운영체제 1차 프로젝트 보고서

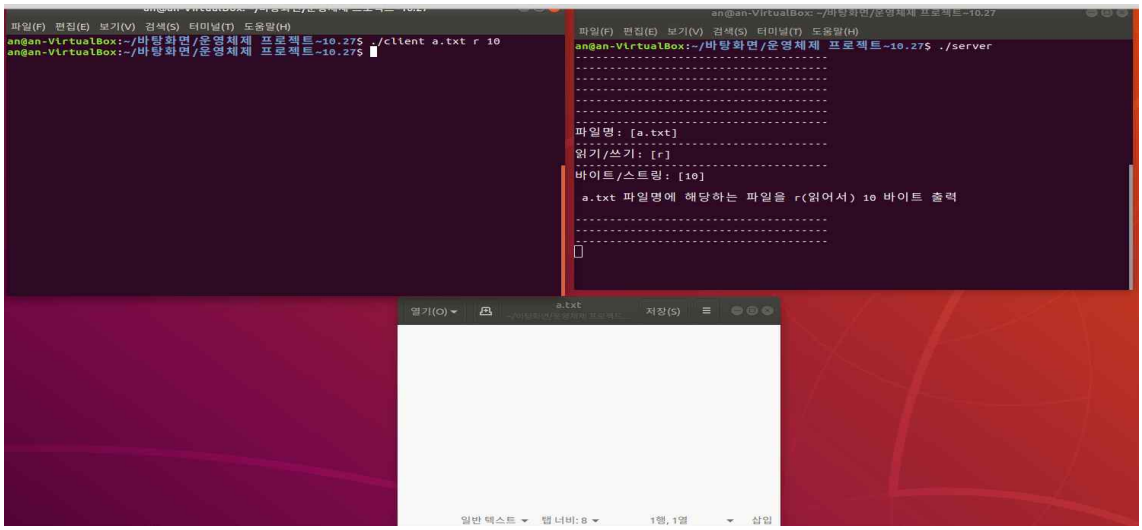
소프트웨어학과

2017 23272

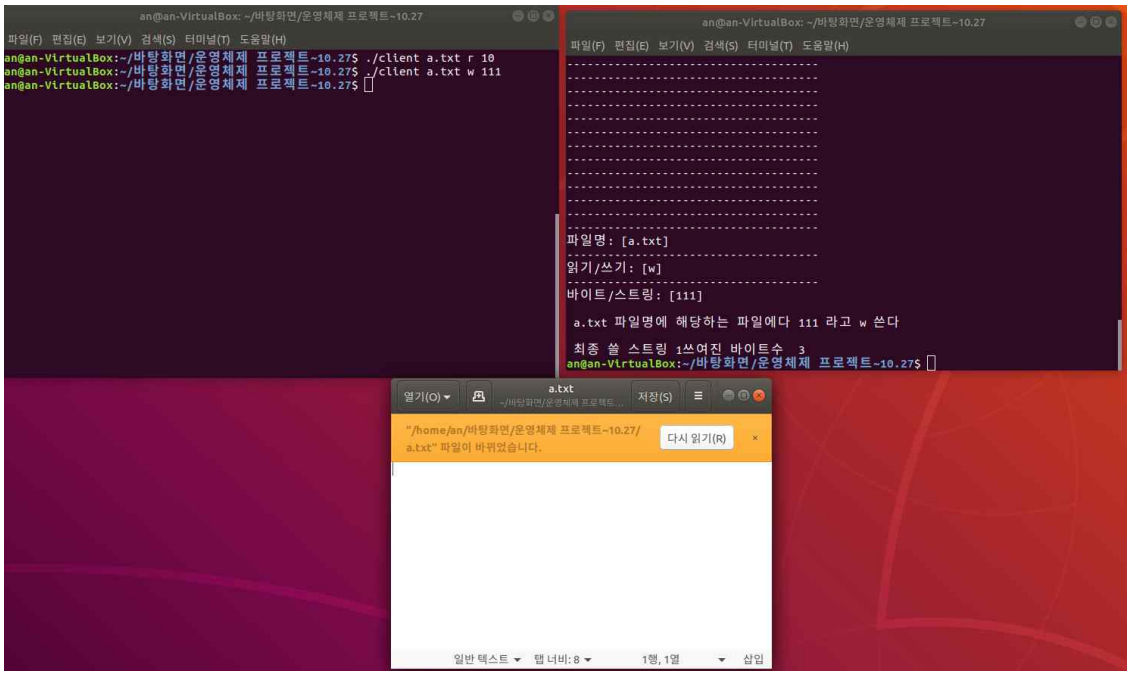
안창희

목차:

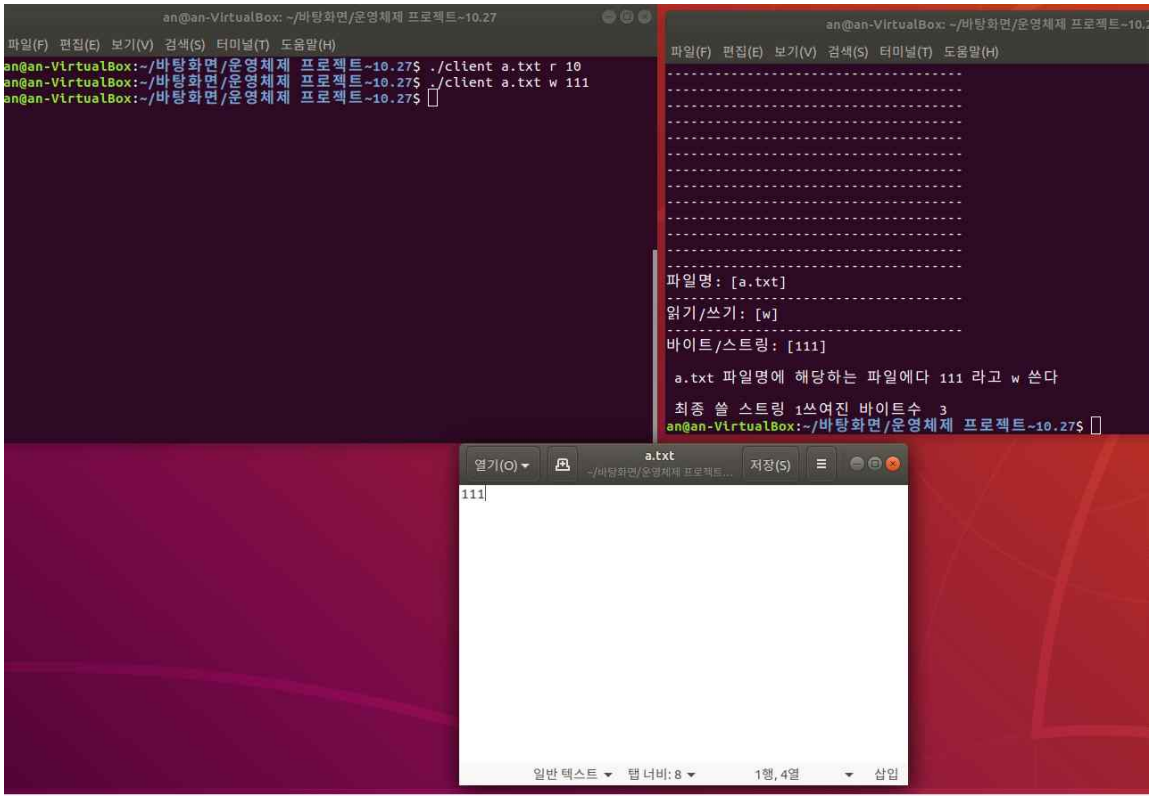
2. 동작시험 결과



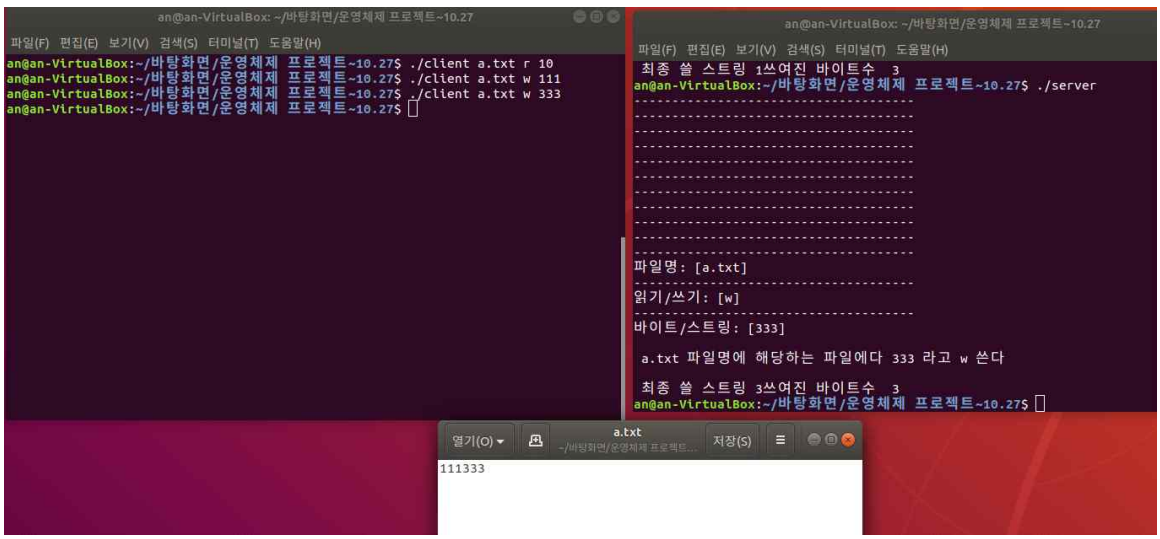
(a.txt 파일에는 아무것도 적혀있지 않은 상태)
서버가 기동된 상태에서 클라이언트가 a.txt의 파일을 10바이트만큼 읽어 들이라는 요청을 보냈다.
서버는 요청을 받아 분석한 후 그대로 파일의 내용을 출력했으며, 파일의 내용이 없어 출력된 내용이 보이지 않는다.
과정을 나타내는 출력문을 통해 파일내용의 출력이 이루어 졌음을 알 수만 있다.



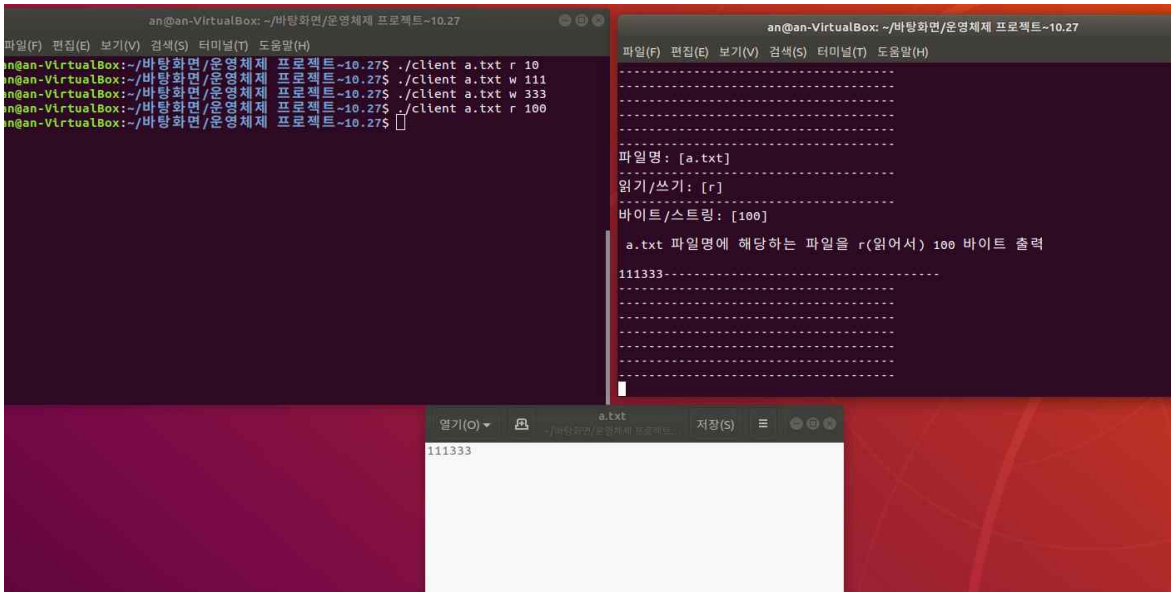
아무것도 쓰여 있지 않던 a.txt 파일에 111이라고 클라이언트가 작성 요청을 전송했다.
서버는 작성의 완료와 쓰인 스트링, 쓰여진 바이트 수를 차례로 출력한다.
=>이후 파일의 작성이 완료되어 텍스트 편집기에서 상태를 갱신하라는 메시지가 출력됐다.



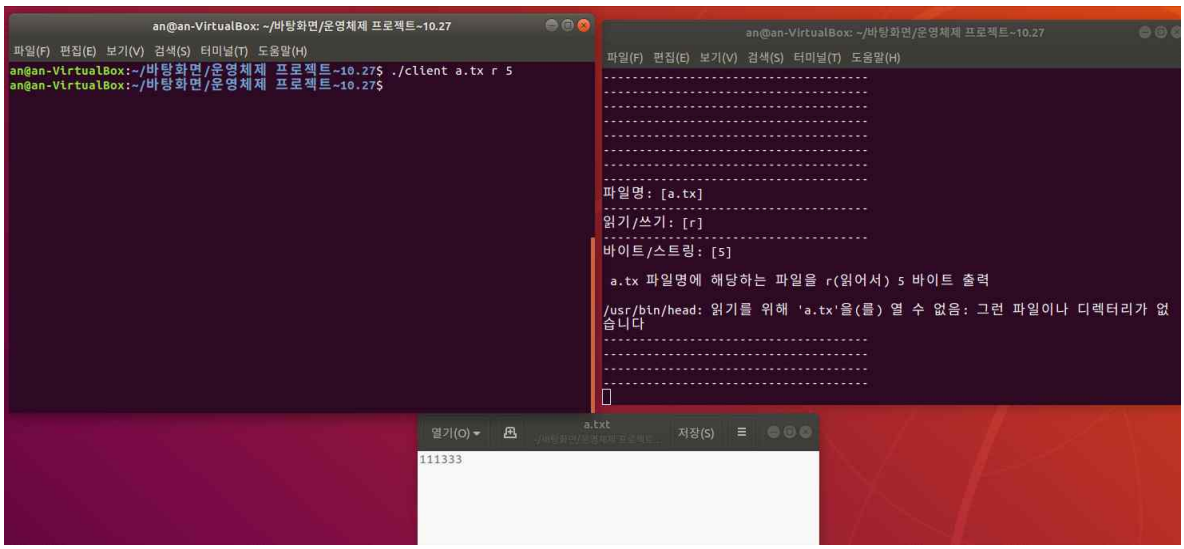
텍스트 편집기의 상태를 갱신하고 난 후 a.txt 파일에는 111로 출력이 완료된 것이 보인다.



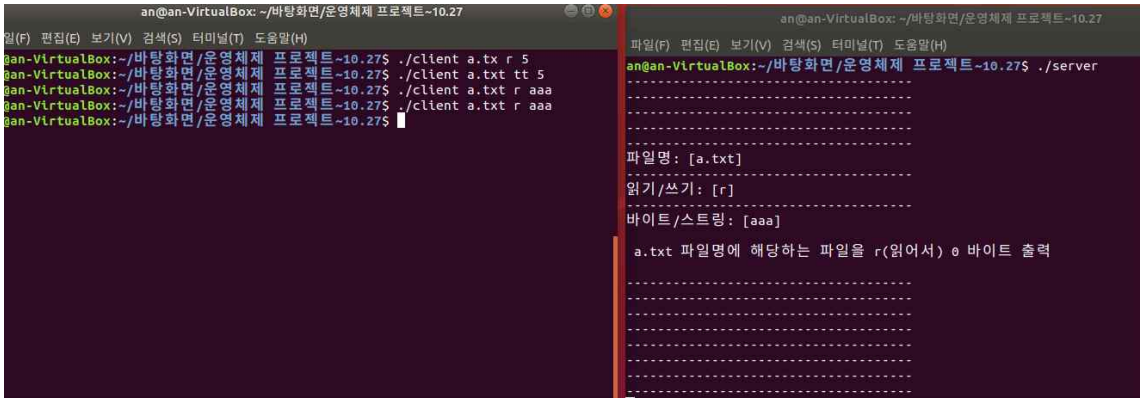
이후 333이라고 한 번 더 파일에 작성한 결과이다.
파일에 대한 스트림 개방 시 O_APPEND 옵션을 추가하여 write() 함수가 파일에서 작성된 부분을 이어서 작성하도록 하였다.



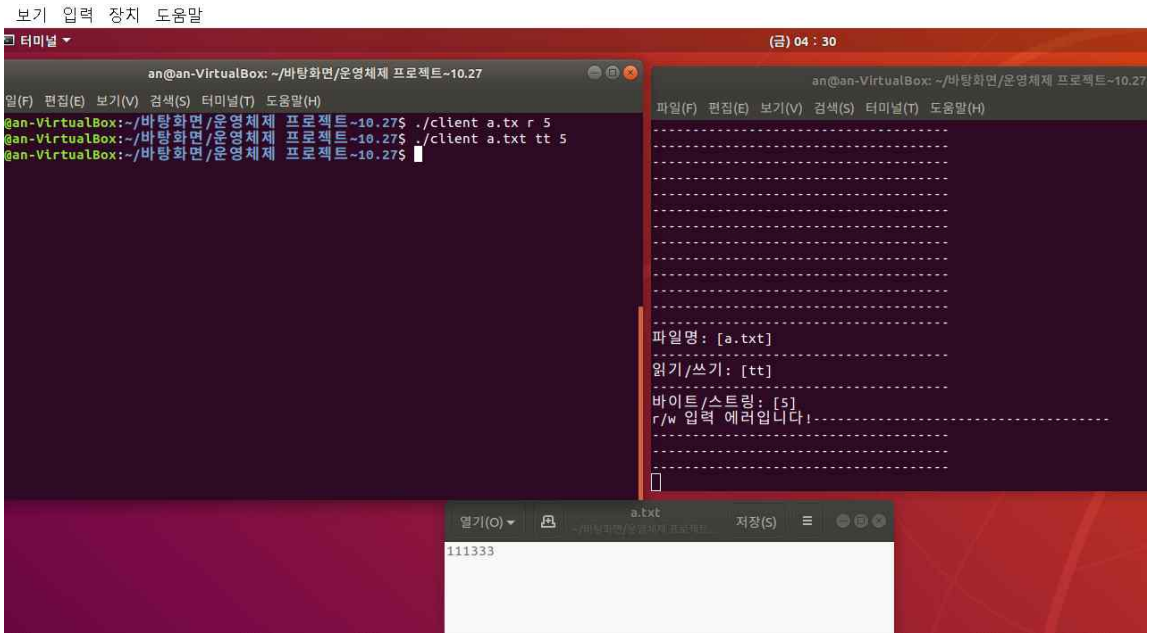
333까지의 파일작성을 완료하고 100바이트만큼의 파일 읽기를 요청한 상황이다.
파일에 작성된 내용은 111333 이므로 100바이트의 읽기를 수행해도 6번째 문자 이후로는 출력되지 않는다.



파일명이 잘못 되었을 경우의 오류문장 출력
이는 head 명령어의 내장된 오류문장 출력 기능이 작동한다.



읽기를 요청하고 바이트수를 잘못된 값으로 입력한 경우의 오류
바이트 수에 해당하는 값을 숫자가 아닌 값은 0바이트로 취급하여 출력하지 않는다.



파일의 입출력 유형에 r이나 w가 아닌 다른 값을 입력한 경우의 에러
r과 w 이외의 다른값이 입력되면 에러문을 출력하고 응답 대기상태로 돌아간다.

자가진단표

평가항목	점수	비고
클라이언트-서버 모델을 잘 이해하고 구현했는가?	8/10	클라이언트 서버 모델의 개념을 잘 숙지했고 네임드 파이프를 통한 send와 receive와 그에 따른 연산을 잘 구현함 하지만 양방향 통신보다 일방통신에 가까움
파일 입출력 개념을 잘 구현 했는가?	9/10	파일입출력 개념을 잘 이해했으며 리눅스 와 C언어 API를 적절하게 활용하여 요구사항에 맞는 기능을 잘 구현함 하지만 코드의 효율과 에러처리는 더 개선할 점이 있다고 생각됨
fork를 통한 parent child 프로세스를 잘 활용했는가?	8/10	fork를 통한 프로세스의 복제와 exec를 통한 자식 프로세스의 활용을 적절하게 배치하였고, 요구사항에 맞게 구현함 하지만 프로그램 구현 중 이 개념을 응용하여 더 효율적인 방안을 모색하지 못해서 아쉬움
각종 에러를 잘 검출했는가?	6/10	생각 가능한 에러에 관하여 구현했지만 리눅스 API와 사용에 대한 숙지가 부족하여 에러 검출에 대한 완성도가 높지 않음
코드의 가독성과 최적화가 적절하게 이루어 졌는가?	5/10	필요한 기능을 모두 구현하긴 했지만 코드 문장들이 적재적소에 배치되었다고 말하기 힘듦 특히 각종 변수와 기능에 대한 부분은 명확히 정의하거나 재사용성을 위해 함수정의를 사용하면 더 좋을 것 같음

3. 토의 및 느낀점

본인은 이번 프로젝트를 수행하면서 리눅스를 사용하는 프로그램을 처음 만들어 보았고,
C언어를 사용할 줄은 알았지만 처음 사용해보는 API가 생각보다 많아서 그것에 익숙해지고 공부하는데 시간과 노력이 많이 들었던 것 같다.

네임드 파이프 개념이나 파일의 입출력에 대한 기본 이론은 알고 있었지만 막상 과제를 수행하기 위해 코드를 짜는 과정에서는 문법만이 아니라 에러처리, 입출력 형식등 맞추어 하는 조건이 많았고,

자료구조, 객체지향 프로그램에서와 같이 코딩 기반으로 배우지 않았던 것에 대해 독학하며 구현하는 과정이 처음엔 매우 버거웠었다,

하지만 관련 예제와 책, 검색을 통해 하나씩 배워가면서 구현하는 과정에서 그동안 해왔던 프로그래밍 공부와는 다른 프로세스 간 통신과 시스템 콜, 그리고 파일 입출력 개념에 대해서 직접 코딩을 하면서 이해할 수 있어서 좋았다.

특히 mkfifo를 통해 파이프를 생성하는 과정중 실제로 입력된 파이프 이름을 가진 파일이 생성되는 것을 확인하면서 놀람과 깨달음을 동시에 얻을 수 있었다.

이외에도 execl 함수를 통해 프로세스를 초기화 하는 과정에서 각종 리눅스 명령어들을 포함하고 있는 bin 폴더와 usr/bin 폴더의 API들의 종류를 열람하고 그것들의 기능과 옵션을 공부하는 과정은 정말 흥미로웠다.

C언어에서만 지원하는 기능 외의 리눅스 API와 시스템콜을 사용해 요구사항에 맞는 기능을 설계하는 과정은 기존의 프로그래밍 과정보다 한차원 높은 프로그래밍을 경험하는 듯 한 느낌이었다.

하지만 그동안 구현해왔던 프로그래밍과는 다르게 각 기능을 완전히 이해한 것을 전제로 구현하는 게 아니어서 코드의 가독성이나 효율성과 같은 면이 스스로 생각하는 기대치 만큼 부합하지 못했던 것도 사실이다.

정말 힘들었던 프로젝트이고 스스로 생각하기에 잘 했다고 할 만큼 구현하지 못했지만, 다음 프로젝트를 위해서 앞으로는 강의시간에 배운 이론 공부를 미리 혼자서 실습해 보면서 이론적으로만 배우려 하지 않고 능동적으로 내것으로 만들어야겠다고 느꼈다.

이상으로 프로젝트 보고서를 마친다.