dreamcatcher-zp
博览而约取，厚积而薄发

malloc 底层实现及原理

阅读目录

摘要：偶尔看到面试题会问到 malloc 的底层原理，今天就来记录一下，毕竟学习要“知其所以然”，这样才会胸有成竹。

注：下面分析均是基于 linux 环境下的 malloc 实现。步骤是：先总结结论，再逐步展开

回到顶部

结论

1）当开辟的空间小于 128K 时，调用 brk（）函数，malloc 的底层实现是系统调用函数 brk（），其主要移动指针 _enddata(此时的 _enddata 指的是 Linux 地址空间中堆段的末尾地址，不是数据段的末尾地址)

2）当开辟的空间大于 128K 时，mmap（）系统调用函数来在虚拟地址空间中（堆和栈中间，称为“文件映射区域”的地方）找一块空间来开辟。

回到顶部

具体内容

当一个进程发生缺页中断的时候，进程会陷入核心态，执行以下操作：

1）检查要访问的虚拟地址是否合法

2）查找/分配一个物理页

3）填充物理页内容（读取磁盘，或者直接置0，或者什么都不做）

4）建立映射关系（虚拟地址到物理地址的映射关系）

5）重复执行发生缺页中断的那条指令

如果第3布，需要读取磁盘，那么这次缺页就是 majfit(major fault：大错误),否则就是 minflt(minor fault：小错误)

回到顶部

内存分配的原理

从操作系统角度看，进程分配内存有两种方式，分别由两个系统调用完成：brk 和 mmap (不考虑共享内存)

1）brk 是将数据段（.data）的最高地址指针 _edata 往高地址推

2）mmap 是在进程的虚拟地址空间中（堆和栈中间，称为“文件映射区域”的地方）找一块空闲的虚拟内存。

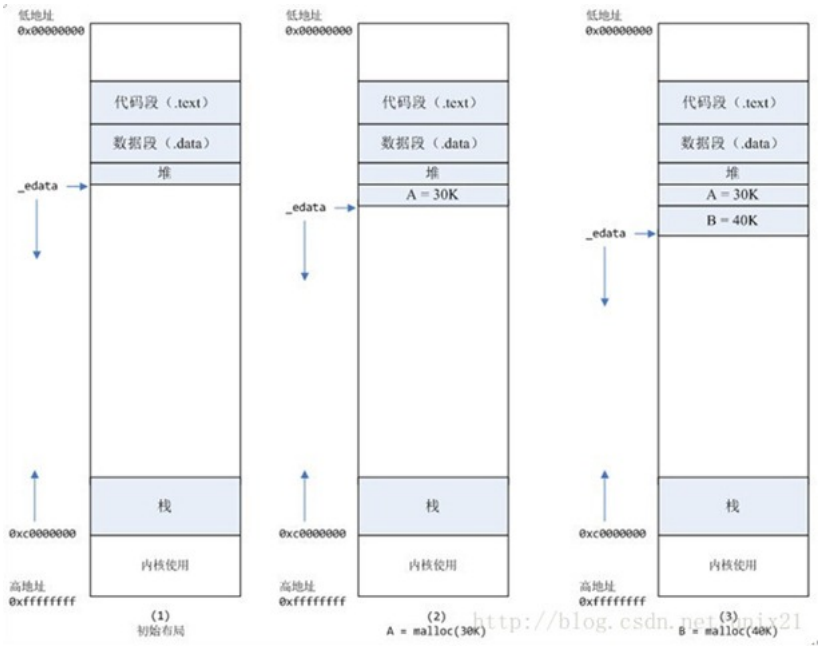
这两种方式分配的都是虚拟内存，没有分配物理内存。在第一次访问已分配的虚拟地址空间的时候，发生缺页中断，操作系统负责分配物理内存，然后建立虚拟内存和物理内存之间的映射关系。

回到顶部

具体分配过程

情况一：malloc 小于 128K 的内存，使用 brk 分配

将_edata往高地址推(只分配虚拟空间，不对应物理内存(因此没有初始化)，第一次读/写数据时，引起内核缺页中断，内核才分配对应的物理内存，然后虚拟地址空间建立映射关系)，如下图：



1，进程启动的时候，其（虚拟）内存空间的初始布局如图1所示

2，进程调用A=malloc(30K)以后，内存空间如图2：

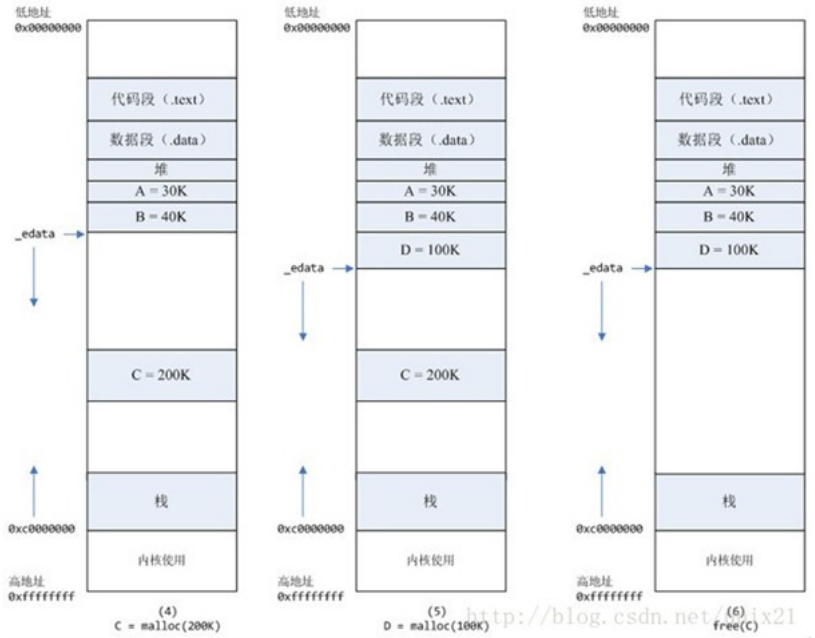
malloc函数会调用brk系统调用，将_edata指针往高地址推30K，就完成虚拟内存分配

你可能会问：难道这样就完成内存分配了？

事实是：_edata+30K只是完成虚拟地址的分配，A这块内存现在还是没有物理页与之对应的，等到进程第一次读写A这块内存的时候，发生缺页中断，这个时候，内核才分配A这块内存对应的物理页。**也就是说，如果用malloc分配了A这块内容，然后从来不访问它，那么，A对应的物理页是不会被分配的。**

3，进程调用B=malloc(40K)以后，内存空间如图3

情况二：malloc 大于 128K 的内存，使用 mmap 分配（munmap 释放）



4，进程调用C=malloc(200K)以后，内存空间如图4

默认情况下，**malloc函数分配内存，如果请求内存大于128K（可由M_MMAP_THRESHOLD选项调节），那就不是去推_edata指针了，而是利用mmap系统调用，从堆和栈的中间分配一块虚拟内存**

这样子做主要是因为:

brk分配的内存需要等到高地址内存释放以后才能释放（例如，在B释放之前，A是不可能释放的，因为只有一个_edata 指针，这就是内存碎片产生的原因，什么时候紧缩看下面），而mmap分配的内存可以单独释放。

当然，还有其它的好处，也有坏处，再具体下去，有兴趣的同学可以去看glibc里面malloc的代码了。

5，进程调用D=malloc(100K)以后，内存空间如图5

6，进程调用free(C)以后，C对应的虚拟内存和物理内存一起释放

"在B释放之前，A是不
一个_edata 指针，这
因。"这句话和下面的
理内存都没有释放，
针，如果往回推，那

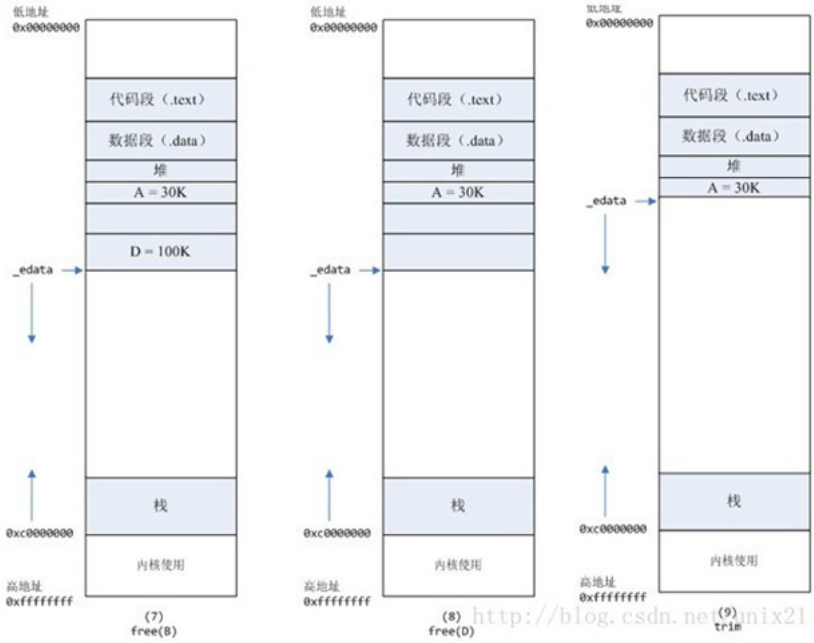
2. Re:dockerfile---
Unable to locate p
@ xh龙渊哈哈，一起

3. Re:dockerfile---
Unable to locate p
最近也在学阳哥的do

阅读排行榜
1. C++---类成员变
2. malloc 底层实现
3. error LNK2019
lass std::basic_os
d::char_traits<ch
4. docker---Docke
5. c++ 中的数字和

评论排行榜
1. dockerfile---apt
ble to locate packa
2. malloc 底层实现

推荐排行榜
1. malloc 底层实现
2. 如何在 main() 执
(4)
3. Error:stray '\24
4. dockerfile---apt
ble to locate packa
5. one_day_one_li



7，进程调用free(B)以后，如图7所示

B对应的虚拟内存和物理内存都没有释放，因为只有一个_edata指针，如果往回推，那么D这块内存怎么办呢？当然，B这块内存，是可以重用的，如果这个时候再来一个40K的请求，那么malloc很可能就把B这块内存返回回去了

8，进程调用free(D)以后，如图8所示

B和D连接起来，变成一块140K的空闲内存

9，默认情况下：

当最高地址空间的空闲内存超过128K（可由M_TRIM_THRESHOLD选项调节）时，执行内存紧缩操作（trim）。在上一个步骤free的时候，发现最高地址空闲内存超过128K，于是内存紧缩，变成图9所示

参考博客：
https://www.cnblogs.com/dongzhiquan/p/5621906.html

分类：c/c++

标签：malloc

爱笑的张飞
关注 - 6
粉丝 - 6
+加关注

4 0

« 上一篇： C++---拷贝构造函数和赋值构造函数
» 下一篇： MySQL---MVCC机制

posted on 2019-05-04 16:57 爱笑的张飞 阅读(4607) 评论(1) 编辑 收藏

评论

#1楼 2020-08-17 20:08 打不死的黄妖精

"在B释放之前，A是不可能释放的，因为只有一个_edata 指针，这就是内存碎片产生的原因。"这句话和下面的"B对应的虚拟内存和物理内存都没有释放，因为只有一个_edata指针，如果往回推，那么D这块内存怎么办呢？"感觉有些矛盾啊，既然A不能先于B释放，那么B为何又能先于D释放呢?

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。

相关博文:
· malloc
· malloc，realloc和calloc的区别
· vma
· malloc/free与new/delete
· [C]链接和生存周期
» 更多推荐...

最新 IT 新闻:
· 为什么大家都讨厌「VIP 通货膨胀」?
· 苹果若真的封杀Epic，就是和整个游戏界作对?
· 在豆瓣上有4万人 每天都假装活在1980-2000年
· 不含算法的TikTok值得买吗？专家：买家难以复刻其魔力
· 理想汽车回应再次"断轴"：以51km/h时速撞到水泥花坛 已澄清
» 更多新闻...

历史上的今天:
2019-05-04 C++---拷贝构造函数和赋值构造函数