

Lab4 实验报告

PB21000033 赵奕

一、实验目标

这次实验要为我们的 OS 内核实现内存管理，为用户提供 malloc/free 接口，并用我们写的 shell 进行测试

附加实验 (kmalloc, kfree): 也已实现。具体实现方式在**第三节**的问题 1 中有所展现

二、源代码说明

如何维护空闲链表

dPartition: 空闲块的块首用一个结构体来描述这个空闲块

```
typedef struct EMB {
    unsigned long size;
    union {
        unsigned long nextStart;    // if free: pointer to next block
        unsigned long userData;    // if allocated, belongs to user
    };
} EMB; // 共占 8 个字节
```

需要注意，我的实现中 size 包括了 EMB 块自身。考虑到在内存分配过程中 size 变量不应该被覆盖，因此每一个 EMB 块对应的实际可用存储空间大小为 $size - 4$

eFPartition: 这种方式一个很大的区别是，每个块的 size 固定，故空闲块首的结构体只需要保存 next_start 即可。需要注意：虽然这里没有写 union 体，但是本质上和 EMB 是一样的 (如果这个块被使用了，那么 next_start 会被 user_data 覆盖)

```
typedef struct EEB {
    unsigned long next_start;
} EEB; // 共占 4 个字节
```

其他链表实现细节类似，不过在处理上会简便很多。另外，eFP 的方式每个结构体所管辖的范围是一定的，因此在初始时整条链表就已经全部建立，后续分配空间的过程是从链表中删除节点的过程。

释放后合并空间

dPartition: 这次实现的链表是单向链表，因此处理释放空间没有那么简单，因为我们需要先找到 blockPre 和 blockNxt 即待释放空间前后的 EMB 块

找到后，我们选择先测试与后面的链接 (即是否能与后方空闲 EMB 块合并) 再测试与前面的链接 (是否能与前方空闲 EMB 块合并)

需要注意，如果 blockPre 不存在，可能需要修改相应 dpHandler 的相关属性 firstFreeStart

eFPartition: 同样需要找到前后的空闲 EEB 块。与 dPartition 简化之处在于，不需要考虑合并空闲块的操作，只需要简单插入节点到链表中就可以了。

三、问题回答（思考题）

1. malloc 接口的具体实现

首先简述 malloc/free 和 kmalloc/kfree 如何相关隔离。

在 `pMemInit.c` 里面，获取了可用的内存大小之后，我先将所有空闲空间作为一个整体调用 `dPartitionInit`，管理句柄保存在变量 `pMemHandler` 中。

接下来,我从 pMemHandler 中申请了两块空间,大小分别为 $\frac{pMemSize}{3}$ 和 $\frac{pMemSize}{2}$,前者用作 kernel 内存分配,后者用作 user 内存分配。

对这两块空间分别进行 `dPartitionInit` 并分别保存句柄于变量 `kMemHandler`, `uMemHandler` 中即可。

然后描述 malloc/free 接口的调用过程

malloc: (kmalloc 同理, 除了第一步中改用 kMemHandler 句柄) 1. malloc: 调用 dPartitionAlloc(uMemHandler, size) (使用 dPartition 管理空间, 空间管理句柄为 uMemHandler) 2. dPartitionAlloc: 调用 dPartitionAllocFirstFit(dp, size) (使用 first fit 分配算法申请空间)

free: (kfree 同理, 除了第一步中改用 kMemHandler 句柄) 1. free: 调用 dPartitionFree(uMemHandler, start) (使用 dPartition 管理空间, 空间管理句柄为 uMemHandler) 2. dPartitionFree: 调用 dPartitionFreeFirstFit(dp, start) (first fit 分配算法对应的回收空间函数)

2. 运行结果

基础指令 cmd 列出所有可执行命令, clear 命令清屏

```
Machine View
brealid@oslab4:~$ cmd
oslab4 bash, These shell commands are defined internally.
Type 'cmd' to see this list.

cmd
help
clear
whoami
echo
time
testMalloc1
testMalloc2
maxMallocSizeNow
testDP1
testDP2
testDP3
testDP4
testDP5
testDP6
brealid@oslab4:~$ clear
clear
brealid@oslab4:~$ cmd
cmd
oslab4 bash, These shell commands are defined internally.
Type 'cmd' to see this list.

cmd
help
clear
whoami
echo
time
testMalloc1
testMalloc2
maxMallocSizeNow
testDP1
testDP2
testDP3
testDP4
testDP5
testDP6
brealid@oslab4:~$
```

先后运行: maxMallocSizeNow, testMalloc1, testMalloc2, maxMallocSizeNow
可以发现空间正常由请释放赋值。

且可以由 `testMalloc2` 的运行结果（申请的地址）发现 `testMalloc1` 申请的空间有被正常释放。这一点也可以从两次 `maxMallocSizeNow` 中看出

```
Machine View
breald@oslab4:~$ maxMallocSizeNow
MAX_MALLOC_SIZE: 0x3f7d000 (with step = 0x1000);
breald@oslab4:~$ testMalloc1
We allocated 2 buffers.
BUF1(size=19, addr=0x2b5a37c) filled with 17(+): *****
BUF2(size=24, addr=0x2b5a394) filled with 22(#): *****
breald@oslab4:~$ testMalloc2
We allocated 2 buffers.
BUF1(size=9, addr=0x2b5a37c) filled with 9(+): *****
BUF2(size=19, addr=0x2b5a38c) filled with 19(.): *****
breald@oslab4:~$ maxMallocSizeNow
MAX_MALLOC_SIZE: 0x3f7d000 (with step = 0x1000);
breald@oslab4:~$

Unknown interrupt

breald@oslab4:~$ clear
clear
breald@oslab4:~$ maxMallocSizeNow
maxMallocSizeNow
MAX_MALLOC_SIZE: 0x3f7d000 (with step = 0x1000);
breald@oslab4:~$ testMalloc1
testMalloc1
We allocated 2 buffers.
BUF1(size=19, addr=0x2b5a37c) filled with 17(+): *****
BUF2(size=24, addr=0x2b5a394) filled with 22(#): *****
breald@oslab4:~$ tet
tet
Command 'tet' not found
breald@oslab4:~$ clear
clear
breald@oslab4:~$ maxMallocSizeNow
maxMallocSizeNow
MAX_MALLOC_SIZE: 0x3f7d000 (with step = 0x1000);
breald@oslab4:~$ testMalloc1
testMalloc1
We allocated 2 buffers.
BUF1(size=19, addr=0x2b5a37c) filled with 17(+): *****
BUF2(size=24, addr=0x2b5a394) filled with 22(#): *****
breald@oslab4:~$ testMalloc2
testMalloc2
We allocated 2 buffers.
BUF1(size=9, addr=0x2b5a37c) filled with 9(+): *****
BUF2(size=19, addr=0x2b5a38c) filled with 19(.): *****
breald@oslab4:~$ maxMallocSizeNow
maxMallocSizeNow
MAX_MALLOC_SIZE: 0x3f7d000 (with step = 0x1000);
breald@oslab4:~$
```

运行 testdP1, 运行结果符合预期

注意到, 虽然申请了 0x100 大小的空间, 但是由于 dPartition 占据了一定的空间, 因此仅有 0xf8 大小的空间可用, 故 malloc(0x100) 失败, 这是符合预期的

```
Machine View
testMalloc1
testMalloc2
maxMallocSizeNow
testdP1
testdP2
testdP3
testdP4
breald@oslab4:~$ testdP1
We had successfully malloc() a small memBlock (size=0x100, addr=0x2b5a37c);
It is initialized as a very small dPartition;
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
dPB(start=0x2b5a384, size=0xf8, nextStart=0x0)
Alloc a memBlock with size 0x10, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x20, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x40, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x80, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x100, failed!
Now, converse the sequence.
Alloc a memBlock with size 0x100, failed!
Alloc a memBlock with size 0x80, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x40, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x20, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x10, success(addr=0x2b5a388).....Released;
breald@oslab4:~$

Unknown interrupt

cmd
oslab4 bash, These shell commands are defined internally.
Type 'cmd' to see this list.

cmd
help
clear
whoamt
echo
time
testMalloc1
testMalloc2
maxMallocSizeNow
testdP1
testdP2
testdP3
testdP4
breald@oslab4:~$ testdP1
testdP1
We had successfully malloc() a small memBlock (size=0x100, addr=0x2b5a37c);
It is initialized as a very small dPartition;
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
dPB(start=0x2b5a384, size=0xf8, nextStart=0x0)
Alloc a memBlock with size 0x10, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x20, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x40, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x80, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x100, failed!
Now, converse the sequence.
Alloc a memBlock with size 0x100, failed!
Alloc a memBlock with size 0x80, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x40, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x20, success(addr=0x2b5a388).....Released;
Alloc a memBlock with size 0x10, success(addr=0x2b5a388).....Released;
breald@oslab4:~$
```

testdP2 的输出长度比较长, 查看不到的部分选择在串口中查看

testdP2 的过程正如输出所描绘, 以这样的形式展现:

- -> A:- -> A:B:- -> A:B:C:- -> -:B:C:- -> -:C:- -> -

输出符合这一情况

```
Machine View
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
Now, A:B:C:- ==> -:B:C:- ==> -:C:- ==> -
Alloc memblock A with size 0x10: success(addr=0x2b5a380)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a390)
EMB(start=0x2b5a390, size=0xe4, nextStart=0x0)
Alloc memblock B with size 0x20: success(addr=0x2b5a39c)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a3bc)
EMB(start=0x2b5a3bc, size=0xc0, nextStart=0x0)
Alloc memblock C with size 0x30: success(addr=0x2b5a3c0)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xdc, nextStart=0x0)
Now, release A.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xdc, nextStart=0x0)
Now, release B.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xe4, nextStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xc0, nextStart=0x0)
At last, release C.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
breall@goslab4:~$
Unknown interrupt
```

```
testMalloc1
testMalloc2
maxMallocSizeNow
testDP1
testDP2
testDP3
testFP
breall@goslab4:~$ testDP2
testDP2
We had successfully malloc() a small memBlock (size=0x100, addr=0x2b5a37c);
It is initialized as a very small dPartition:
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
Now, A:B:C:- ==> -:B:C:- ==> -:C:- ==> -
Alloc memblock A with size 0x10: success(addr=0x2b5a380)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a390)
EMB(start=0x2b5a390, size=0xe4, nextStart=0x0)
Alloc memblock B with size 0x20: success(addr=0x2b5a39c)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a3bc)
EMB(start=0x2b5a3bc, size=0xc0, nextStart=0x0)
Alloc memblock C with size 0x30: success(addr=0x2b5a3c0)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xdc, nextStart=0x0)
Now, release A.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xdc, nextStart=0x0)
Now, release B.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xe4, nextStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xc0, nextStart=0x0)
At last, release C.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
breall@goslab4:~$
Unknown interrupt
```

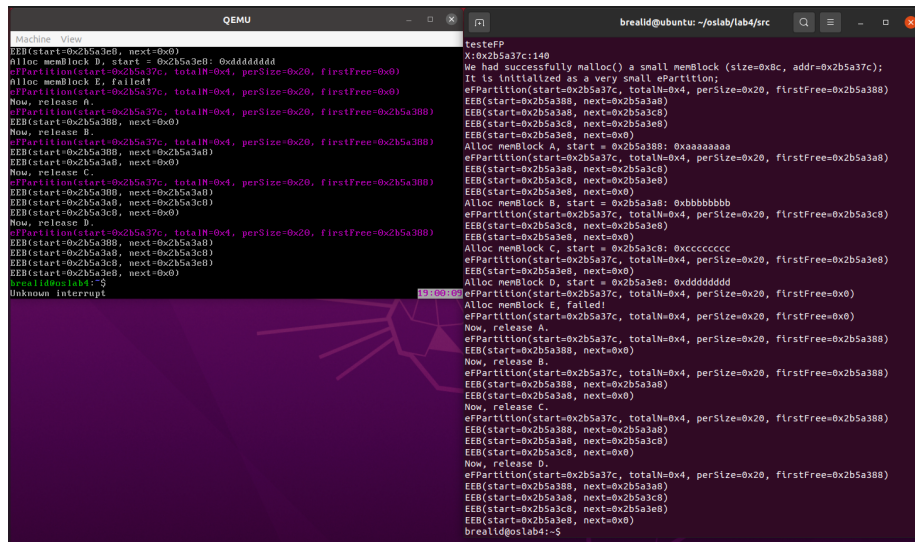
testDP3 将释放的过程反了过来，因此我们看不到多个 EMB 共存的情况，输出同样符合预期 testDP3 的过程：

- -> A:- -> A:B:- -> A:B:C:- -> A:B:- -> A:- -> -

```
Machine View
We had successfully malloc() a small memBlock (size=0x100, addr=0x2b5a37c);
It is initialized as a very small dPartition:
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
Now, A:B:C:- ==> -:B:C:- ==> -:C:- ==> -
Alloc memblock A with size 0x10: success(addr=0x2b5a380)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a390)
EMB(start=0x2b5a390, size=0xe4, nextStart=0x0)
Alloc memblock B with size 0x20: success(addr=0x2b5a39c)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a3bc)
EMB(start=0x2b5a3bc, size=0xc0, nextStart=0x0)
Alloc memblock C with size 0x30: success(addr=0x2b5a3c0)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xdc, nextStart=0x0)
At last, release C.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
Now, release B.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a390)
EMB(start=0x2b5a390, size=0xe4, nextStart=0x0)
Now, release A.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
breall@goslab4:~$
Unknown interrupt
```

```
EMB(start=0x2b5a384, size=0x14, nextStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xdc, nextStart=0x0)
Now, release B.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0x30, nextStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xdc, nextStart=0x0)
At last, release C.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
breall@goslab4:~$ testDP3
testDP3
We had successfully malloc() a small memBlock (size=0x100, addr=0x2b5a37c);
It is initialized as a very small dPartition:
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
Now, A:B:C:- ==> -:B:C:- ==> -:C:- ==> -
Alloc memblock A with size 0x10: success(addr=0x2b5a380)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a390)
EMB(start=0x2b5a390, size=0xe4, nextStart=0x0)
Alloc memblock B with size 0x20: success(addr=0x2b5a39c)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a3bc)
EMB(start=0x2b5a3bc, size=0xc0, nextStart=0x0)
Alloc memblock C with size 0x30: success(addr=0x2b5a3c0)!
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a3f0)
EMB(start=0x2b5a3f0, size=0xdc, nextStart=0x0)
At last, release C.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
Now, release B.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a390)
EMB(start=0x2b5a390, size=0xe4, nextStart=0x0)
Now, release A.
dPartition(start=0x2b5a37c, size=0x100, firstFreeStart=0x2b5a384)
EMB(start=0x2b5a384, size=0xf8, nextStart=0x0)
breall@goslab4:~$
```

testeFP 测试了 eFP 的接口。首先有申请的大小 $140 = 32 * 4 + 12$ ，其中 12 是 struct eFPPartition 句柄的大小，32 是对齐后的块大小
可以看到申请空间就是删除链表的过程，释放空间就是插入链表的过程
考虑到我们的 $n=4$ 故最终仅有 A,B,C,D 空间申请成功，E 申请失败
可以看到对空间的赋值操作等也都是成功的



```
Machine View
EBB(start=0x2b5a3e0, next=0x0)
Alloc memBlock B, start = 0x2b5a3e0: 0xdddddddd
ePartition(start=0x2b5a37c, totalN=0x4, perSize=0x20, firstFree=0x0)
Alloc memBlock E, failed!
ePartition(start=0x2b5a37c, totalN=0x4, perSize=0x20, firstFree=0x0)
Now, release A.
ePartition(start=0x2b5a37c, totalN=0x4, perSize=0x20, firstFree=0x2b5a300)
EBB(start=0x2b5a300, next=0x0)
Now, release B.
ePartition(start=0x2b5a37c, totalN=0x4, perSize=0x20, firstFree=0x2b5a300)
EBB(start=0x2b5a300, next=0x2b5a3a0)
EBB(start=0x2b5a3a0, next=0x0)
Now, release C.
ePartition(start=0x2b5a37c, totalN=0x4, perSize=0x20, firstFree=0x2b5a300)
EBB(start=0x2b5a300, next=0x2b5a3a0)
EBB(start=0x2b5a3a0, next=0x2b5a3c0)
EBB(start=0x2b5a3c0, next=0x0)
Now, release D.
ePartition(start=0x2b5a37c, totalN=0x4, perSize=0x20, firstFree=0x2b5a300)
EBB(start=0x2b5a300, next=0x2b5a3a0)
EBB(start=0x2b5a3a0, next=0x2b5a3c0)
EBB(start=0x2b5a3c0, next=0x0)
Now, release E.
ePartition(start=0x2b5a37c, totalN=0x4, perSize=0x20, firstFree=0x2b5a300)
EBB(start=0x2b5a300, next=0x2b5a3a0)
EBB(start=0x2b5a3a0, next=0x2b5a3c0)
EBB(start=0x2b5a3c0, next=0x0)
breallid@oslab4:~$
Unknown interrupt
```

四、遇到的问题 and 解决方法

string.h 编译 warning: 将 src 的数据类型改成 const unsigned char * 即可 (否则会产生 const 修饰符丢失的错误)

初始 init 的一些信息无法显示: 注释 main.c 中的 clear__screen(); 即可看到信息显示