

2018 시스템 프로그래밍
- Lab 04 -

제출일자	2018.10.17
분 반	01
이 름	함지희
학 번	201702087

실습 1 - 프로그램 작성

조건 1 student 라는 이름의 struct 정의

```
struct.c (~) - VIM
b201702087@2018-sp:~$ vi struct.c
1  include <stdio.h>
2
3  struct student{
4      int sn;
5      char* initial;
6  };
7
8  void swap(struct student* arg1, struct student* arg2){
9      struct student temp;
10     temp = *arg1;
~/struct.c [utf-8,unix] [c] 1,1/32 Top
```

student라는 이름의 구조체를 정의 해준다.

sn은 학번이고 int형으로 받게 선언한다. initial은 이니셜이고 char*를 이용해 String형인 이니셜을 받을 수 있게 한다.

조건 2 조교와 학생의 student 구조체를 생성 후 정보 입력

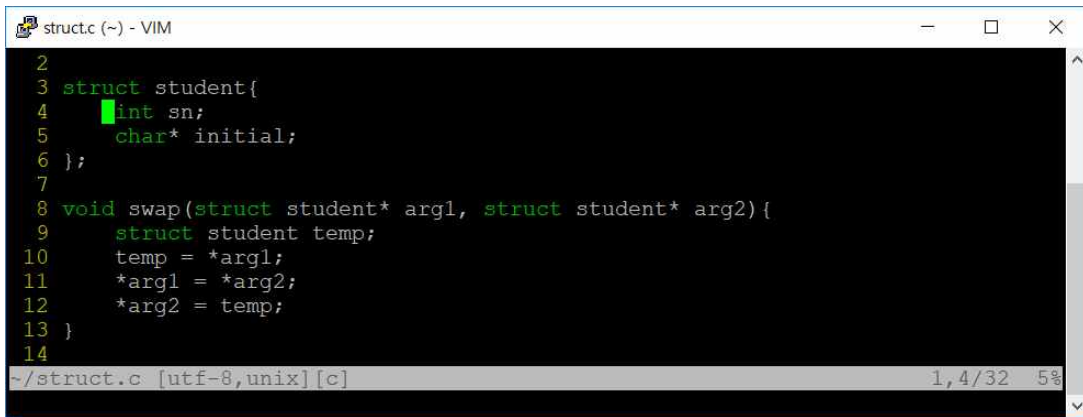
```
struct.c (~) - VIM
13 }
14
15 int main(){
16     struct student ham;
17     struct student t;
18     ham.sn = 201702087;
19     ham.initial = "hjh";
20     t.sn = 1;
21     t.initial = "hdh";
22
~/struct.c [utf-8,unix] [c] 0,22/32 54%
```

student 구조체 ham(자신)과 t(조교)를 생성한다.

ham의 sn에 201702087을 저장하고 ham의 initial에 hjh를 저장한다.

마찬가지로 t의 sn에 1을 저장하고 initial에 hdh를 저장한다.

조건 3 | swap 함수



```
struct.c (~) - VIM
2
3 struct student{
4     int sn;
5     char* initial;
6 };
7
8 void swap(struct student* arg1, struct student* arg2){
9     struct student temp;
10    temp = *arg1;
11    *arg1 = *arg2;
12    *arg2 = temp;
13 }
14
~/struct.c [utf-8,unix] [c] 1,4/32 5%
```

구조체 student 포인터 arg1 , arg2를 매개변수로 받는 swap함수를 만든다.

call by reference를 사용하기 위해 포인터를 사용한다.

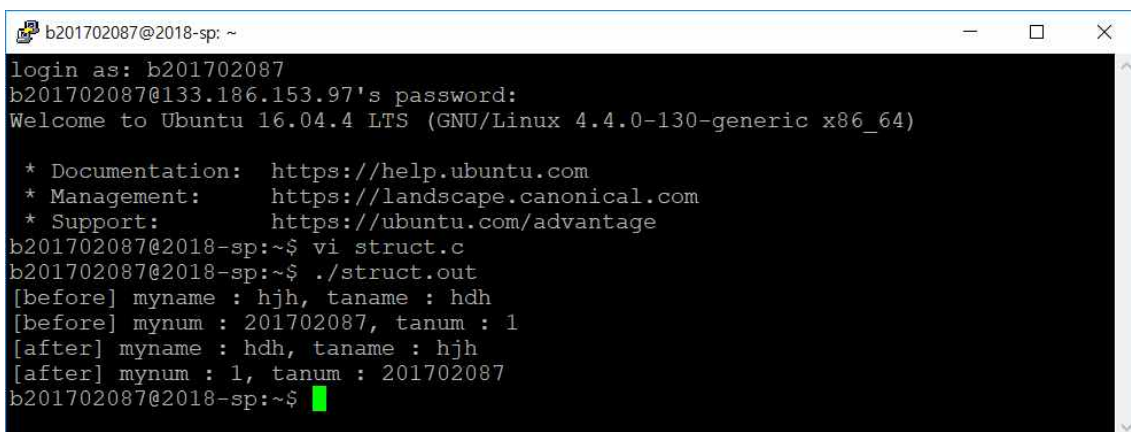
구조체 student형의 temp변수를 선언해준다.

우선 temp에 arg1포인터로 접근한 데이터를 넣어준다.

*arg1에 arg2포인터로 접근한 데이터를 쓴다.

* arg2에 temp값을 쓴다.

결과화면 | 셸창에서 결과 화면에 대한 출력



```
b201702087@2018-sp: ~
login as: b201702087
b201702087@133.186.153.97's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-130-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
b201702087@2018-sp:~$ vi struct.c
b201702087@2018-sp:~$ ./struct.out
[before] myname : hjh, taname : hdh
[before] mynum : 201702087, tanum : 1
[after] myname : hdh, taname : hjh
[after] mynum : 1, tanum : 201702087
b201702087@2018-sp:~$
```

실습 2

1 학생, 조교 저장 데이터 확인

```
b201702087@2018-sp: ~
Reading symbols from struct.out...done.
(gdb) b 22
Breakpoint 1 at 0x400616: file struct.c, line 22.
(gdb) b 27
Breakpoint 2 at 0x40065a: file struct.c, line 27.
(gdb) run
Starting program: /home/sys01/b201702087/struct.out

Breakpoint 1, main () at struct.c:23
warning: Source file is more recent than executable.
23      printf("[before] myname : %s, taname : %s \n", ham.initial, t.in
      itial);
(gdb) info local
ham = {sn = 201702087, initial = 0x400738 "hjh"}
t = {sn = 1, initial = 0x40073c "hdh"}
(gdb) c
Continuing.
[before] myname : hjh, taname : hdh
[before] mynum : 201702087, tanum : 1

Breakpoint 2, main () at struct.c:28
28      printf("[after] myname : %s, taname : %s \n", ham.initial, t.ini
      tial);
(gdb) info local
ham = {sn = 1, initial = 0x40073c "hdh"}
t = {sn = 201702087, initial = 0x400738 "hjh"}
(gdb)
```

gdb를 하기 위해 struct.c를 gcc -g -o struct.out struct.c로 컴파일 해주었다.

gdb struct.out을 해주어 gdb를 실행시켰다.

그 다음엔 swap전의 조교와 학생의 데이터를 확인해주기 위해 22행에 breakpoint를 넣어주었다. 22행은 struct.c의 main함수에서 조교와 학생의 데이터를 입력한 뒤의 행이다.

그 다음엔 swap이후의 데이터를 확인하기 위해 27행에 breakpoint를 넣어주었는데 27행은 swap함수를 호출한 직후의 행이다.

run으로 struct.out을 실행시키면 첫 번째 b인 22행에서 멈추게 된다. 화면에서는 23행에서 멈췄지만 그 이유는 내 코드에서 22행은 빈칸이라서 그런다. info local을 사용하여 현재 변수의 데이터 값을 출력한다.

c를 눌러 다음 b까지 실행시켜주면 2번째 b인 27행에서 멈추게 되는데 내 코드는 거기가 빈칸이라 28행에서 멈췄다. 거기서 info local을 해주면 swap이후의 변수의 데이터 값이 출력된다.

```

b201702087@2018-sp: ~
[before] mynum : 201702087, tanum : 1

Breakpoint 1, swap (arg1=0x7fffffff500, arg2=0x7fffffff510) at struct.c:10
warning: Source file is more recent than executable.
10      temp = *arg1;
(gdb) display temp
1: temp = {sn = 0, initial = 0x7ffff7ffe168 ""}
(gdb) display *arg1
2: *arg1 = {sn = 201702087, initial = 0x400738 "hjh"}
(gdb) display *arg2
3: *arg2 = {sn = 1, initial = 0x40073c "hdh"}
(gdb) s
11      *arg1 = *arg2;
1: temp = {sn = 201702087, initial = 0x400738 "hjh"}
2: *arg1 = {sn = 201702087, initial = 0x400738 "hjh"}
3: *arg2 = {sn = 1, initial = 0x40073c "hdh"}
(gdb) s
12      *arg2 = temp;
1: temp = {sn = 201702087, initial = 0x400738 "hjh"}
2: *arg1 = {sn = 1, initial = 0x40073c "hdh"}
3: *arg2 = {sn = 1, initial = 0x40073c "hdh"}
(gdb) s
13  }
1: temp = {sn = 201702087, initial = 0x400738 "hjh"}
2: *arg1 = {sn = 1, initial = 0x40073c "hdh"}
3: *arg2 = {sn = 201702087, initial = 0x400738 "hjh"}
(gdb)

```

8행에 breakpoint를 걸어서(b swap으로 해도 됨) swap함수 전까지 실행시킨 후 s로 swap함수를 한 행씩 실행시키면서 변수의 변화를 확인했다.

display temp, display *arg1, display *arg2 로 s로 한 행씩 실행 시킬 때 마다 변수의 상태를 출력하게끔 했다.

1. temp안엔 쓰레기값이 들어있고 *arg1엔 학생의 값이, *arg2엔 조교의 값이 들어있다.
2. temp에 *arg1의 데이터가 입력되고, *arg1은 그대로 학생의 값, *arg2엔 조교의 값이 들어있다.
3. temp는 그대로이다. *arg1에 *arg2의 데이터가 입력되고, *arg2엔 그대로 조교의 값이 들어있다.
4. temp는 그대로이다. *arg1엔 조교의 값, *arg2엔 학생의 값이 됨으로써 성공적으로 swap이 되었다.