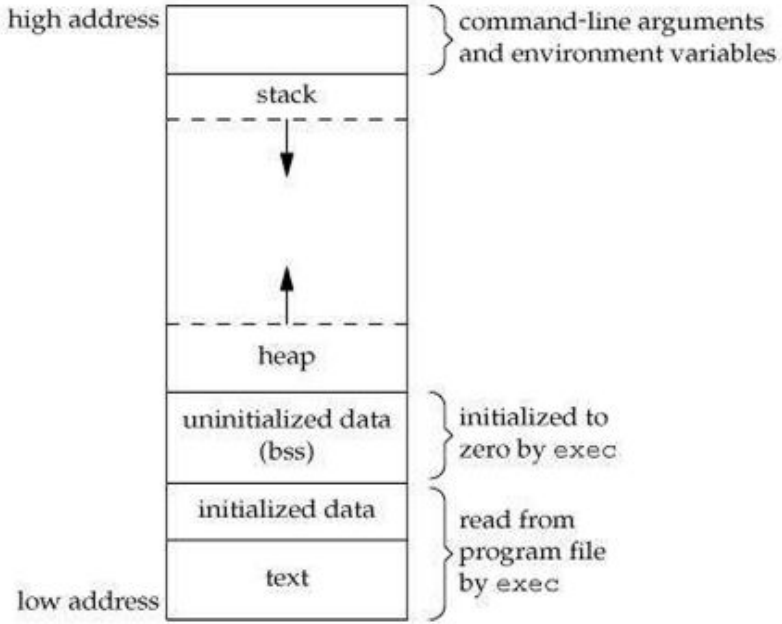
Recitation 3

TA Hanxiong Chen
hc691@rutgers.edu
Hill 206

Memory Layout

- Text or Code Segment
  a. Contains binary code of the compiled program
- Initialized Data Segment
  a. global , static, constant and external variables (declared with extern keyword)
  b. Variables are initialized
- Uninitialized Data Segment (bss)
  a. Global, static variables
  b. Initialize to 0 (even do not have explicit init)
- Heap
  a. For dynamic memory allocation
  b. Growth from low to high
- Stack
  a. Stack frame



Memory Laout Examples

Initialized Data Segment

```
#include <stdio.h>

char c[]="rishabh tripathi";      /*  global variable stored in
Initialized Data Segment in read-write area*/
const char s[]="HackerEarth";     /* global variable stored in
Initialized Data Segment in read-only area*/

int main()
{
    static int i=11;            /* static variable stored in
Initialized Data Segment*/
    return 0;
}
```

Memory Layout Examples (cont.)

Uninitialized Data Segment (bss.)

```c
#include <stdio.h>

char c;                 /* Uninitialized variable stored in bss*/

int main()
{
    static int i;      /* Uninitialized static variable stored in bss
*/
    return 0;
}
```

Memory Layout Examples (cont.)

Heap

```c
#include <stdio.h>
int main()
{
    char *p=(char*)malloc(sizeof(char));     /* memory allocating in
heap segment */
    return 0;
}
```

Metadata

- What is metadata?
- Why we need that?
- Where is it?

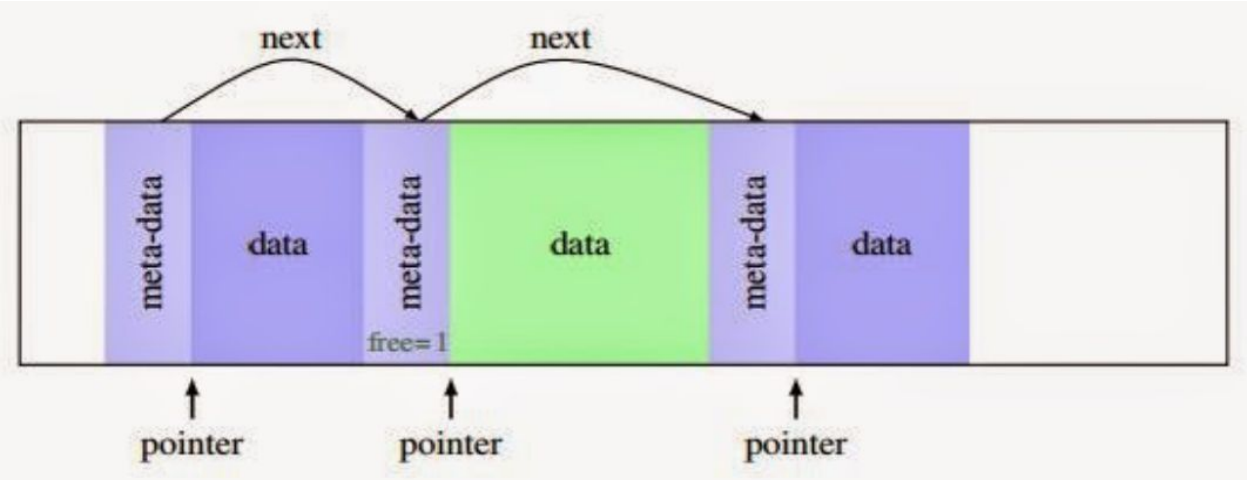
Think about how does free() function work.

Metadata

Example

```
struct metadata {

        size_t size;
        int isFree;
        struct block *next;
        ...
};
```



Question: Can we ignore the “next” pointer?

Memory placement strategy

REMIND ME:

First Fit

Best Fit

Worst Fit

Valgrind

- Valgrind is a programming tool for memory debugging, memory leak detection
- It is already installed on ilab
- To launch valgrind by running:
  - valgrind ./xxx
  - You can add --tool to use tools: --leak-check=yes / --leak-check=full
  - Use valgrind --help to get help
- It will run your program in their virtual environment. Your program will run very slow under this environment.
- Go to http://valgrind.org for more info and guidance.

HW3

1. What are the differences between strlen and sizeof a string in C? Why?
2. Write the function: replace(char string[], char from[], char to[])

   Which finds the string from in the string string and replaces it with the string to. You may assume that from and to are the same length. For example, the code:

   ```
   Char string = “recieve”;
   replace(string, “ie”, “ei”);
   ```

   Should change string to “receive”.

HW3 (cont.)

3. Write a short program to read two lines of text, and concatenate them using strcat. Since strcat concatenates in-place, you will have to make sure you have enough memory to hold the concatenated copy. For now, use a char array which is twice as big as either of the arrays you use for reading the two lines. Use strcpy to copy the first string to the destination array, and strcat to append the second one.

Sample code is uploaded on sakai.

Others

Don’t trust your calculation. Use sizeof() to get the size of a type.

Eg1:

```
struct A {
        char a;
        char b;
        int c;
        char d;
};
```

The sizeof(struct A) is 12.

The compiler will padding extra space to separate different types.

Eg2:

```
struct B {
        char a;
        char b;
        char c;
        int d;
};
```

The sizeof(struct B) is 8.

Others

Why this happens in our experiment?

```
char *s = “Hello”;
s[0] = ‘F’;
```
→ segmentation fault

```
char s[] = “Hello”;
s[0] = ‘F’;
```
→ string s becomes “Fello”

This is because the string “Hello” is in some read-only memory area. In the first example, you give this read-only memory address to the pointer s. That’s the reason why when you try to modify the content would get error.

In the second example, the string “Hello” is also in a read-only memory area. But s[] this time is a char array on stack. This time “char s[] = “Hello”; ” is to copy the string “Hello” to a new memory space on stack. Then if you do “s[0] = ‘F’;” is to modify the copied string on stack instead of the original string “Hello”. That’s the reason why we would not get error.

So we know that “Hello” is an anonymous const char array. We cannot modify the literal directly. We can do as the second example, or we have to do malloc and then do memcpy to copy the string to heap.

Reference

1. https://www.hackerearth.com/practice/notes/memory-layout-of-c-program/
2. http://tharikasblogs.blogspot.com/p/how-to-write-your-own-malloc-and-free.html
3. https://stackoverflow.com/questions/1704407/what-is-the-difference-between-char-s-and-char-s