


2018 시스템 프로그래밍
- Lab 04 -

제출일자	2018.10.19
분 반	02
이 름	박상현
학 번	201702012

실습 1 - 프로그램 작성

조건 1 student 라는 이름의 struct 정의

소스코드 스크린샷
(putty 창이 보이게)



```
a.c (~/C_GDB_silsub/silsub1) - VIM
1 #include <stdio.h>
2
3 struct student{
4     char *myname;
5     int mynum;
6 };
```

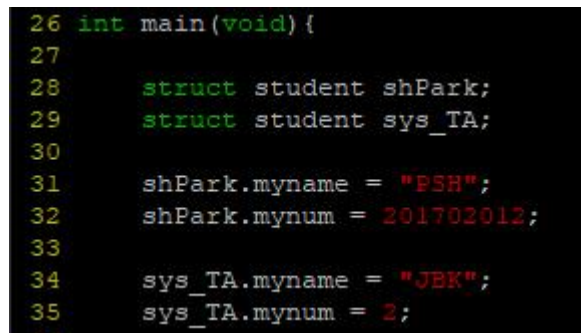
소스 코드에 대한 설명

student 이름의 struct를 정의하는 코드입니다.
myname을 char형 포인터 변수로 선언하고, mynum을 int 변수로 선언하였습니다.

조건 2 조교와 학생의 student 구조체를 생성 후 정보 입력

스크린 샷

- student 구조체를 2개 생성 (자신, 조교)
- 생성한 구조체에 각각 정보 입력



```
26 int main(void) {
27
28     struct student shPark;
29     struct student sys_TA;
30
31     shPark.myname = "PSH";
32     shPark.mynum = 201702012;
33
34     sys_TA.myname = "JBK";
35     sys_TA.mynum = 2;
```

설명

struct student (구조체이름)
을 통하여 student 구조체를 2개 생성합니다. 그 후, 각각의 구조체에 . 으로 접근하여
값을 입력합니다.

조건 3	swap 함수
------	---------

스크린 샷

```
9 void swap(struct student *arg1, struct student *arg2){
10
11     char *name_t;
12     int num_t;
13     name_t = arg1->myname;
14     arg1->myname = arg2->myname;
15     arg2->myname = name_t;
16
17     num_t = arg1->mynum;
18     arg1->mynum = arg2->mynum;
19     arg2->mynum = num_t;
20
21     printf("swap Function call!!\n");
22
23 }
```

설명

arg1과 arg2에 각각 구조체 주솟값이 들어있습니다.
이를 통해 각각의 구조체의 정보들에 접근하는데, struct pointer이기 때문에 . 이 아닌,
->를 사용하여 접근합니다.
자세한 swap과정은 실습2의 2.swap과정 설명 에 적어놓았습니다.

결과화면	셸창에서 결과 화면에 대한 출력
------	-------------------

```
c201702012@2018-sp: ~/C_GDB_silsub/silsub1
c201702012@2018-sp:~/C_GDB_silsub/silsub1$ gcc -g -o a.out a.c
c201702012@2018-sp:~/C_GDB_silsub/silsub1$ ls
a.c  a.out
c201702012@2018-sp:~/C_GDB_silsub/silsub1$ ./a.out
myname : PSH
mynumber : 201702012
taname : JBK
class : 2
[before] myname : PSH, taname : JBK
[before] mynum : 201702012, tanum : 2
swap Function call!!
[after] myname : JBK, taname : PSH
[after] mynum : 2, tanum : 201702012
c201702012@2018-sp:~/C_GDB_silsub/silsub1$
```

실습 2

1 학생, 조교 저장 데이터 확인

```
c201702012@2018-sp: ~/C_GDB_silsub/silsub1
c201702012@2018-sp:~/C_GDB_silsub/silsub1$ ls
a.c  a.out
c201702012@2018-sp:~/C_GDB_silsub/silsub1$ gdb a.out
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...done.
(gdb)
```

```
(gdb) b main
Breakpoint 1 at 0x400641: file a.c, line 26.
(gdb) b a.c : 45
Breakpoint 2 at 0x4006f3: file a.c, line 45.
(gdb) b a.c : swap
Breakpoint 3 at 0x4005e6: file a.c, line 13.
(gdb) b a.c : 14
Breakpoint 4 at 0x4005f1: file a.c, line 14.
(gdb) b a.c : 15
Breakpoint 5 at 0x4005ff: file a.c, line 15.
(gdb) b a.c : 17
Breakpoint 6 at 0x40060a: file a.c, line 17.
(gdb) b a.c : 18
Breakpoint 7 at 0x400614: file a.c, line 18.
(gdb) b a.c : 19
Breakpoint 8 at 0x400622: file a.c, line 19.
(gdb) info break
Num      Type             Disp Enb Address            What
1        breakpoint        keep y   0x0000000000400641 in main at a.c:26
2        breakpoint        keep y   0x00000000004006f3 in main at a.c:45
3        breakpoint        keep y   0x00000000004005e6 in swap at a.c:13
4        breakpoint        keep y   0x00000000004005f1 in swap at a.c:14
5        breakpoint        keep y   0x00000000004005ff in swap at a.c:15
6        breakpoint        keep y   0x000000000040060a in swap at a.c:17
7        breakpoint        keep y   0x0000000000400614 in swap at a.c:18
8        breakpoint        keep y   0x0000000000400622 in swap at a.c:19
(gdb)
```

실습 수행에 대한 설명
최대한 자세하게 작성
(gdb 실행 및 break point 설정)

gcc -g -o a.out a.c를 통해 컴파일을 한 후에,
a.c의 실행파일인 a.out을 디버깅하기 위해
gdb a.out을 입력하여 gdb를 실행합니다.
그 후에, break point들을 거는데, 저는 main에 먼저 걸고, swap함수를 호출하는
45line에 걸었습니다.
그 후에는, swap함수 내의 swap과정들을 나타내는 line들에 break point를
설정하였습니다.

```

(gdb) run
Starting program: /home/sys02/c201702012/C_GDB_silsub/silsub1/a.out

Breakpoint 5, main () at a.c:26
26      int main(void){
(gdb) c
Continuing.
myname : PSH
mynumber : 201702012
taname : JBK
class : 2
[before] myname : PSH, taname : JBK
[before] mynum : 201702012, tanum : 2

Breakpoint 6, main () at a.c:45
45      swap(&shPark, &sys_TA);
(gdb) info locals
shPark = {myname = 0x4007fd "PSH", mynum = 201702012}
sys_TA = {myname = 0x400801 "JBK", mynum = 2}
(gdb) display shPark.myname
1: shPark.myname = 0x4007fd "PSH"
(gdb) display shPark.mynum
2: shPark.mynum = 201702012
(gdb) display sys_TA.myname
3: sys_TA.myname = 0x400801 "JBK"
(gdb) display sys_TA.mynum
4: sys_TA.mynum = 2
(gdb) s

Breakpoint 7, swap (arg1=0x7fffffff4d0, arg2=0x7fffffff4e0) at a.c:13
13      name_t = arg1->myname;
(gdb) info locals
name_t = 0x7ffff7ffe168 ""
num_t = 0
(gdb) display name_t
5: name_t = 0x7ffff7ffe168 ""
(gdb) display num_t
6: num_t = 0
(gdb) s

Breakpoint 8, swap (arg1=0x7fffffff4d0, arg2=0x7fffffff4e0) at a.c:14
14      arg1->myname = arg2->myname;
5: name_t = 0x4007fd "PSH"
6: num_t = 0
(gdb) s

Breakpoint 9, swap (arg1=0x7fffffff4d0, arg2=0x7fffffff4e0) at a.c:15
15      arg2->myname = name_t;
5: name_t = 0x4007fd "PSH"
6: num_t = 0
(gdb) s

Breakpoint 10, swap (arg1=0x7fffffff4d0, arg2=0x7fffffff4e0) at a.c:17
17      num_t = arg1->mynum;
5: name_t = 0x4007fd "PSH"
6: num_t = 0
(gdb) s

Breakpoint 11, swap (arg1=0x7fffffff4d0, arg2=0x7fffffff4e0) at a.c:18
18      arg1->mynum = arg2->mynum;
5: name_t = 0x4007fd "PSH"
6: num_t = 201702012
(gdb) s

Breakpoint 12, swap (arg1=0x7fffffff4d0, arg2=0x7fffffff4e0) at a.c:19
19      arg2->mynum = num_t;
5: name_t = 0x4007fd "PSH"
6: num_t = 201702012
(gdb) s
21      printf("swap Function call!!\n");
5: name_t = 0x4007fd "PSH"

```

```

6: num_t = 201702012
(gdb) s
swap Function call!!
23     }
5: name_t = 0x4007fd "PSH"
6: num_t = 201702012
(gdb) s
main () at a.c:47
47     printf("[after] myname : %s, taname : %s\n", shPark.myname, sys_TA.myname);
1: shPark.myname = 0x400801 "JBK"
2: shPark.mynum = 2
3: sys_TA.myname = 0x4007fd "PSH"
4: sys_TA.mynum = 201702012
(gdb) s
[after] myname : JBK, taname : PSH
48     printf("[after] mynum : %d, tanum : %d\n", shPark.mynum, sys_TA.mynum);
1: shPark.myname = 0x400801 "JBK"
2: shPark.mynum = 2
3: sys_TA.myname = 0x4007fd "PSH"
4: sys_TA.mynum = 201702012
(gdb) s
[after] mynum : 2, tanum : 201702012
51     }
1: shPark.myname = 0x400801 "JBK"
2: shPark.mynum = 2
3: sys_TA.myname = 0x4007fd "PSH"
4: sys_TA.mynum = 201702012
(gdb) c
Continuing.
[Inferior 1 (process 25643) exited normally]
(gdb) █

```

설명

먼저, run을 입력하여 디버깅을 실행합니다.

c를 입력하여 break point가 걸린 swap함수로 이동하여 함수를 호출합니다.

그 전에, info locals로 변수들을 확인하고, 각각 display shPark.myname 등으로 display를 설정해줍니다.

swap함수에 들어가서도 info locals와 display를 설정해줍니다.

swap함수는 두 개의 struct 주소값을 입력받는데, 각각 제 정보가 들어있는 구조체인 shPark 와 조교님 정보가 들어있는 구조체인 sys_TA 의 주소값을 &을 붙여 전달합니다.

주소값을 전달받은 swap함수는 포인터매개변수인 arg1과 arg2를 통해 각각 두 구조체에 접근합니다.

swap함수에서는 두 개의 변수를 선언하였는데, 각각 char포인터형 변수인 name_t와 int형 변수인 num_t입니다. 뒤에 t는 temp를 의미하는 뜻에서 붙였습니다.

swap이 작동하는 방식을 설명하겠습니다.

먼저, struct 포인터인 arg1과 arg2를 접근하기 위해 ->를 사용합니다.

arg1->myname으로 myname에 접근한 값을 temp변수인 name_t에 저장합니다. arg1의 myname에 arg2의 myname값을 저장하고, name_t의 값을 arg2의 myname에 저장합니다. 이로써 struct들의 myname이 swap이 되었습니다. 또한, 이를 info locals를 통해 swap함수의 local variables들의 값 대입을 확인할 수 있습니다. 다만, info locals를 통해서는 main에서 정의된 구조체들의 값을 확인할 수는 없었습니다. 이는 swap함수를 빠져나와 확인해보았습니다.(물론 정상적으로 값이 바뀌었습니다.=>call by reference)

myname을 swap한 방식과 동일하게 mynum을 swap해줍니다. 마지막으로, ppt처럼 "swap Function call!!"를 print해줍니다.

s를 통해 한줄씩 넘길때마다 swap함수의 local variable들의 변화가 display를 통해 확인됩니다.

이로써 swap함수는 마무리되었고, 빠져나와 main으로 다시 돌아갑니다.

main으로 돌아가 main의 local variable들의 display를 확인합니다. 처음의 info locals와 비교하였을 때, 각각의 주솟값들과 값들이 각각 swap된 것을 확인합니다.(call by reference)

※ a.c 의 전체 스크린샷입니다.

```
a.c (~/.C_GDB_silsub/silsub1) - VIM
1  #include <stdio.h>
2
3  struct student{
4      char *myname;
5      int mynum;
6  };
7
8
9  void swap(struct student *arg1, struct student *arg2){
10
11     char *name_t;
12     int num_t;
13     name_t = arg1->myname;
14     arg1->myname = arg2->myname;
15     arg2->myname = name_t;
16
17     num_t = arg1->mynum;
18     arg1->mynum = arg2->mynum;
19     arg2->mynum = num_t;
20
21     printf("swap Function call!\n");
22
23 }
24
25
26 int main(void) {
27
28     struct student shPark;
29     struct student sys_TA;
30
31     shPark.myname = "SSH";
32     shPark.mynum = 201702012;
33
34     sys_TA.myname = "JBK";
35     sys_TA.mynum = 2;
36
37     printf("myname : %s\n", shPark.myname);
38     printf("mynumber : %d\n", shPark.mynum);
39     printf("taname : %s\n", sys_TA.myname);
40     printf("class : %d\n", sys_TA.mynum);
41
42     printf("[before] myname : %s, taname : %s\n", shPark.myname, sys_TA.myname);
43     printf("[before] mynum : %d, tanum : %d\n", shPark.mynum, sys_TA.mynum);
44
45     swap(&shPark, &sys_TA);
46
47     printf("[after] myname : %s, taname : %s\n", shPark.myname, sys_TA.myname);
48     printf("[after] mynum : %d, tanum : %d\n", shPark.mynum, sys_TA.mynum);
49
50
51 }
```