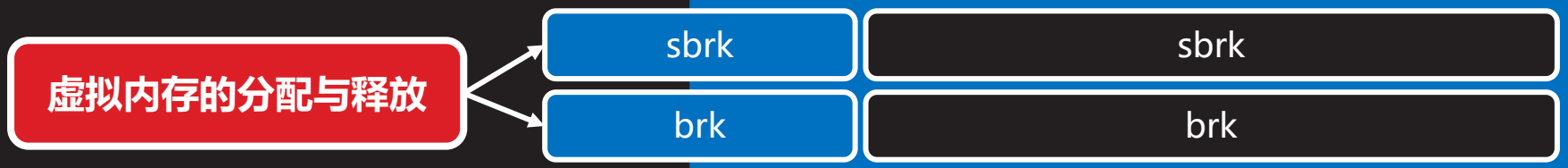
Technology Tarena 达内科技

Unix系统高级编程

内存的分配与释放 Unit06

虚拟内存的分配与释放



sbrk

sbrk

- 以相对方式分配和释放虚拟内存

```
#include <unistd.h>

void* sbrk (intptr_t increment);

        成功返回上次调用sbrk/brk后的堆尾指针，失败返回-1
```

  - *increment*：虚拟内存增量(以字节为单位)
    - >0 - 分配虚拟内存
    - <0 - 释放虚拟内存
    - =0 - 当前堆尾指针

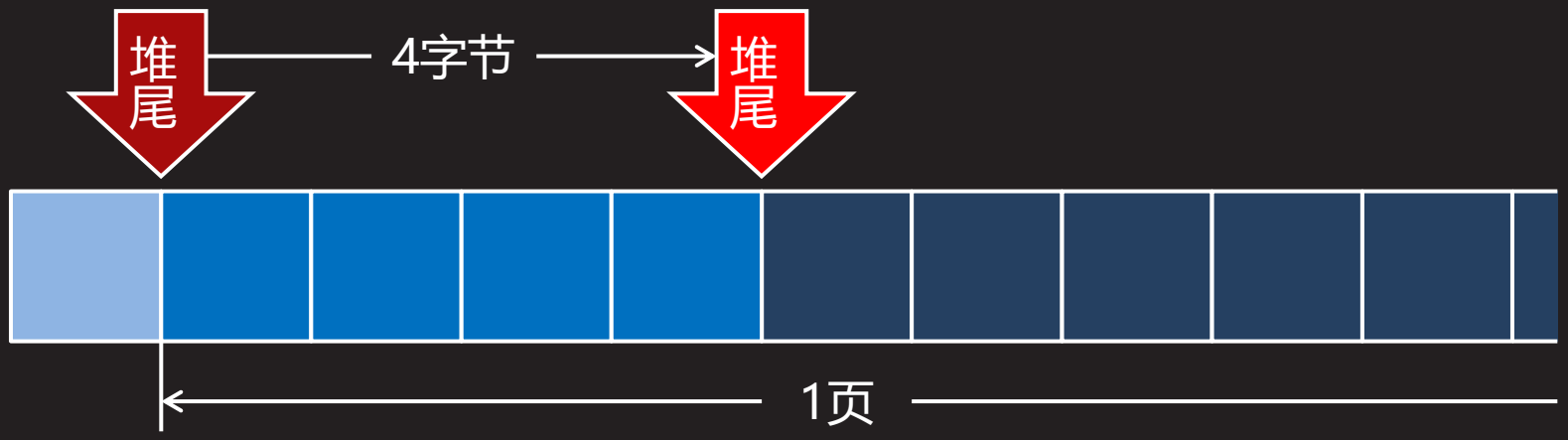
知识讲解

sbrk（续1）

- 系统内部维护一个指针，指向当前堆尾，即堆区最后一个字节的下一个位置，sbrk函数根据增量参数调整该指针的位置，同时返回该指针在调整前的位置，其间若发现内存页耗尽或空闲，则自动追加或取消内存页的映射

知识讲解

```
void* p = sbrk (4);          p = sbrk (0);
```



sbrk（续2）

- 例如

```
— p1 = sbrk ( 4); // BBBB ^--- ---- ---- -
  p2 = sbrk ( 4); // BBBB BBBB ^--- ---- -
  p3 = sbrk ( 4); // BBBB BBBB BBBB ^--- -
  p4 = sbrk ( 4); // BBBB BBBB BBBB BBBB ^
  p5 = sbrk ( 0); // BBBB BBBB BBBB BBBB ^
  p6 = sbrk (-8); // BBBB BBBB ^--- ---- _
  p7 = sbrk (-8); // ^--- ---- ---- ---- -
  // B: 已分配的字节
  // -: 未分配的字节
  // _: 函数返回指针
  // ^: 当前堆尾指针
```

知识讲解

Technology Tarena 达内科技

sbrk

【参见：sbrk.c】

课堂练习

- sbrk

Technology Tarena 达内科技

brk

brk

- 以绝对方式分配和释放虚拟内存

```
#include <unistd.h>

int brk (void* end_data_segment);
```

成功返回0，失败返回-1

  - *end_data_segment*：堆尾指针的新位置
    - >堆尾指针的原位置：分配虚拟内存
    - <堆尾指针的原位置：释放虚拟内存
    - =堆尾指针的原位置：什么也没有做

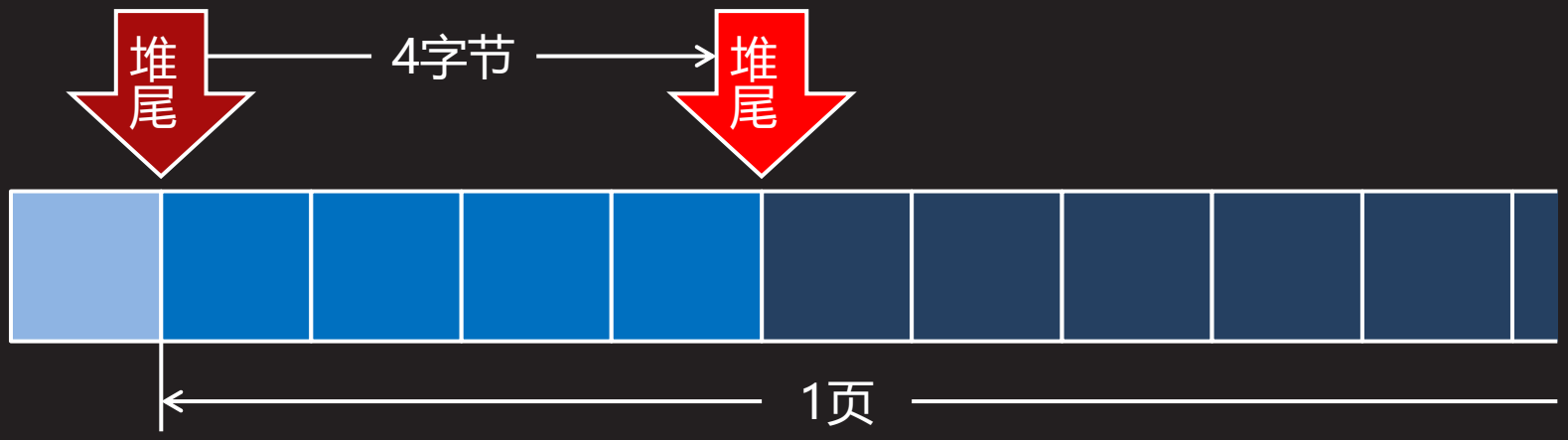
知识讲解

brk（续1）

- 系统内部维护一个指针，指向当前堆尾，即堆区最后一个字节的下一个位置，brk函数根据指针参数设置该指针的位置，其间若发现内存页耗尽或空闲，则自动追加或取消内存页的映射

知识讲解

```
void* p = sbrk (0);   brk (p+4);
```



brk（续2）

- 例如

```
— p1 = sbrk (0);      // ^--- ---- ---- ---- -
  brk (p2 = p1 + 4); // BBBB ^--- ---- ---- -
  brk (p3 = p2 + 4); // BBBB BBBB ^--- ---- -
  brk (p4 = p3 + 4); // BBBB BBBB BBBB ^--- -
  brk (p5 = p4 + 4); // BBBB BBBB BBBB BBBB ^
  brk (p3);          // BBBB BBBB ^--- ---- -
  brk (p1);          // ^--- ---- ---- ---- -
  // B：已分配的字节
  // -：未分配的字节
  // _：函数返回指针
  // ^：当前堆尾指针
```

brk（续3）

- sbrk和brk
  - 事实上，sbrk和brk不过是移动堆尾指针的两种不同方法，移动过程中还要兼顾虚拟内存和物理内存之间映射关系的建立和解除(以页为单位)
  - 用sbrk分配内存比较方便，用多少内存就传多少增量参数，同时返回指向新分配内存区域的指针，但用sbrk做一次性内存释放比较麻烦，因为必须将所有的既往增量进行累加
  - 用brk释放内存比较方便，只需将堆尾指针设回到一开始的位置即可一次性释放掉之前分多次分配的内存，但用brk分配内存比较麻烦，因为必须根据所需要的内存大小计算出堆尾指针的绝对位置
  - 用sbrk分多次分配适量内存，最后用brk一次性整体释放

brk

【参见：brk.c】

课堂练习

- brk

内存映射的建立与解除

Technology Tarena 达内科技

建立内存映射

建立内存映射

- 建立虚拟内存到物理内存或文件的映射

```
#include <sys/mman.h>

void* mmap (void* start, size_t length, int prot,
    int flags, int fd, off_t offset);
```

成功返回映射区内存起始地址，失败返回MAP_FAILED(-1)

  - ***start***：映射区内存起始地址，NULL系统自动选定后返回
  - ***length***：映射区字节长度，自动按页(4K)圆整

知识讲解

建立内存映射（续1）

- 创建虚拟内存到物理内存或文件的映射
  - *prot*：映射区访问权限，可取以下值

    `PROT_READ` - 映射区可读

    `PROT_WRITE` - 映射区可写

    `PROT_EXEC` - 映射区可执行

    `PROT_NONE` - 映射区不可访问

知识讲解

建立内存映射（续2）

- 创建虚拟内存到物理内存或文件的映射
  - ***flags***：映射标志，可取以下值

    MAP_ANONYMOUS - 匿名映射，将虚拟内存映射到物理内存而非文件，忽略***fd***和***offset***参数

    MAP_PRIVATE - 对映射区的写操作只反映到缓冲区中，并不会真正写入文件

    MAP_SHARED - 对映射区的写操作直接反映到文件中

    MAP_DENYWRITE - 拒绝其它对文件的写操作

    MAP_FIXED - 若在***start***上无法创建映射，则失败(无此标志系统会自动调整)

    MAP_LOCKED - 锁定映射区，保证其不被换出

知识讲解

建立内存映射（续3）

- 创建虚拟内存到物理内存或文件的映射
  - ***fd***：文件描述符
  - ***offset***：文件偏移量，自动按页(4K)对齐
- 例如
  - char* p = (char*)mmap (NULL, 8192,
    PROT_READ | PROT_WRITE,
    MAP_ANONYMOUS | MAP_PRIVATE, 0, 0);
    if (p == MAP_FAILED) {
    perror ("mmap");
    exit (EXIT_FAILURE); }
  - strcpy (p, "Hello, Memory !");
    printf ("%s\n", p);

知识讲解

Technology Tarena 达内科技

解除内存映射

解除内存映射

- 解除虚拟内存到物理内存或文件的映射

```
#include <sys/mman.h>

int munmap (void* start, size_t length);

                                成功返回0，失败返回-1
```

  - ***start***：映射区内存起始地址，必须是页的首地址
  - ***length***：映射区字节长度，自动按页(4K)圆整

知识讲解

解除内存映射（续1）

- 例如
  - if (munmap (p, 4096) == -1) {
    perror ("munmap");
    exit (EXIT_FAILURE);
    }
    strcpy (p += 4096, "Hello, Memory !");
    printf ("%s\n", p);
    if (munmap (p, 4096) == -1) {
    perror ("munmap");
    exit (EXIT_FAILURE);
    }
- munmap允许对映射区的一部分解映射，但必须按页

知识讲解

解除内存映射（续2）

- mmap/munmap底层不维护任何东西，只是返回一个首地址，所分配内存位于堆中
- brk/sbrk底层维护一个指针，记录所分配的内存结尾，所分配内存位于堆中，底层调用mmap/munmap
- malloc底层维护一个线性链表和必要的控制信息，不可越界访问，所分配内存位于堆中，底层调用brk/sbrk
- 每个进程都有4G的虚拟内存空间，虚拟内存地址只是一个数字，在与实际物理内存建立映射之前是不能访问的
- 所谓内存分配与释放，其本质就是建立或解除从虚拟内存到物理内存的映射，并在底层维护不同形式的数据结构，以把虚拟内存的占用与空闲情况记录下来

知识讲解

内存映射的建立与解除

【参见：mmap.c】

课堂练习

- 内存映射的建立与解除

Technology Tarena 达内科技

总结和答疑