[506489] System Programming (F'18)

Term Project (Week 3, Final) Report

팀 정보

No.	이름	학번
1	정균모	20145165
2	박인근	20145128
3	유재혁	20145144
4		
팀 이름: 인성삼인조		

<목차>

* 보고서 작성이 완료되면 <목차>를 업데이트 해 주세요. (방법: "목차" 클릭>"목차 업데이트">"목차 전체 업데이트")

결과물 제출

- SmartCampus 에 업로드 해야 하는 제출물 목록은 다음과 같습니다.
  - Report(프로젝트 보고서), PDF 형식
  - 프로젝트 발표자료 (ppt, pptx, pdf 등)
  - GitHub 프로젝트를 다운로드 한 zip 파일
- 제출 기한을 넘기면 자동으로 0 점 처리됩니다.
- 팀 과제이며, 팀별로 한 명만 제출하면 됩니다.

* 보고서 및 발표자료도 GitHub 에서 관리해야 합니다!!!

보고서 작성 가이드

- Week 3 시작: 2018. 12. 12. (Wed) 1:00pm
- Week 3 마감: 2018. 12. 17. (Mon) 12:59pm
- 매 주차별로 반드시 1 회 이상 GitHub 에 commit 한 내역이 있어야 합니다. (팀별로 1 회 이상, 개인별 1 회 아님). GitHub 에 commit 한 내역이 없을 경우, 큰 감점을 받게 됩니다.

본 보고서는 총 3 개의 파트로 구성되어 있습니다.

- Part I: 최종 보고서
  - 1 주차 보고서 내용 전체를 포함하여 작성하고, 그 동안 추가/수정된 내용이 있다면 그에 맞게 보고서를 수정해야 합니다.
- Part II: 데모/시연 결과물
  - 개발한 프로그램이 정상적으로 구동된다는 것을 보여주세요.
  - 데모/시연을 위해 사용한 시나리오를 자세하게 설명하고, 해당 시나리오에 대한 단계별 데모 결과물(예: 스크린샷)을 첨부하거나 또는 데모 영상을 촬영해서 첨부파일로 제출해도 됩니다.
- Part III: 프로젝트 진행중 발생한 문제점 및 해결방안
  - 2 주차 보고서 내용중 "프로젝트 진행 중 발생하는 문제점 및 해결방안"에 대한 내용을 포함하는 내용으로 작성하세요.
  - 그 외에 추가로 발생한 문제점이 있었다면 해당 문제점 및 해결방안을 기술하세요.
- Part IV: 개선방향
  - 개발 결과물을 앞으로 어떻게 개선할 수 있을지에 대한 의견을 서술하세요.

Part I: 최종 보고서

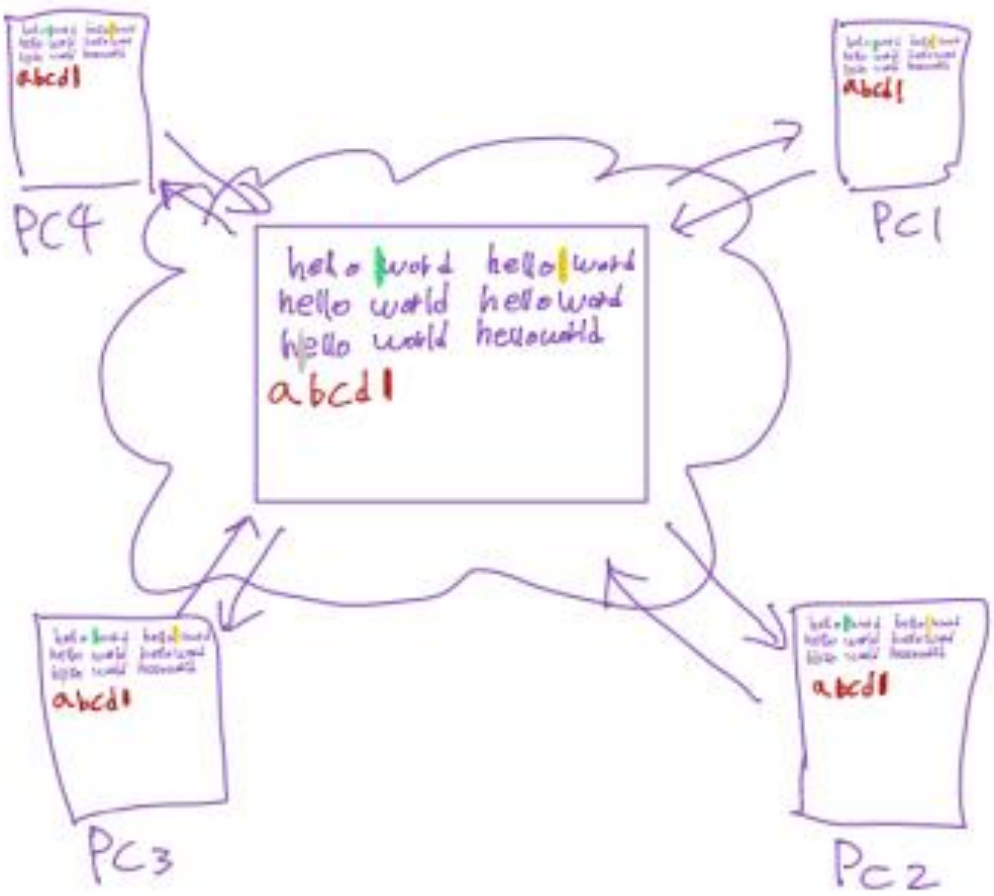
저희의 개발목표는 리눅스에서 같은 작업을 할 때도 정보 공유 및 협업이 힘들다는 것을 여러번 느꼈습니다. 따라서 저희는 사람마다 따로 하는 것이 아닌 같은 공간에서 작업할 수 있는 프로그램을 만드는 것이 목표입니다.

저희는 서버와 클라이언트간의 소통을 하기 위해 소켓프로그램을 해야했습니다. 서버는 아무 IP 에서나 들어올 수 있게 INADDR_ANY 방식을 사용하였고, 포트번호는 4000 번을 사용하였습니다. 클라이언트에서는 서버를 구현하는데 어려움이 있어서 루프백 IP 인 127.0.0.1 을 사용하여 서버와 통신하는 시나리오를 구성하였습니다. 서버는 클라이언트의요청을 기다리다가 클라이언트의 요청이 오면 fork 하여 새로운 프로세스를 할당해 주고 다시 block 상태로 돌아갑니다. 자식 프로세스는 클라이언트와 1:1 로 통신하며 클라이언트의 요청을 서버에서 수행합니다.

클라이언트들은 서버에있는 파일들을 모두 원격으로 사용할 수 있도록 디렉토리 프로그래밍을 지원하며 다른 클라이언트들과 실시간으로 문서 작업을 할 수 있는 환경이 제공됩니다. 지원되는 서버의 기능은 add 를 사용하여 사용자가 사용할 수 있는 실시간 에디터 파일을 생성해주고 ./[파일이름] 을 사용하여 사용자가 만들어냈거나 기존에 있던 파일을 실행시켜줍니다. 디렉토리 프로그래밍을 위해 cd 기능을 사용하여 서버의 폴더를 자유자제로 이동할 수 있으며 ls 명령어로 서버 폴더에 있는 파일 리스트들을 불러 올 수 있습니다. 또한 사용자들의 편의성을 위해서 clear, help, rm 기능을 추가하였습니다.

클라이언트들의 문서작업은 gui 로 구현되며 gtk+라이브러리를 사용하여 에디터를 제공합니다. 에디터는 클라이언트의 수정이 있을때마다 저장되며 저장이 수정이 없을시 업데이트만 진행하여 사용자들이 실시간으로 작업하고 있다고 느끼도록 만들었습니다.

흔히 협업이라고 하면 실시간으로 함께 작업할 수 있는 환경 여건이 조성되어야 하지만 온라인상에서 그러한 환경이 많이 이루어지고 있지 않다는 단점이 있습니다. 따라서 저희가 진행한 프로젝트는 서버만 제공된다면 장소와 시간의 제약없이 어디서든 동시에 프로젝트 작업을 진행 할 수 있을 것 입니다. 또한 더 나아가 문서편집기 뿐만 아니라 PC 로 작업되는 컴파일러나 오피스 프로그램 등 모든 생산물들을 동시에 작업할 수 있는 환경의 밑거름이 될 것 입니다.


abcd!
PC4
abcd!
PC1
hello world hello world
hello world hello world
hello world helloworld
abcd!
abcd!
PC3
abcd!
PC2

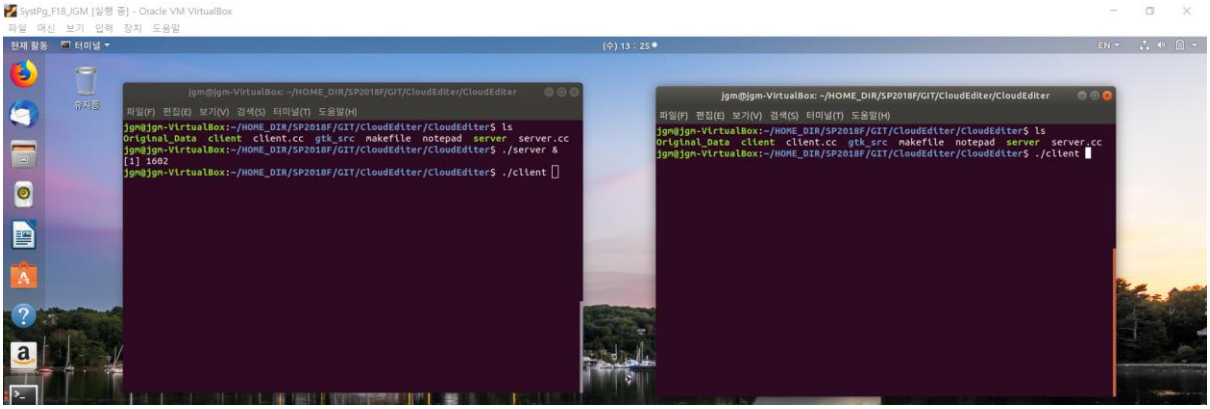
1주차 : 1. 서버, 클라이언트 소켓 프로그래밍
2. 클라이언트가 사용할 수 있는 명령어 목록.
3. 깃허브를 이용한 형상 관리 주소

2주차 : 1. 클라이언트 명령어 최적화 작업,
2. 클라이언트의 서버측 디렉토리 프로그래밍
3. GUI 프로그래밍으로 에디터 만들기

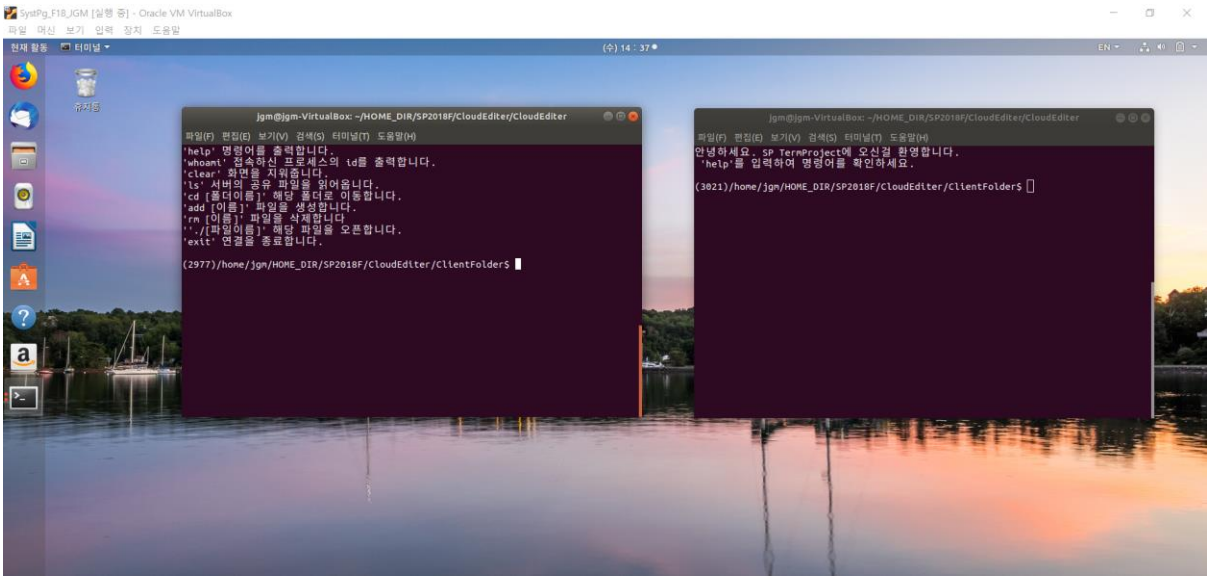
3주차 : 1. 클라이언트와 실시간 편집 내용을 확인할 수 있게 에디터 프로그래밍.
2. 커서의 위치 까지 이동시켜 주는 프로그래밍

Part II: 데모/시연 시나리오 및 결과물

1. 클라이언트는 공유 중인 서버에서 작업을 하기 위해 서버에 접속합니다.



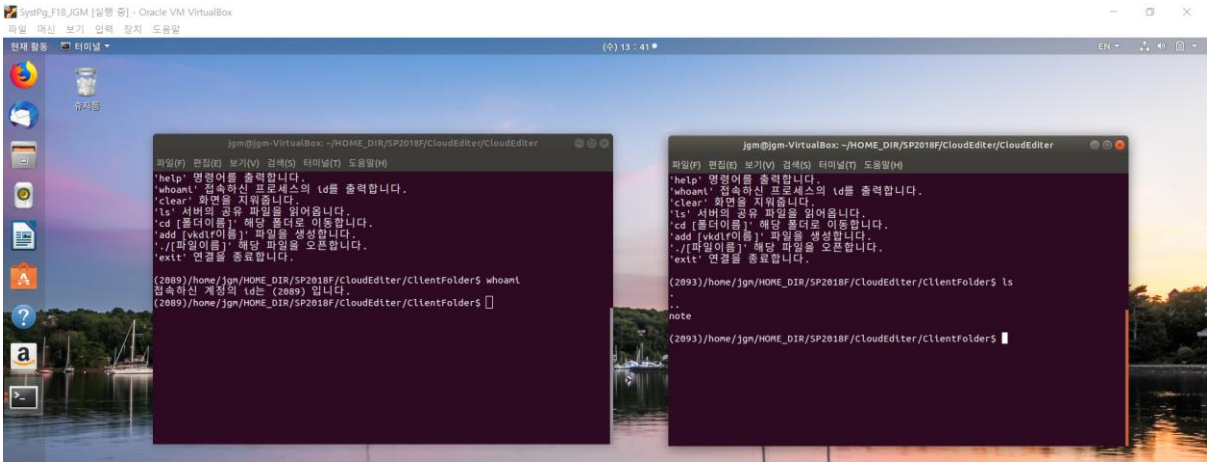
2. 접속 후 클라이언트가 실행할 수 있는 명령어를 확인하기 위해 help 명령어 입력 합니다.



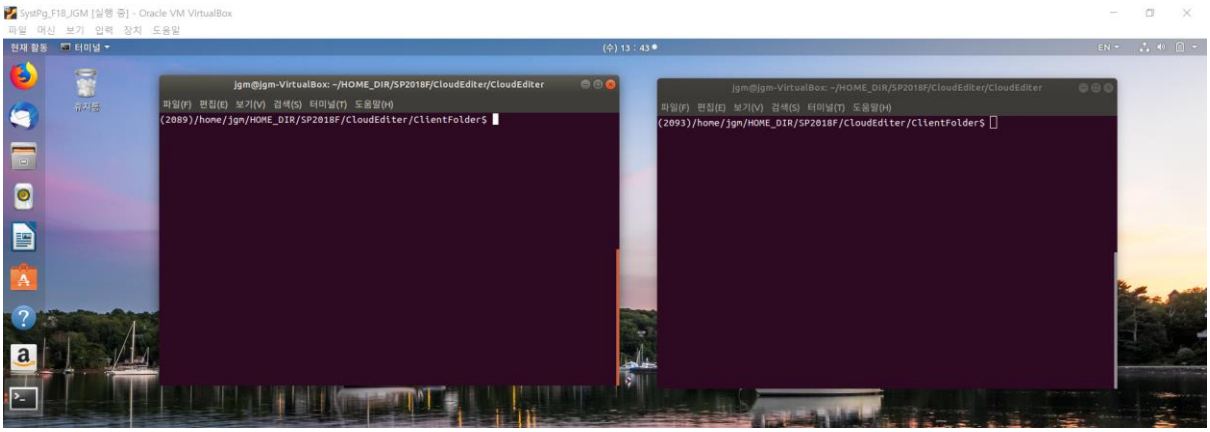
3. 본인이 어떤 프로세스로 접속했는지 궁금하면 Whoami를 입력해 id를 확인 할 수 있습니

다.

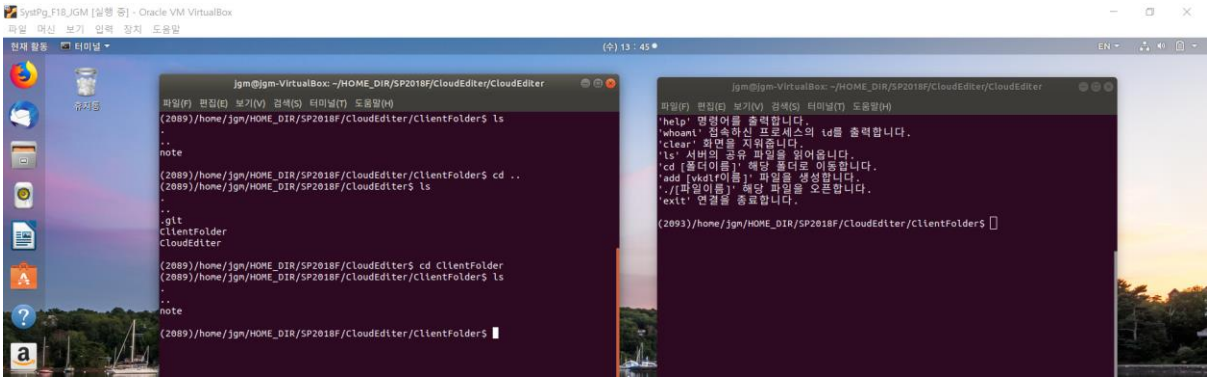
ls 명령어 입력을 입력해 서버의 공유 파일들을 확인합니다.



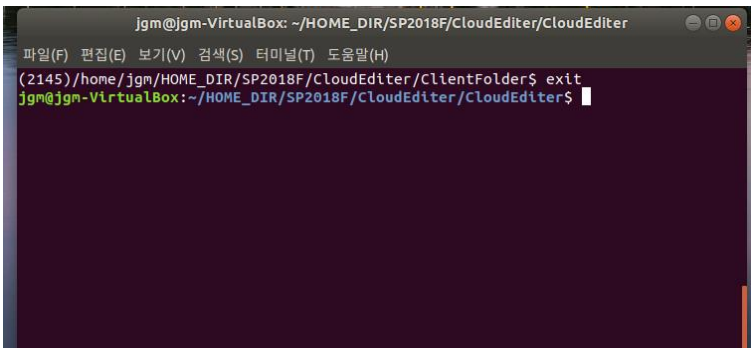
4. 클라이언트의 터미널이 복잡하여 Clear 명령어 사용해 깨끗하게 정리합니다.



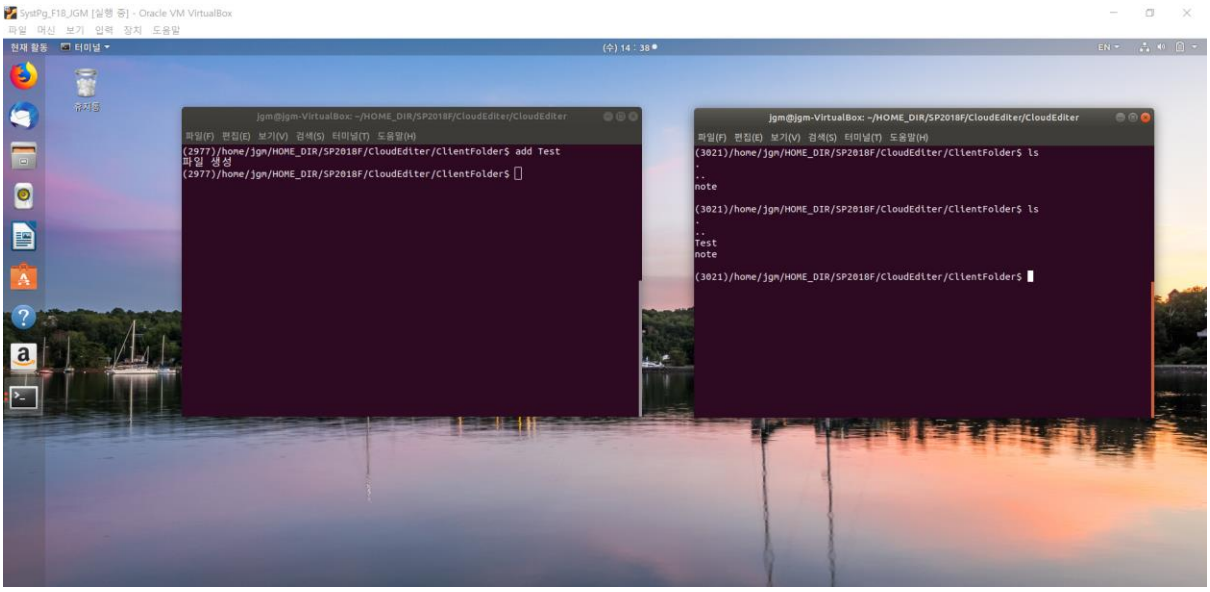
5. Cd 명령어로 폴더 이동 다른 공유 폴더를 확인하기 위해 cd 명령을 사용할 수 있습니다.



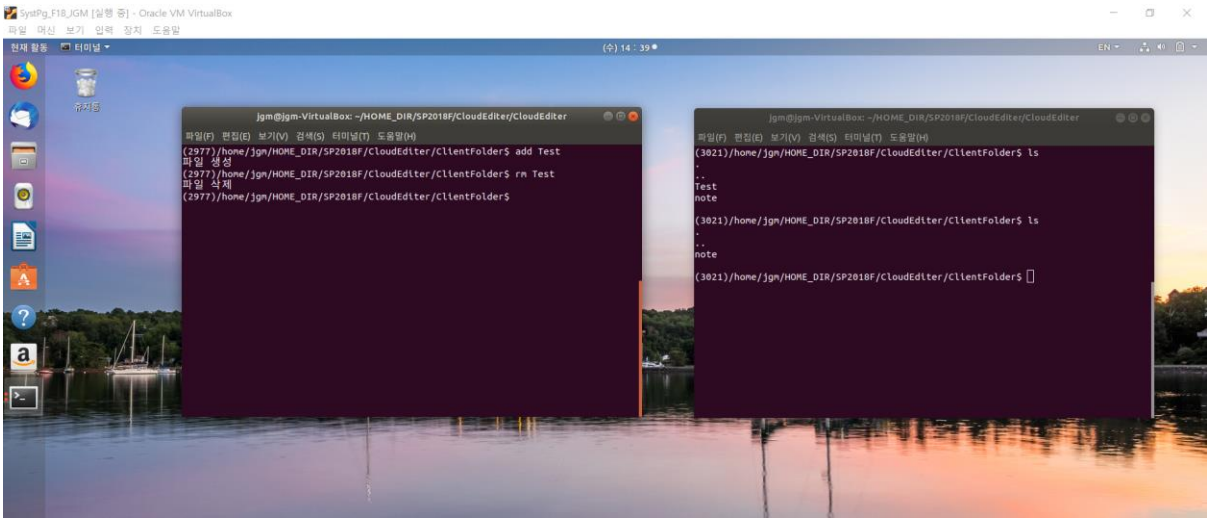
6. 작업을 그만하고 싶을 때 Exit 명령어를 사용해 클라이언트가 종료



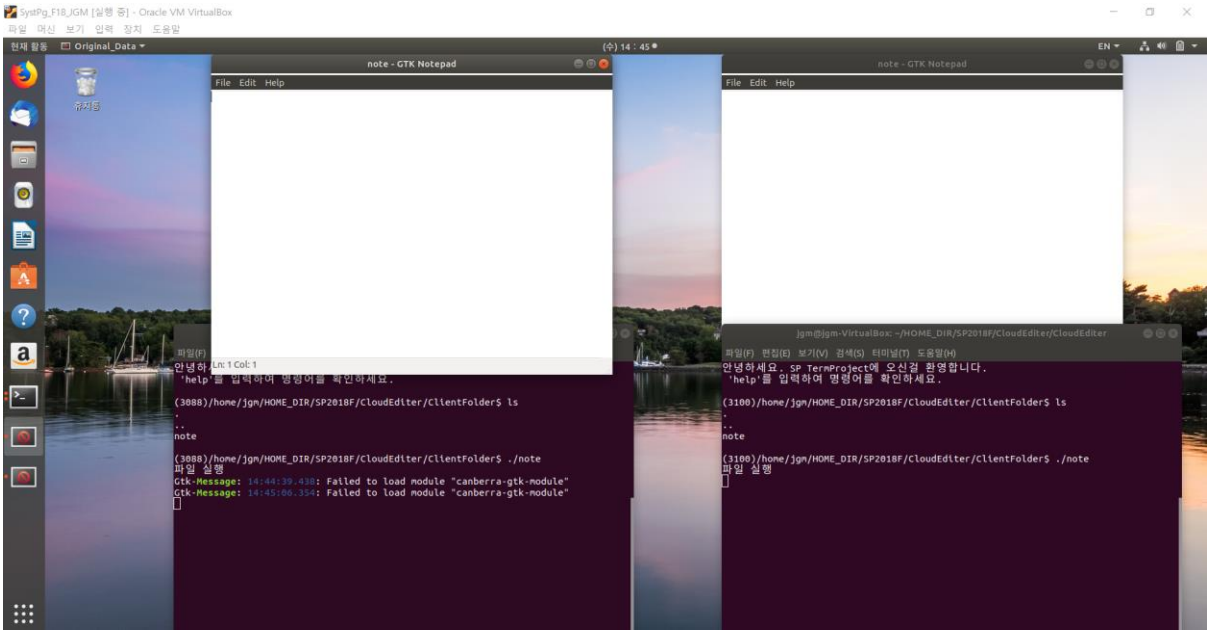
7. Add 명령어로 공유 문서 파일을 생성합니다.



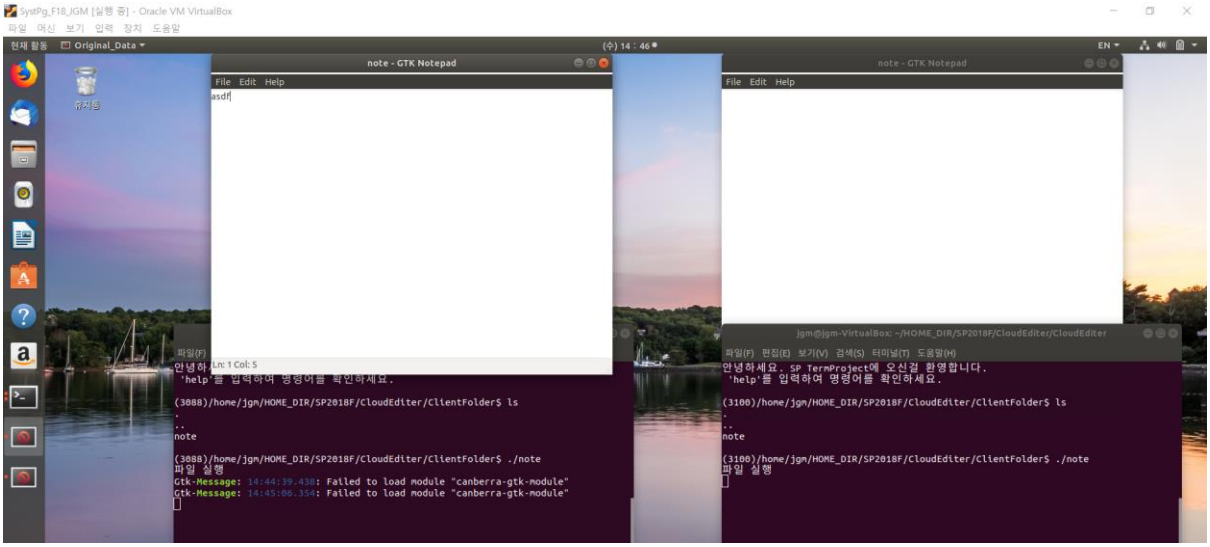
8. Rm 명령어로 공유중인 파일 삭제



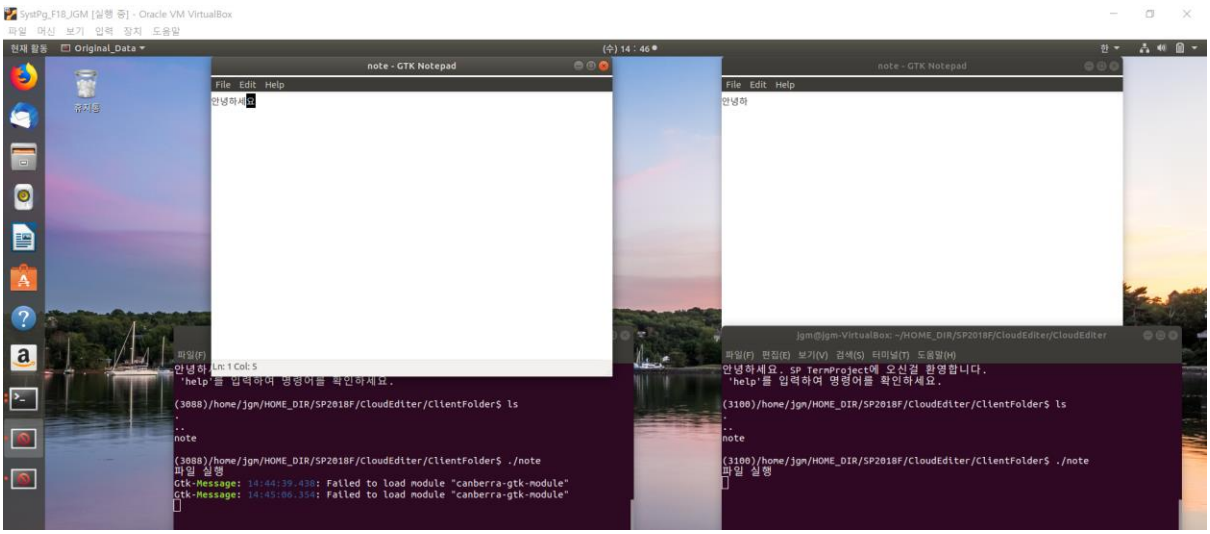
9. ./ 명령어를 통해 공유중인 파일 실행합니다.



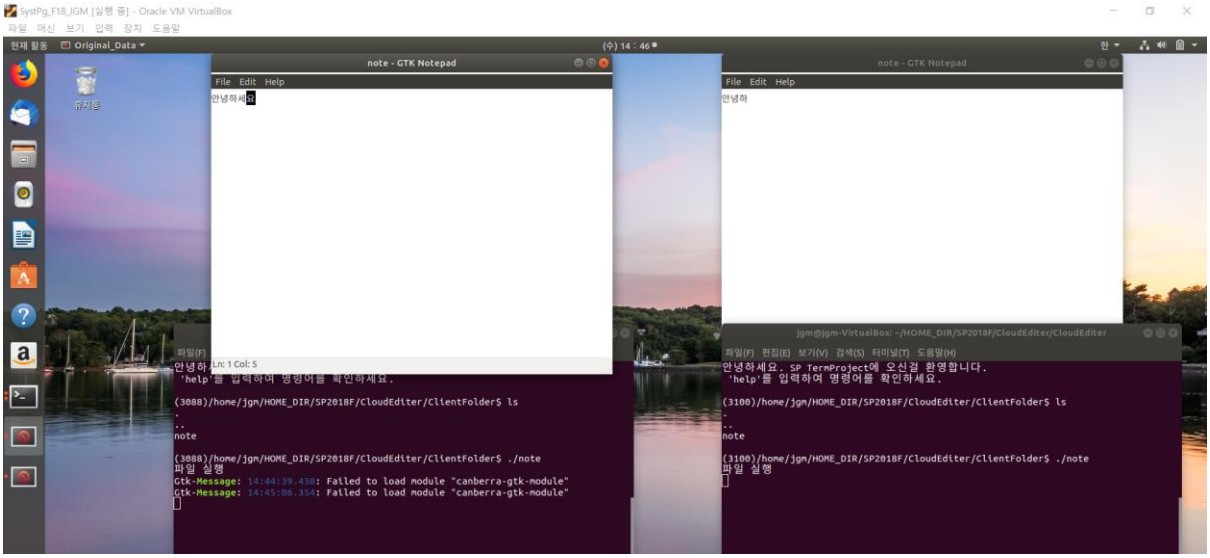
10. 두 개의 터미널을 사용해 문서를 편집합니다.



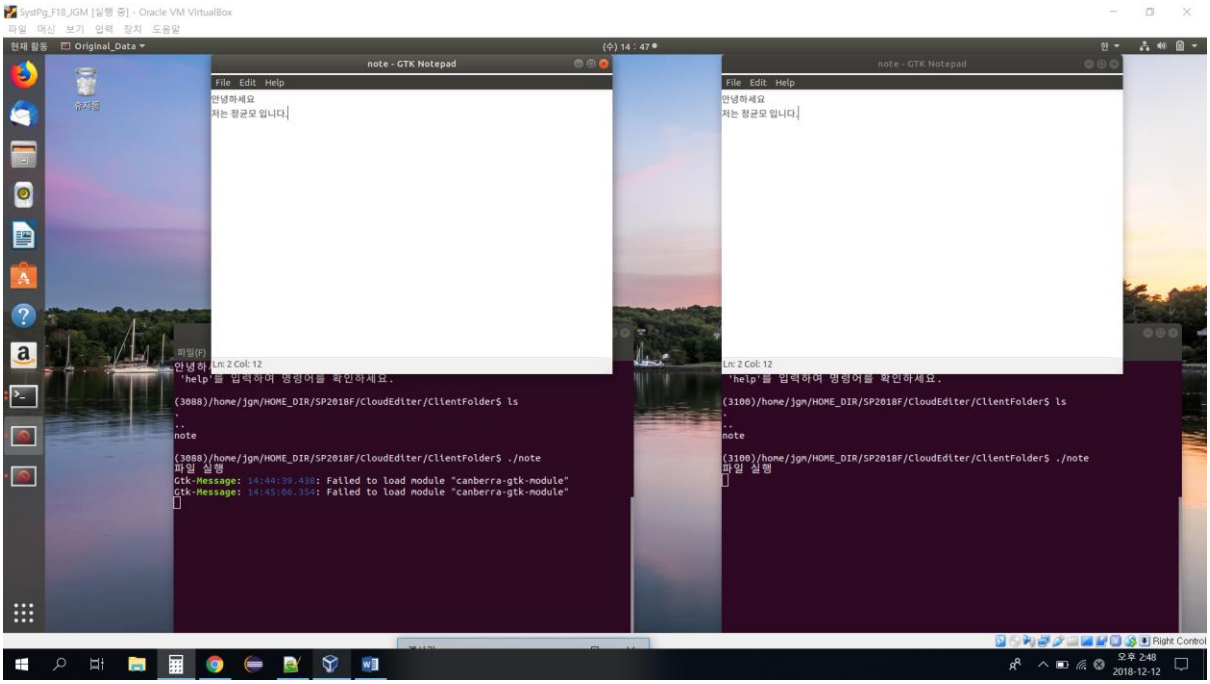
입력을 하면 다른 터미널에 글자가 나타는 것을 확인할 수 있습니다.



다른 터미널에서도 입력시 확인이 가능합니다.



11. 결과



Part III: 프로젝트 진행중 발생한 문제점 및 해결방안

1. 서버와 클라이언트간의 통신을 하는 도중 폴더의 파일리스트를 벡터에 저장하여클라이언트에게 벡터 전체를 보내주려는 작업을 하려 하였지만 문제가 발생하였습니다.

string 타입은 가변문자열이라는 문제와 각 문자가 메모리에 참조되어있는 상태라 클래스 직렬화라는 작업을 통해 보내줘야 한다는 것을 뒤늦게 알았습니다. 벡터의 모든 내용과 그 문자열을 모두 긁어와 클라이언트에게 보내는게 상당히 복잡한 문제이므로 관련 라이브러리를 알아보았지만 다음에 사용해 보도록 하고 일단 불러온 파일리스트 각각을 하나씩 보내주는 작업으로 진행하였습니다.

2. gtk+에서 제공하는 라이브러리를 사용하기 위해 sample 을 보고 에디터 형식으로 만드는 과정이 복잡하였고 앞으로 코딩할 때 실시간으로 저장하는 작업을 어떻게 해야할 지 고민을 했습니다.

사용자로부터 에디터가 실행되는 순간부터 매 초마다 시그널 알람을 발생시키며 파일을 쓰고 읽는 작업을 진행합니다. 이 과정에서 클라이언트들이 서로 수정사항이 있을때 만 저장하는 작업이 필요했고 저장시 다른 사용자들은 그 즉시 저장한 내용을 확인 할 수 있는 업데이트 작업이 필요했습니다. 그 결과 keyboard 의 이벤트(사용자의 키입력)이 발생하면 dirtybit 라는 전역변수를 1 로 바꿔주고 저장한 내용이 없으면 dirtybit 를 0 으로 두어 내용을 업데이트해주는 알람 핸들러 작업으로 문제를 해결하였습니다.

Part IV: 개선방향

저희가 개발한 프로그램은 최종적으로 원격지에서 사용자들이 문서를 작성하거나 수정하는 등의 기능을 하는 프로그램입니다. 하지만 저희가 개발하고자 했던 프로그램은 같은 서버에서 실시간으로 사용자들이 문서 내용을 공유, 작성, 수정 등의 협업을 할 수 있는 프로그램이었습니다. 쉽게 예를 들자면, "Git Hub" , "Google Docs" 와 같은 프로그램이 있습니다. "Google Docs" 를 저희 프로그램과 비교해서 살펴보자면 먼저 '모두가 함께 한 문서에서 동시에 작업을 진행' 할 수 있다는 점과 '모든 변경사항이 입력 즉시 자동으로 저장, 업데이트 된다는 점' 에서는 저희도 개발하고 싶었고 성공했습니다. 하지만 "Google Docs" 는 사용자가 "Google Docs"에 필요한 웹 브라우저를 설치를 하게 되면 서버와 클라이언트 사이에서 웹 브라우저를 통해 간단하게 메시지를 주고 받기만 하는 방식으로 다수의 사용자들이 공용 문서를 손쉽게 이용할 수 있다는 점이 있습니다. 그에 반해 저희는 서버에서만 실행되며 각 클라이언트에게는 GUI 파일을 보내는 것을 구현하지 못했습니다. 따라서 저희는 이 점을 개선해야 할 것입니다. 또한 문서가 길어지게 되면 각 사용자가 원하는 정보를 찾을 수 있는 방법 또한 개발해야 할 것입니다. 그리고 최종적으로는 글자 뿐만이 아닌 이미지, 그림 등을 포함하는 문서를 개발하는 방향으로 개선할 것이며 컴퓨터 뿐만 아니라, 태블릿, 스마트폰 등까지 호환 가능한 프로그램으로 방향을 잡아야 할 것입니다.

[끝]