버전을 **중앙에서 관리하는 시스템**
- CVS, Subversion, Perforce와 같은 시스템
- 파일에 대한 모든 정보를 서버에서 관리
- 클라이언트(개발자)는 **변경 정보를 서버에서 받아서 적용**
- 관리자에 의한 **중앙 집중식 관리**가 가능
- 모든 클라이언트가 관리된 동일한 파일을 전달 받을 수 있음

- 문제점

- **서버에 문제가 발생할 경우**, 해결할 수 있는 방법이 존재하지 않음
- 개발 시스템 전체가 정지되므로 **개발 업무 자체가 중단되는 사태**가 발생
- 서버 디스크에 문제가 발생한 경우, **파일에 대한 전체 History 정보를 잃을 수 있음**



# 분산 버전 관리



- **DVCS**(Distributed VCS)
  - 파일 History와 **파일 변경의 모든 정보를 분산하여 저장**하는 구조
  - 관리 서버의 모든 정보를 **클라이언트도 동일하게 보관**
  - 모든 정보가 클라이언트로 복제되어 저장
  - 서버에 문제 발생 시 가장 최신 버전을 갖는 개발자 정보로 복원이 가능
    - 완벽하게 복원하는 것은 어려우나, 주요 History는 복원 가능
  - 파편화된 정보와 주요 정보가 여러 군데 나뉘어 관리
  - 비슷한 구조로 Block-Chain의 정보 보관 방식이 있음





# 최종 형태의 버전 관리



- git

- **Linux Kernel 소스 관리**를 위해 개발됨

- 1991~2002년 동안 Linux Kernel은 Patch 파일과 단순 압축 파일로 관리
    - 2002년 BitKeeper라는 DVCS 프로그램을 이용하여 관리를 시작
    - 2005년 BitKeeper의 상용화로 소스 관리 도구의 필요성이 각인됨

- Linux Torvalds가 직접 개발에 참여

- 빠른 속도와 단순한 구조로 설계
    - 비선형적인 개발 지원
    - 완벽한 분산 데이터 저장
    - 대형 프로젝트에서도 문제 없도록 다양한 전략을 제시

- 전 세계적인 개발자 필수 교양 도구

- 다양한 개발 도구에 에드온/플러그인 또는 기본으로 지원

- 너무 유연한 기능을 제공하는 바람에 오히려 어려움



# 이번 학기 목표 - 1



- **버전 관리 시스템과 공개 소프트웨어 학습**
  - git 소프트웨어 사용 방법과 활용 방법
  - git을 이용한 개인 프로젝트 작성 과정에 대한 실습
  - git을 이용한 협업 프로젝트 작성 과정과 문제 해결에 대한 실습
  - 원활한 파일 이력 관리를 위한 주석 입력 방법과 활용
    - doxygen 공개 소프트웨어를 활용한 주석 입력
  - 포트폴리오 작성을 위한 마크 업 언어 학습
    - HTML과 비슷하나 더 간단하고 편하게 작성 가능
  - github 사이트의 기능 활용
    - github에서 활용할 수 있는 다양한 SW 개발 관련 기능
    - 개발 효율성 증가를 위한 github 기능 확인
    - 추후 활용을 위한 개발 관련 기능들 학습



## 이번 학기 목표 - 2



- **학과에서 처음으로 진행하는 수업**
  - 이전에는 별도의 수업이 존재하지 않음
  - 처음 진행하기 때문에 수업 시간이나 시험 관련 정보가 전무
- **소프트웨어 개발 지원 도구**
  - 소프트웨어를 직접적으로 개발하는 도구가 아님
  - 최대한 많이 그리고, 천천히 진행하면서 수업을 진행
  - 과제도 매 시간마다 작성하거나 거의 없을 수 있음
  - 도구를 배우기 때문에 생각보다 쉬울 수 있음
- **개발자 교양 수업**
  - 소프트웨어 개발자의 기본기이면서 교양 필수
  - 회사에서도 다양한 분야에 사용되고 있음