Function Pointers : 함수가 메모리의 어디에 저장되어 있는지 가르킬 때 사용

함수와 함수 포인터

```
void f1()
{
    return;
}

int f2(char i)
{
    return i + 1;
}
```

```
int main()
{
    void (*pf1)() = f1;
    //void (*pf1)() = &f1;

    int (*pf2)(char) = f2;

    (*pf1)();//call f1 via pf1
    //pf1();

    int a = pf2('A');
    //int a = (*pf2)('A');

    printf("%d\n", a);//66
```

변수에 대한 포인터에서 포인터 연산을 하기 위해
자료형의 크기를 미리 알아야 했던 것 처럼
예) int *a에서 a++는 sizeof(int)만큼 주소값 증가

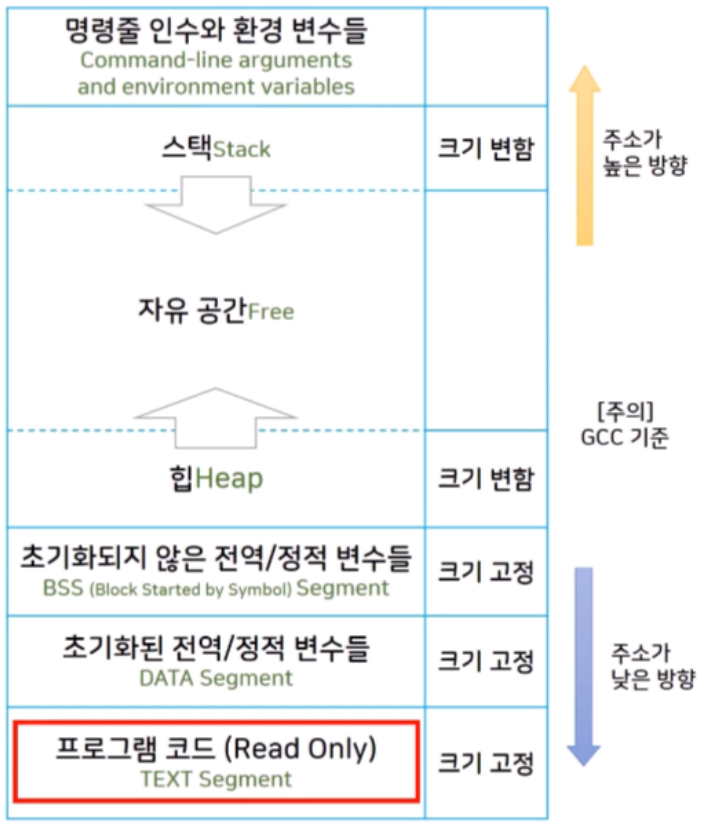
함수 포인터의 성질

프로그램이 시작될 때

```
#include <stdio.h>

void func()
{
    int i = 123;
    printf("%lld\n", (long long)&i);
}

int main()
{
    const char* message = "Banana";
    printf("Apple and %s", message);
    printf("\n");

    void (*f_ptr)() = func; // address of a function

    printf("%lld\n", (long long)&message);
    printf("%lld\n", (long long)&f_ptr);
    printf("%lld\n", (long long)message);
    printf("%lld\n", (long long)f_ptr);
    printf("%lld\n", (long long)main);

    func();

    return 0;
}
```

```
Apple and Banana
140733053998720
140733053998728
4196090
4195718
4195758
140733053998700
```

GCC, 리눅스 기준

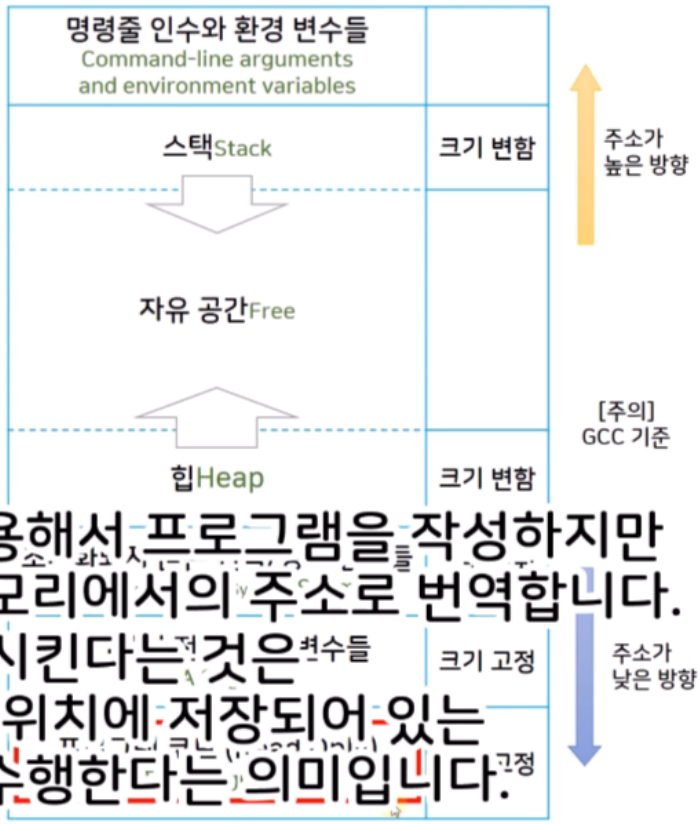


## 프로그램이 시작될 때

```c
#include <stdio.h>

void func()
{
    int i = 123;
    printf("%lld\n", (long long)&i);
}

int main()
{
    const char* message = "Banana";
    printf("Apple and %s", message);
    printf("\n");

    void (*f_ptr)() = func; // address of a function
    printf("%lld\n", (long
    printf("%lld\n", (long

    func();

    return
}
```

```
Apple and Banana
140733053998720
140733053998728
4196090
4195718
4195758
140733053998700
```

GCC, 리눅스 기준




프로그래머는 함수의 이름을 이용해서 프로그램을 작성하지만
컴파일러는 이름(식별자)들을 메모리에서의 주소로 번역합니다.
즉, 함수를 실행시킨다는 것은
메모리에서 함수의 주소 위치에 저장되어 있는
명령어들을 순차적으로 수행한다는 의미입니다.