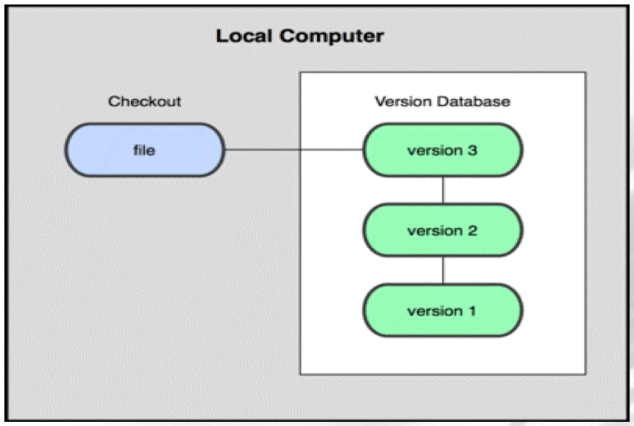
버전 관리 시스템

버전 관리 시스템

- 버전 관리 시스템
  - ✓ 파일의 변화를 시간에 따라 기록한 후 과거 특정 시점의 버전을 다시 불러올 수 있는 시스템
  - ✓ 모든 컴퓨터 파일이 버전 관리의 대상
  - ✓ 이미지나 레이아웃을 수정할 때마다 각각의 형태를 모두 보존하고 싶은 그래픽 디자이너나 웹 디자이너라면 버전 관리 시스템(Version Control System; VCS)을 사용하는 것이 현명한 선택이 될 수 있음
  - ✓ VCS를 사용하면 개별 파일 혹은 프로젝트 전체를 이전 상태로 되돌리거나 시간에 따른 변경 사항을 검토할 수 있으며, 문제가 되는 부분을 누가 마지막으로 수정했는지, 누가 언제 이슈를 만들어냈는지 등을 알 수 있기 때문
  - ✓ 파일을 잃어버리거나 무언가 잘못되어도 쉽게 복구 가능

버전 관리 시스템

- ❖ 종류
  - ✓ 로컬 버전 관리 시스템
    - 대부분의 사람들이 버전 관리를 위해 쓰는 방법은 파일을 다른 디렉토리에 복사하는 것인데 이 방법은 간단하고 자주 사용되는 방법이지만 실수가 발생하기 쉽기 때문에 어느 디렉토리에서 작업하고 있었는지 잊어버리고 엉뚱한 파일을 덮어쓰거나 의도하지 않았던 위치로 복사할 수 있는데 이 문제를 해결하기 위해 오래전부터 프로그래머들은 간단한 데이터베이스에 파일의 변경 사항을 기록하는 로컬 버전 관리 시스템을 사용
    - VCS 도구들 중 현재에도 널리 쓰이는 것으로 RCS라 불리는 시스템이 있는데 그 예로 Mac OS X 운영체제에서는 개발 도구를 설치하면 RCS가 자동으로 설치되는데 RCS의 기본적인 동작 방식은 각 리비전들 간의 패치 세트(patch set)라고 하는 데이터의 차이점들을 특별한 형식의 파일에 저장하고 특정 시점의 파일 내용을 보고 싶을 때 해당 시점까지의 패치들을 모두 더하여 파일을 만들어내는 방식



버전 관리 시스템

- ❖ 종류
  - ✓ 중앙집중식 버전 관리 시스템(클라이언트 – 서버 모델)
    - 중앙집중식 버전 관리 시스템(Centralized Version Control System; CVCS)은 외부에 있는 개발자와의 협업 문제를 해결하기 위해 개발됐는데 CVS, Subversion, Perforce와 같은 시스템들이 여기에 속하며 CVCS에서는 버전 관리되는 모든 파일을 저장하는 하나의 서버 와 이 중앙 서버에서 파일들을 가져오는(checkout) 다수의 클라이언트가 존재
    - 오랫동안 사용된 이 방식은 지금까지도 버전 관리의 대표적인 방식
    - CVCS는 로컬 VCS에 비해 장점이 많은데 누구나 다른 사람들이 무엇을 하고 있는지 알 수 있고 관리자는 누가 무엇을 할 수 있는지 꼼꼼하게 관리할 수 있으며 CVCS를 관리하는 것은 수많은 클라이언트의 로컬 데이터베이스를 관리하는 것보다 훨씬 쉬움
    - CVCS는 심각한 단점이 있는데 중앙 서버가 잘못되면 모든 것이 잘못된다는 것
    - 서버가 다운될 경우 서버가 다시 복구될 때까지 다른 사람과의 협업을 하는 것을 포함해서 진행 중이던 작업을 버전 관리하는 것도 불가능해지며 중앙 데이터베이스가 저장된 하드디스크에 오류가 발생하고 백업도 없다면 사람들이 각자 자신의 컴퓨터에 가지고 있던 스냅샷 외에는 그 동안 쌓인 프로젝트의 이력을 모두 잃게 되기 때문에 로컬 VCS 시스템과 같은 문제가 발생할 수 있는데 이는 프로젝트의 모든 이력이 한곳에만 있을 경우 피할 수 없는 문제

버전 관리 시스템

- ❖ 종류
  - ✓ 분산 버전 관리 시스템
    - 분산 버전 관리 시스템(Distributed Version Control System; DVCS)은 앞서 말한 문제를 해결하기 위해 개발
    - Git, Mecurial, Bazaar, Darcs 등 DVCS에서는 클라이언트가 파일들의 마지막 스냅샷을 가져오는 대신 저장소(repository)를 통째로 복제해서 서버에 문제가 생겨도 어느 클라이언트든 복제된 저장소를 다시 서버로 복사하면 서버가 복구되는 형태로 체크아웃(checkout)을 할 때마다 전체 백업이 일어나는 형태
    - DVCS에서는 다수의 원격 저장소(remote repository)를 갖는 것이 가능하기 때문에 동시에 여러 그룹과 여러 방법으로 함께 작업할 수 있어서 계층 모델(hierarchical model) 등 중앙 집중 시스템에서는 할 수 없는 다양한 작업 방식(workflow)들을 사용해 볼 수 있음

버전 관리 시스템

- 실례
  - ✓ CVS
    - Concurrent Versions System 을 의마하는 말로 클라이언트-서버 방식의 버전관리 시스템
    - 공식 웹 사이트: https://savannah.nongnu.org/projects/cvs
  - ✓ 서브버전
    - CVS의 여러 단점을 개선한 '클라이언트-서버'모델의 자유 소프트웨어 버전 관리 시스템
    - 공식 웹 사이트 : https://subversion.apache.org/
  - ✓ 머큐리얼
    - 구글에서 관리를 지원하고 있는 분산 모델의 버전 관리 시스템
    - Git은 필요한 기능을 골라서 사용하지만 머큐리얼은 버전 관리 시스템에 필요한 모든 기능을 한 번에 통합해서 제공
    - 파이썬으로 개발

버전 관리 시스템

- 실례
  - ✓ Git
    - 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템 또는 이러한 명령어
    - Git은 2005년에 리눅스 커널 개발을 위해 초기 개발에 기여한 다른 커널 개발자들과 함께 2005년에 Linus Torvals 가 처음 개발
    - Git은 GNU 일반 공중 사용 허가서 v2 하에 배포되는 자유 소프트웨어
    - Git의 장점
      - 전 세계의 수많은 사용자가 사용 중
      - git을 사용한 저장소를 공유 사이트인 GitHub 웹 사이트의 존재
      - 사용자 수에서 나오는 많은 숫자의 Tutorial 과 Project 가 존재