Memory API (Chapter 14)

shinma

This Chapter

This chapter is a short chapter which explains the different types of memory APIs in C.

Application program interface (API) is a set of routines, protocols, and tools for building software applications.

- API specifies how software components should interact.

Types of Memory

Stack memory (sometimes known as automatic memory)

Heap memory

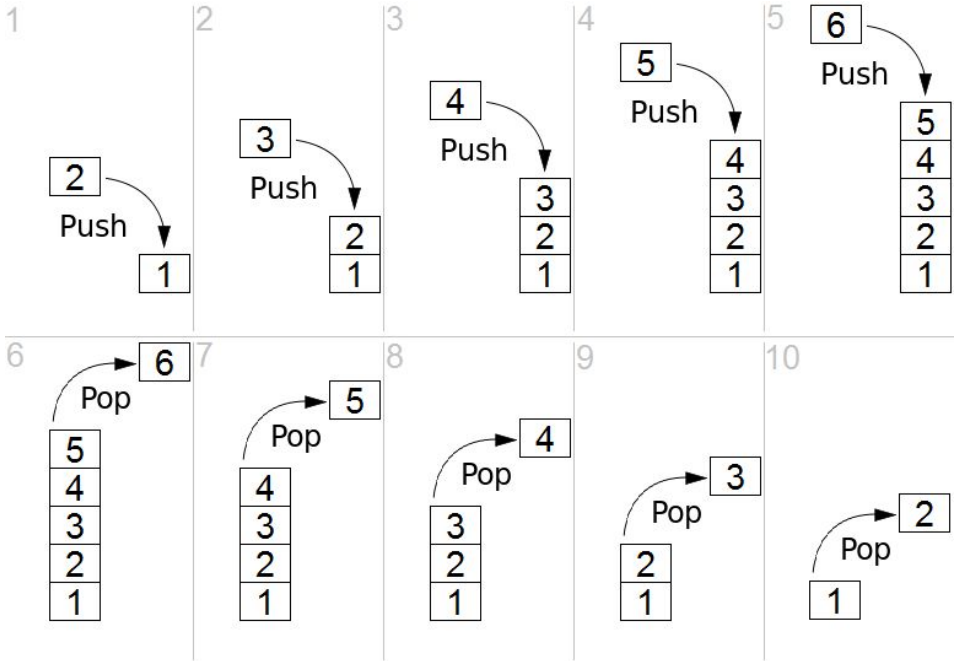
C dynamic memory allocation

- Malloc call
- Free call

Stack memory

Stack is an abstract data type = **Last in first out.**

- Usually has a pop and push feature. (adding and subtracting)



Stack memory

In C programming, stack-based memory allocation

- When a function executes, it may add some of its state data to the top of the stack; when the function exits it is responsible for removing that data from the stack
  - When calling a function, your compiler will reserve space for your function to run.
- Another name being **Automatic memory** as the compiler does all of this without your consent.
- Remember. Information on stack **disappears** after the program finishes running!

```
void func() {
    int x; // declares an integer on the stack
    ...
```

Heap memory

Heap is for dynamic memory allocation

- Heap memory is the opposite of stack memory as it depends on the program processing on stored data.

A **heap** is an area or folder in the computer's main storage (memory) where a program is able to process data in some variable amount.

Heap data is writable and managed by the programmer.

Heap memory

This code has both stack and heap memory called inside.

To explain the code:

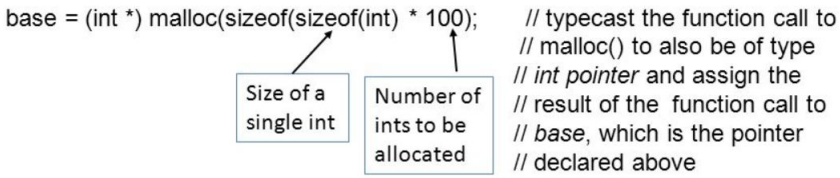
- The function makes the compiler reserves memory for an integer **BUT**
- When the program calls malloc(), it requests space for an integer on the heap
- It goes to the address of the integer and runs of successfully found. If the address is failed to be found, the code will not run.

```
void func() {
    int *x = (int *) malloc(sizeof(int));
    ...
}
```

The malloc() Call

The function malloc is used to preserve a certain amount of memory when running a program.

The malloc function will request a block of memory from the heap and the operating system will reserve the requested amount of memory.



Using The malloc() Call (Chapter 14 examples)

First off, the header file stdlib.h is going to be used malloc.

- But this writing is unnecessary as in C library, which all C programs link with by default.

The single parameter malloc() takes is of type size t which simply describes how many bytes you need.

However, most programmers do not type in a number here directly (such as 10) but instead a malloc.

```
#include <stdlib.h>
...
void *malloc(size_t size);
```

The free() Call

When the amount of memory is not needed anymore, you must return it to the operating system by calling the function free. (Free ())

```
if(base -- NULL)
{
     fprintf(stderr, "malloc failed.  Exitin\n");
     exit (1);
     . . .
     // process  //
     . . .

    free(base) ;   // remember to always free any dynamically allocated memory
```

Other calls (not mentioned in detail in chapter 14)

The calloc() function

The function calloc() will allocate a block of memory for an array. All of these elements are size bytes long. During the initialization all bits are set to zero.

The realloc() function

The function realloc() reallocates a memory block with a specific new size. If you call realloc() the size of the memory block pointed to by the pointer is changed to the given size in bytes. This way you are able to expand and reduce the amount of memory you want to use.

Thank you