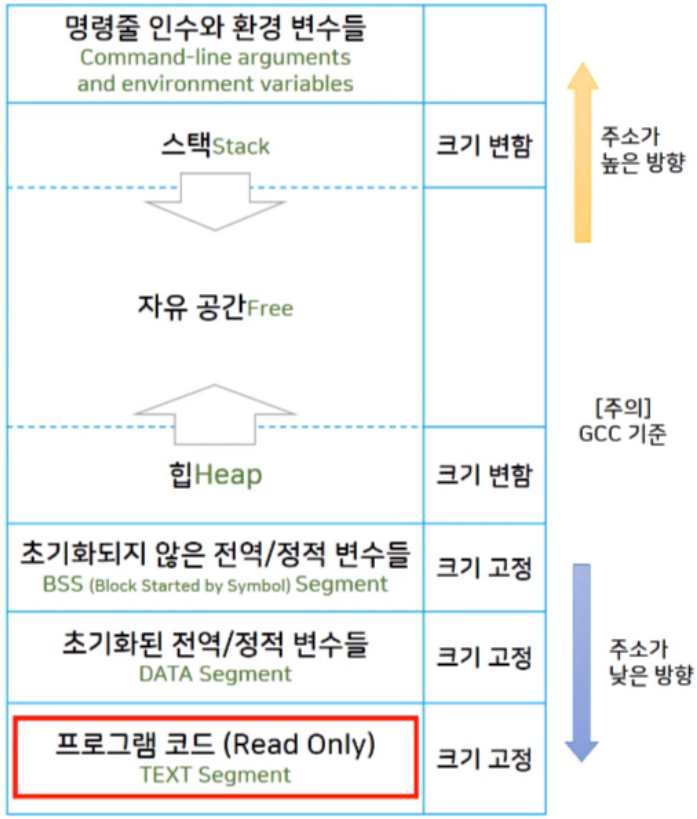
프로그램이 시작될 때

```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 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;
}
```

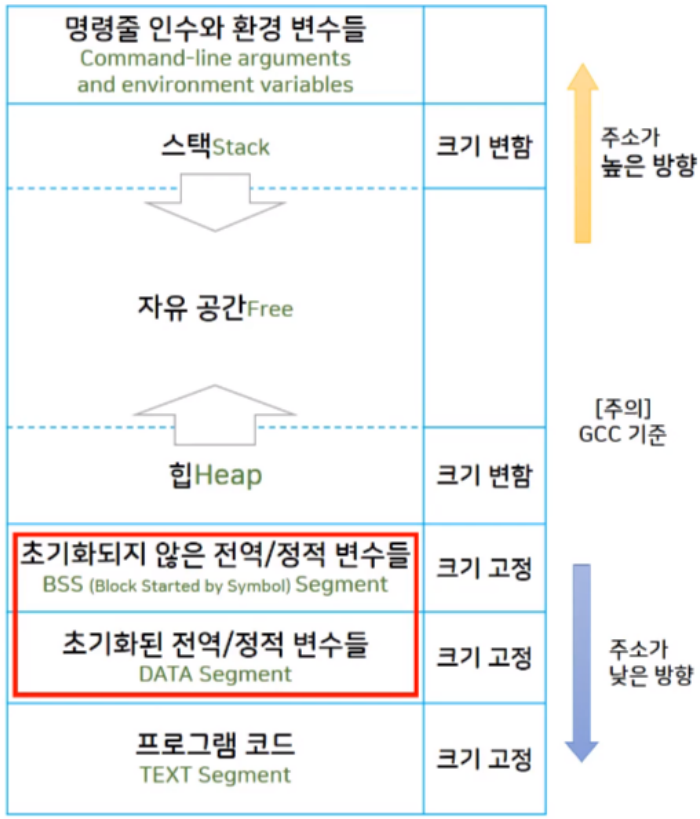
```
Apple and Banana
140733053998720
140733053998728
4196090
4195718
4195758
140733053998700
```

GCC, 리눅스 기준



프로그램 전체에서 계속 사용되는 변수들

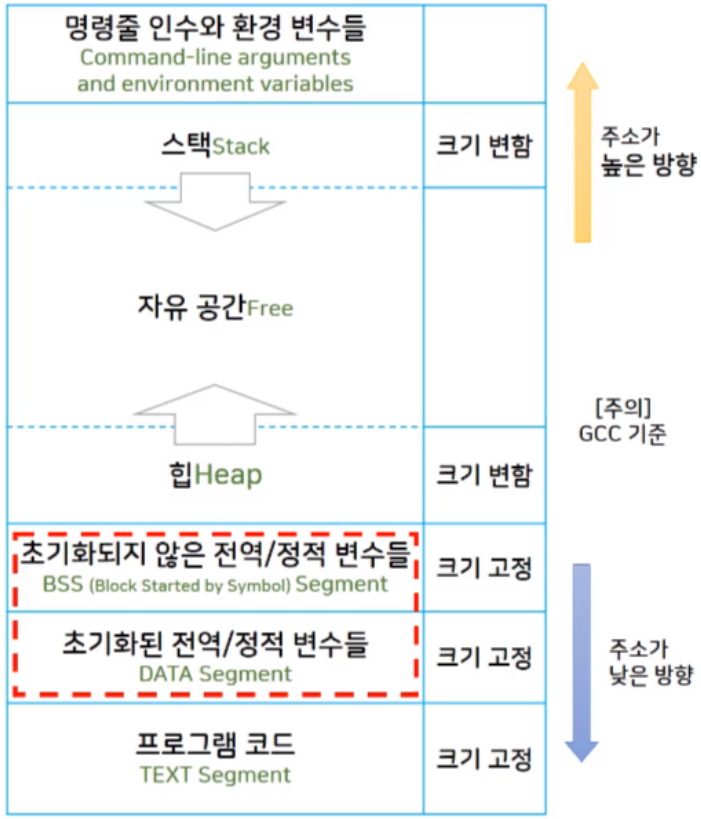
```c
#include <stdio.h>

int g_i = 123;      // global variable
int g_j;            // global variable

void func1()
{
    g_i++;          // uses g_i
}

void func2()
{
    g_i += 2;       // uses g_i
}

int main()
{
    int local = 1234;

    func1();
    func2();

    printf("%d", g_i);// uses g_i

    return 0;
}
```



- 전역 변수들 또는 함수들

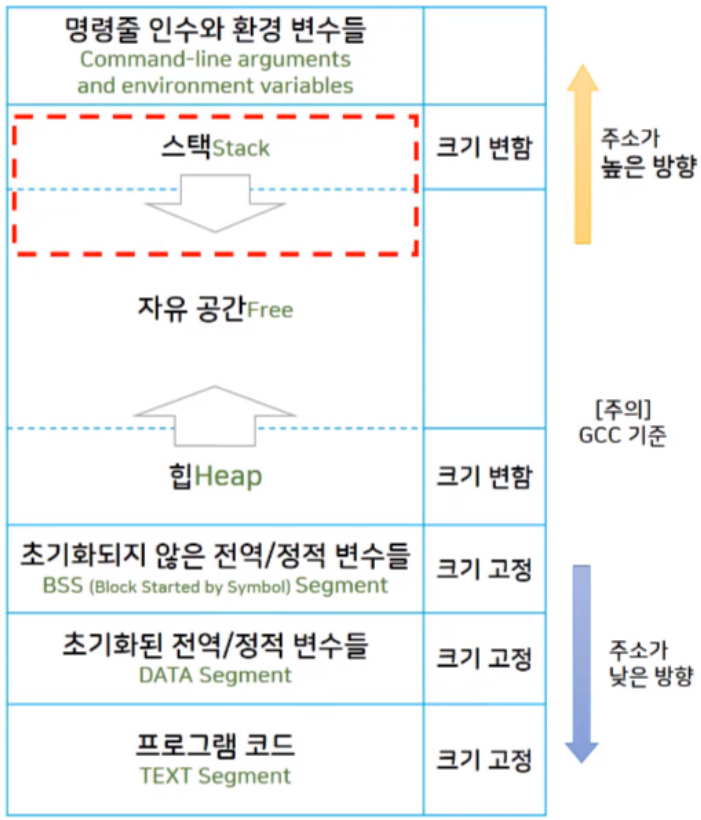
프로그램의 일부에서 큰 메모리가 필요한 경우

```c
#include <stdio.h>

#define MAX 1000

// 1. Use global variable
int g_arr[MAX];

int main()
{

/*
    Use g_arr
    ...
    Do NOT use g_arr
    ...
    Use g_arr
    ...
*/

    return 0;
}
```



프로그램의 일부에서 큰 메모리가 필요한 경우

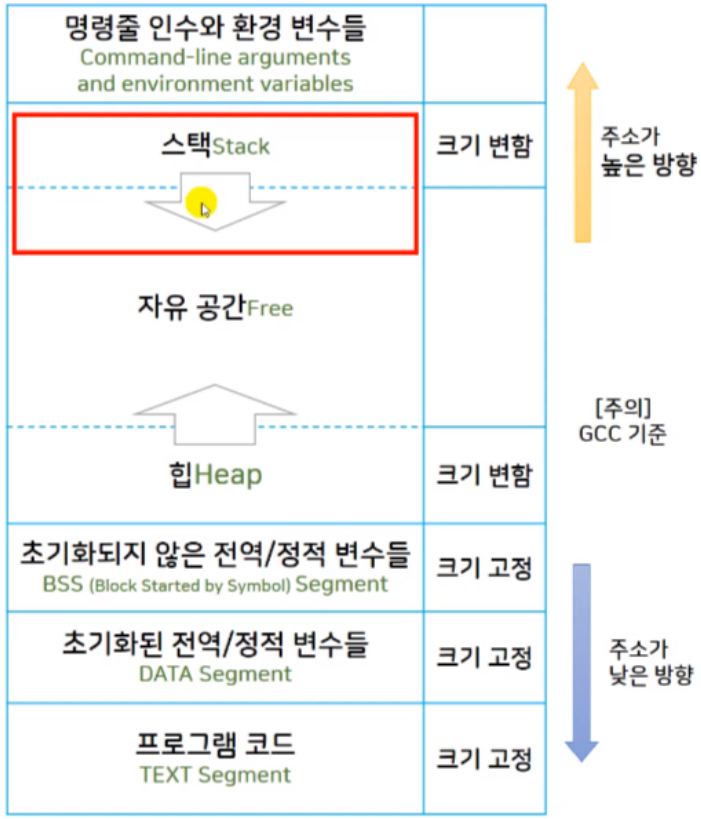
```c
#include <stdio.h>

#define MAX 1000

int main()
{
    // 2. Use local in main()
    int l_arr[MAX] = { 0, };

/*
    Use l_arr
    ...
    Do NOT use l_arr
    ...
    Use l_arr
    ...
*/

    return 0;
}
```



- 지역 변수들 - 단, main()에 있을 경우 계속 메모리에 머물러 있는 것처럼 보여짐
- 스택을 이용해 메모리를 효율적으로 사용 가능
- 물론, 단점도 존재

프로그램의 일부에서 큰 메모리가 필요한 경우

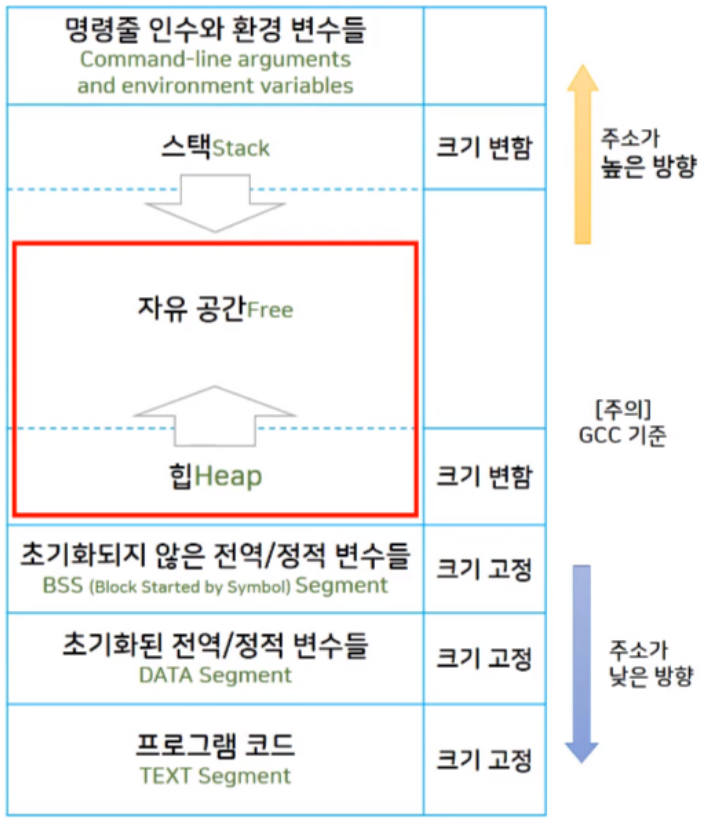
```c
#include <stdio.h>

#define MAX 1000

// 3. Use local in smaller function
void func()
{
    int l_arr[MAX] = { 0, };
}

int main()
{
/*
    Call func()
    ...
    Call func()
    ...
*/
    return 0;
}
```



- 스택에 선언된 지역변수들은 속도가 빠름

필요한 메모리의 크기를 미리 알 수 없을 경우

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n = 0;

    // n from files, internet, scanf, etc.

    char* arr = (char*)malloc(sizeof(char) * n);

    // ...

    free(arr);

    return 0;
}
```



- 가상 주소 공간을 사용하여 스택과 힙이 충돌하는 것을 방지
- 단점은 주소 배정 시 운영체제를 거쳐야하고, 이로 인해 속도가 느려진다.