

TI DSP, MCU, Xilinx Zynq FPGA 기반의 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – 김형주

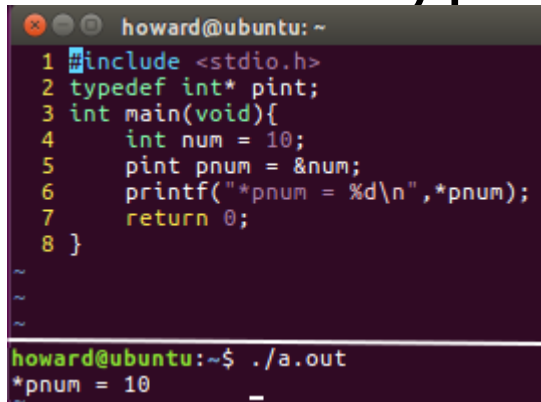
mihaelkel@naver.com

What I learned (18.02.28)

- How to use some functions
typedef, malloc(), free(), struct
- Stack memory, Heap memory
- function pointer

How to use typedef

- Declare : `typedef type customed_type;`



```
howard@ubuntu: ~  
1 include <stdio.h>  
2 typedef int* pint;  
3 int main(void){  
4     int num = 10;  
5     pint pnum = &num;  
6     printf("*pnum = %d\n",*pnum);  
7     return 0;  
8 }  
~  
~  
~  
howard@ubuntu:~$ ./a.out  
*pnum = 10  
_
```

- Description : You can use typedef when you want to transform type name to other name. especially, declaring struct type.

How to use malloc()

- Use : `arr = (type*)malloc(sizeof(type)*num);`

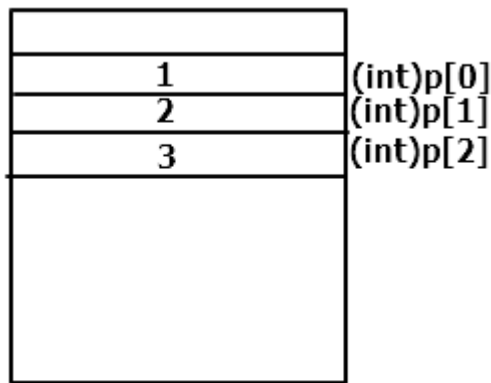
```
howard@ubuntu: ~  
1 #include <stdio.h>  
2 #include <stdlib.h>  
3 int main(void){  
4     int* arr;  
5     int size = 20, i;  
6     arr = (int*)malloc(sizeof(int)*size);  
7     for(i=0;i<size;i++)  
8         arr[i]=i;  
9     for(i=0;i<size;i++)  
10        printf("arr[%d] = %d\n",i,arr[i]);  
11  
12  
13     free(arr);  
14     return 0;  
15 }
```

```
arr[0] = 0  
arr[1] = 1  
arr[2] = 2  
arr[3] = 3  
arr[4] = 4  
arr[5] = 5  
arr[6] = 6  
arr[7] = 7  
arr[8] = 8  
arr[9] = 9  
arr[10] = 10  
arr[11] = 11  
arr[12] = 12  
arr[13] = 13  
arr[14] = 14  
arr[15] = 15  
arr[16] = 16  
arr[17] = 17  
arr[18] = 18  
arr[19] = 19
```

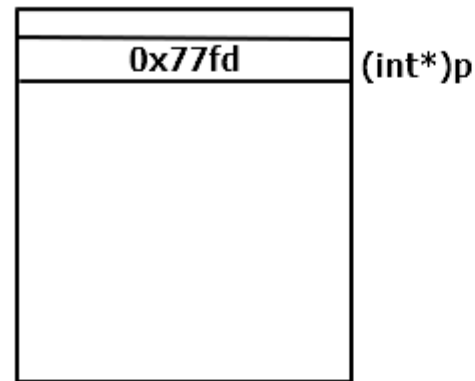
- Description : You can use `malloc()`, `free()` when you allocate memory dynamically. Assuming that you don't know how many data will come in. if you allocated memory statically, The data could be lost.

Stack memory&Heap memory

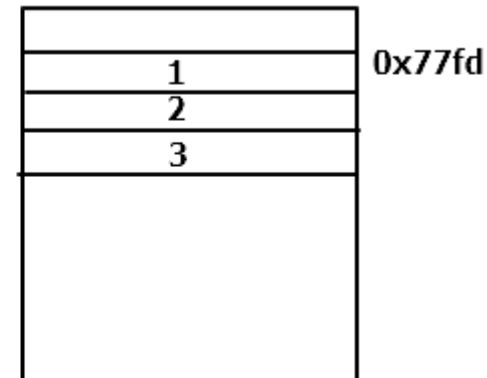
- Both are in virtual memory in CPU



stack memory



stack memory



Heap memory

- Description : Declaring data statically, The data be allocated like figure 1. Dynamically, figure 2. Using dynamic allocation In Embedded Stytem, You should consider how much heap memory in cpu.

Function pointer

- Use : `returntype (*functionname)(factor, factor);`

```
howard@ubuntu: ~  
1 #include <stdio.h>  
2 #include <stdlib.h>  
3 int add_int(int x,int y);  
4 int main(void){  
5     int x = 1, y = 2, res;  
6     int (*p)(int,int);  
7     p = add_int;  
8     res = p(x,y);  
9     printf("res = %d\n",res);  
10  
11     return 0;  
12 }  
13 int add_int(int x,int y){  
14     return x+y;  
15 }
```

res = 3

- Description : remind that all variable(including dynamic variable, array variable) has address. A function also does. The function's name is an address as array and pointer's are. A function's prototype consists of 'return value', 'function name', 'factor'. In oop(object oriented programming), there are method, object, and class, etc.. Using function pointer and struct, you can make those in C. So, If you are an expert in C, learning other language be a piece of cake.

Make class in C

- Entire code

```
howard@ubuntu: ~  
1 #include <stdio.h>  
2  
3 typedef struct _status{  
4     int Str;  
5     int Int;  
6     int Dex;  
7     int Life;  
8 }status;  
9 typedef struct _character{  
10     status stat;  
11     char input;  
12     void (*skill)(char);  
13 }character;  
14  
15 void use_Skill(char key);  
16 void init_character(character* ch,int Str,int Int,int Dex,int Life);  
17  
18 int main(void){  
19     char key;  
20     //create a character : magician  
21     character magician;  
22  
23     //Initiate the character(str 5, int 10, dex 6, life 30)  
24     init_character(&magician,5,10,6,30);  
25  
26
```

Make class in C

```
27 //Display the character status
28 printf("Str : %d\nInt : %d\nDex : %d\nLife : %d\n",
29     magician.stat.Str,
30     magician.stat.Int,
31     magician.stat.Dex,
32     magician.stat.Life
33 );
34
35 //play game
36 printf("Use your skill (q,w,e,r) : ");
37 scanf("%c",&key);
38 magician.skill(key);
39
40 return 0;
41 }
42
43 void use_Skill(char key){
44     switch(key){
45         case 'q':
46             printf("q skill used!\n");
47             break;
48         case 'w':
49             printf("w skill used!\n");
50             break;
51         case 'e':
52             printf("e skill used!\n");
53             break;
54         case 'r':
55             printf("r skill used!\n");
56             break;
57         default :
58             break;
59     }
60 }
61 void init_character(character* ch,int Str,int Int,int Dex,int Life){
62     ch->skill = use_Skill;
63
64     ch->stat.Str=5;
65     ch->stat.Int=10;
66     ch->stat.Dex=6;
67     ch->stat.Life=30;
68 }
```


Make class in C

- Result

```
Str : 5  
Int : 10  
Dex : 6  
Life : 30  
Use your skill (q,w,e,r) : w  
w skill used!
```

- Class : character
- Method in character class : skill(key)
- Object in character class : status