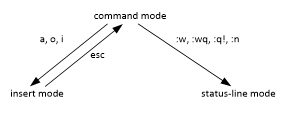
수업

vi 에 관한 파일 생성 내용



: ? 이면 status-line 모드입니다.

cusor moving: j(down), k(up), h(left), l(right)

deletion: x(delete one character), dd(delete a line)

copy and paste:

3yy : copy 3 lines starting from the current line

p   : paste them after the current line

recover

u   : recover what you have just deleted

:q! : quit without saving

:w : write

:wq : save and exit

/pat : search the pattern in "pat"

:set number : display line numbers

:40 : go to line 40

vi 에서 각 커맨드를 쓰는 방법

리눅스 환경에서 c파일 생성하기

gcc = c프로그램을 컴파일 하는 명령어

ex1.c - c프로그램 텍스트

ex1 – 위 c코드가 어셈블리 – 기계어로 변환된 내용이 들어있음

g++ = c++프로그램을 컴파일 하는 명령어

xxd = 아스키 코드를 출력 – 컴파일된 어셈블리어 ex1 내용 보여줄수는’ 있음

cat = 내용 자체 그대로를 출력 = chr 형 – ex1 내용 볼수 없음 – 아그키 값이아니면 찍을수 없음

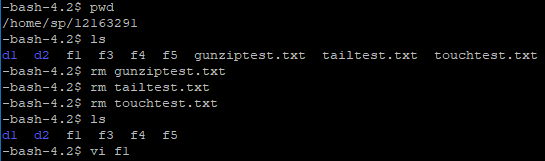
 : vi로 ex1의 내용확인하는 방법

HW

1) Make a file with vi.

vi f1

(when you open a file with vi, sometimes you see Open, Edit, ... etc. at the bottom of the screen, in that case just type "e" in order to start editing)



pwd 현재 로그인 위치임을 확인한 후,

ls로 자식 파일이 무엇이 있나 확인하여 보았습니다.

쓸모없어보이는 .txt파일들을 삭제 한후,

vi f1으로 f1 파일을 열어보았습니다.





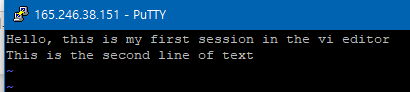
그 상태에서 e를 누르니 아무일도 일어나지 않았습니다.

i를 누르니 그제서야 수정이 가능하였습니다.

2) Start insertion with ‘i’ key and type following.

**Hello, this is my first session in the vi editor.**

**This is the second line of text.**





i로 f1파일의 내용을 위와 같이 수정하였습니다.

:wq를 통해 이 내용을 저장하였습니다.

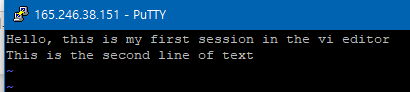


그후 cat f1을 통해 내용을확인하여 보니, 잘 생성되어있는 것을 확인할 수 있었습니다.

3) Return to command mode with ‘esc’. Save and exit with ‘:wq’

esc

:wq





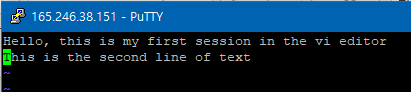
이는 앞서 3)번의 내용에서 확인한 내용입니다.

:wq 를 통해 파일의 내용을 저장하면서, quit 할 수 있습니다.

:q 는 저장하지 않고 quit합니다.

4) Reopen the file

vi f1

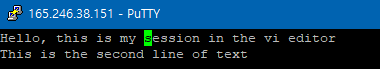


다시 열어보니, 내용이 그대로 잘 저장되어있는 것을 확인할 수 있었습니다.

5) Use ‘j’, ‘k’, ‘h’, ‘l’ to move the cursor around. Move the cursor to the word “first”. Use ‘x’ key to delete the word “first”. The result should be as follows.

**Hello, this is my session in the vi editor.**

**This is the second line of text.**



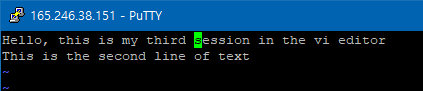
x로의 삭제는 왼쪽에서 > 뒤로 이루어 지므로

커서를 first 단어 앞에 두고 x를 누르면 first를 지울 수 있게됩니다

6) Insert “third” as follows.

**Hello, this is my third session in the vi editor.**

**This is the second line of text.**



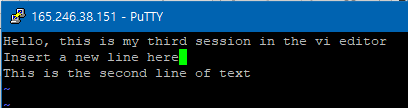
i를 누르고 third를 입력해 주었습니다.

7) Add a new line as follows. Use ‘o’ key.

**Hello, this is my third session in the vi editor.**

**Insert a new line here.**

**This is the second line of text.**



**o**를 입력하여 보니, 한줄 공백이 바로 아랫 줄에 생기며 커서가 그줄의 맨 앞에 있는 것을 확인할 수 있었습니다.

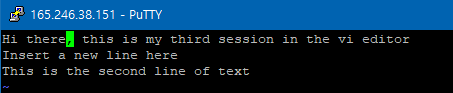
그 자리에 새로운 내용을 입력해주었습니다.

8) Change the beginning as follows. Use ‘x’ and ‘i’ key.

**Hi there, this is my third session in the vi editor.**

**Insert a new line here.**

**This is the second line of text.**



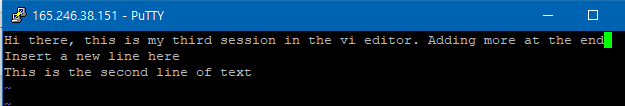
내용을 수정하였습니다

9) Add more at the end. Use ‘a’ key.

**Hi there, this is my third session in the vi editor. Adding more at the end.**

**Insert a new line here.**

**This is the second line of text.**



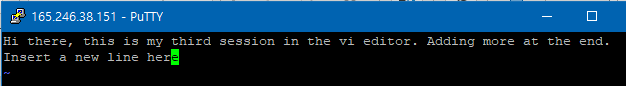
a 로 첫줄 맨 뒤의 내용을 입력해주었습니다.

이는 i와 비슷한 역할을 하는 커맨드 같습니다.

10) Delete the last line. Use ‘dd’.

**Hi there, this is my third session in the vi editor. Adding more at the end.**

**Insert a new line here.**



esc 를 누르고, 커맨드 모드로 만든 다음,

dd로 한줄을 없애주었습니다.

11) Add few more lines. Use ‘o’.

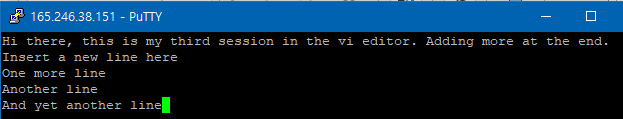
**Hi there, this is my third session in the vi editor. Adding more at the end.**

**Insert a new line here.**

**One more line.**

**Another line.**

**And yet another line.**



o를 통해 새로운 줄 + 새로운 내용을 추가해 주었습니다.

12) Change the last line.

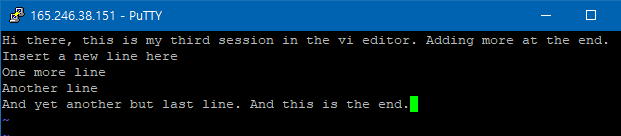
**Hi there, this is my third session in the vi editor. Adding more at the end.**

**Insert a new line here.**

**One more line.**

**Another line.**

**And yet another but last line. And this is the end.**



저는 dd로 한줄을 없애준 후, o 를 통해 한줄 추가 + 새로운 내용 추가를 하였습니다.

13) Copy and paste as follows. Use ‘2yy’ to copy two lines at the current cursor; move the cursor to another location and use ‘p’ to paste them at that location.

**Hi there, this is my third session in the vi editor. Adding more at the end.**

**One more line.**

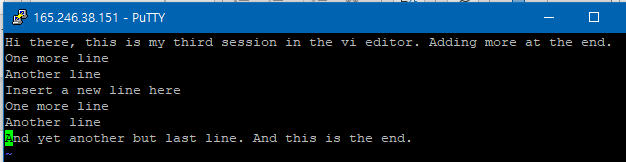
**Another line.**

**Insert a new line here.**

**One more line.**

**Another line.**

**And yet another but last line. And this is the end.**



리눅스 커서 방향은 왼 > 오 이므로,

3번째 줄에서 2yy로 3-4 줄을 cp한 다음

p(paste) 를 1번쨰 줄에서 해주면

2-3 줄에 3-4 줄의 내용이 잘 복.붙 된 것을 확인할 수 있습니다.

14) Go to line 6 with ‘:6’ and make change as follows.

**Hi there, this is my third session in the vi editor. Adding more at the end.**

**One more line.**

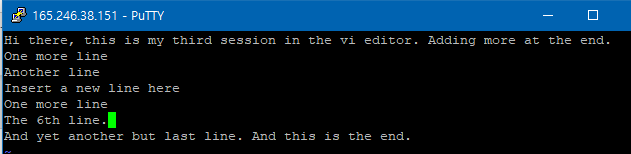
**Another line.**

**Insert a new line here.**

**One more line.**

**The 6th line.**

**And yet another but last line. And this is the end.**



**:6**으로, 6번째 줄에 가준 후,

dd를 통해 Another line을 없애 줍니다.

I 로 6th line 내용을 추가해 줍니다.

15) Write a simple c program and compile and run.

Write a program:

$ vi ex1.c

#include <stdio.h>

void main(){

printf("hi there\n");

}

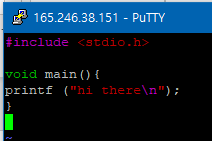
Compile:

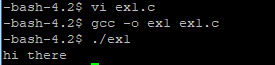
$ gcc –o ex1 ex1.c

Run:

$ ./ex1

hi there





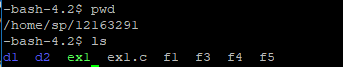
vi ex1.c 를 통해, c파일의 내용을 수정해주고, (확장명이 .c 이므로 c언어 파일입니다)

gcc -o ex1 ex1.c 를 통해, ex1.c 파일을 ex1파일에 컴파일 해줍니다.

./ex1 를 통해 컴파일 결과값 – 코드 실행 결과 값을 확인합니다.

lec4

1) Login to the system. Show the current directory. Show what files are in your directory.

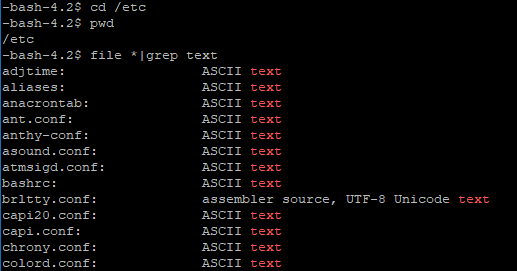


로그인 위치에서

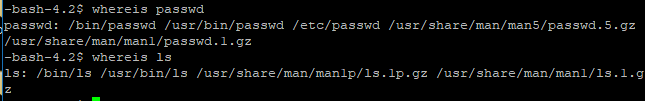
ls로 자식 파일들을 확인하여 보았습니다.

2) Go to “/etc” directory. "file \*" will show the information for all files in the current directory. Combine "file \*" and "grep" using the pipe symbol(|) to display file information only for text files.

$ file \* | grep text



3) Find the location of the password file (“passwd”), the location of C header files such as "stdio.h", and the location of utility programs (or Linux commands) such as “ls”. Use "whereis" commad.



4) Go to your login directory ("cd" without arguments will move you to your login directory). Make ex1.c using vi. Compile with "gcc" and run.

vi ex1.c

#include <stdio.h>

void main(){

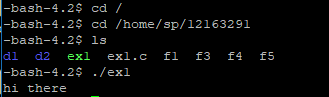
printf("hello\n");

}

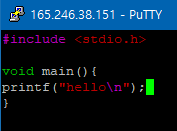
gcc -o ex1 ex1.c

./ex1

hello



이미 ex1 파일이 있어 실행하여보니 내용이 hi there 이었습니다.



내용을 hello로 변경해 주었습니다.



로 컴파일 해보니 내용이 바뀐 것을 확인할 수 있었습니다

To compile with g++, change void main() => int main()

#include <stdio.h>

int main(){

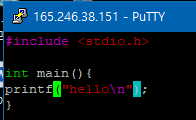
printf("hello\n);

}

g++ -o ex1 ex1.c

./ex1

hello

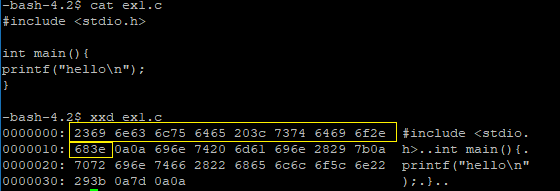


c++ 파일로 컴파일 하기 위해서는 main의 형식을 void 에서 int 로 변경해주어야합니다.



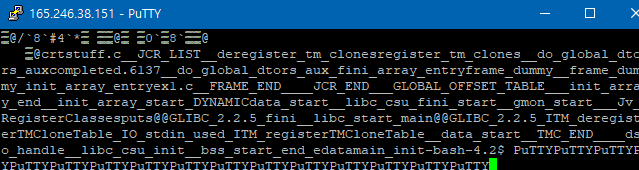
g++ 로 컴파일이 되는 것을 확인할 수 있었습니다.

5) Display the contents of ex1.c using cat and xxd. With xxd, you can see the ascii code for each character in ex1.c. Find the ascii codes for the first line of the program: "#include <stdio.h>".

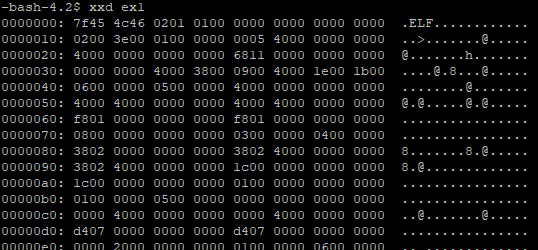


아스키 코드에서 i = 69 이므로 xxd로 ex1.c 파일을 연결과 아스키 코드로 제대로 출력되고 있는 것을 확인할 수 있습니다.

6) Display the contents of ex1 (the executable file). You cannot use "cat" to see ex1. Why?



ex1 은 ex1.c를 컴퓨터 기계에 맞춘 기계어-어셈블리어로 구성되어있으므로, char 로 내용을 표현할 수 없기 때문에 위와 같이 char의 내용인 부분은 읽을 수 없게됩니다.



cat에 반해 , 아스키코드로 볼 수 있는 xxd로 표현하는 경우에는 읽을 수 있습니다.

6-1) The compiler has translated the C statements in ex1.c into machine instructions and stored in ex1:

55 -- push rbp

48 89 e5 -- mov rbp, rsp

bf f0 05 40 00 -- mov edi, 0x400df0

.........

Find these machine instructions in ex1 with xxd. Use "/pattern" command in vi to search for a string.

$ xxd ex1 > x

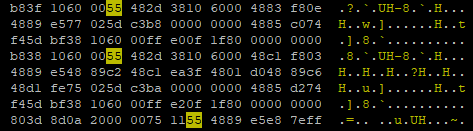
$ vi x

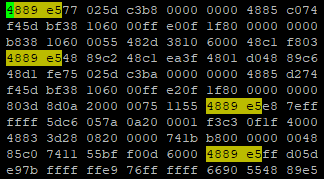


이렇게 xxd > 를 사용하게되면 , 어셈블리 어로 이루어진 파일도 확인할 수 있게됩니다.

vi로 ex1 파일을 아스키 코드 내용을 복사한 파일인 x파일을 연 후,

/ 커맨드로 찾고싶은 내용을 찾아보았습니다.

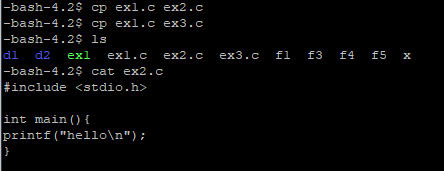




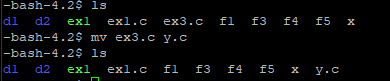
7) Copy ex1.c to ex2.c, ex3.c, and ex4.c. Remove ex2.c. Rename ex3.c to y.c.

copy 커맨드는 cp

rename 커맨드는 mv, rename 이 있습니다.



cp이후, cat으로 복사가 성공적으로 되었는지 확인해 보았습니다.



mv를 이용해 ex3.c 파일의 이름을 바꿔주었습니다.

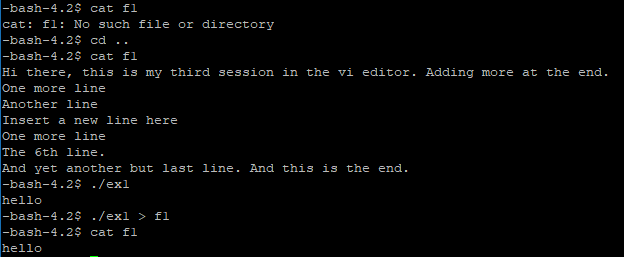
8) Make a subdirectory. Copy y.c in this subdirectory.





9) Redirect the output of ex1 to another file using ">" symbol.

$ ./ex1 > f1



우선 만든 디렉토리에서 cd ..로 12163291 로 한칸 올라간다음

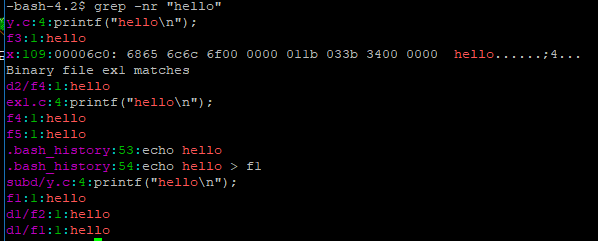
./ex1 > f1을 통해

원래 Hi there … 내용이 들어있던 f1파일의 내용을

./ex1의결과인 hello로 바꿔줍니다.

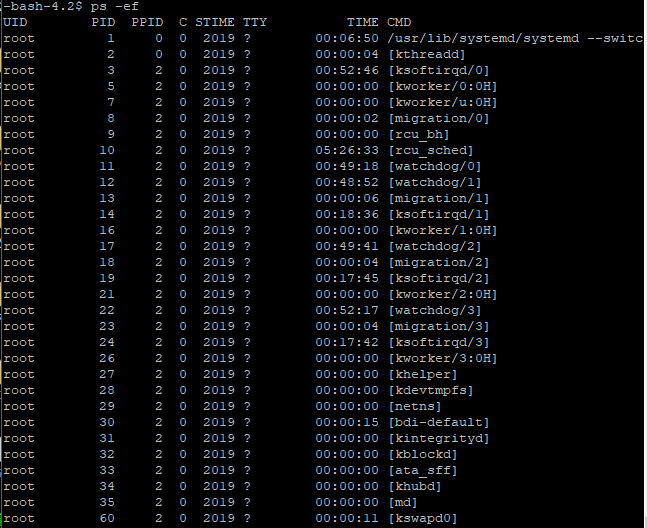
ex1.c의 내용이아닌, 그 컴파일 결과인 hello 가 f1에 들어가게 됩니다.

10) Use grep to search "hello" in all files (use -nr option).



grep 커맨드로 hello라는 내용이 들어있는 것들은 모두 찾을 수 있는 것을 확인할 수 있습니다.

11) Find out what processes exist in your system. Use "ps -ef".



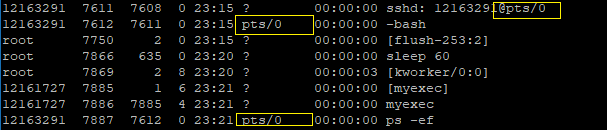
ps -ef를 통해 프로세스들을 확인할 수 있습니다.

12) "ps -ef" shows all the processes in the system. How do you know which ones are yours? Use "tty" for this purpose. Note that when a user logs in, the system allocates a terminal, and you can find the terminal number with "tty" command. What is your terminal number?



tty 커맨드를 통해 제게 할당된 터미널 넘버를 찾아보았습니다.

제 터미널#는 0이었습니다.



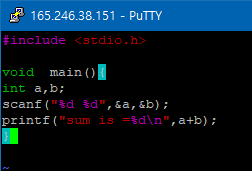
정확히는 모르겠으나, tty부분이 pts/0 인 부분을 찾아보니

12163291 – 제 학번이 할당된 부분인 것을 확인할 수 있었습니다.

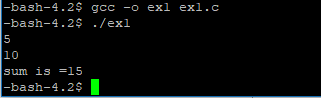
캡쳐본의 첫 줄을 보면

@pts/0 으로 pts/0 이 할당(?)되고 있는 것을 확인할 수 있습니다.

13) Modify ex1.c so that it receives two numbers from the user and prints the sum. Use scanf() for this.



위와 같이 ex1.c 파일을 수정했습니다.



결과가 잘 출력되는 것을 확인할 수 있었습니다.

14) Modify ex1.c so that it contains an infinite loop after printing "hello".

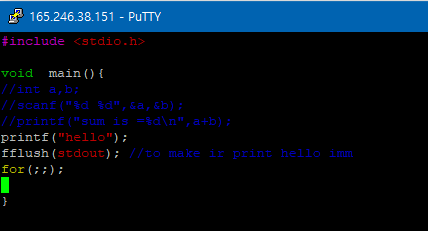
........

printf("hello");

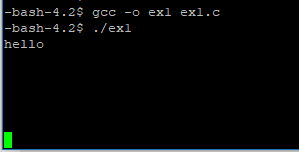
fflush(stdout); // to make it print hello immediately

for(;;);

.......



결과를 확인하여 보니



hello 이후에 커맨드라인이 나오지 않고, 계속 공백이 출력되는 것을 확인할 수 있었습니다

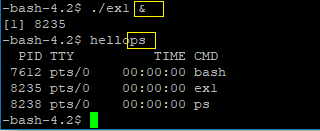


Ctrl + C로 종료해 줍니다.

15) Run the program with & at the end, and use ps to check its status. "&" puts the process in the background so that you can type next command.

$ ./ex1 &

$ ps



./ex1 뒤에 &를 붙여주니, 14)와 달리, 커맨드 라인이 나오게되는 것을 확인할 수 있었습니다.

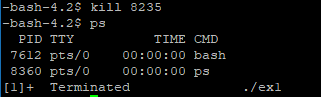
&를 통해 뒤에 커맨드라인의 등장을 확실히 할 수 있는 것을확인할 수 있었습니다.

16) Kill it with "kill" command.



kill 커맨드로 킬 해주었습니다.

kill은 PID값이 필요하므로,

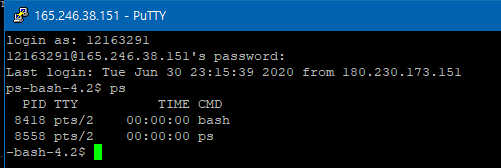


앞서 확인한 ex1 의 PID값으로 돌아가고있던 ex1 프로세스를 kill해줍니다.

17) Run the program again without & at the end. Open another login window, find out the process ID of the process running in the first window, and kill it.



뒤에 &없이 ./ex1 프로그램을 실행해주었습니다.



그후, putty로 새창을 열어 로그인한 후, ps로 현재 진행되는 프로세스를 확인하였습니다.

저는 ex1의 PID가 여기에 떠서,

kill PID값을 해서 다른 창의 ex1 프로세스를 끝낼 수 있을것이라 예상하였는데,

ex1의 PID값이 뜨지 않아,

이 문제를 어떻게 풀어야 할지 모르겠습니다.

- Ctrl C로 강제 종료 해주었습니다.

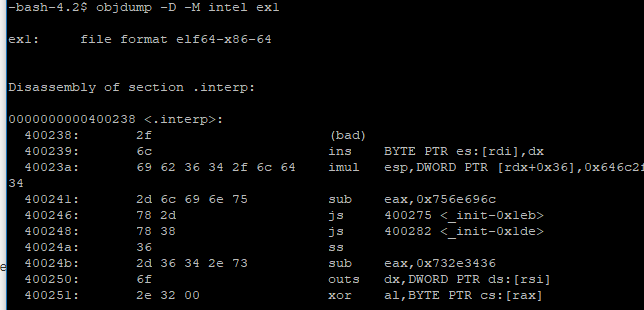
18) Use "objdump -D -M intel ex1" to dump the assembly code of ex1.c. Find <main>.

objdump는 GNU 바이너리 유틸리티의 일부로서, 라이브러리, 컴파일된 오브젝트 모듈, 공유 오브젝트 파일, 독립 실행파일등의 바이너리 파일들의 정보를 보여주는 프로그램입니다.

[-D|--disassemble-all]

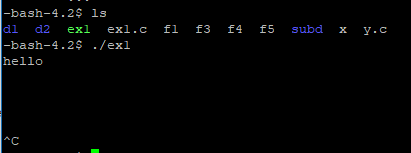
[-M options|--disassembler-options=options]

이므로, 커맨드 설명상, ex1.c의 어셈블리 코드인 ex1을 디스 어셈블 하게 될 것이라고 예상됩니다.



엄청 긴 코드가 출력되었습니다.

… Disassembly… 내용으로, 위의 예상대로, 디스어셈블이 되는 것이라 생각됩니다.



이후에도 ex1은 돌아가는데, 디스어셈블에 대한 이해가 더욱필요하다 생각됩니다.

19) Run following and tell the difference between gets and fgets

#include <stdio.h>

#include <string.h>

int main(){

char buf[20];

printf("enter a sentence\n");

gets(buf);

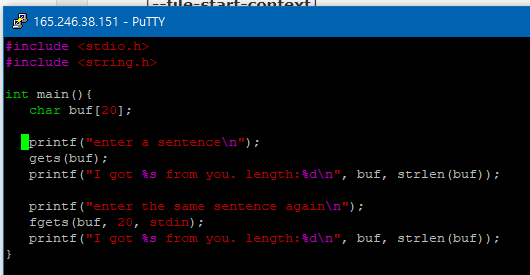
printf("I got %s from you. length:%d\n", buf, strlen(buf));

printf("enter the same sentence again\n");

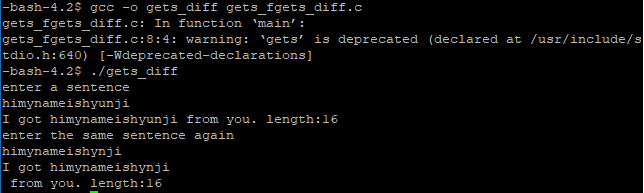
fgets(buf, 20, stdin);

printf("I got %s from you. length:%d\n", buf, strlen(buf));

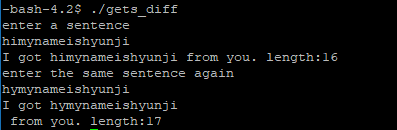
}



컴파일 후 실행해보았습니다.



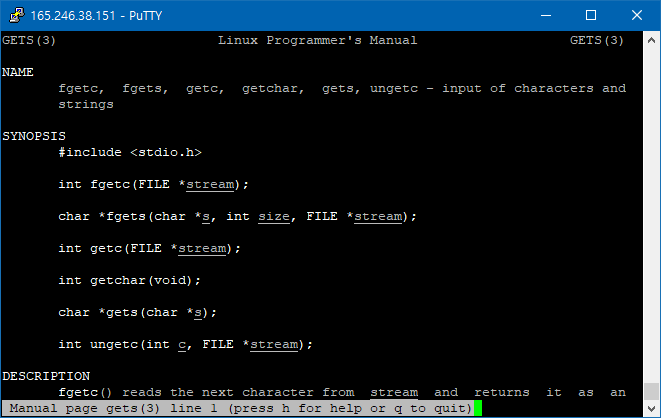
컴파일시 워닝이 뜨는데, 에러가 아니므로 무시합니다.

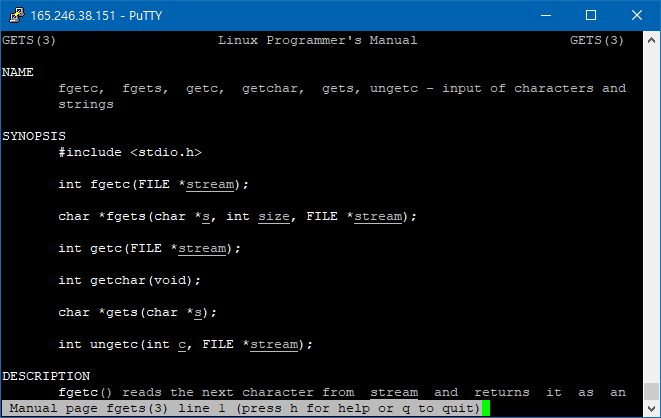


get는 줄바꿈이 일어나지 않았고, fgets는 fgets로 받은 내용이 출력된 이후에, 저절로 줄바꿈이 생성되었습니다.

따라서, 길이도 fgets 가 +1이 됩니다.

20) Write a program to read a sentence and echo it as follows. Use gets() or fgets(). Do "man gets" or "man fgets" to find out the usage of them.



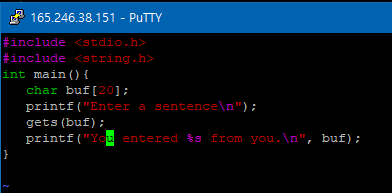


man 커맨드로 각 gets의 쓰임새를 확인하였습니다.

Enter a sentence

aa bcd e e ff aa bcd bcd hijk lmn al bcd

You entered aa bcd e e ff aa bcd bcd hijk lmn al bcd





숙제내용대로 출력해 보았습니다.