LEC

file을 어떻게 다루는지에 대해서

file번호 - 3번부터 순서대로 부여됩니다.

0: 키보드, 1.2: 화면

open()으로 받은 파일 번호를 통해

read()에 넣음으로써 파일data 읽을 수 있습니다.

read = 읽은 만큼 rtn됩니다. 즉, 0이되면 다 읽은 것이고, 읽기 시작하는 위치는 맨뒤의 byte 값에 따라 유동적입니다.

보통 loop를 사용하여 read를 다룹니다.

writ에서. .. “hello” – “hello”의 주소로 읽히게 됨

0 = 키보드

1 = 화면입니다 . write 에서 1로 같이쓰면 그냥 화면에 쓰는 것

6-0, 6-1 등에 있는 프로그램 만들 때 파일 이름을 f1.c 로 하고 싫행파일 이름을 f1 등으로 하면 안됩니다. 6-0 등에 있는 프로그램은 f1 이라는 파일을 오픈해서 그 안의 내용을 읽거나 쓰는 프로그램인데 파일 이름 충돌이 일어납니다. 프로그램 이름은 ex1.c 등으로 하기 바랍니다.gcc -o f1 f1.c하면 f1.c 에 있는 코드를 컴파일해서 f1 이라는 실행파일을 만들라는 의미입니다.따라서 f1 파일에 I have a dreadm... 등을 저장한 후 위 명령을 하면 기존의 f1 파일은 지워지고 f1.c 에 있는 c 코드가 컴파일되어 그 기계어가 f1을 덮어쓰게 됩니다.

kchang@inha.ac.kr에서 모두에게: 10:28 AM

프로그램 이름으로 ex.c 도 피하기 바랍니다.

ex 는 리눅스 명령어 중의 하나이므로 프로그램 이름으로 사용할 경우 시스템과 충돌을 일으킵니다.

HW

\*\* About Indentation \*\*

When you are writing a program, please do indentation. **No point for code with no indentation!**

void main(){

int x;

if (x < 100){

int y;

y=x;

}

===> (After indentation)

void main(){

int x;

if (x < 100){

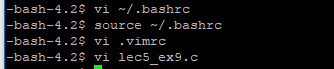
int y;

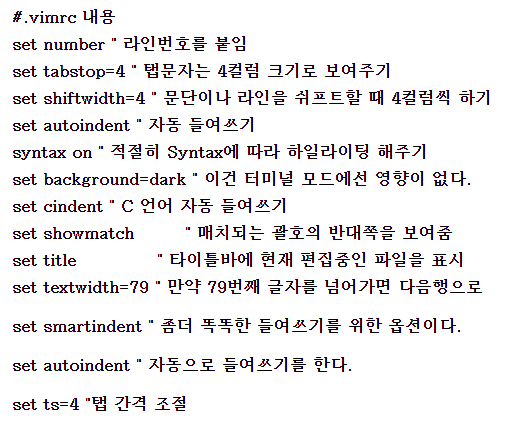
y=x;

}

}

+





위와같이, .bashhrc 실행후, .virmc파일을 만들어 원하는 옵션대로 내용 수정후, 실행하면

들여쓰기가 자동으로 이루어 질 것입니다.

1) Make a file, "f1", and fill it with more than 20 bytes.

$vi f1

I have a dream

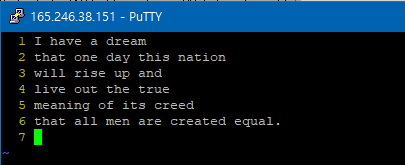
that one day this nation

will rise up and

live out the true

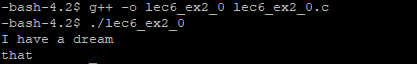
meaning of its creed

that all men are created equal.

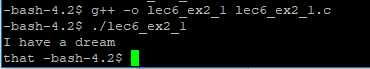


2) Try the code in 6-0), 6-1), 6-2), 6-3).

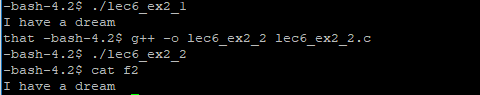
- 6-0) Reading 20 bytes from file "f1" and display them in the terminal using printf.



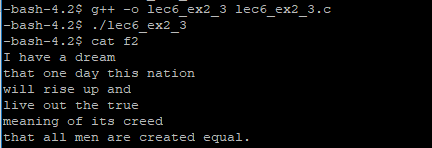
- 6-1) Reading 20 bytes from file "f1" and display them in the terminal using write.



- 6-2) Copying 20 bytes from file "f1" to "f2". Check "f2" with "cat f2".



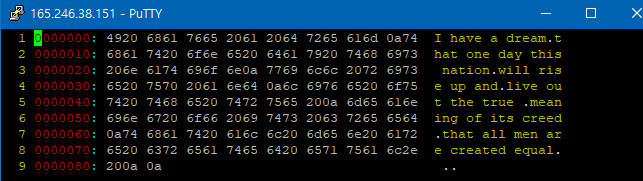
- 6-3) Same as above, but copy all data from f1 to f2. Check "f2" with "cat f2".



3) Find the byte size of f2 with “ls –l f2”. Use xxd to find out the actual data stored in f2.







4) Write a program "hw4.c" that opens f2 and shows each byte of it in hexadecimal number, decimal number, and character. Use printf("%x %d %c\n", ...............) to display a number in various format.

int x, y; char buf[20];

x=open("f2", O\_RDONLY, 00777); // open f2 for reading

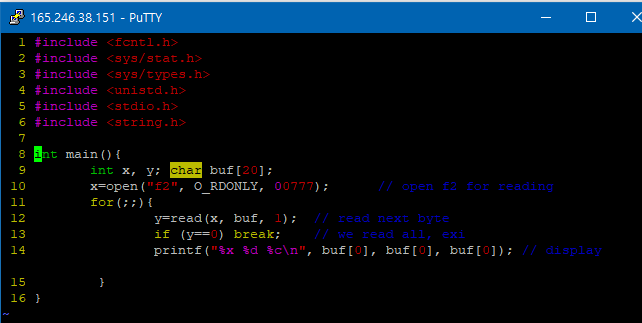
for(;;){

y=read(x, buf, 1); // read next byte

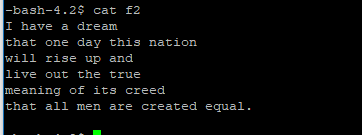
if (y==0) break; // we read all, exit the loop

printf("%x %d %c\n", buf[0], buf[0], buf[0]); // display

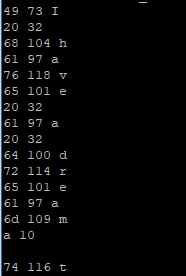
}



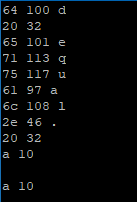
코드는 동일하게 작성하여 주었습니다.



f2의 내용은 위와 같습니다.



…



각 16진수값, -x // 10진수 아스키값 -d // 원본 알파벳값 – c 로

잘 출력되고 있는 것을 확인할 수 있었습니다.

5) Compile hw4.c with –g option and run gdb to execute each instruction one by one. Use “p” or “x” to check the value of a variable.

$ gcc -g -o hw4 hw4.c

$ gdb hw4

gdb) b main -- stop at main

gdb) r -- run

............

9 x=open("f2", O\_RDONLY, 00777); -- next line to execute

gdb) list -- show code list

gdb) n -- execute current line

11 y=read(x, buf, 1); -- line 9 has been executed. next is line 11

gdb) p x -- show x

$1 = 7 -- f2 is now file number 7

gdb) n

..........

gdb) p y

$2 = 1 -- we have read 1 byte

gdb) p buf[0]

$4 = 73 'I' -- assume we have 'I' in buf[0]

gdb) x/4xb buf -- show 4 bytes at buf in hexadecimal num

0x7fffffffe470: 0x49 0x06 0x40 0x00 -- we have 0x49=73='I' in buf[0]

gdb) n

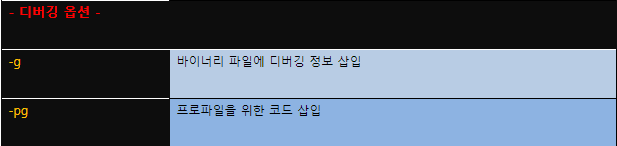
............ –- repeat a few times

gdb) list -- show rest of code

gdb) b 15 -- break point at line 15 (after loop)

gdb) c -- continue to that break point

gdb) q -- stop gdb



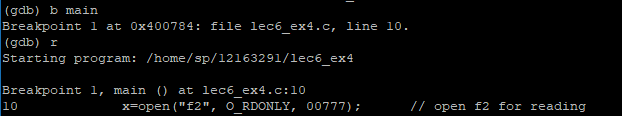
-g 는 디버깅 옵션값 입니다.

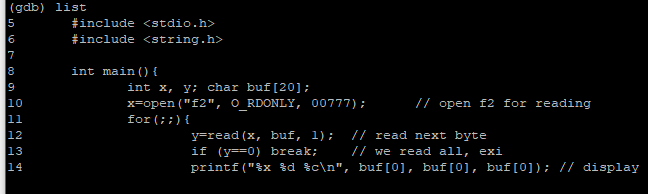


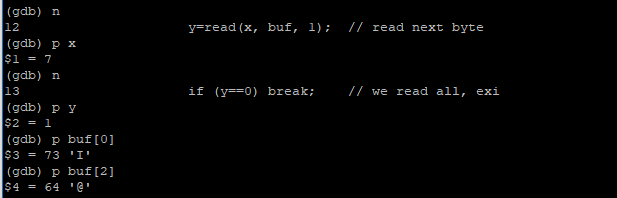
-g 옵션값을 추가해 줍니다.

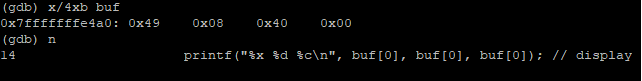


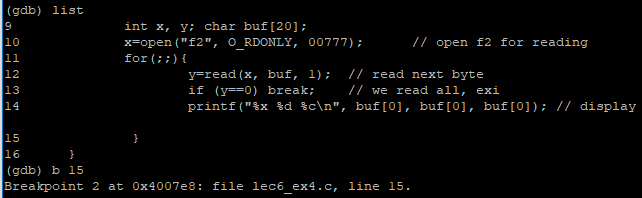
gdb 커맨드를 통해 디버깅을 시작해 줍니다.

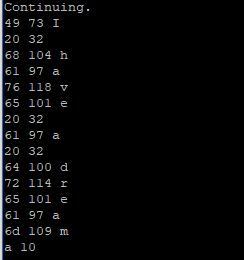


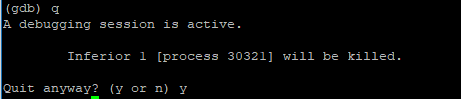








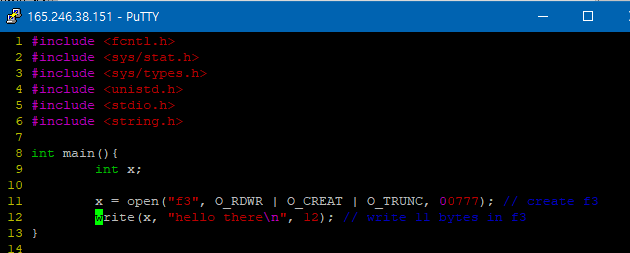




6) Write a program that creates a file and writes “hello there”in it. Use open() and write(). Confirm the result with "cat".

x = open("f3", O\_RDWR | O\_CREAT | O\_TRUNC, 00777); // create f3

write(x, "hello there", 11); // write 11 bytes in f3



줄 바꿈을 추가해 바이트를 11 에서 12로 하나 늘려주었습니다.



7) Write a program that makes a copy for file "hw4.c" into another file "cphw4.c". Use open(), read(),and write(). Confirm that they are same with "cat" and "ls -l".

x1 = open hw4.c as O\_RDONLY

x2 = open cphw4.c as O\_RDWR | O\_CREAT | O\_TRUNC

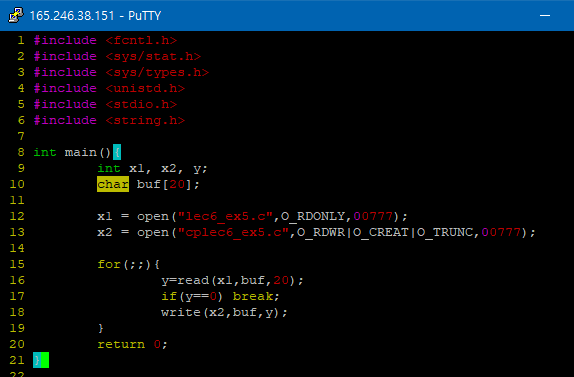
for(;;){

y = read max 20 bytes from x1 into buf

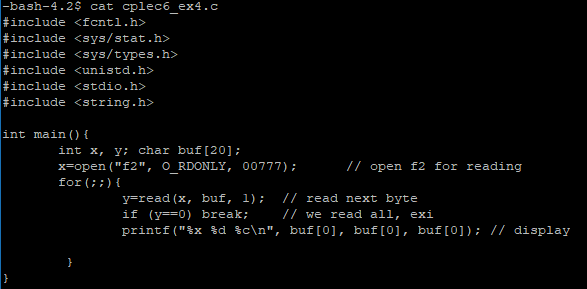
if y is 0, break

write y bytes of buf into x2

}



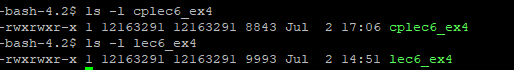






cplec6\_ex4.c 파일이

제대로 생성된 것을 확인할 수 있었습니다.



8) Write a program that makes a copy for file "hw4" (the executable file for "hw4.c) into another file cphw4. Confirm that they are same with "xxd" and "ls -l".

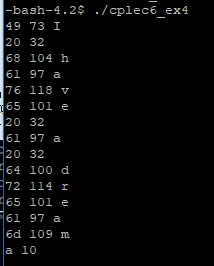
Execute cphw4 to see if it runs ok.



위왁 같이 컴파일 파일 이름을 (.c 빠진 이름)

을 넣어 cp를 진행 해 주었습니다.





성공적으로 돌아가고 있습니다.

9) Repeat 7). But get the name of the files from the user. Confirm that the result of copy with "cat" and "ls -l".

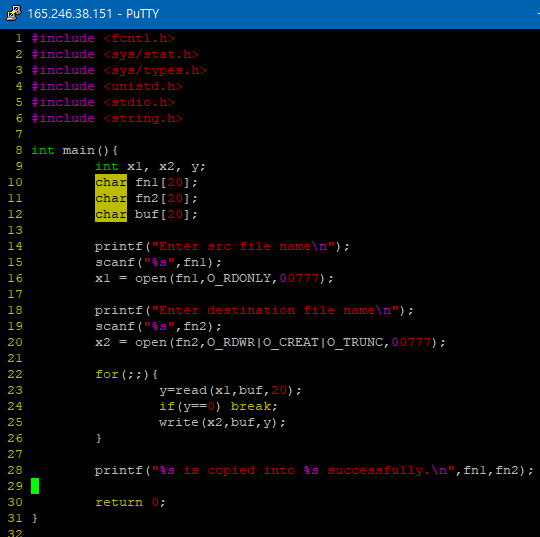
Enter src file name

hw4.c

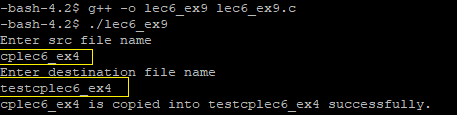
Enter dest file name

newhw4.c

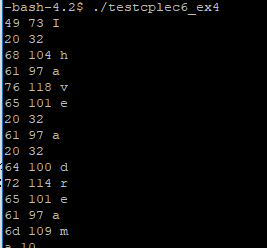
hw4.c is copied into newhw4.c successfully.



코드 내용은위와 같습니다.



앞서 사용하였던 cplec6\_ex4 의 내용을 그대로 카피하게 하였습니다.



같은 결과가 cp된 것을 확인할 수 있었습니다.

10) Write "mycat" that displays the contents of a user-input file in the terminal in characters. Give a text file and a non-text file to mycat and explain the difference.

$./mycat

Enter file name

f1

The content of f1 is :

I have a dream

that one day this nation

will rise up and

live out the true

meaning of its creed

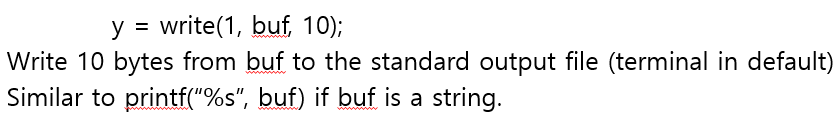
that all men are created equal.

$./mycat

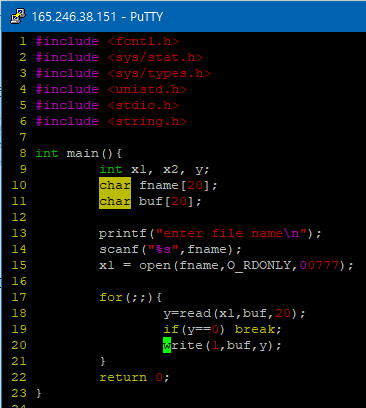
Enter file name

hw4

.............

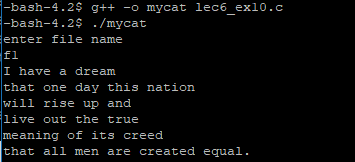


강의노트 내용 참고하였습니다.



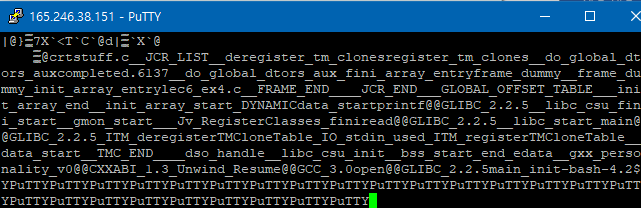
코드는 위와같이 fname으로 파일 이름을 받고,

write + 1 을하여, print를 바로 해줄 수 있도록 하였습니다.



잘 출력되었습니다.

mycat + lec6\_ex4 를 해보았습니다.



결과를 보니, 저번 과제에서 하였던, 어셈블리어를 읽을수 없는 경우에 나타나는 에러와 동일하였습니다.

따라서 이를 xxd로 or 아스키코드로 읽게되면 읽을 수 있게 될 것이라 생각됩니다.

11) Write "myxxd" that displays the contents of a user-input file in the terminal in hexadecimal numbers. Give a text file and a non-text file to myxxd and explain the difference. You need to use printf(“%x “, buf[i]) to display a byte in a hexadecimal number. Also declare the buffer as an array of unsigned char. Compare the result from the result of xxd.

$./myxxd

Enter file name

f1

The content of f1 is :

49 20 68 61 ..........

$ xxd f1

..................

$./myxxd

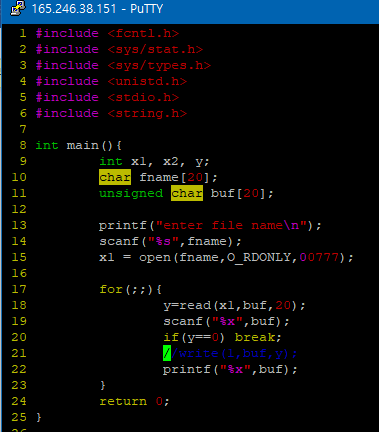
Enter file name

hw4

.............

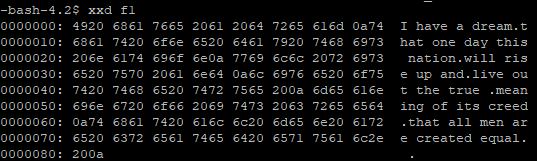
$ xxd hw4

...............

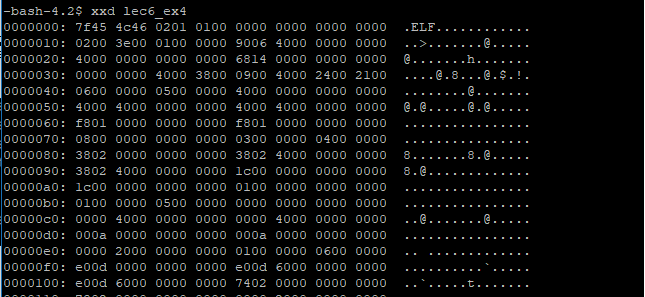


..!

xxd + 파일이름



xxd+기계어 파일



12) Run following code and display f8 with cat and xxd respectively. Explain the results.

x=open("f8", O\_CREAT|O\_RDWR|O\_TRUNC, 00777);

write(x, "ab", 2);

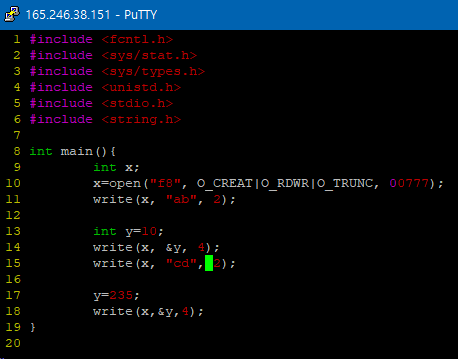
int y=10;

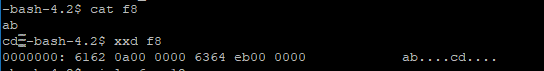
write(x, &y, 4);

write(x, "cd", 2);

y=235;

write(x, &y, 4);





line 10: f8이라는 파일을 만들어/열어

line 11: 그 파일안에 “ab”라는 내용을 2바이트 크기로 넣어줍니다.

line 14: int는 4바이트 정수이므로 4바이트 크기로 y=10 정보가 담긴 위치의 주소값을 넣어줍니다.

line 15: 그 다음에 “cd”라는 내용을 2바이트 크기로 넣어줍니다.

line 18: int는 4바이트 정수이므로 4바이트 크기로 [ y=235 ] 정보가 담긴 위치의 주소값을 넣어줍니다.



그럼 f8 파일의 내용은 강의 노트와 같이 이렇게 순서대로 채워지게 됩니다.

cat f8로 읽을 수 있는 내용은 char 이므로 ab, cd는 제대로 보이게 되고, 주소값은 안보입니다.

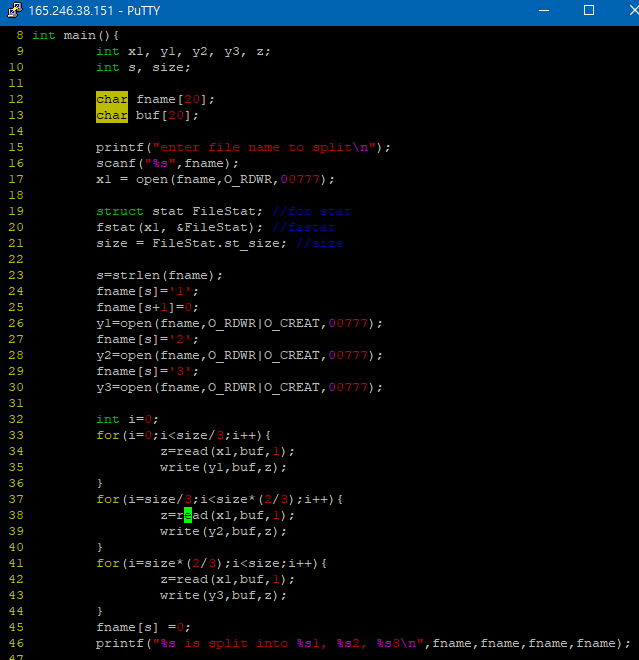
xxd는 이진 정보도 읽을 수 있으므로 결과가 잘 보이게 됩니다.

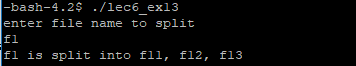
13) Write a program that divides a given file into three small files of roughly equal size. Use fstat() to find out the size of a file.

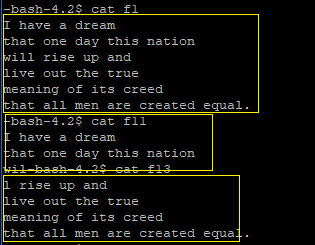
Enter file name to split

f9

f9 is split into f91, f92, and f93.







잘 나눠 졌습니다.

14) What is wrong with following program?

char temp0[20], \*temp1[10], \*temp2[10];

printf(“enter src file name\n”);

gets(temp0);

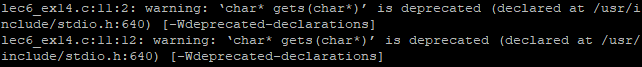
temp1[0]=temp0;

printf(“enter dest file name\n”);

gets(temp0);

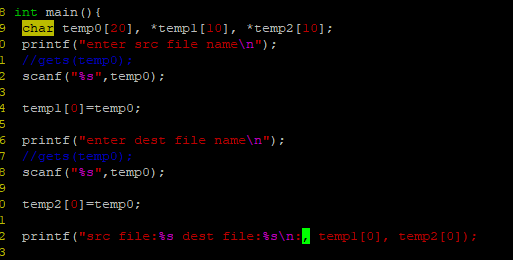
temp2[0]=temp0;

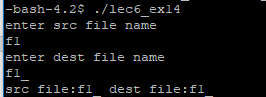
printf(“src file:%s dest file:%s\n”, temp1[0], temp2[0]);



get의 사용에서 경고가 발생하여,

이를 scanf 함수로 대체하여주었습니다.





그후 결과가 잘 출력되는 것을 확인할 수 있었습니다.

//처음에는 마지막에 포인터 값을 string으로 출력하려고 하는과정에서 생기는 오류인줄 착각하여 temp1[0] 앞에 &를 붙여주었었습니다.

ㄴ 상관없습니다.

15) What is wrong with following program. Find the problem with GDB ( gdb + -g ) and fix it.

int x, x1, y;

x=open(“f1”, O\_RDONLY, 00777);

x1=open(“f2”, O\_WRONLY|O\_CREAT|O\_TRUNC,00777);

char buf[512];

int cnt=0;

for(;;){

y=read(x,buf,1);

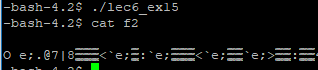
if (y==0) break;

cnt++;

}

write(x1, buf, cnt);

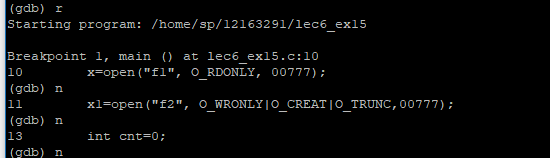
컴파일 이후,



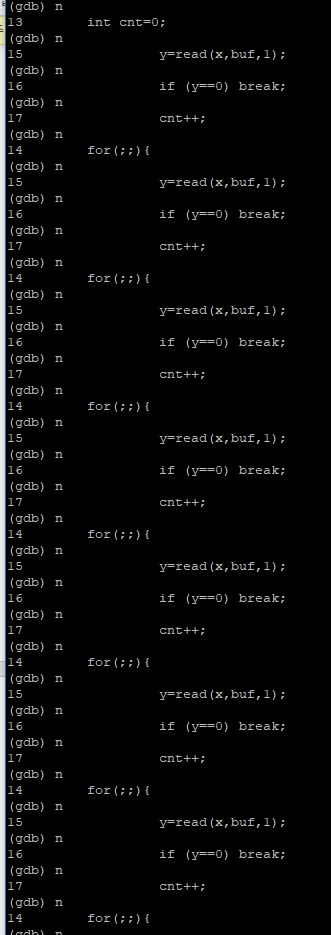
결과를 보니 제대로 값이 옮겨가지 않았습니다.



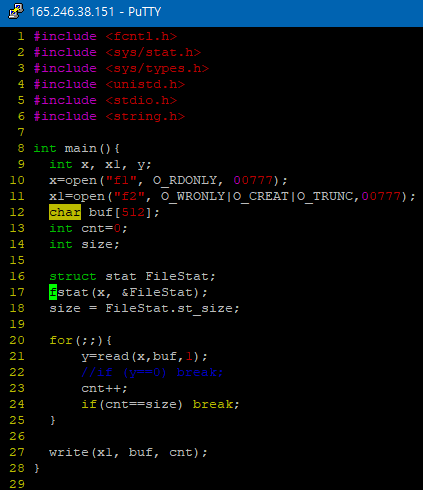
디버깅을 시작합니다.



디버깅을 하다보니,



무한루프 지점이 생기는 것을 확인하였습니다. ( y==0인 지점이 만들어지지 않는 것입니다.



저는 파일의 길이를 알아내서, 그 길이만큼의 write를 진행할 수 있도록,

또한 cnt가 높아지다가, 읽어오는 파일의 길이가 되었을 때, for문을 빠져나갈 수 있도록 하였습니다.

하지만 다른 방법으로 접근해야하는 것 같습니다.