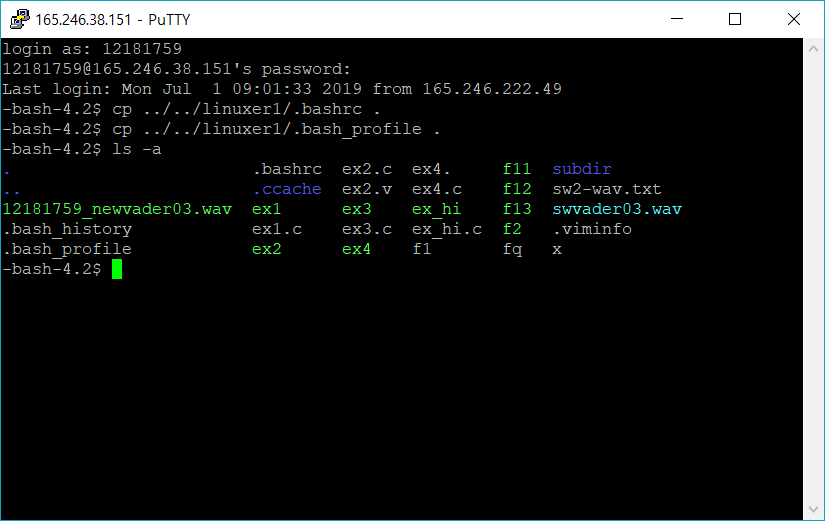
**시스템 프로그래밍**

**6주차 과제**

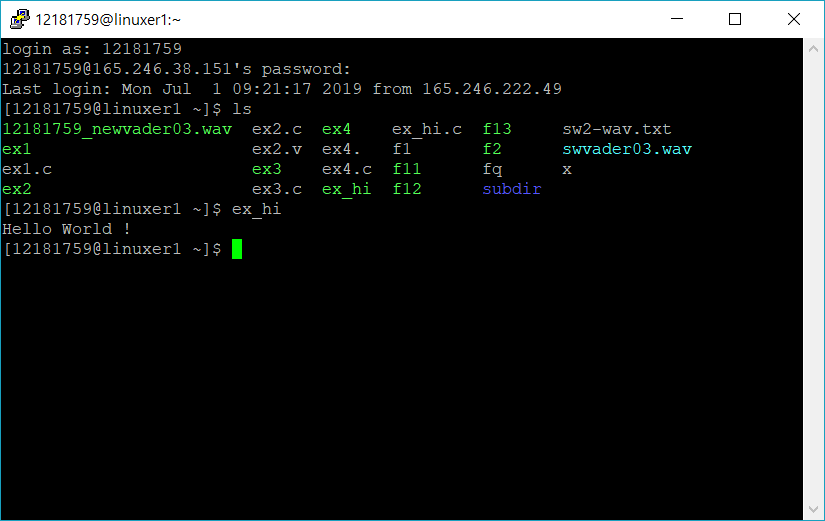
**12181759**

**김태완**

0)



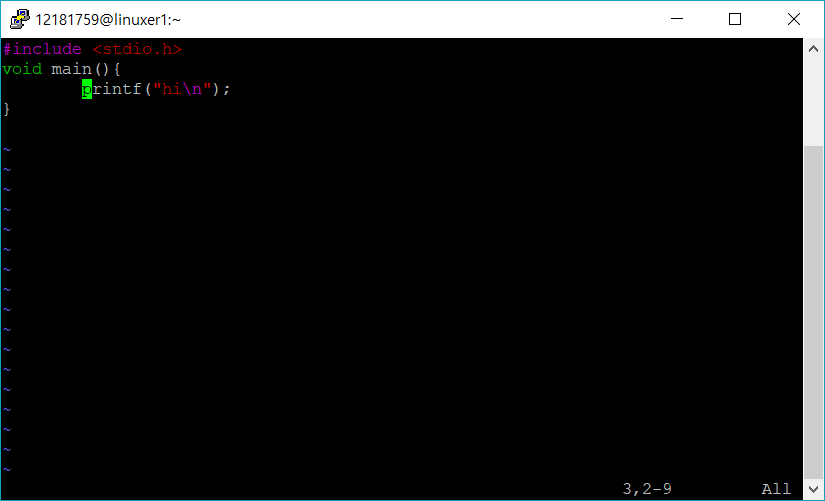
제시된 대로, Set up .bash\_profile을 동작한 후, ls -a로 모두 copy 되어있음을 확인한 후 터미널을 재 실행했다.



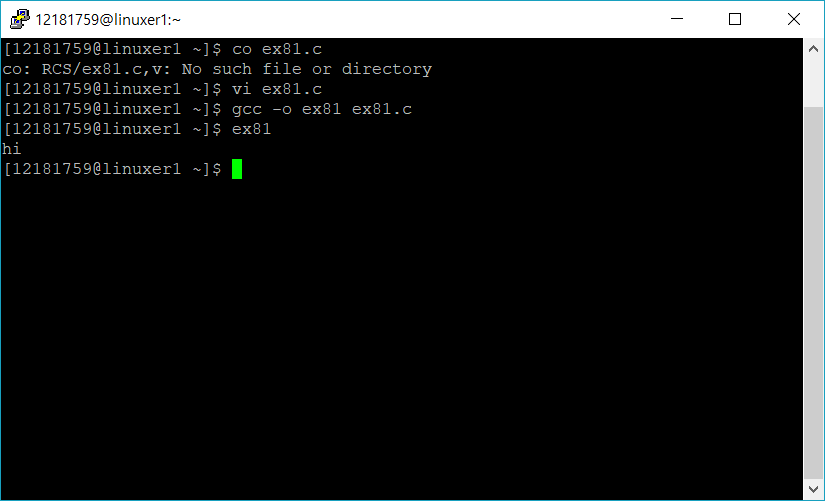
터미널이 [12181759@linuxer1 ~]로 변했고, ./ 없이도 실행파일이 동작됨을 확인했다.

모두 올바로 Set up 되었다.

1-ex1)

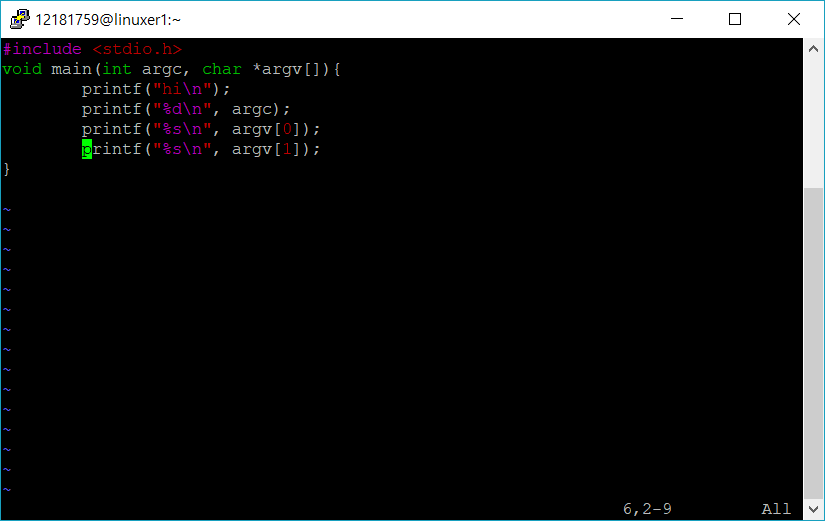


문제에서 제시된 코드를 ex81.c 에 작성하였다.

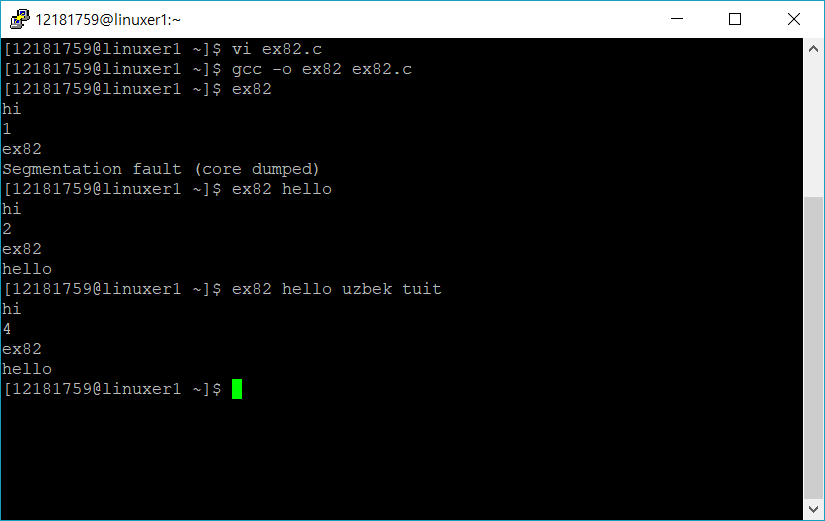


컴파일 후 실행 하자, ./ 명령없이도 정상적으로 동작함을 확인하였다.

1-ex2)

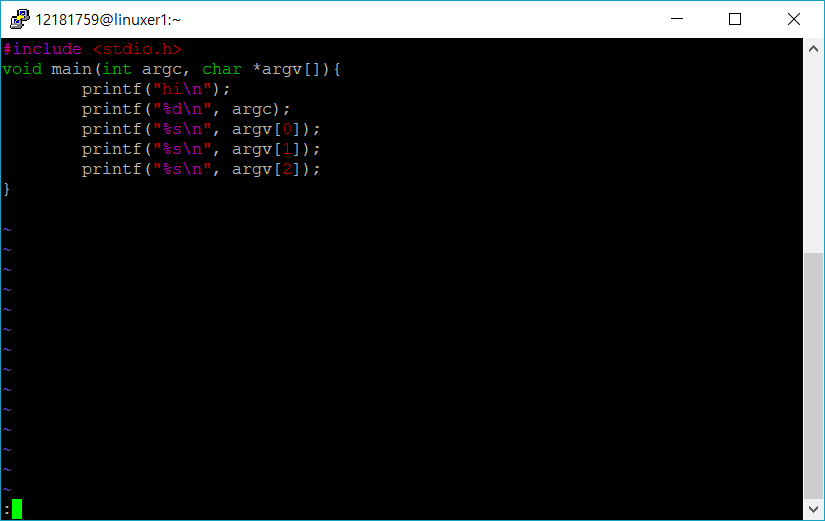


제시된 코드를 ex82.c 에 입력했다. 코드는 argc에 command line argument의 개수를 입력받고 argv에 그 string의 포인터를 저장한다.

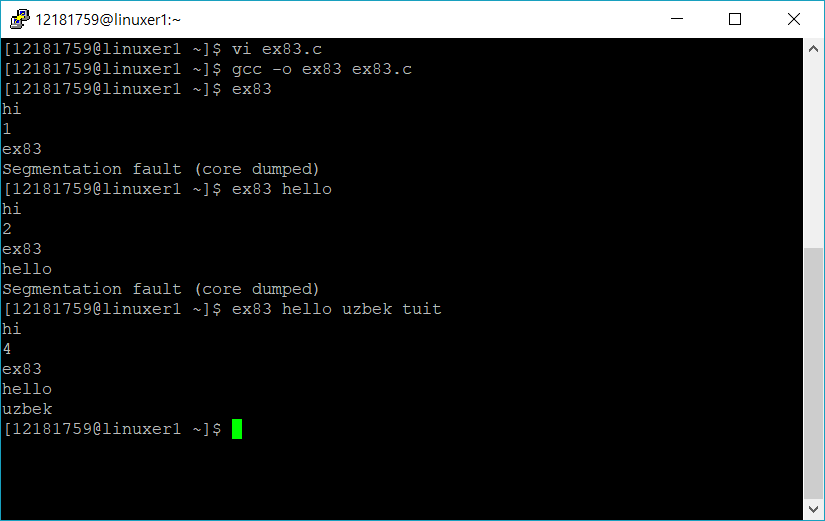


컴파일 후 실행한 모습이다. 먼저 $ ex82 를 시행한다. 이는 오류이다. main에서 출력으로 배열을 호출하기 때문에,반드시 하나 이상의 command line argument를 포함해야 하기 때문이다. 빈 배열을 호출하기에 오류가 난다. 다음으로 $ex82 hello 를 시행한다. argc의 개수가 2개로 늘었고, 출력 또한 내가 한 argument를 포함한 2개로 늘어난 모습이다. 마지막으로, $ ex2 hello uzbek tuit 을 시행하면, 출력은 위와 같이, 그 아래가 무시된다. 그 이유는 프로그램은 argv[1] 까지만 출력하도록 되어있기 때문이다. argc는 정상적으로 count 했다.

1-ex3)

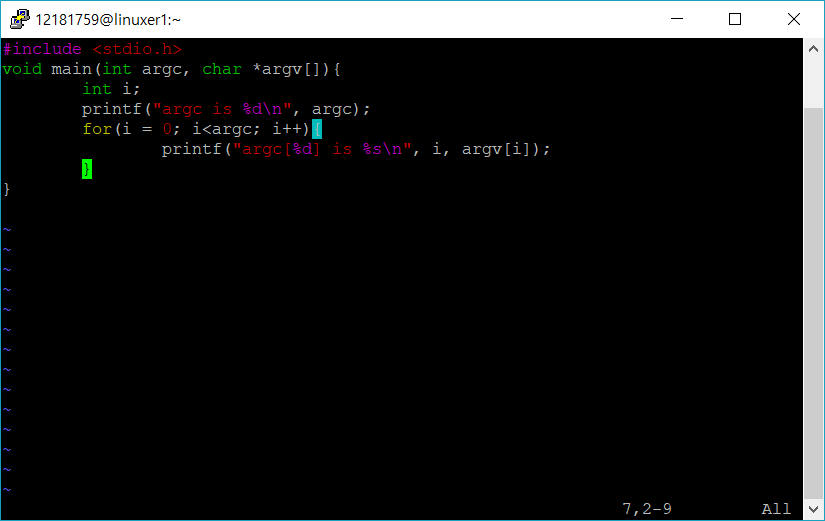


제시된 코드를 작성하였다. 코드는 ex2의 변형인데 ex2는 argument를 1개 출력하는데, ex3은 2개까지 출력하도록 작성되어 있다.

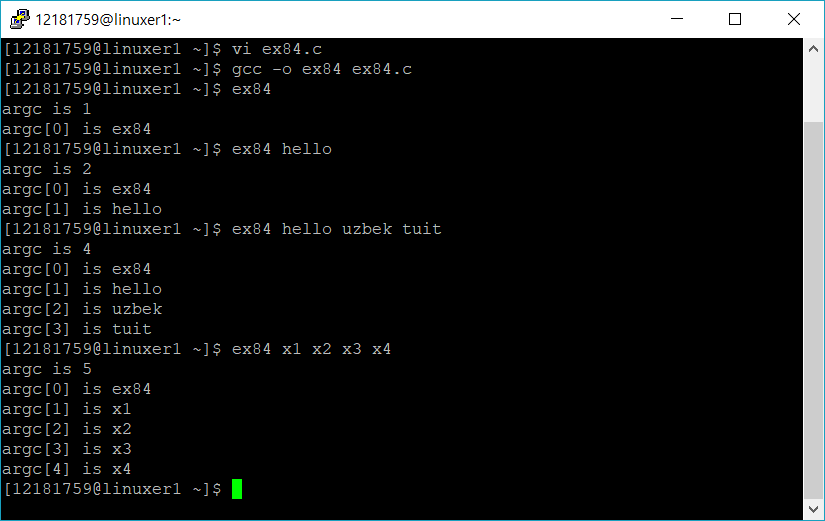


ex2에서 시행한 기능을 모두 동일하게 시행했다. 1번째 기능은 동일하게 수행된다. 2번째부터 그 차이가 보이는데, ex2는 argument를 1개만 출력하기에, 최소 1개의 argument만 있으면 충분했지만 ex3은 2개의 argument를 출력해야 해서 최소 2개의 argument가 필요하다. 따라서 1개의 argument만 포함한 $ex83 hello 도 오류인 것이다. 3번째 기능에서는 ex2는 hello 까지만 출력되었던데 반해, uzbek 까지 출력되었다. 코드가 2개의 argument를 출력하도록 작성되어있기 때문이다. 따라서 3번째 argument인 tuit는 무시된다. argc는 정상적으로 count되어 4이다.

1-ex4)

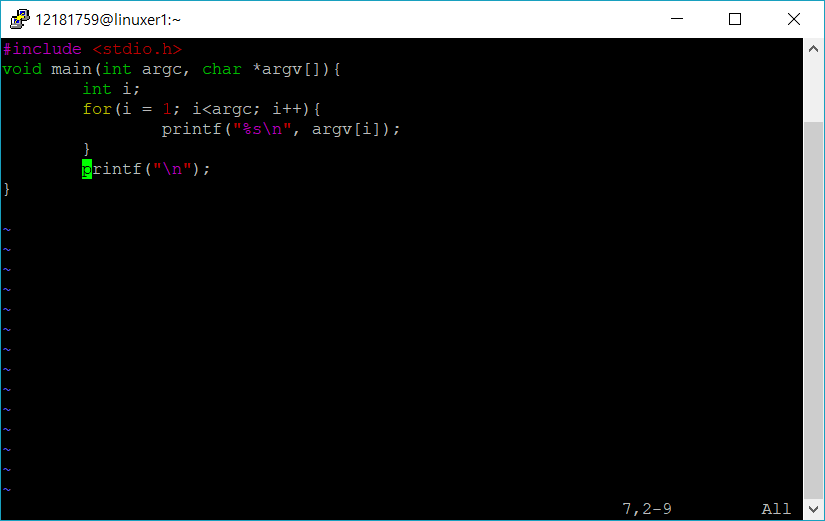


제시된 코드를 ex84.c에 작성하였다. 이 코드는 입력받은 argument가 몇 개이든 다 출력하도록 되어있다. argc에서 count된 argument의 개수만큼 for문을 돌리면서 argv 문자형포인터 배열에 저장된 string을 모두 출력한다.

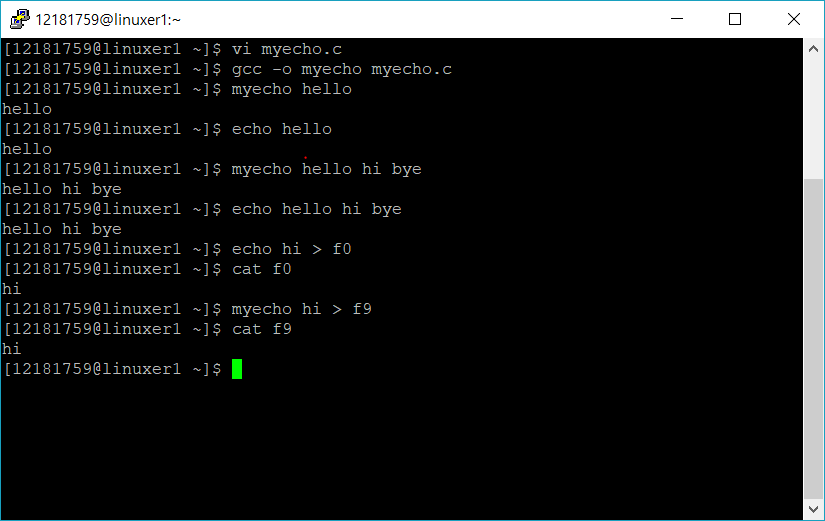


ex2, ex3에서 시행한 기능들을 돌려보았다. $ex84, $ex84 hello 모두 정상적으로 동작한다. 반드시 n개의 argument를 출력한다는 위의 예제들과 달리, argc에서 count된 만큼만 for문을 돌며 argv를 출력하기 때문에 오류가 나지 않는다. $ex84 hello uzbek tuit, $ex84 x1 x2 x3 x4 와 같이 더 많은 argument를 주어도 for문을 도는 횟수만 늘어날 뿐, 모두 정상적으로 출력되며 argc도 모두 정상적으로 count 된다.

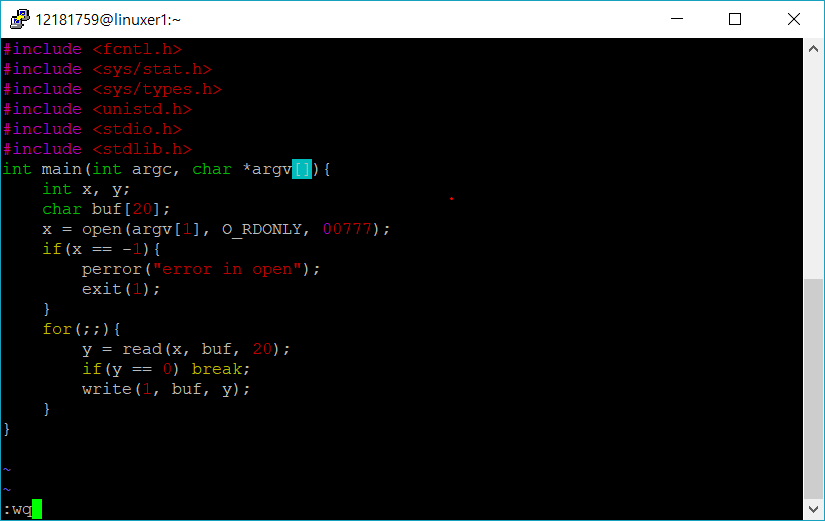
1-myecho)



제시된 코드를 작성하였다. 코드는 먼저 입력된 argument의 개수, 즉 argc만큼 for문을 돌면서 argv 문자열 포인터 배열에 저장된 string을 출력한다. echo 명령과 같은 기능을 수행한다.

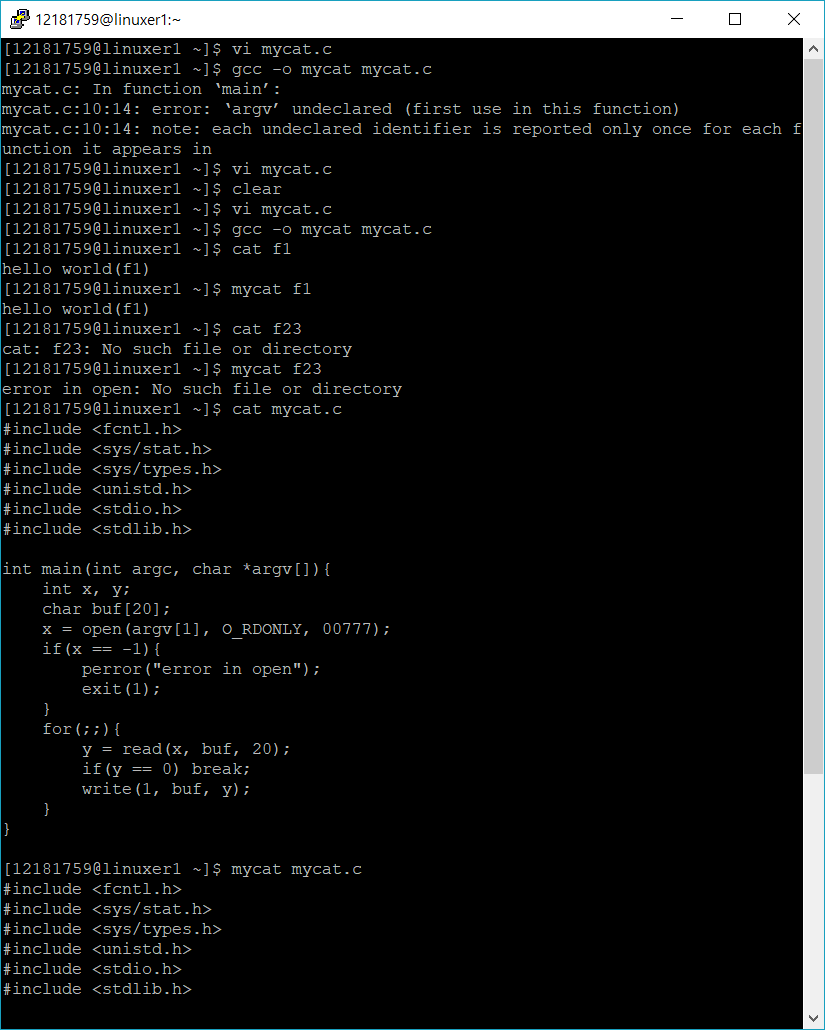


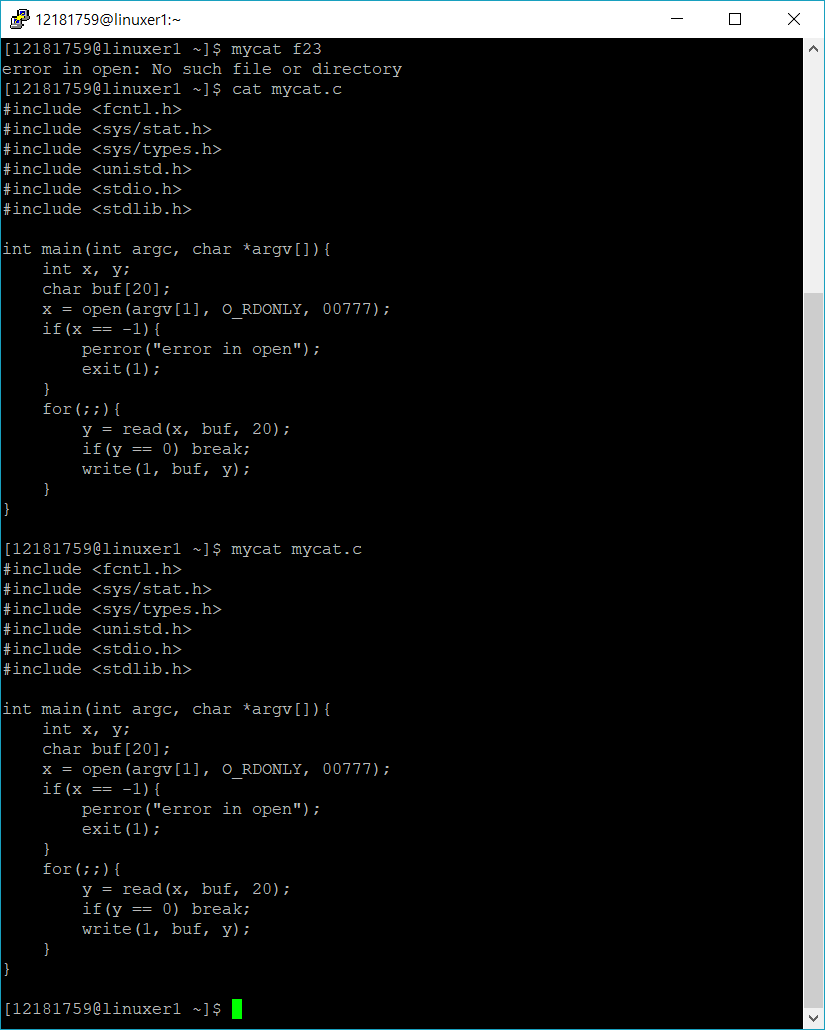
강노에 제시된 예제처럼 myecho와 echo를 번갈아가며 그 기능을 비교해보았다. 모든 기능이 동일하게 정상적으로 구현됨을 확인하였다.

1-mycat) 

제시된 코드를 작성했다. 코드는 먼저 x에 첫번째 argument로 입력된 이름의 파일을 open하고, 만약 이것이 open되지 않으면 error를 출력한다. 다음으로 무한 루프를 돌면서 그 파일안에 있는 문자를 20 byte씩 읽어와 std 1, 즉 화면에 출력한다. 만약 y가 0이 되면, 무한루프를 멈추고 프로그램을 종료한다.

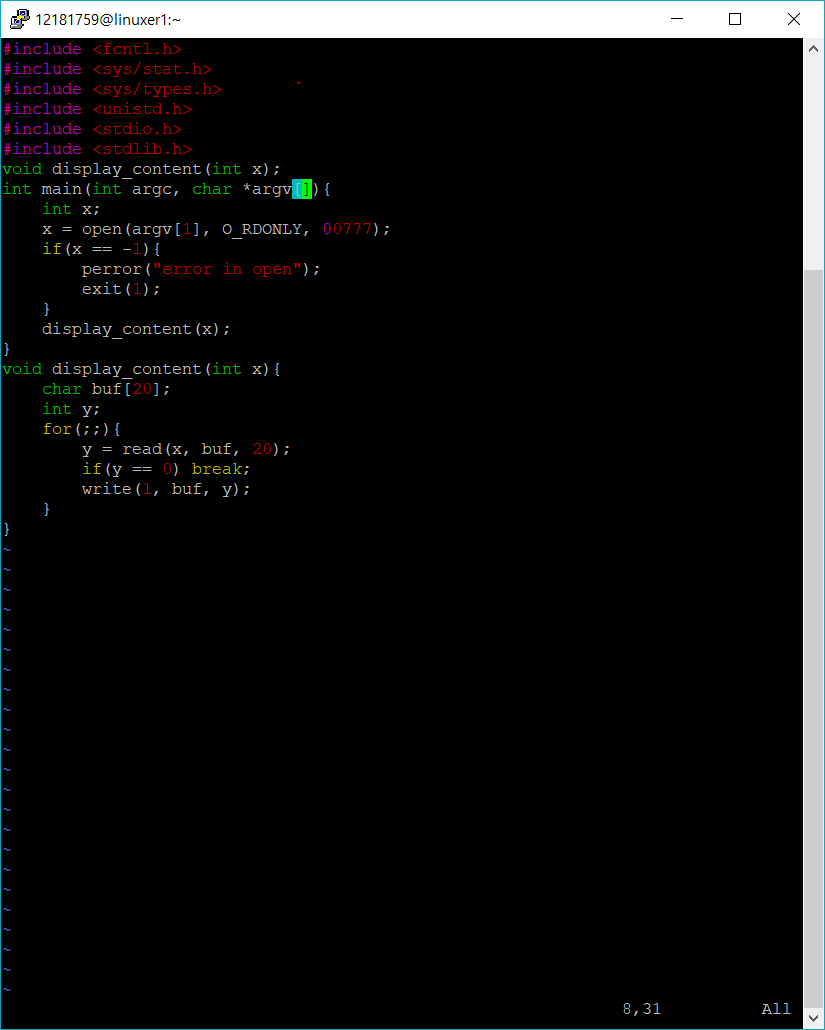
아래는 강노의 예제를 시행해본 모습이다.



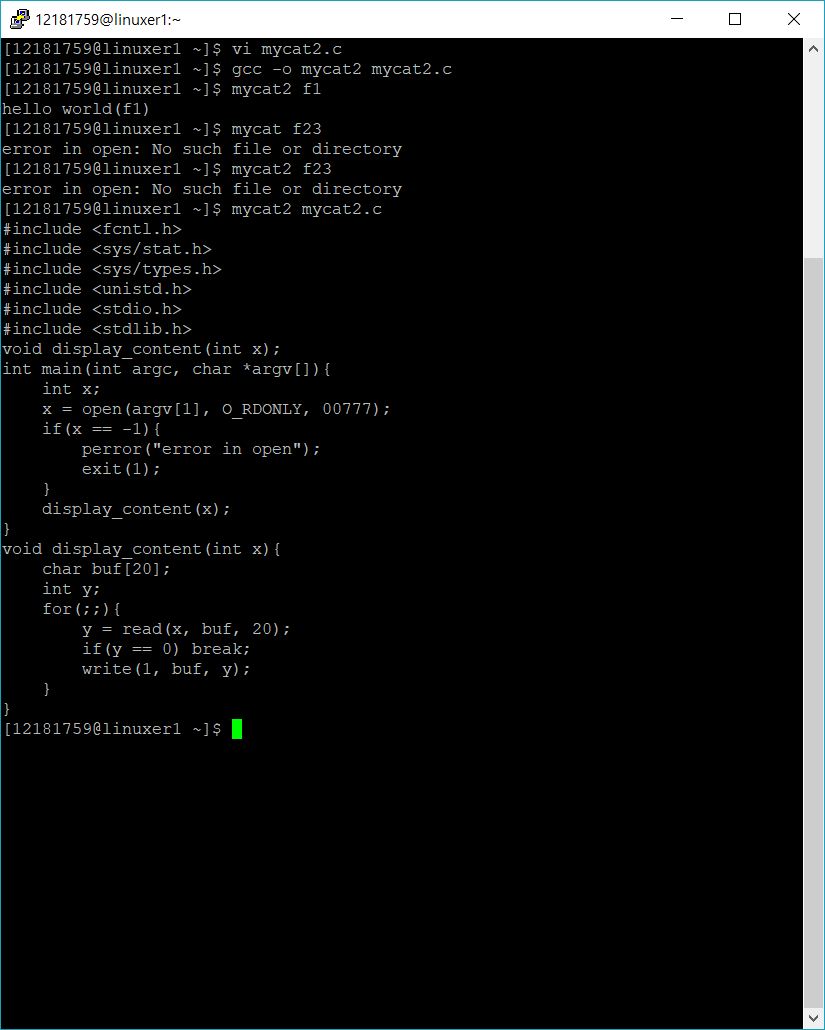


본인은 file1에 hello world(f1) 을 입력해 놓았는데, 이것이 cat과 mycat 모두에서 올바로 출력됨을 확인하였습니다. 다음으로 없는 파일에 대해서는 error msg가 정상적으로 출력됨도 보았습니다. mycat mtcat.c로 자기 자신도 자기 자신을 프로그램으로 하는 파일로 열린 다는 사실이 조금 신기했지만 생각해보면 실행파일과 c파일인데 당연한 것이었습니다.

1-mycat2)

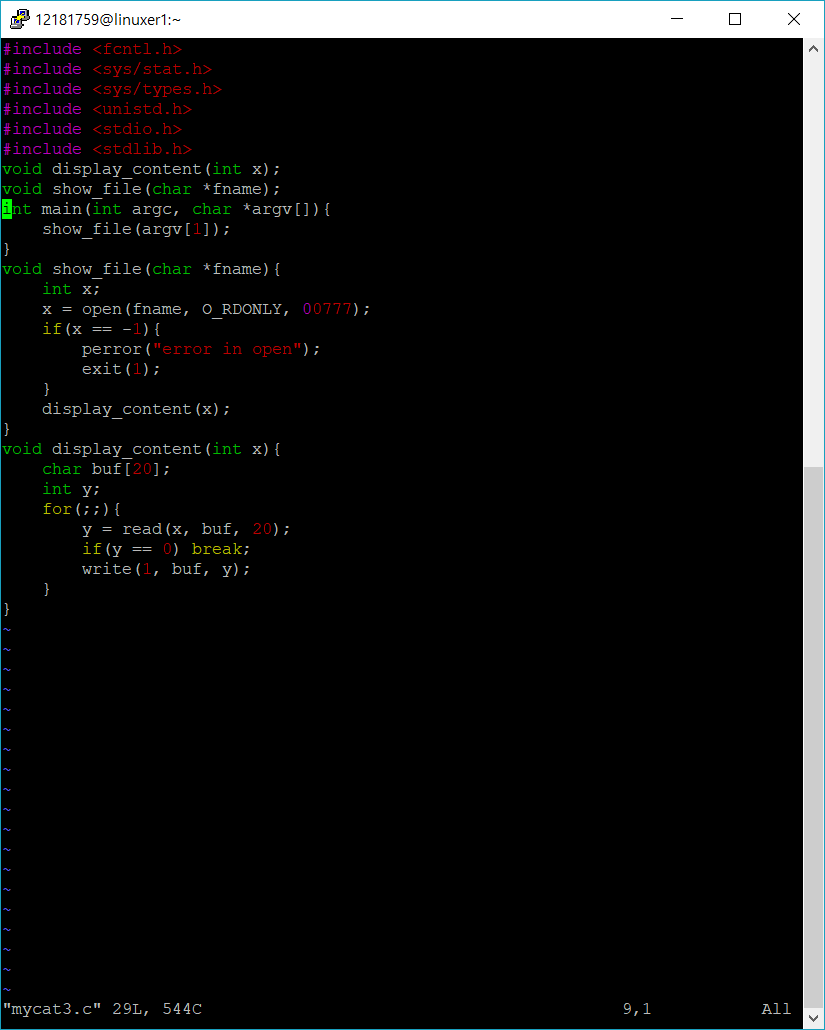


제시된 mycat2 를 작성했습니다. mycat2는 위의 mycat 코드에서 x 파일에서 20바이트씩 읽어와 화면에 출력하고, 더 이상 byte를 읽지 못하면 프로그램을 종료하는 부분을 display\_content라는 함수로 만들어서 작성한 파일입니다. 당연히 mycat과 동일한 기능을 수행합니다.

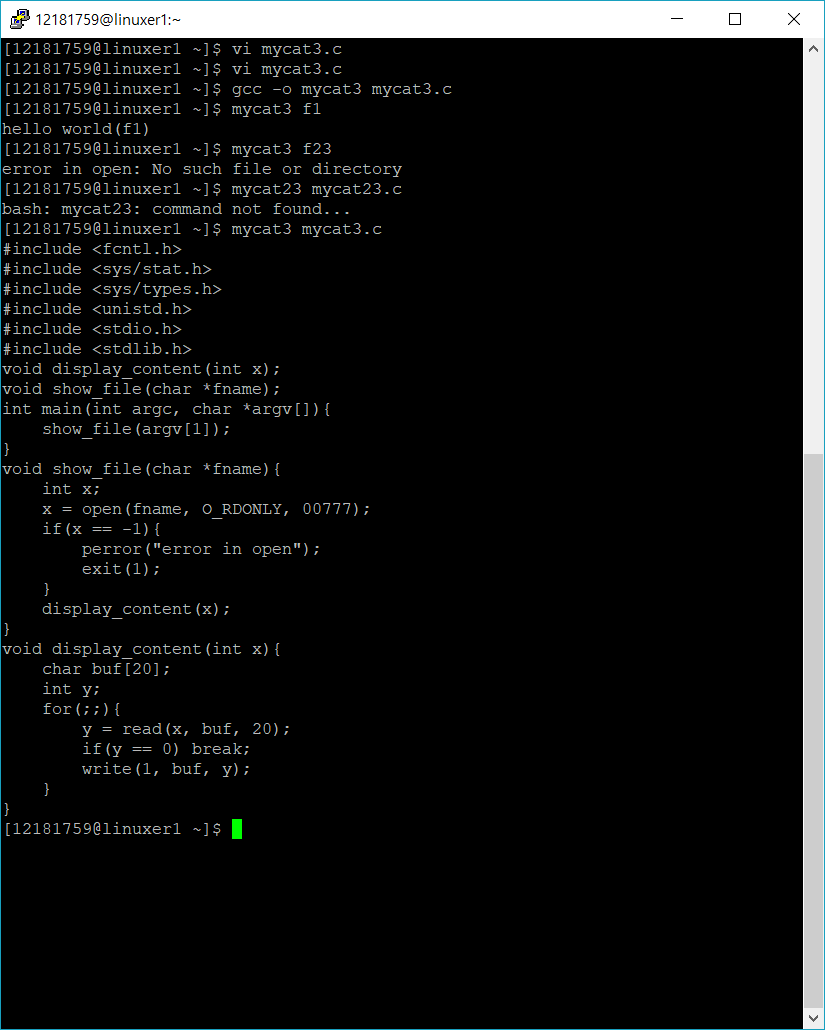


mycat에서 수행한 기능들을 실행해본 모습, 모든 기능이 mycat과 동일하게 정상적으로 수행됨.

1-mycat3)

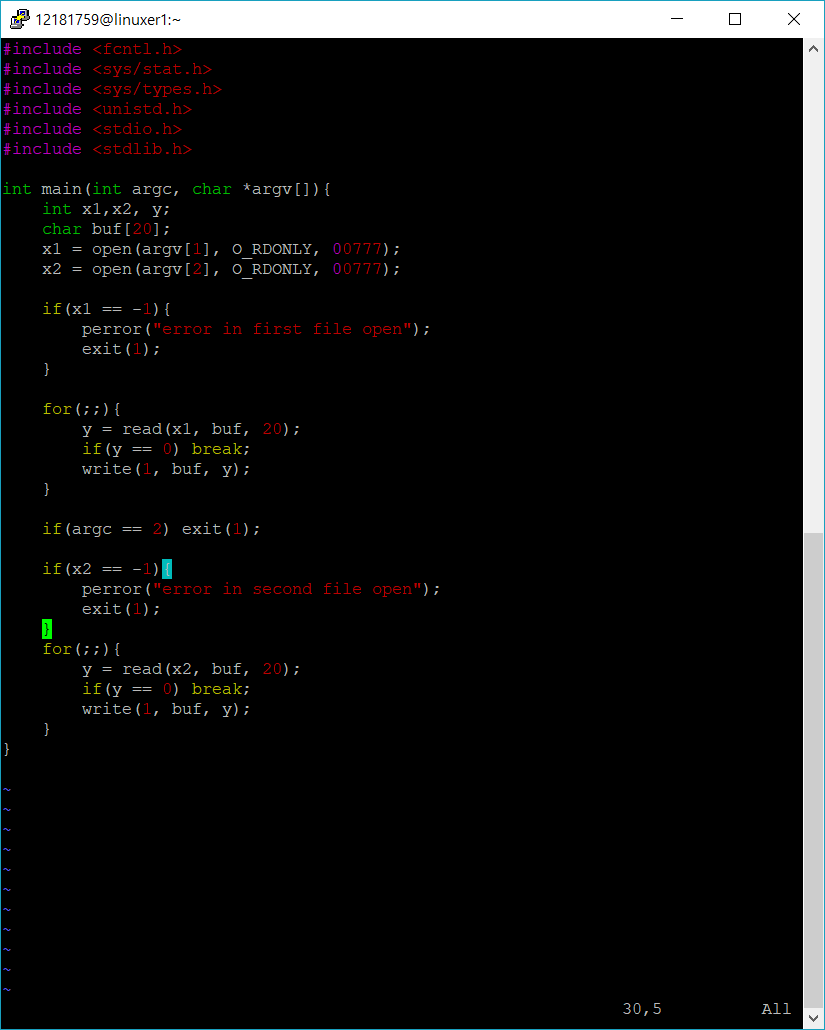


제시된 코드를 작성하였다. 이 코드는 mycat2 에서 main에 남아있는 부분을 show\_file 함수로 작성하고, 이 함수 안에 display\_content를 다시 받아와서 이 함수 하나만으로 main을 구성하고, 이 만으로 프로그램이 동작되게 되어있다. 모든 기능의 수행은 mycat, mycat2, cat과 같아야 한다.



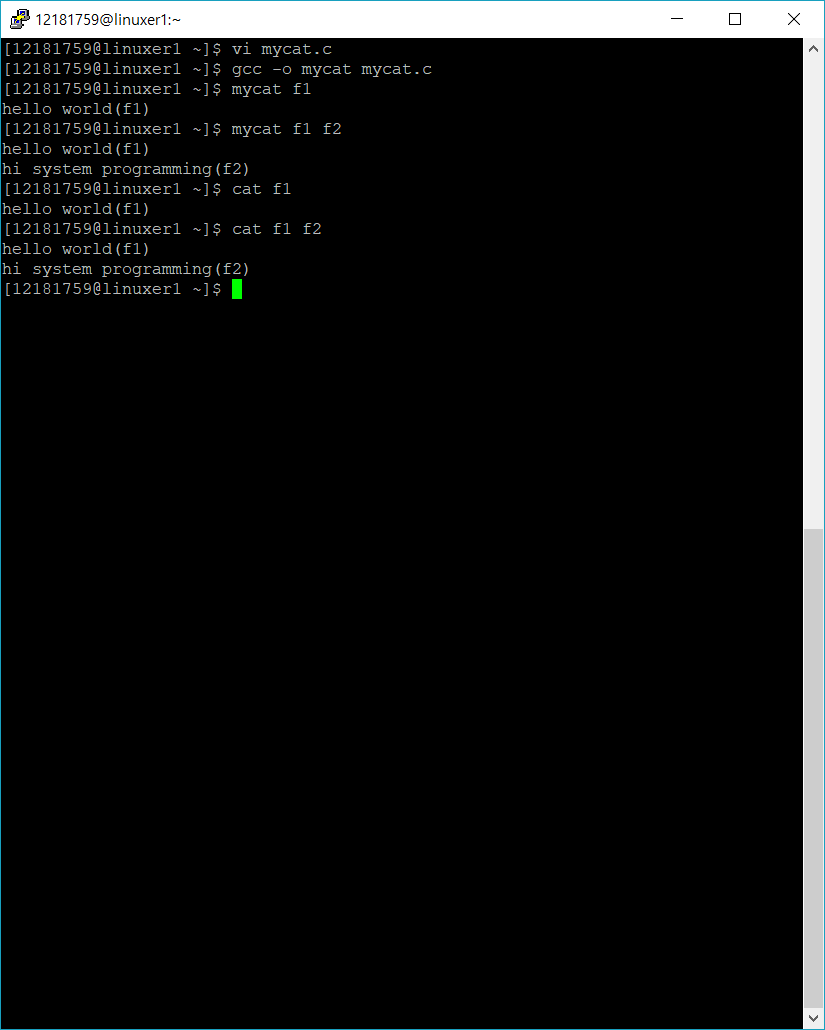
모든 기능이 정상적으로 동작함을 확인하였다.

2)



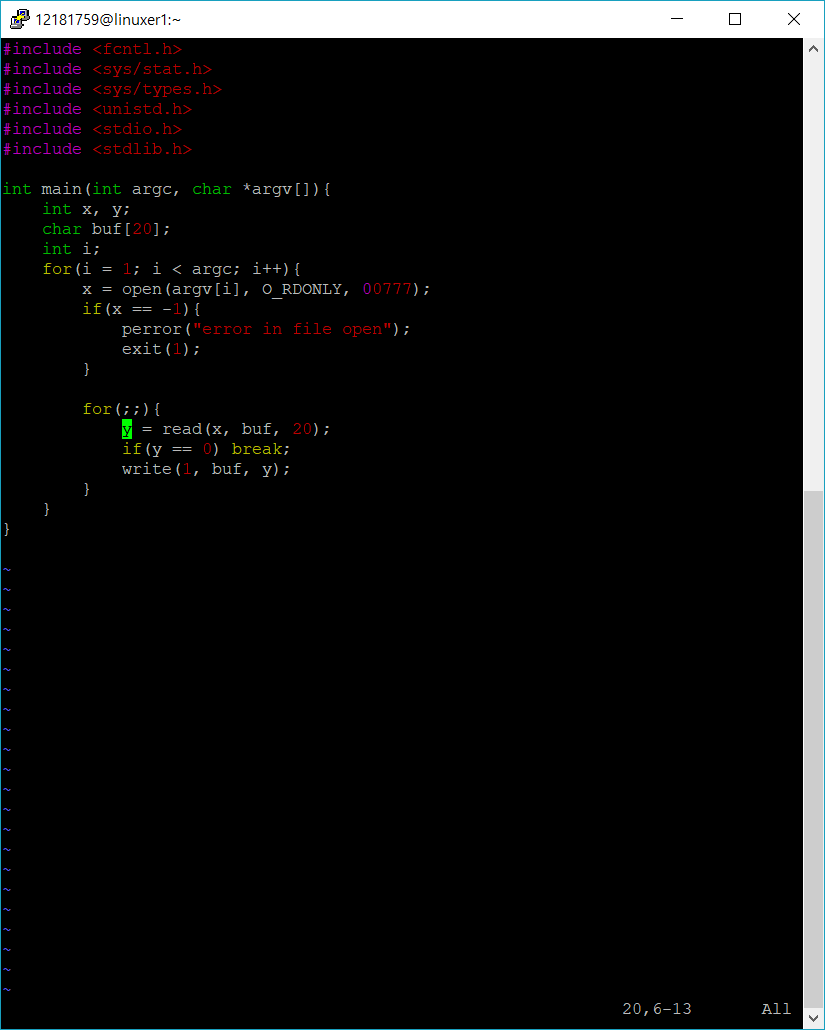
문제에서 요구하는 프로그램을 작성하였다. 먼저 x1, x2 개를 파일을 저장할 변수로 선언한다.

그 다음에 x1에 첫 argument로 들어온 file, x2에 두번째 argument로 들어온 file을 저장하고 연다. 예외처리는 각각 해주었다. x1이 읽어오지 못하면 first file이 error이다 라 해주고 x2가 읽어오지 못하면 second file 이 error이라 해준다. 출력은 먼저 x1 에 있는 file부터 출력한 후, 혹시 argc가 2, 즉 argument가 1개만 있는 상태라면 거기서 프로그램을 종료한다. 만약 아니라면, 그 다음 argument 하나에 대해 동일한 기능을 수행한다.

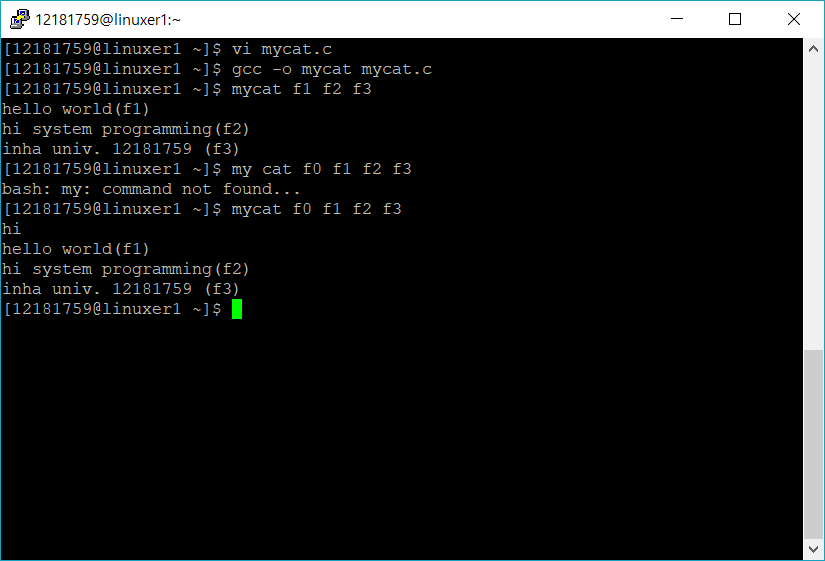


모든 기능이 제시된 대로 잘 구현되었다.

3)

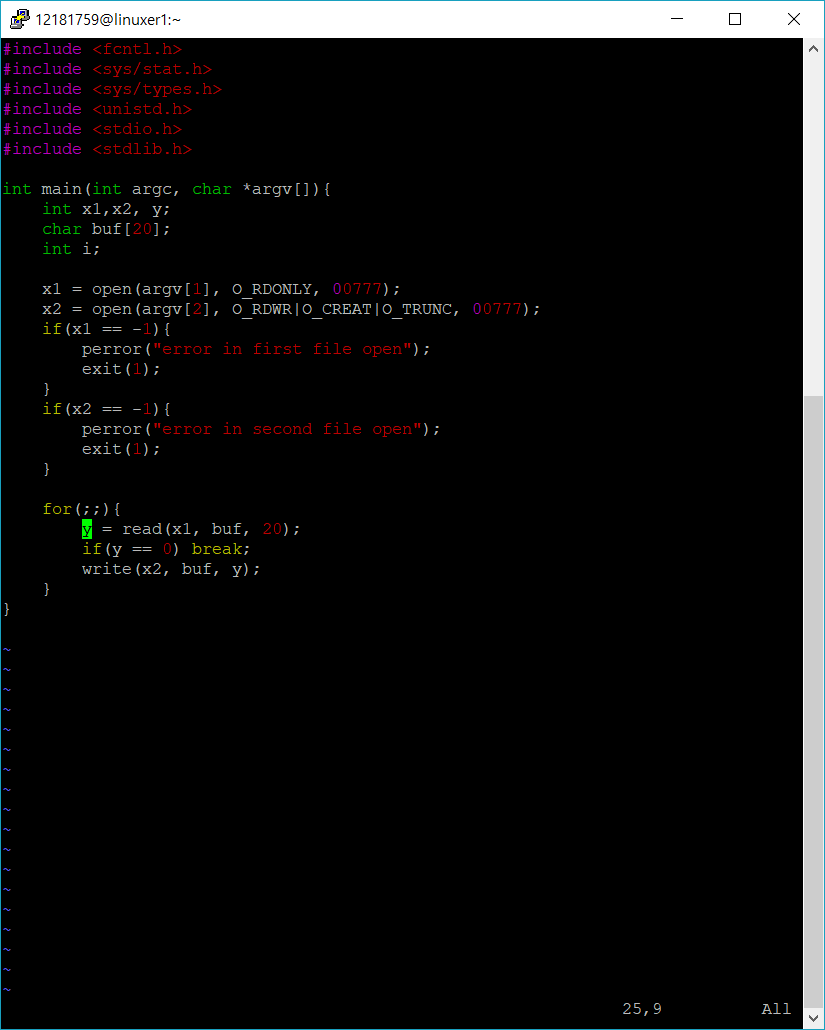


문제에서 말한대로 코드를 작성한다. argc의 값 만큼 for문을 1부터 돌며 argv[i] 번째의 파일을 open 하고 이를 출력한다. 이렇게 하면 argc의 값이 곧 argument의 개수 이므로 입력된 모든 파일을 돌 수 있다.

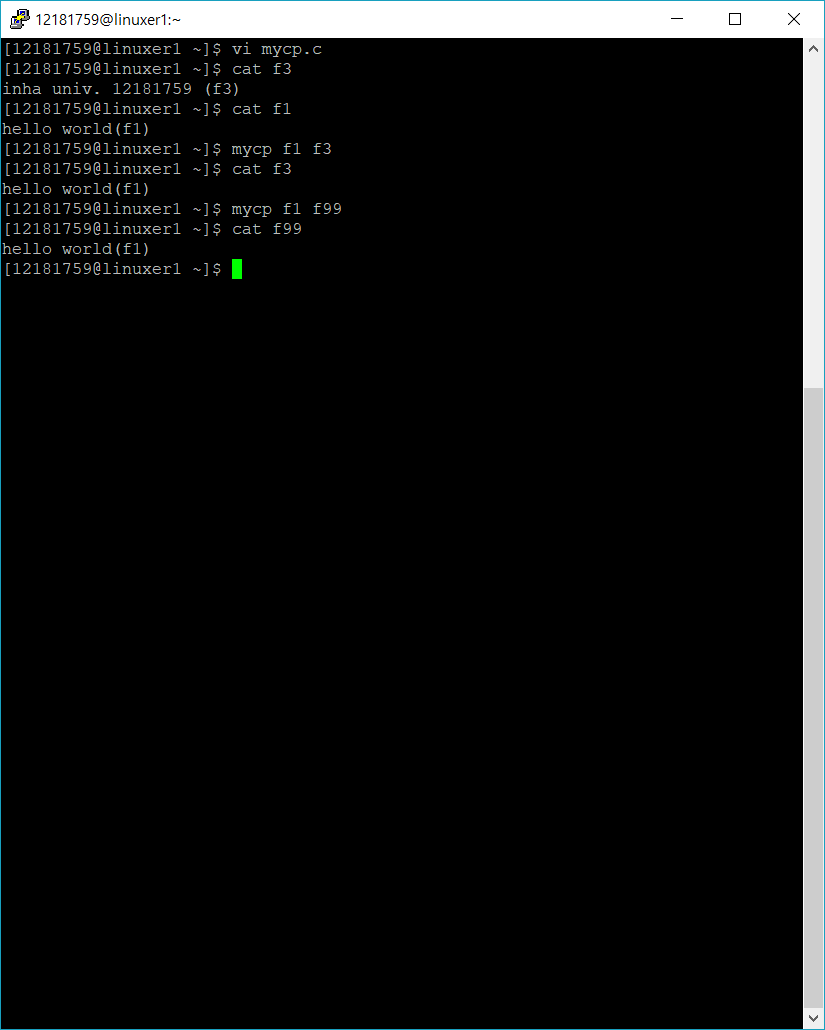


모든 기능이 정상적으로 구현됨을 확인 하였다.

4)

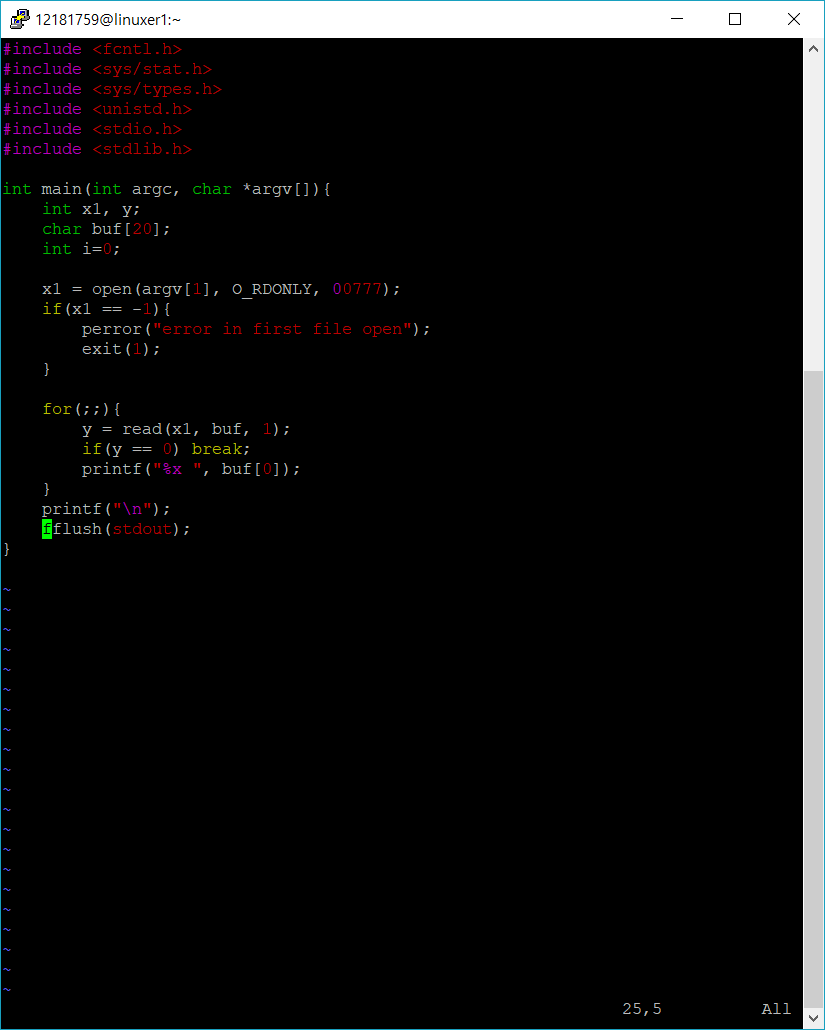


문제에서 요구한 프로그램을 작성하였다. 코드를 설명하면 먼저 argument로 2개를 받는데 첫번째로 복사될 내용을 가진 파일명, 두번째로 그 내용으로 작성될 파일명 이다. cp 는 만약 없는 파일이라면 새로 만들어지고 이미 존재하는 파일이라면 그 내용이 덮어 써지므로 O\_CREAT|O\_TRUNC 로 한다. 다음으로 무한 루프를 돌면서 x1 파일의 내용을 20byte씩 끝까지 읽고 이를 x2에 write한다. 만약 모두 읽어 y가 0이되면, 프로그램은 종료된다.

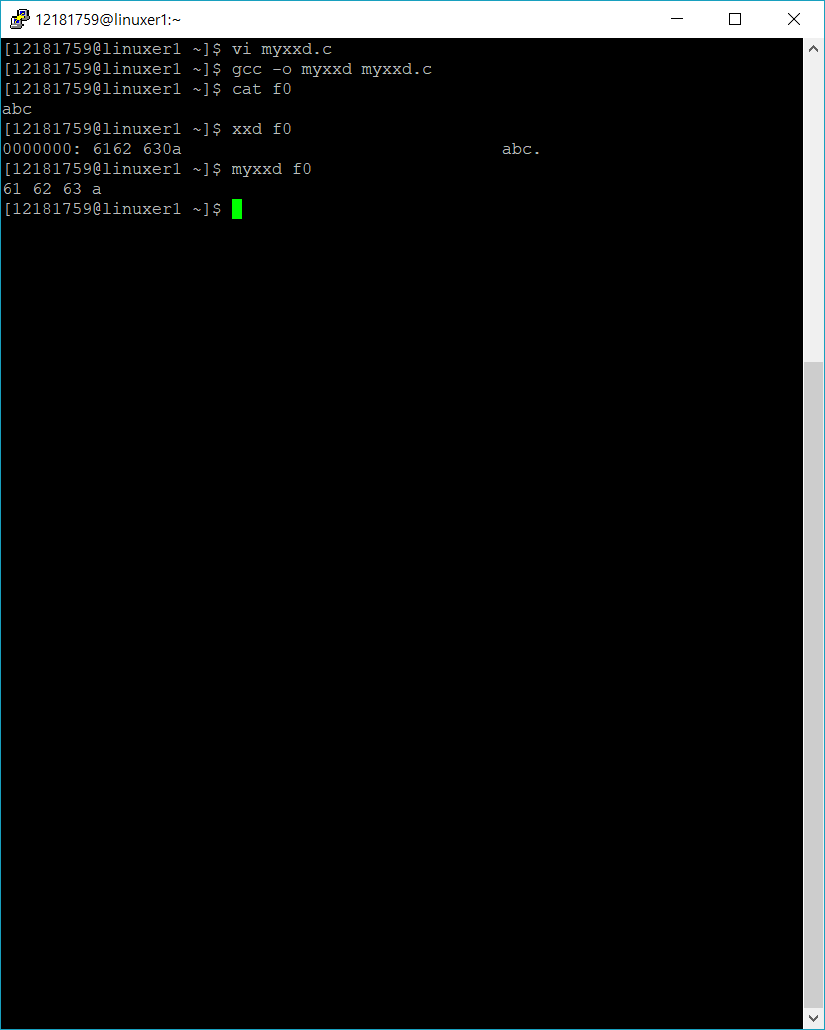


모든 기능이 cp와 동일하게 정상적으로 동작함을 확인 하였다.

5)

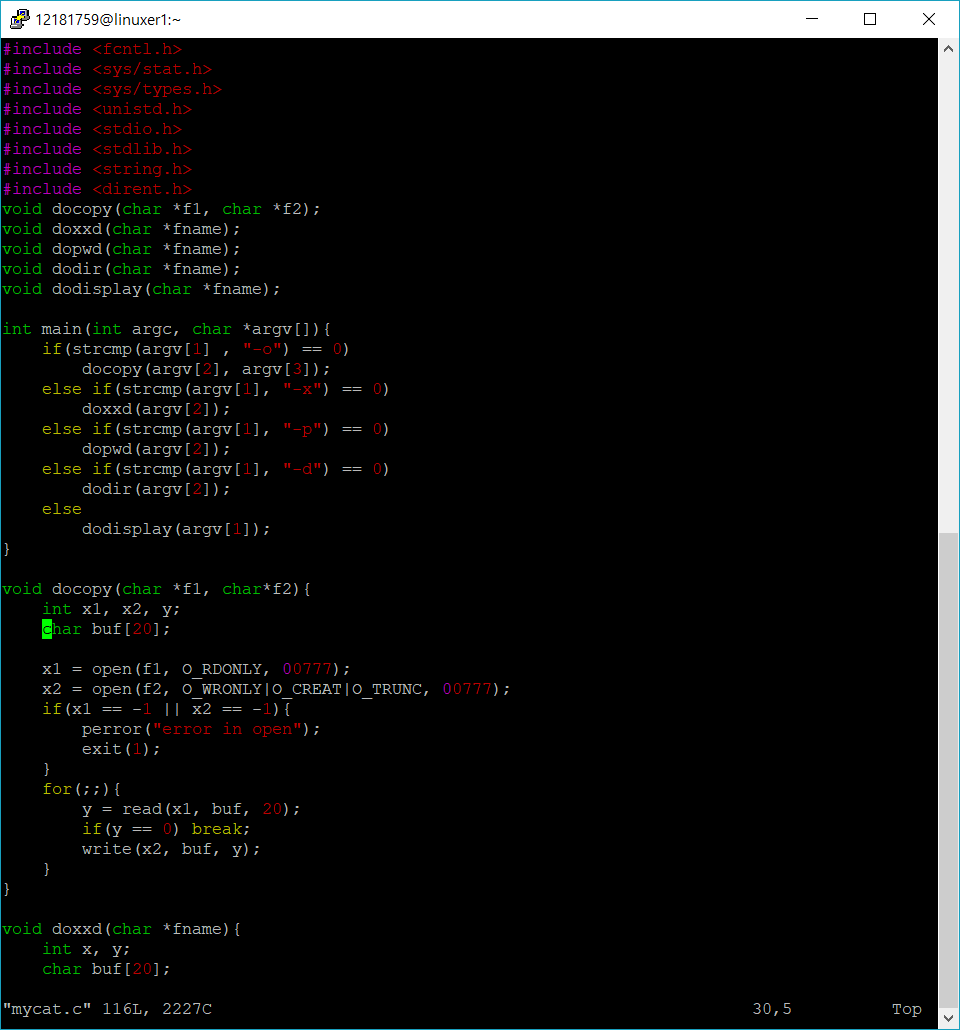


문제에서 요구한대로 프로그램을 작성했다. 기능은 x1 파일에서 1byte씩 읽어와서 이를 16진수 형태로 출력하는 것이다. xxd 와 유사하게 동작한다.

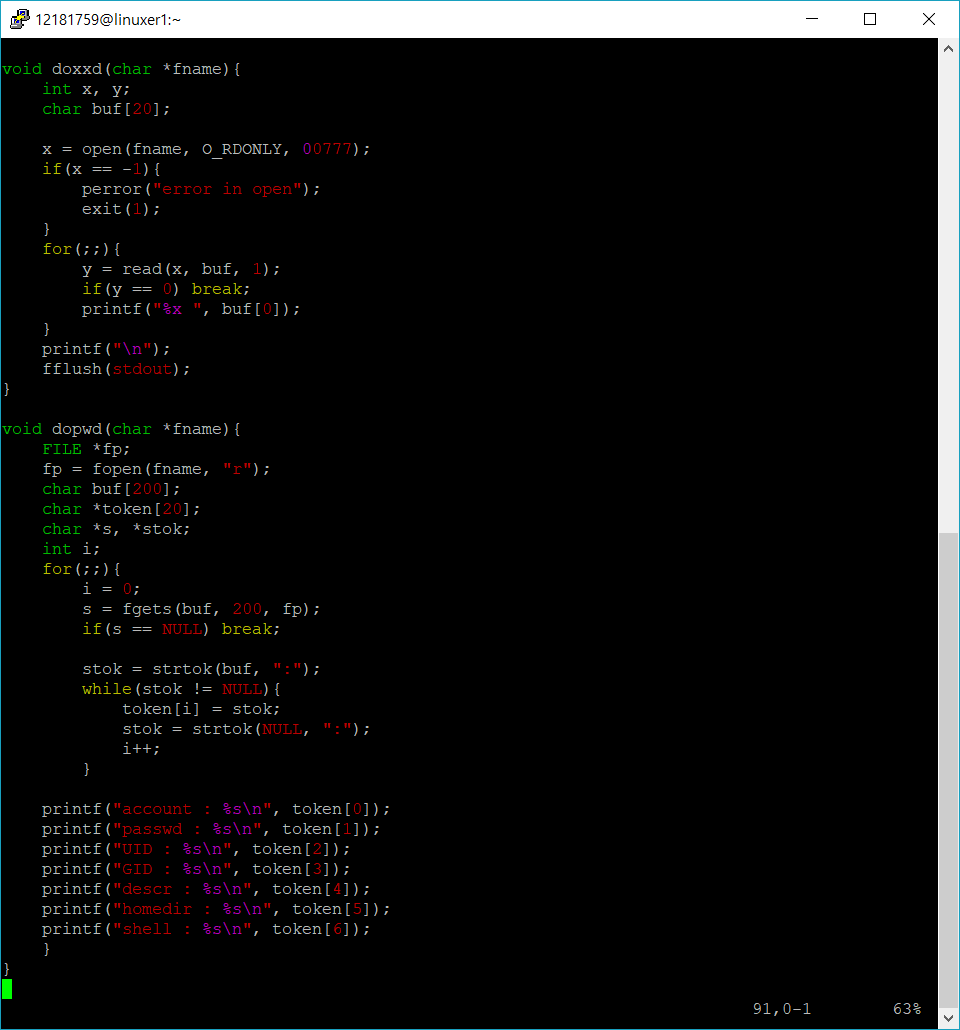


모든 기능이 정상적으로 동작함을 확인했다.

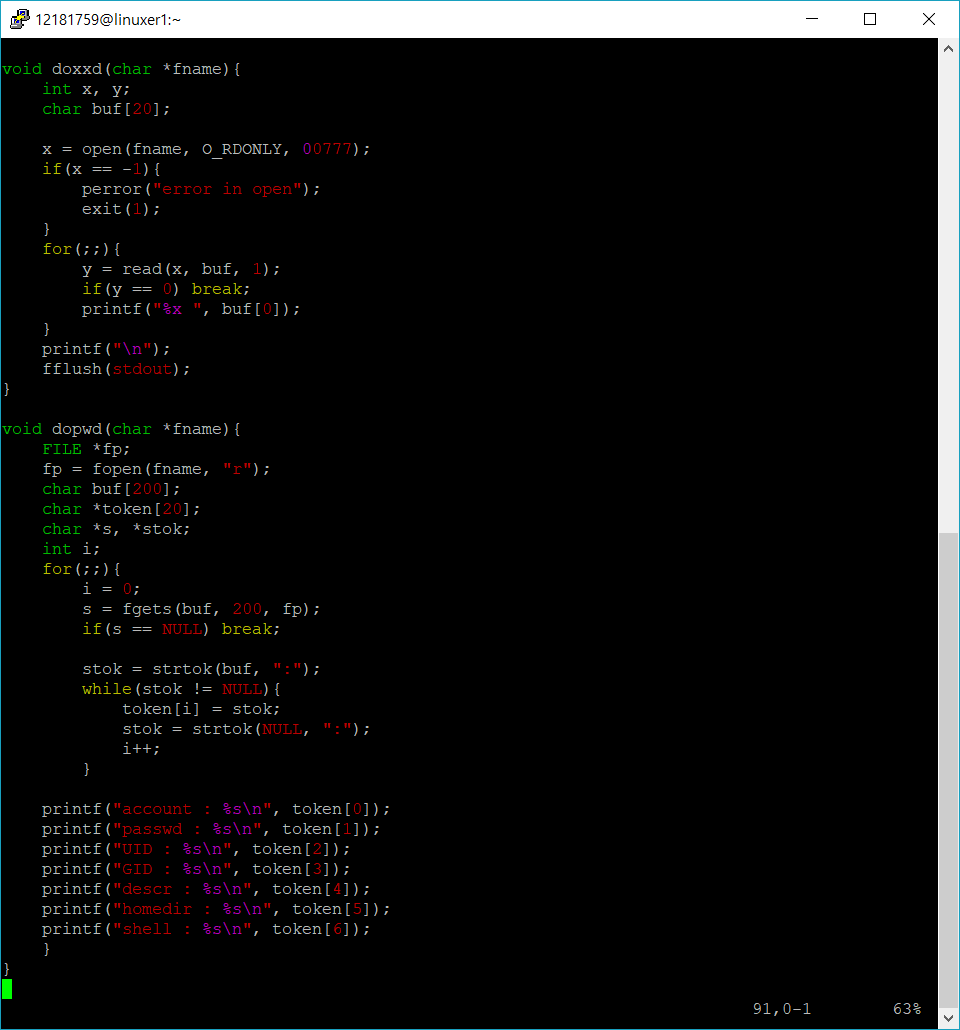
6)

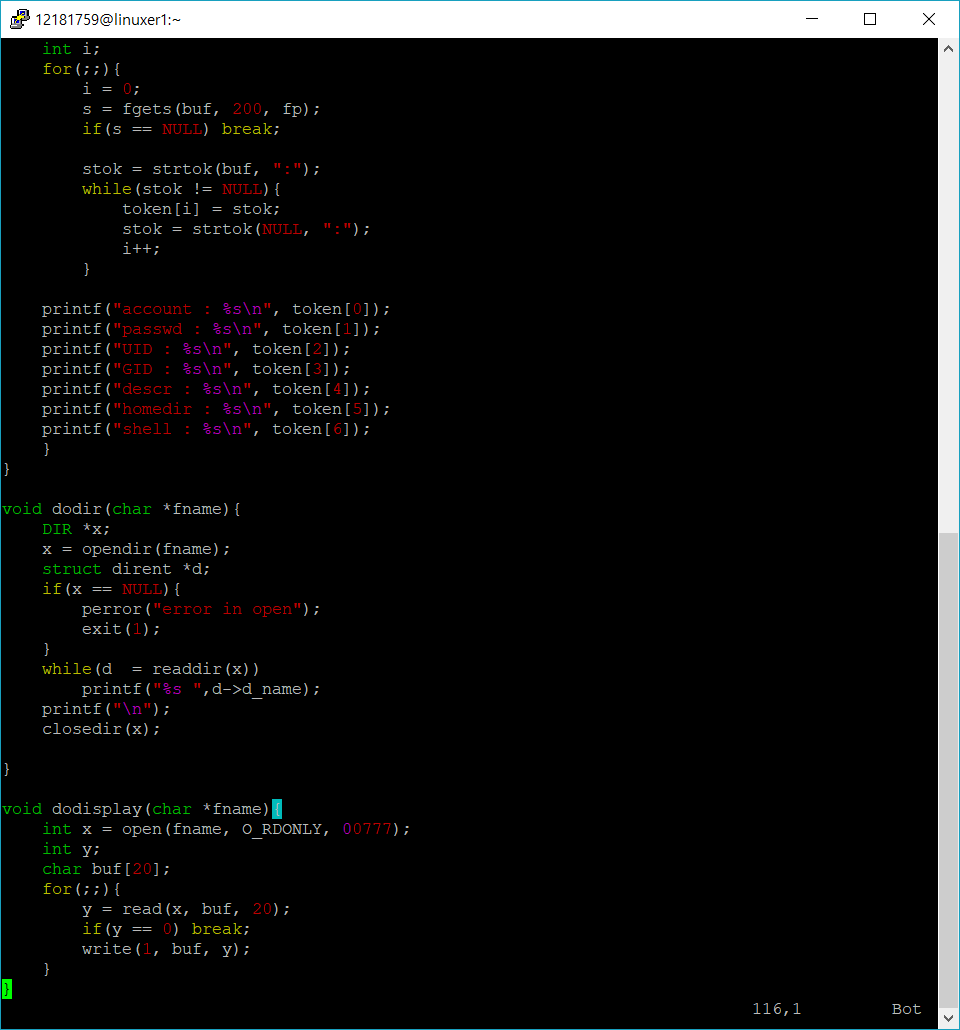


문제에서 구현하라는 대로 프로그램을 작성했다. main 안에서는 argv[1]에 들어온 argument가 만약 옵션이라면, 그 기능에 맞는 함수를 시행하고 만약 옵션이 아니라면, 기본 cat 기능을 수행하는 dodisplay 함수를 시행한다.



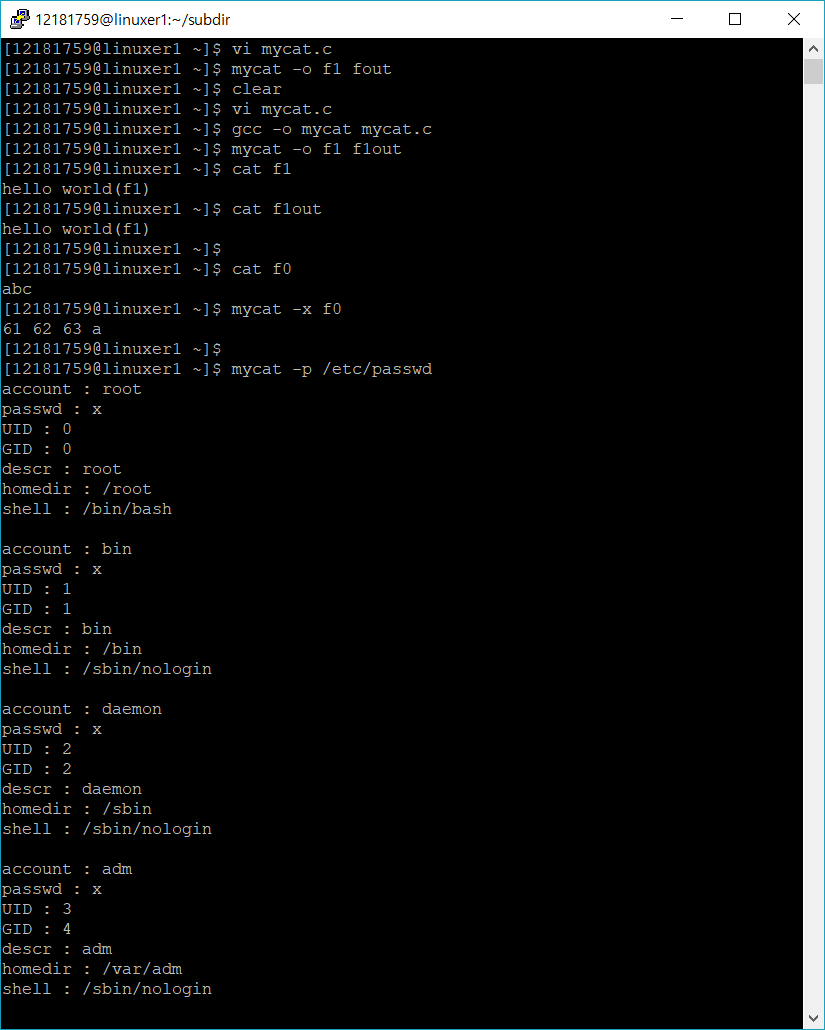
각각 -o, -x 옵션에 따르는 docopy 함수와 doxxd 함수는 과제 위에서 작성한 프로그램을 가져왔다. 각각 cp와 xxd 와 동일하거나 유사한 기능을 수행한다.



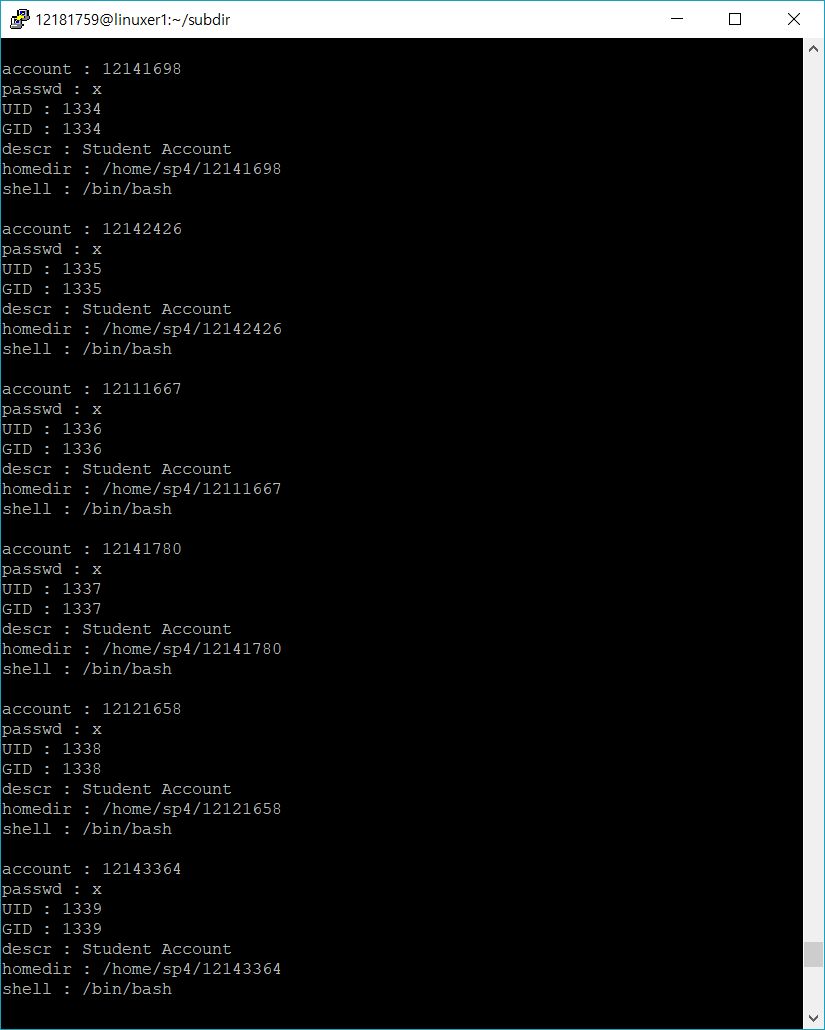
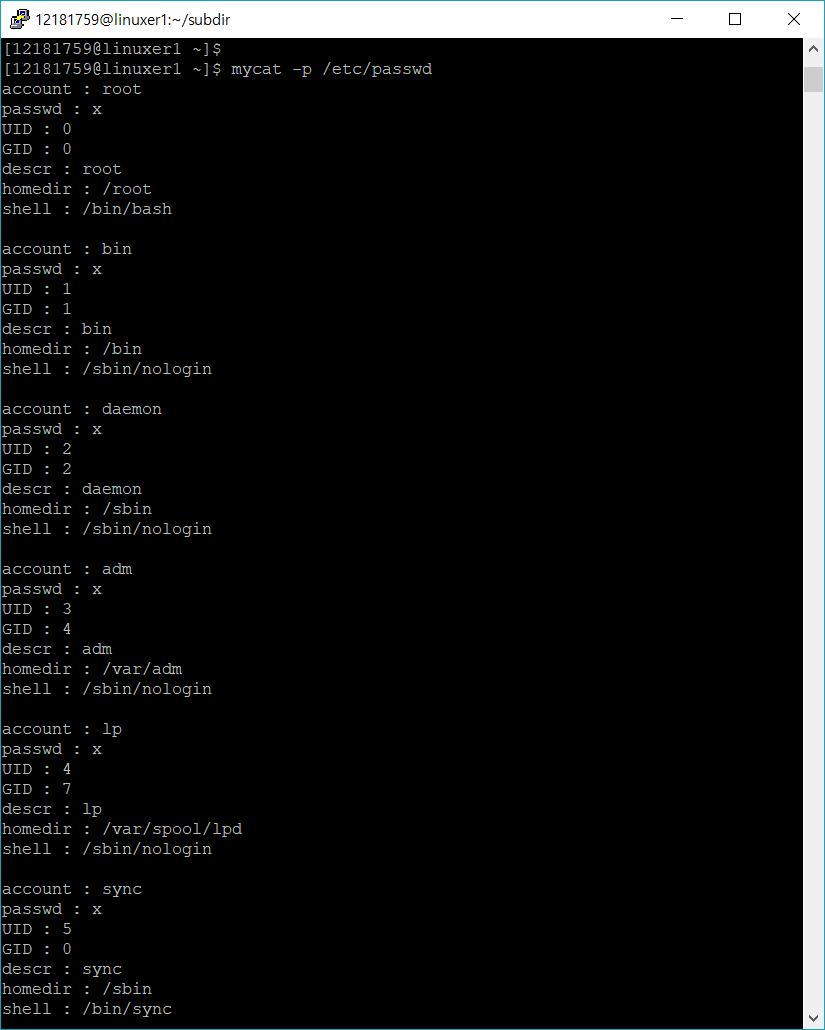


-p 옵션인 dopwd 함수는 passwd에 들어있는 정보를 끊어서 출력하도록 하는 함수이다. 이를 위해 file 포인터를 이용한다. 먼저 파일을 열고 한 string을 읽어올 포인터 변수 s에 buf로 fget한다. 읽어온 string 한 줄을 strtok 함수를 이용하여 이를 “ : “를 기준으로 끊어 token에 저장한다. 그 방법은 전에 string에서 strtok 함수 연습에 사용한 방법과 동일하게 구현했다. token을 넘어가며 읽으면서 그 전을 NULL로 만드는 것이다. token 들은 각각 account, passwd, UID, GID, descry, student account home dir, shell 을 가르킨다. 그리고 이들을 각각 출력하고, 다시 그 다음줄을 읽어온다. 읽어오는 줄 s 가 NULL일 때 까지 시행한다.

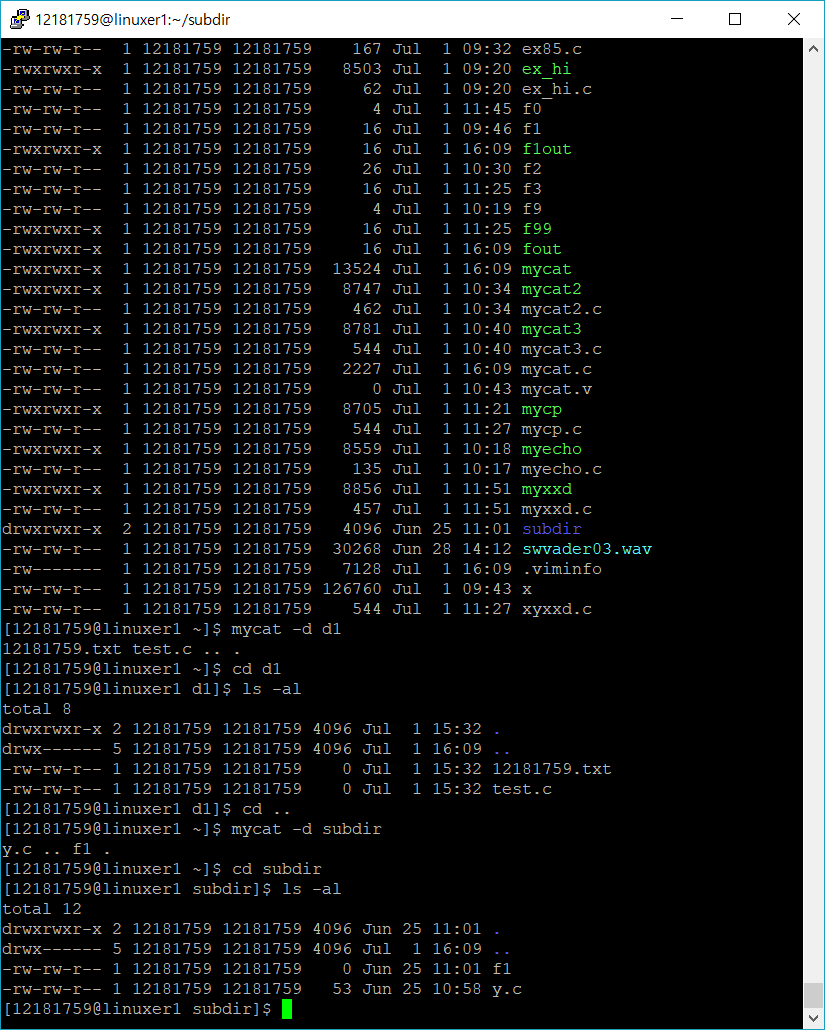
다음은 -d 옵션인 dodir 함수이다. dodir 함수를 사용하기 위해 opendir, readdir 함수의 사용법을 참고했다. dir을 쓰기위해서 우선 dirent struct를 include 해주어야 한다. 그다음에 dir x와 strict dirent d를 선언한다. d가 파일 이름을 next 하면서 읽는 동안 이들의 이름을 모두 출력한다. readdir 함수는 dir 안에 있는 모든 파일과 디렉토리의 이름을 넘어가면서 읽는다. 모두 읽었으면 closedir 한다. dodisplay 함수는 기본 cat 기능을 수행한다. file 내부의 모든 문장을 화면에 출력한다.



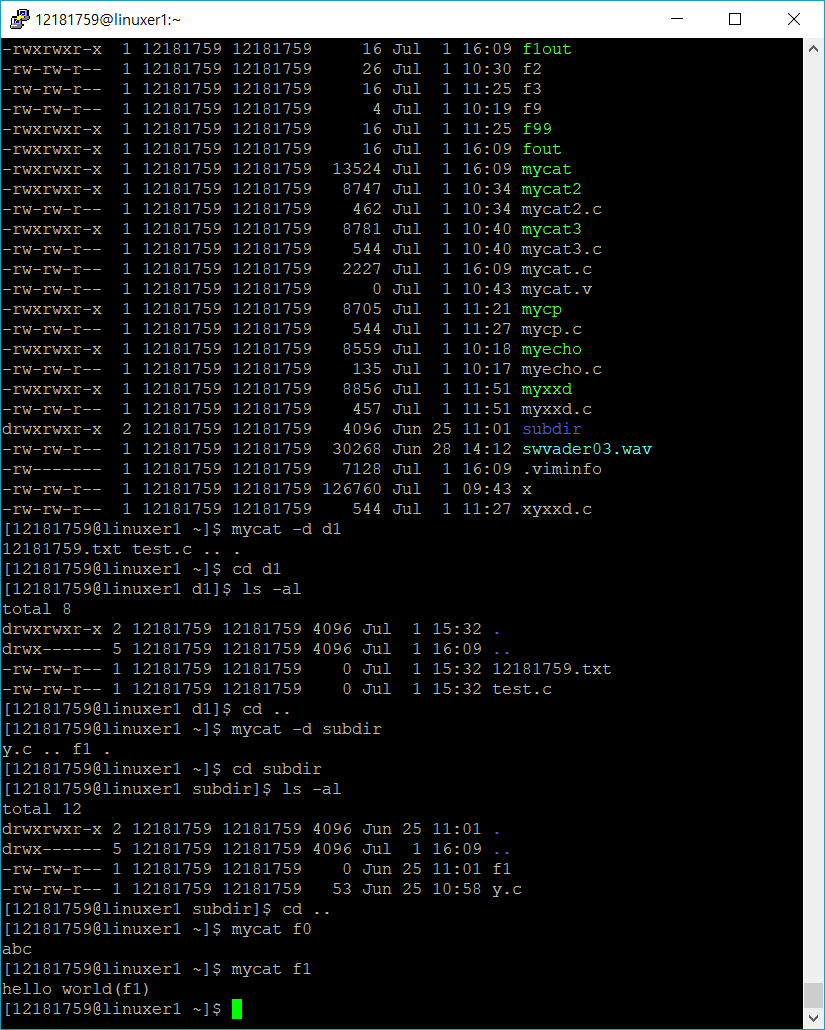
-o, -x 옵션을 시행해본 모습이다. 모든 기능이 정상적으로 동작함을 확인 할 수 있었다.



-p 옵션을 시행해본 모습이다. 모든 기능이 정상적으로 동작하고, 모두 올바로 잘 나누어져 출력되었음을 확인 하였다.



-d 옵션을 시행해본 모습이다. 파일 이름은 모두 잘 출력된다. ,이름과 함께 ‘ .. ‘ ‘ . ‘ 이 출력되는 이유는 ls -al 을 해보면 알 수 있는데, 숨겨져 있는 디렉토리 파일 명이다. readdir은 이들도 모두 읽어낸다.



마지막으로, 아무것도 옵션도 주지 않았을 때, 정상적으로 파일명만 출력됨을 확인할 수 있었다.