Pointer More

王慧妍
why@nju.edu.cn

南京大学 计算机科学与技术系 计算机软件研究所

回顾上周

- 指针含义
  - *
  - &

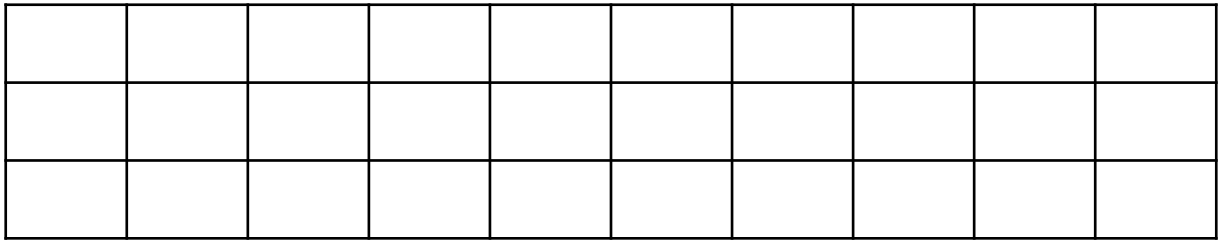
```
#include<stdio.h>
#define N_VALUES 5

int main(){
    float array[N_VALUES];
    float *vp;
    for(vp = &array[0]; vp < &array[N_VALUES];)
        *vp++ = 0;
    for(vp = &array[N_VALUES]; vp > &array[0];)
        *--vp = 0;
      //may UB

      for(vp = &array[N_VALUES-1]; vp >= &array[0]; vp --)
        *vp = 0;
}
```

高级指针

- 再撕：二维数组+指针
  - int matrix[3][10];
    - matrix
    - matrix+1
    - *(matrix + 1)
    - *(matrix+1)+5
    - *(*(matrix+1)+5)



动态内存分配

动态数组

- 回顾VLA：可变长数组int array[n]
  - 不推荐

- malloc和free
  - C函数库提供malloc和free，分别用于执行动态内存的分配与释放
  - 申请后需要释放

  - int *a = NULL;
  - a = (int *)malloc(n*sizeof(int));
  - free(a);

malloc

- #include<stdlib.h>
- void *malloc(size_t _Size);
- 向malloc申请的空间的大小是以字节为单位的
- 返回类型默认是void *

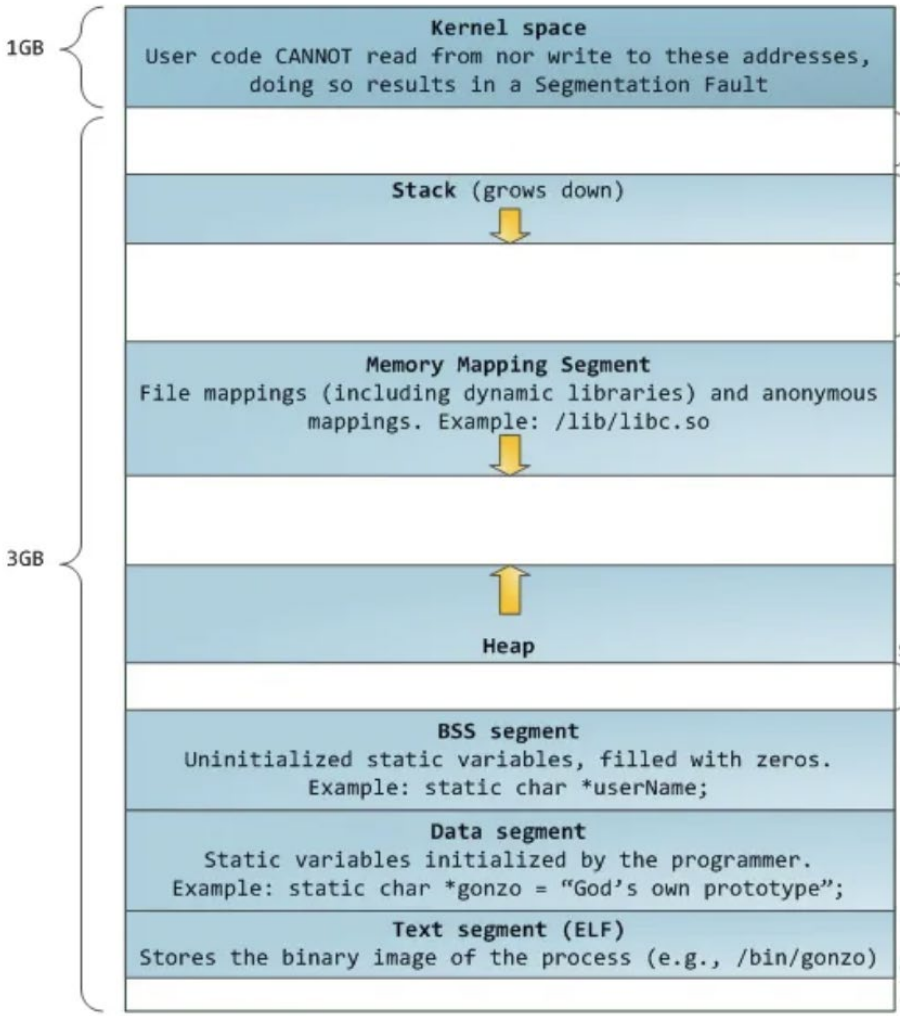
- 样例
  - int *array = (int *) malloc(len*sizeof(int))
  - int *array = malloc(len*sizeof(int))
  - int *array = malloc(len*sizeof(*array))

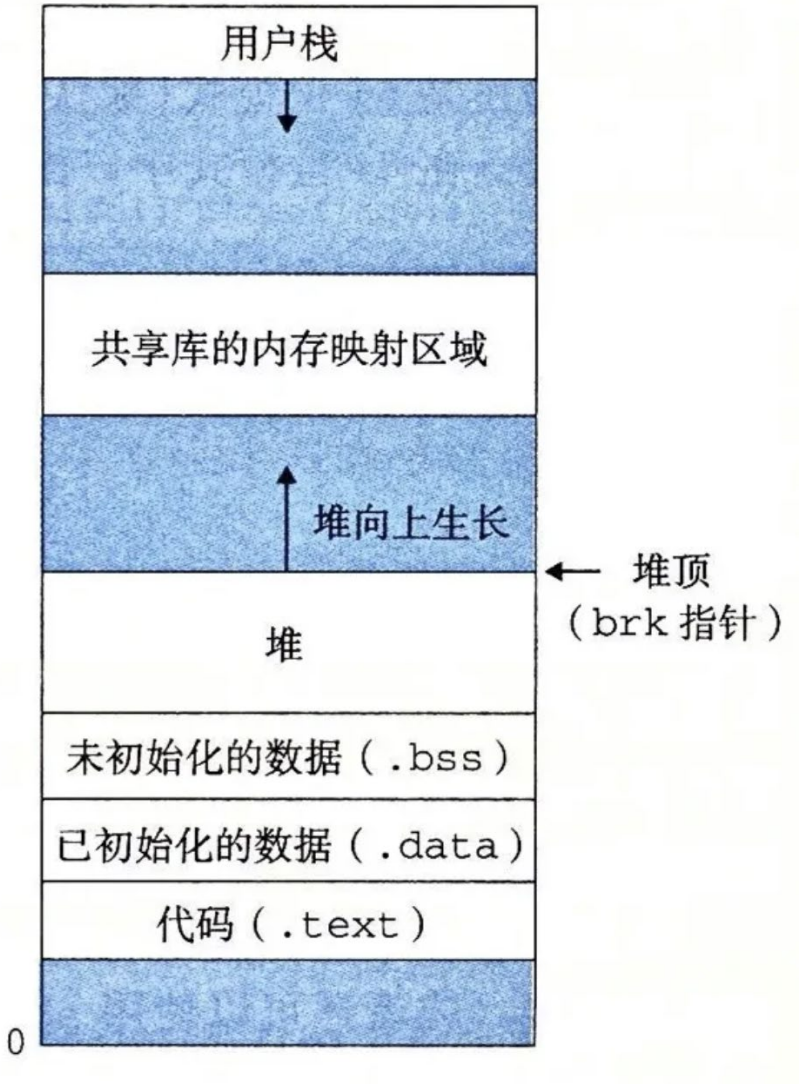
- 警惕：分配失败返回NULL

free

- #include<stdlib.h>
- void free(void *pointer);
- free(a):释放指针指向内存，指针变量依然存在（野指针）
  - 释放后a = NULL

- 需要释放之前动态申请的内存，一对一配对使用
  - 内存泄漏memory leak
  - 出来混，迟早要还的
  - malloc_space.c


1GB
3GB
Kernel space
User code CANNOT read from nor write to these addresses, doing so results in a Segmentation Fault
Stack (grows down)
Memory Mapping Segment
File mappings (including dynamic libraries) and anonymous mappings. Example: /lib/libc.so
Heap
BSS segment
Uninitialized static variables, filled with zeros. Example: static char *userName;
Data segment
Static variables initialized by the programmer. Example: static char *gonzo = "God's own prototype";
Text segment (ELF)
Stores the binary image of the process (e.g., /bin/gonzo)


用户栈
共享库的内存映射区域
堆向上生长
堆顶
（brk 指针）
堆
未初始化的数据（.bss）
已初始化的数据（.data）
代码（.text）
0

动态内存分配的常见错误

- 申请了没有free
- 对NULL指针进行解引用
- 对分配的内存越界操作
- 释放并非动态分配的内存（段错误）
- 试图释放动态分配的内存的部分
- 释放后依旧试图继续使用

一些其他memory allocation函数

C | Dynamic memory management

Dynamic memory management

Functions

Defined in header <stdlib.h>

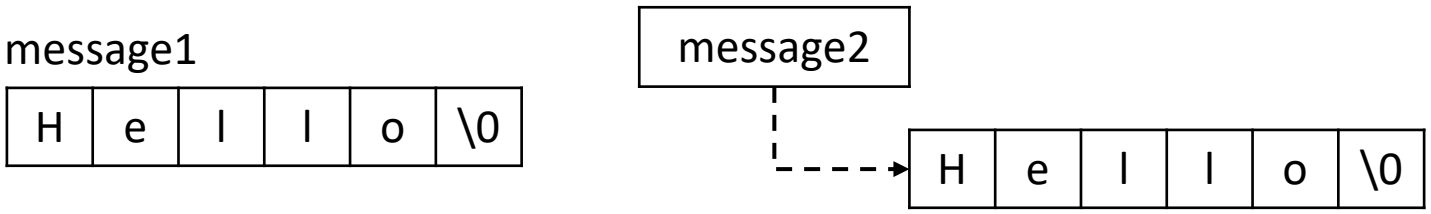
malloc	allocates memory (function)
calloc	allocates and zeroes memory (function)
realloc	expands previously allocated memory block (function)
free	deallocates previously allocated memory (function)
aligned_alloc (C11)	allocates aligned memory (function)

字符串和字符数组

字符串指针和字符数组

- `char msg1[] = {'H', 'e', 'l', 'l', 'o', '\0'};`
- `char msg1[] = "Hello";`

- `char *msg2 = "Hello";`



一些例子string.h

- strlen.c

- strcpy.c

string.h

- 常见的字符串函数
  - 不受限制的字符串函数
    - `size_t strlen (char const *string);`
    - `char *strcpy (char *dst, char const *src);`
    - `char *strcat (char *dst, char const *src);`
    - `int strcmp (char const *s1, char const *s2);`
  - 长度受限的字符串函数
    - `char *strncpy (char *dst, char const *src, size_t len);`
    - `char *strncat (char *dst, char const *src, size_t len);`
    - `int strncmp (char const *s1, char const *s2, size_t len);`

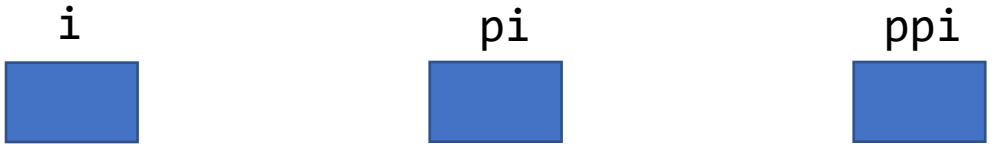
string.h

- 常见的字符串函数
  - 查找字符或子串函数
    - `char *strchr(char const *str, int ch);`
    - `char *strrchr(char const *str, int ch);`
    - `char *strpbrk(char const *str, char const *group);`
    - `char *strstr(char const *s1, char const *s2);`
  - 查找计数
    - `size_t *strspn(char const *str, char const *group);`
    - `size_t *strcspn(char const *str, char const *group);`
  - 查找标记
    - `char *strtok(char *str, char const *sep);`
      - strtok.c

高级指针

指针plus

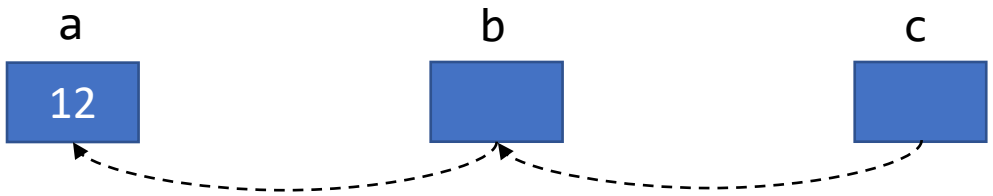
- 例子
  - int i;
  - int *pi;
  - int **ppi;




```
    printf("%d\n", ppi);
    printf("%x\n", &ppi);
    printf("%p\n", &ppi);
    *ppi = 5;
```

指针plus

- 例子
  - `int a = 12;`
  - `int *b = &a;`
  - `c = &b;` //c是什么类型?




```
int i;
int *pi;
int **ppi;
```

进一步理解*

- C Operator Precedence - cppreference.com
  - `int *p, q;`
  - `*p++;`
  - `++*p;`

  - `int *p[5];`
  - `int *f();`

  - `int f()[];`
  - `int f[]();`

指针数组排序

- 指针数组 vs 数组指针
  - `int *p1[10];`
  - `int *(p2[10]);`
  - `int (*p3)[10];`

指针的指针的另外用途

- 传递命令行参数
  - echo.c
  - `./echo.c hello world`

```
int
main(int argc, char **argv){

}
```

  - `./echo.c hello \t world –e`

End

- Keep coding!