实验四，内存管理

李卓 pb19000064

实验目的

1. 实现内存检测，确定动态内存的范围。
2. 实现内存的动态分区管理机制和等大小分区管理机制。
3. 提供 kmalloc/kfree 及 malloc/free 两套接口，供内核和用户使用。
4. 提供 addNewCmd()函数，用来增加新的命令行指令。

实验内容

内存检测，确定动态内存的范围

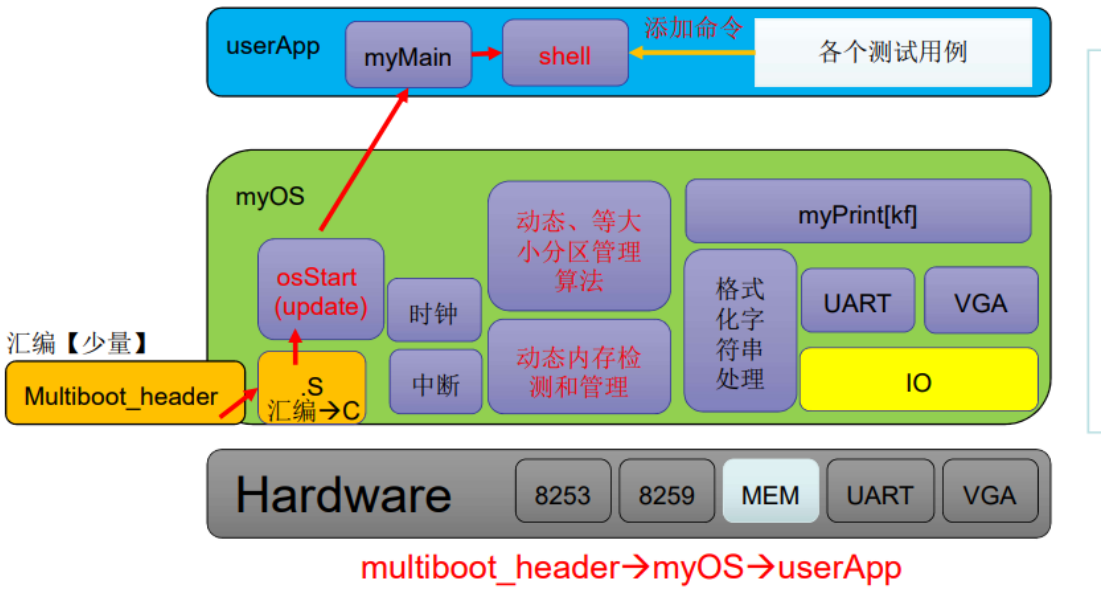
提供动态分区管理机制 dPartition

提供等大小固定分区管理机制 ePartition

使用动态分区管理机制来管理所有动态内存

提供 kmalloc/kfree 和 malloc/free 两套接口，分别 提供给内核和用户

实验框架



内核：内存（算法、检测、管理） 用户：新功能测试

被测功能：动态内存管理功能、两种算法

自测：userApp 他测：替换 userApp 或增加 shell 命令

实验流程

1. 在 multiboot_header 中完成系统的启动。

2. 在 start32.S 中做好准备，调用 osStart.c 进入 c 程序。

3. 在 osStart.c 中完成初始化 8259A，初始化 8253，清屏及内存初始化等操作，调用 myMain， 进入 userApp 部分。

4. 运行 myMain 中的代码，进行时钟设置，shell 初始化，内存测试初始化等操作，启动 shell。

5. 进入 shell 程序，等待命令的输入

实验原理

1.内存检查

设从 1M 开始–,假设只有 1 块连续的内存空间，通过检测得到这个内存块的大小

检测算法 void memTest(unsigned long start, unsigned long grainSize)

- 从 start 开始，以 grainSize 为步长，进行内存检测
- 检测方法： 1）读出 grain 的头 2 个字节

2）覆盖写入 0xAA55，再读出并检查是否是 0xAA55，若不是则检测结束；

3）覆盖写入 0x55AA，再读出并检查是否是 0x55AA，若不是则检测结束；

4）写回原来的值

5）对 grain 的尾 2 个字节，重复 2-4

6）步进到下一个 grain，重复 1-5，直到检测结束

2. 动态分区管理

dPartitionInit()实现动态分区的初始化，将待分配的内存块，作为一块进行管理。 dPartitionAlloc()实现分配，对相关的内存块大小，及表示下一空闲块的指针进行调整。 dPartitionFree()实现释放，将指定内存进行释放，并对管理的数据结构进行调整。 dPartitionAlloc()及 dPartitionFree()经过包装后，即得到 kmalloc(),kfree(),malloc()及 free() 函数

3. 等大小分区管理

eFPartitionTotalSize()实现对传入的大小的对齐，返回实际需要的大小。

eFPartitionInit()实现将所有块按顺序连接成链，便于分配和回收。

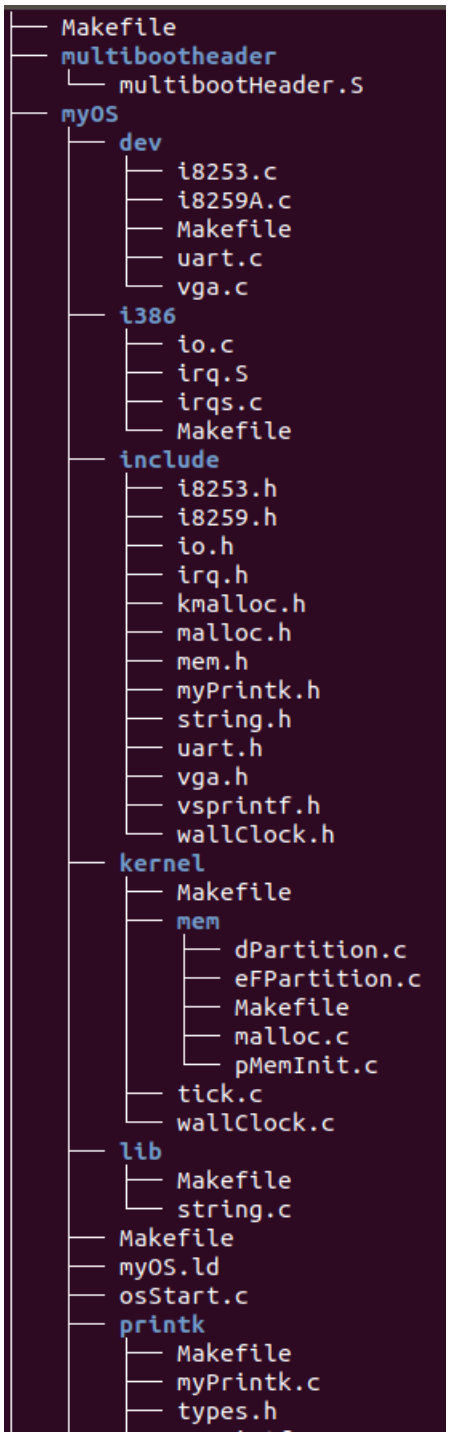
eFPartitionAlloc()实现块的分配，对链表进行重新链接，返回供使用的地址。

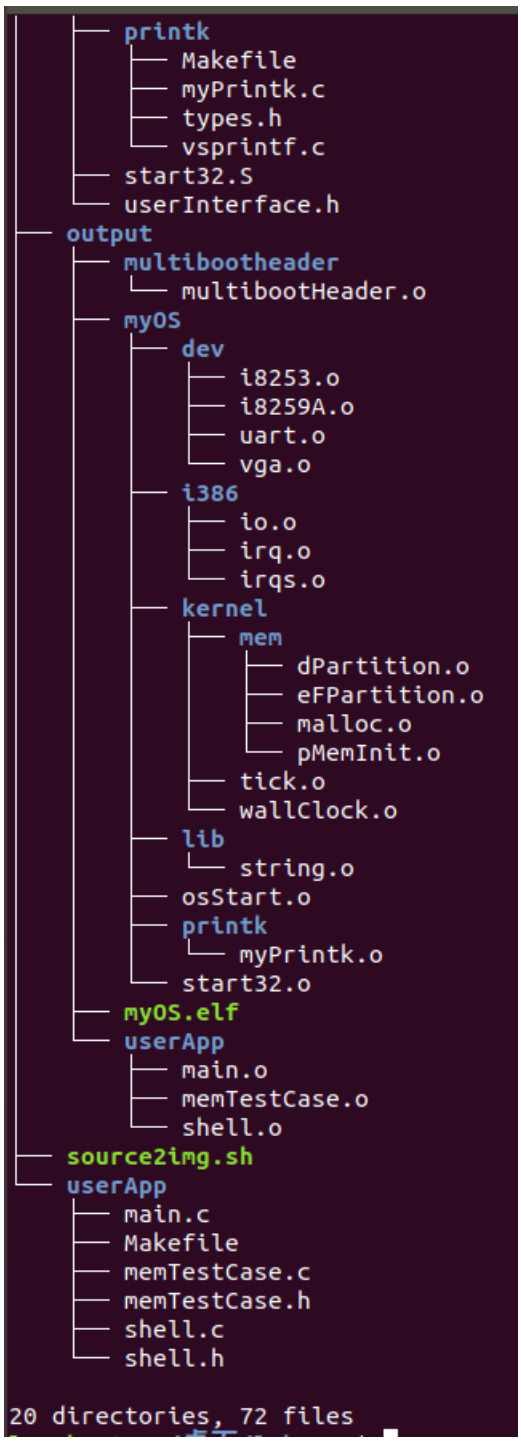
eFPartitionFree()实现块的逐一回收并重新链接至链表上。

4. addNewCmd

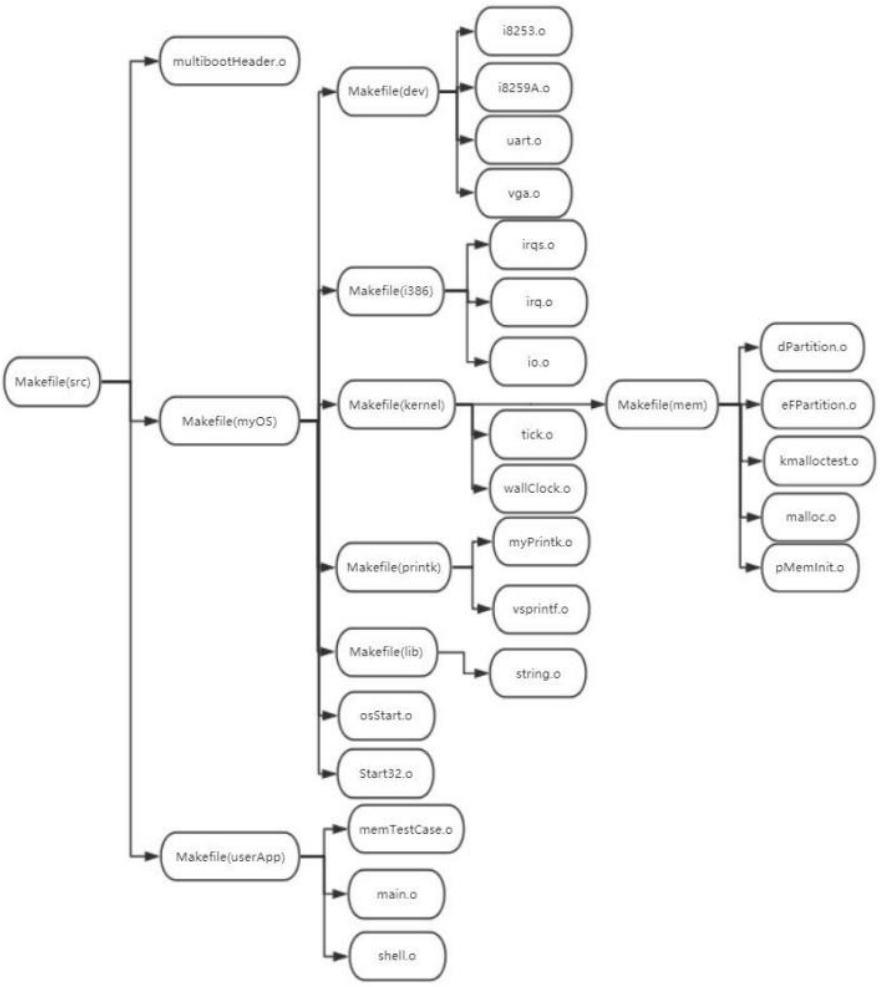
维护一个 cmd 链表，是实现 shell 中的命令的动态添加

文件目录组织


Makefile
multibootheader
multibootHeader.S
myOS
dev
i8253.c
i8259A.c
Makefile
uart.c
vga.c
i386
io.c
irq.S
irqs.c
Makefile
include
i8253.h
i8259.h
io.h
irq.h
kmalloc.h
malloc.h
mem.h
myPrintk.h
string.h
uart.h
vga.h
vsprintf.h
wallClock.h
kernel
Makefile
mem
dPartition.c
eFPartition.c
Makefile
malloc.c
pMemInit.c
tick.c
wallClock.c
lib
Makefile
string.c
Makefile
myOS.ld
osStart.c
printk
Makefile
myPrintk.c
types.h


printk
Makefile
myPrintk.c
types.h
vsprintf.c
start32.S
userInterface.h
output
multibootheader
multibootHeader.o
myOS
dev
i8253.o
i8259A.o
uart.o
vga.o
i386
io.o
irq.o
irqs.o
kernel
mem
dPartition.o
eFPartition.o
malloc.o
pMemInit.o
tick.o
wallClock.o
lib
string.o
osStart.o
printk
myPrintk.o
start32.o
myOS.elf
userApp
main.o
memTestCase.o
shell.o
source2img.sh
userApp
main.c
Makefile
memTestCase.c
memTestCase.h
shell.c
shell.h
20 directories, 72 files

makefile 组织



地址空间布局

Section	Offset (Base = 1M)	align
.multiboot_header	0	8
.text(代码段)	16	8
.data(数据段)	16+.text section	16
.bss	当前	16
堆栈(动态内存空间)	当前	

编译过程说明

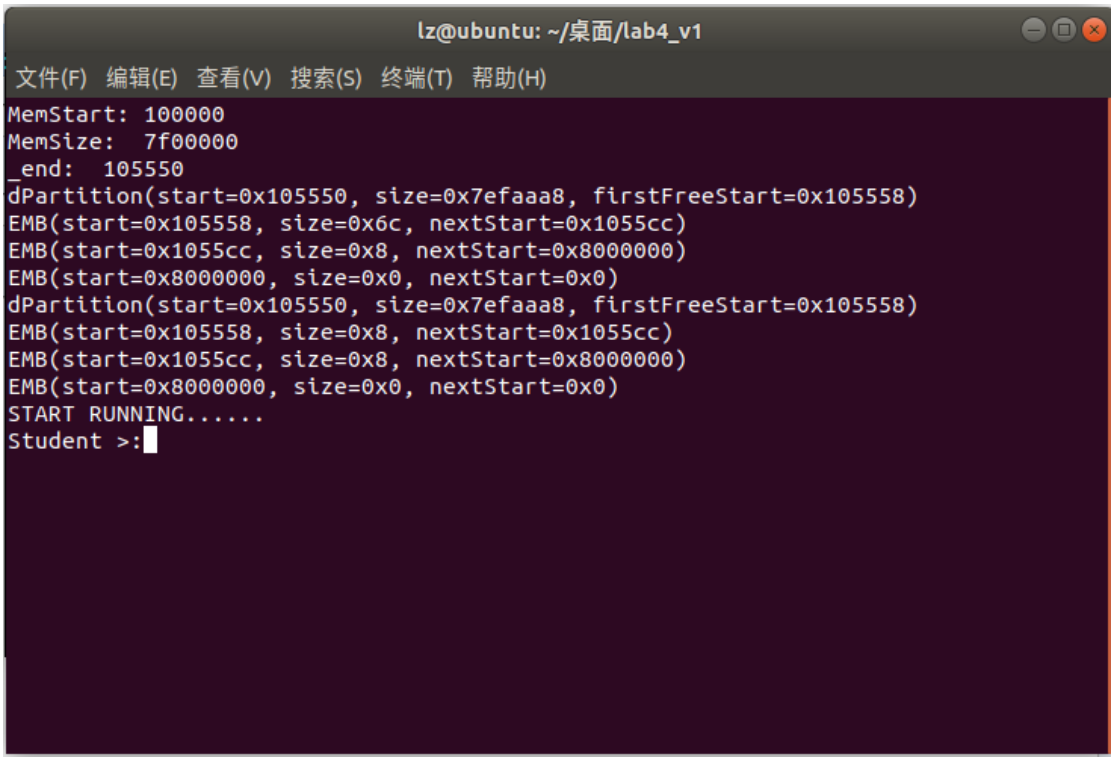
默认方式，链接生成 myOS.elf 文件

chmod 777 source2run.sh

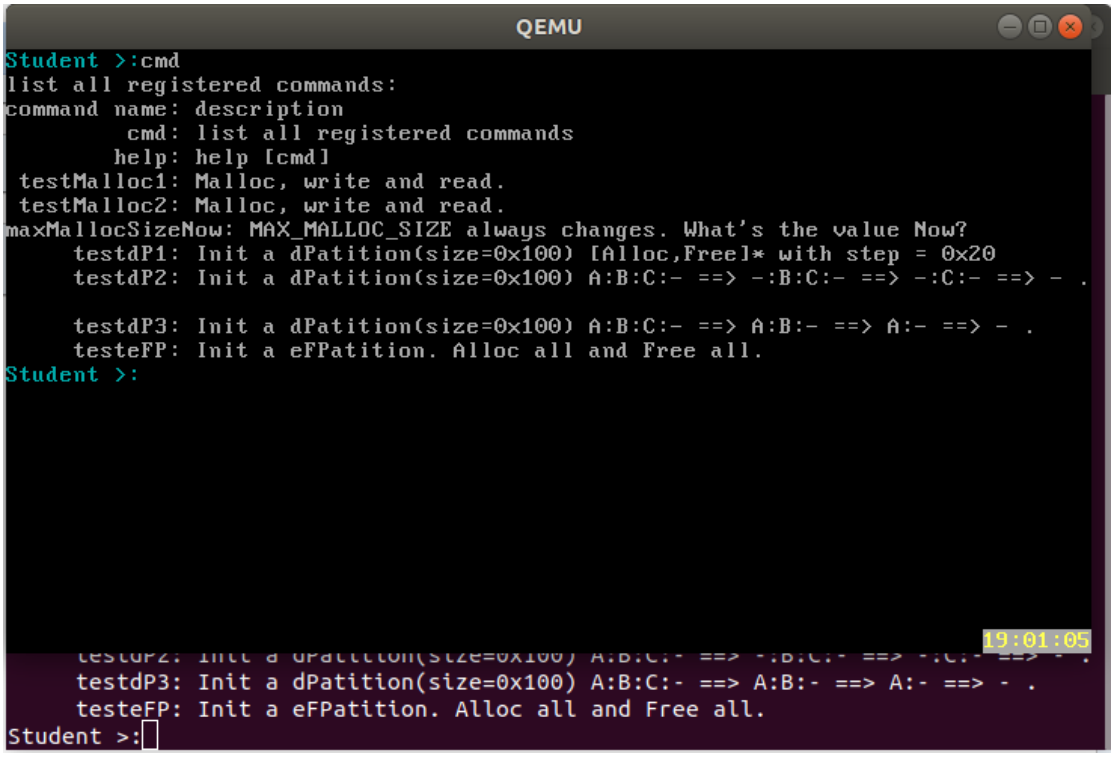
./source2run.sh

sudo screen /dev/pts/1

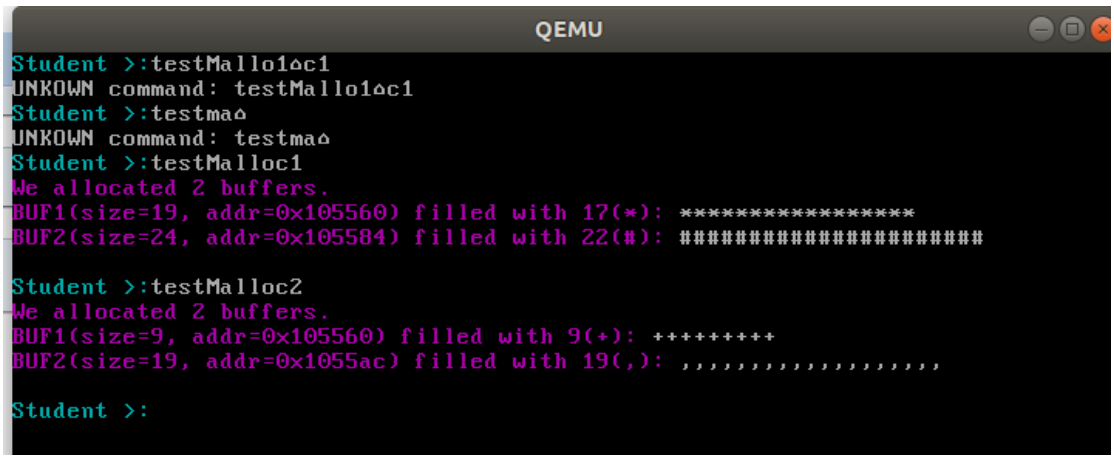
运行结果



cmd



testMalloc1 testMalloc2

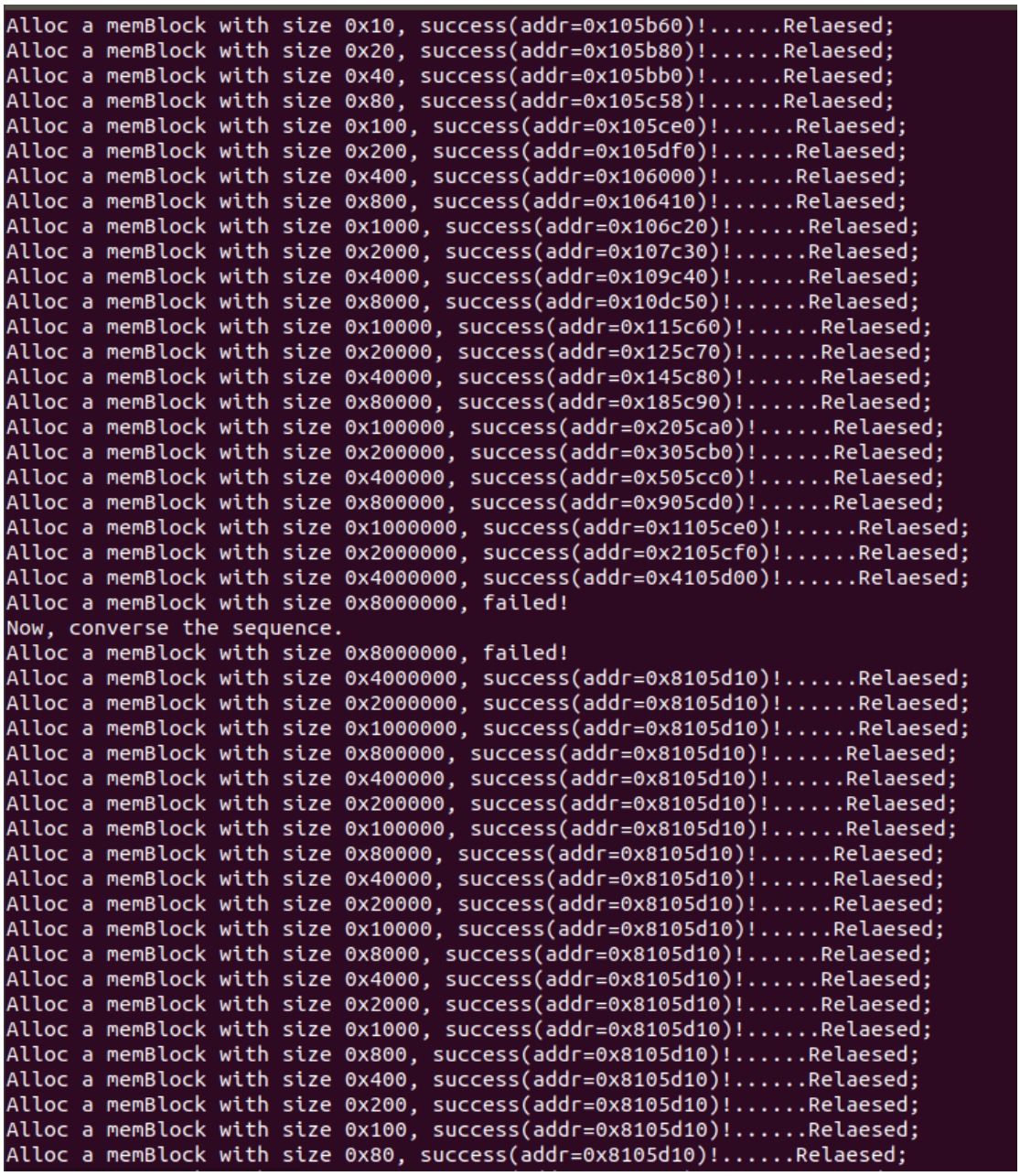

QEMU
Student >:testMalloc1
UNKOWN command: testMalloc1
Student >:testmao
UNKOWN command: testmao
Student >:testMalloc1
We allocated 2 buffers.
BUF1(size=19, addr=0x105560) filled with 17(*): ******************
BUF2(size=24, addr=0x105584) filled with 22(#): ########################
Student >:testMalloc2
We allocated 2 buffers.
BUF1(size=9, addr=0x105560) filled with 9(+): +++++++++
BUF2(size=19, addr=0x1055ac) filled with 19(,): ,,,,,,,,,,,,,,,,,,
Student >:

testeFP

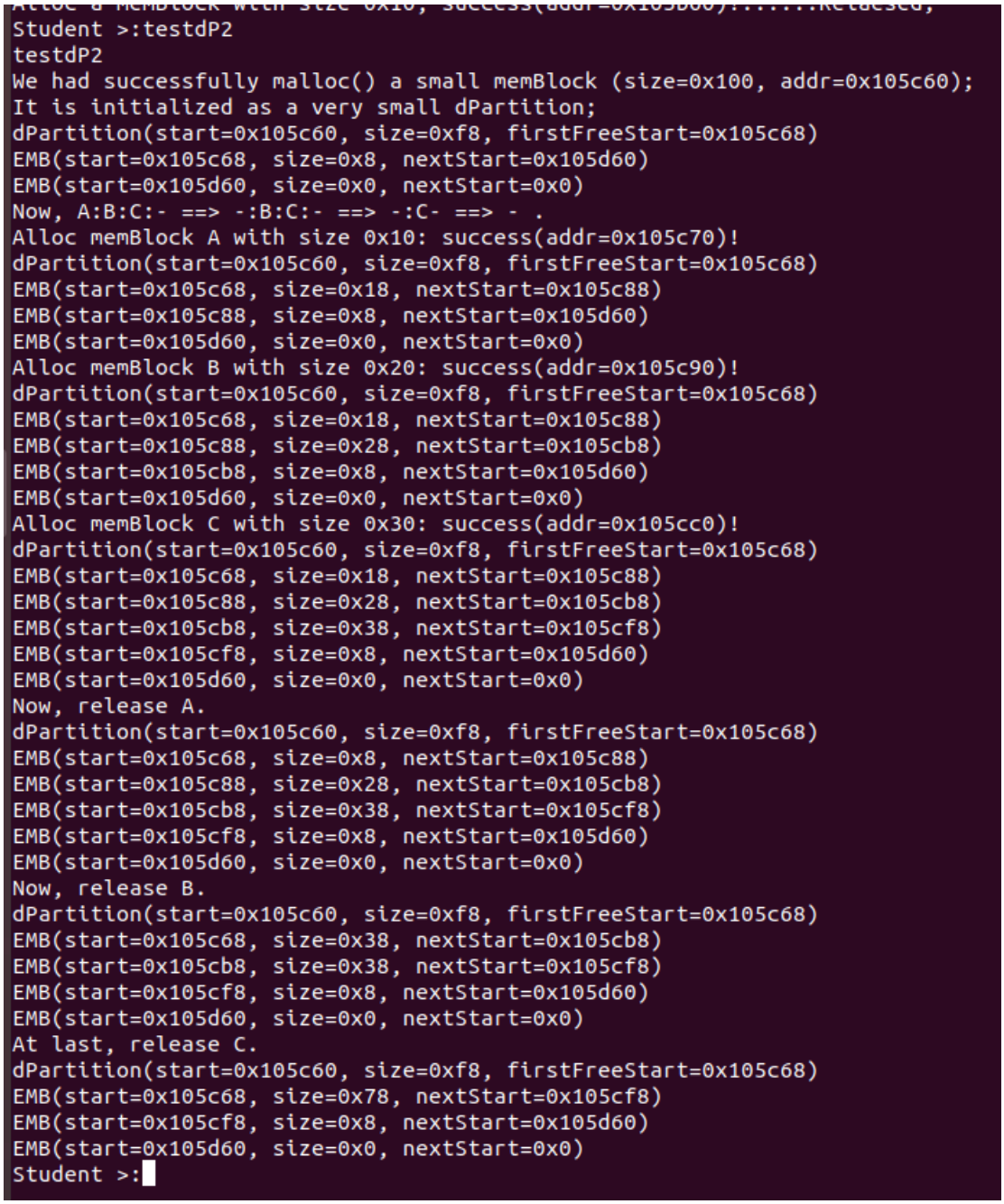
```
Student >:testeFP
testeFP
X:0x105560:12
We had successfully malloc() a small memBlock (size=0xc, addr=0x105560);
It is initialized as a very small ePartition;

persize32
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x10556c)
EEB(start=0x10556c, next=0x10558c)
EEB(start=0x10558c, next=0x1055ac)
EEB(start=0x1055ac, next=0x1055cc)
EEB(start=0x1055cc, next=0x0)
Alloc memBlock A, start = 0x10556c: 0xaaaaaaaa
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x10558c)
EEB(start=0x10558c, next=0x1055ac)
EEB(start=0x1055ac, next=0x1055cc)
EEB(start=0x1055cc, next=0x0)
Alloc memBlock B, start = 0x10558c: 0xbbbbbbbb
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x1055ac)
EEB(start=0x1055ac, next=0x1055cc)
EEB(start=0x1055cc, next=0x0)
Alloc memBlock C, start = 0x1055ac: 0xcccccccc
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x1055cc)
EEB(start=0x1055cc, next=0x0)
Alloc memBlock D, start = 0x1055cc: 0xdddddddd
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x0)
Alloc memBlock E, failed!
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x10559c)
EEB(start=0x10559c, next=0x30)
EEB(start=0x30, next=0xf000d71e)
EEB(start=0xf000d71e, next=0x0)
Now, release A.
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x10556c)
EEB(start=0x10556c, next=0x10559c)
EEB(start=0x10559c, next=0x30)
EEB(start=0x30, next=0xf000d71e)
EEB(start=0xf000d71e, next=0x0)
Now, release B.
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x10558c)
EEB(start=0x10558c, next=0xbbbbbbbb)
EEB(start=0xbbbbbbbb, next=0x0)
Now, release C.
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x1055ac)
EEB(start=0x1055ac, next=0xcccccccc)
EEB(start=0xcccccccc, next=0x0)
Now, release D.
eFPartition(start=0x105560, totalN=0x4, perSize=0x20, firstFree=0x1055cc)
EEB(start=0x1055cc, next=0xdddddddd)
EEB(start=0xdddddddd, next=0x0)
```

testdP1


Alloc a memBlock with size 0x10, success(addr=0x105b60)!......Relaesed;
Alloc a memBlock with size 0x20, success(addr=0x105b80)!......Relaesed;
Alloc a memBlock with size 0x40, success(addr=0x105bb0)!......Relaesed;
Alloc a memBlock with size 0x80, success(addr=0x105c58)!......Relaesed;
Alloc a memBlock with size 0x100, success(addr=0x105ce0)!......Relaesed;
Alloc a memBlock with size 0x200, success(addr=0x105df0)!......Relaesed;
Alloc a memBlock with size 0x400, success(addr=0x106000)!......Relaesed;
Alloc a memBlock with size 0x800, success(addr=0x106410)!......Relaesed;
Alloc a memBlock with size 0x1000, success(addr=0x106c20)!......Relaesed;
Alloc a memBlock with size 0x2000, success(addr=0x107c30)!......Relaesed;
Alloc a memBlock with size 0x4000, success(addr=0x109c40)!......Relaesed;
Alloc a memBlock with size 0x8000, success(addr=0x10dc50)!......Relaesed;
Alloc a memBlock with size 0x10000, success(addr=0x115c60)!......Relaesed;
Alloc a memBlock with size 0x20000, success(addr=0x125c70)!......Relaesed;
Alloc a memBlock with size 0x40000, success(addr=0x145c80)!......Relaesed;
Alloc a memBlock with size 0x80000, success(addr=0x185c90)!......Relaesed;
Alloc a memBlock with size 0x100000, success(addr=0x205ca0)!......Relaesed;
Alloc a memBlock with size 0x200000, success(addr=0x305cb0)!......Relaesed;
Alloc a memBlock with size 0x400000, success(addr=0x505cc0)!......Relaesed;
Alloc a memBlock with size 0x800000, success(addr=0x905cd0)!......Relaesed;
Alloc a memBlock with size 0x1000000, success(addr=0x1105ce0)!......Relaesed;
Alloc a memBlock with size 0x2000000, success(addr=0x2105cf0)!......Relaesed;
Alloc a memBlock with size 0x4000000, success(addr=0x4105d00)!......Relaesed;
Alloc a memBlock with size 0x8000000, failed!
Now, converse the sequence.
Alloc a memBlock with size 0x8000000, failed!
Alloc a memBlock with size 0x4000000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x2000000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x1000000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x800000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x400000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x200000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x100000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x80000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x40000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x20000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x10000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x8000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x4000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x2000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x1000, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x800, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x400, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x200, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x100, success(addr=0x8105d10)!......Relaesed;
Alloc a memBlock with size 0x80, success(addr=0x8105d10)!......Relaesed;

testdP2


Student >:testdP2
testdP2
We had successfully malloc() a small memBlock (size=0x100, addr=0x105c60);
It is initialized as a very small dPartition;
dPartition(start=0x105c60, size=0xf8, firstFreeStart=0x105c68)
EMB(start=0x105c68, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
Now, A:B:C:- ==> -:B:C:- ==> -:C- ==> - .
Alloc memBlock A with size 0x10: success(addr=0x105c70)!
dPartition(start=0x105c60, size=0xf8, firstFreeStart=0x105c68)
EMB(start=0x105c68, size=0x18, nextStart=0x105c88)
EMB(start=0x105c88, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
Alloc memBlock B with size 0x20: success(addr=0x105c90)!
dPartition(start=0x105c60, size=0xf8, firstFreeStart=0x105c68)
EMB(start=0x105c68, size=0x18, nextStart=0x105c88)
EMB(start=0x105c88, size=0x28, nextStart=0x105cb8)
EMB(start=0x105cb8, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
Alloc memBlock C with size 0x30: success(addr=0x105cc0)!
dPartition(start=0x105c60, size=0xf8, firstFreeStart=0x105c68)
EMB(start=0x105c68, size=0x18, nextStart=0x105c88)
EMB(start=0x105c88, size=0x28, nextStart=0x105cb8)
EMB(start=0x105cb8, size=0x38, nextStart=0x105cf8)
EMB(start=0x105cf8, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
Now, release A.
dPartition(start=0x105c60, size=0xf8, firstFreeStart=0x105c68)
EMB(start=0x105c68, size=0x8, nextStart=0x105c88)
EMB(start=0x105c88, size=0x28, nextStart=0x105cb8)
EMB(start=0x105cb8, size=0x38, nextStart=0x105cf8)
EMB(start=0x105cf8, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
Now, release B.
dPartition(start=0x105c60, size=0xf8, firstFreeStart=0x105c68)
EMB(start=0x105c68, size=0x38, nextStart=0x105cb8)
EMB(start=0x105cb8, size=0x38, nextStart=0x105cf8)
EMB(start=0x105cf8, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
At last, release C.
dPartition(start=0x105c60, size=0xf8, firstFreeStart=0x105c68)
EMB(start=0x105c68, size=0x78, nextStart=0x105cf8)
EMB(start=0x105cf8, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
Student >:

testdP3

```
EMB(start=0x105cf8, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
At last, release C.
dPartition(start=0x105c60, size=0xf8, firstFreeStart=0x105c68)
EMB(start=0x105c68, size=0x78, nextStart=0x105cf8)
EMB(start=0x105cf8, size=0x8, nextStart=0x105d60)
EMB(start=0x105d60, size=0x0, nextStart=0x0)
Student >:testdP3
testdP3
We had successfully malloc() a small memBlock (size=0x100, addr=0x105d70);
It is initialized as a very small dPartition;
dPartition(start=0x105d70, size=0xf8, firstFreeStart=0x105d78)
EMB(start=0x105d78, size=0x8, nextStart=0x105e70)
EMB(start=0x105e70, size=0x0, nextStart=0x0)
Now, A:B:C:- ==> -:B:C:- ==> -:C- ==> - .
Alloc memBlock A with size 0x10: success(addr=0x105d80)!
dPartition(start=0x105d70, size=0xf8, firstFreeStart=0x105d78)
EMB(start=0x105d78, size=0x18, nextStart=0x105d98)
EMB(start=0x105d98, size=0x8, nextStart=0x105e70)
EMB(start=0x105e70, size=0x0, nextStart=0x0)
Alloc memBlock B with size 0x20: success(addr=0x105da0)!
dPartition(start=0x105d70, size=0xf8, firstFreeStart=0x105d78)
EMB(start=0x105d78, size=0x18, nextStart=0x105d98)
EMB(start=0x105d98, size=0x28, nextStart=0x105dc8)
EMB(start=0x105dc8, size=0x8, nextStart=0x105e70)
EMB(start=0x105e70, size=0x0, nextStart=0x0)
Alloc memBlock C with size 0x30: success(addr=0x105dd0)!
dPartition(start=0x105d70, size=0xf8, firstFreeStart=0x105d78)
EMB(start=0x105d78, size=0x18, nextStart=0x105d98)
EMB(start=0x105d98, size=0x28, nextStart=0x105dc8)
EMB(start=0x105dc8, size=0x38, nextStart=0x105e08)
EMB(start=0x105e08, size=0x8, nextStart=0x105e70)
EMB(start=0x105e70, size=0x0, nextStart=0x0)
At last, release C.
dPartition(start=0x105d70, size=0xf8, firstFreeStart=0x105d78)
EMB(start=0x105d78, size=0x18, nextStart=0x105d98)
EMB(start=0x105d98, size=0x68, nextStart=0x105e08)
EMB(start=0x105e08, size=0x8, nextStart=0x105e70)
EMB(start=0x105e70, size=0x0, nextStart=0x0)
Now, release B.
dPartition(start=0x105d70, size=0xf8, firstFreeStart=0x105d78)
EMB(start=0x105d78, size=0x88, nextStart=0x105e08)
EMB(start=0x105e08, size=0x8, nextStart=0x105e70)
EMB(start=0x105e70, size=0x0, nextStart=0x0)
Now, release A.
dPartition(start=0x105d70, size=0xf8, firstFreeStart=0x105d78)
EMB(start=0x105d78, size=0x8, nextStart=0x105e08)
EMB(start=0x105e08, size=0x8, nextStart=0x105e70)
EMB(start=0x105e70, size=0x0, nextStart=0x0)
Student >:
```

maxMallocSizeNow

```
Student >:maxMallocSizeNow
MAX_MALLOC_SIZE: 0x7efb000 (with step = 0x1000);
Student >:_
```

运行结果解释

testMalloc1:

malloc 两块长为 19 和 24 的空间,分别用* 和#填充，然后 free,.

由于 firstfit 算法，此时 buffer1 和 buffer2 空间上是连续的. buffer1 逻辑长度为 19 但实际长度计算为 0x24，因为对齐需要, EMB 占位 和隔离带 .可以看到，逻辑地址在 emb 实际地址 8 位后

```
We allocated 2 buffers.
BUF1(size=19, addr=0x1055a0) filled with 17(*): *****************
BUF2(size=24, addr=0x1055c4) filled with 22(#): ######################
```

EMB 如图

```
dPartition(start=0x105590, size=0x7efaa68, firstFreeStart=0x105598)
EMB(start=0x105598, size=0x1c, nextStart=0x1055bc)
EMB(start=0x1055bc, size=0x20, nextStart=0x1055e4)
```

testMalloc2:

和 testMalloc1 同理

```
We allocated 2 buffers.
BUF1(size=9, addr=0x1055e0) filled with 9(+): +++++++++
BUF2(size=19, addr=0x1055fc) filled with 19(,): ,,,,,,,,,,,,,,,,,,,
```

占用的 EMB 如图

```
EMB(start=0x1055d8, size=0x14, nextStart=0x1055f4)
EMB(start=0x1055f4, size=0x20, nextStart=0x10561c)
```

实际空间位 0x14 和 0x20

释放后 EMB 如图

```
dPartition(start=0x1055d0, size=0x7efaa28, firstFreeStart=0x1055d8)
EMB(start=0x1055d8, size=0x30, nextStart=0x105644)
```

得到一个 0x30 大小的空间

maxMallocSizeNow:

步长 0x1000

```
Student >:maxMallocSizeNow
maxMallocSizeNow
MAX_MALLOC_SIZE: 0x7efb000 (with step = 0x1000);
```

testdP1;

从 0x10 开始,步长不断翻倍 申请空间，直到失败,全部释放

```
Alloc a memBlock with size 0x4000000, success(addr=0x8007cf0)!......Relaesed;
Alloc a memBlock with size 0x8000000, failed!
```

最后一次申请，大小 0x4000000，最后的地址为 0x8007cf0

testdP2:

A B C 全部申请后,如图,结果正确

```
Alloc memBlock C with size 0x30: success(addr=0x7f87e30)!
dPartition(start=0x7f87dd0, size=0xf8, firstFreeStart=0x7f87dd8)
EMB(start=0x7f87dd8, size=0x18, nextStart=0x7f87df8)
EMB(start=0x7f87df8, size=0x28, nextStart=0x7f87e28)
EMB(start=0x7f87e28, size=0x38, nextStart=0x7f87e68)
EMB(start=0x7f87e68, size=0x8, nextStart=0x7f87ed0)
EMB(start=0x7f87ed0, size=0x0, nextStart=0x0)
```

释放 A 后

```
Now, release A.
dPartition(start=0x7f87dd0, size=0xf8, firstFreeStart=0x7f87dd8)
EMB(start=0x7f87dd8, size=0x8, nextStart=0x7f87df8)
EMB(start=0x7f87df8, size=0x28, nextStart=0x7f87e28)
EMB(start=0x7f87e28, size=0x38, nextStart=0x7f87e68)
```

释放 B 后:

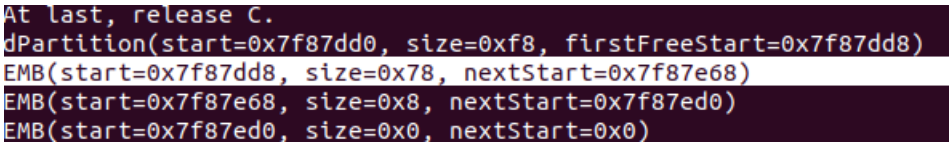
```
Now, release B.
dPartition(start=0x7f87dd0, size=0xf8, firstFreeStart=0x7f87dd8)
EMB(start=0x7f87dd8, size=0x38, nextStart=0x7f87e28)
EMB(start=0x7f87e28, size=0x38, nextStart=0x7f87e68)
EMB(start=0x7f87e68, size=0x8, nextStart=0x7f87ed0)
EMB(start=0x7f87ed0, size=0x0, nextStart=0x0)
```

可以看到,A 和 B 空闲的空间是连续的,所以被合并了,新空间大小为

0x10+0x20+0x08

结果正确

释放 C 后:



同理,空闲空间被合并

testdP3:

和 testdP2 同理

testeFP

如图所示, ABCD 都申请成功，E 失败，结果正确

```
Student >:testeFP
testeFP
X:0x1055e0:12
We had successfully malloc() a small memBlock (size=0xc, addr=0x1055e0);
It is initialized as a very small ePartition;

persize32
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x1055ec)
EEB(start=0x1055ec, next=0x10560c)
EEB(start=0x10560c, next=0x10562c)
EEB(start=0x10562c, next=0x10564c)
EEB(start=0x10564c, next=0x0)
Alloc memBlock A, start = 0x1055ec: 0xaaaaaaaa
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x10560c)
EEB(start=0x10560c, next=0x10562c)
EEB(start=0x10562c, next=0x10564c)
EEB(start=0x10564c, next=0x0)
Alloc memBlock B, start = 0x10560c: 0xbbbbbbbb
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x10562c)
EEB(start=0x10562c, next=0x10564c)
EEB(start=0x10564c, next=0x0)
Alloc memBlock C, start = 0x10562c: 0xcccccccc
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x10564c)
EEB(start=0x10564c, next=0x0)
Alloc memBlock D, start = 0x10564c: 0xdddddddd
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x0)
Alloc memBlock E, failed!
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0xf000ff53)
EEB(start=0xf000ff53, next=0x0)
Now, release A.
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x1055ec)
EEB(start=0x1055ec, next=0xf000ff53)
EEB(start=0xf000ff53, next=0x0)
Now, release B.
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x10560c)
EEB(start=0x10560c, next=0xbbbbbbbb)
EEB(start=0xbbbbbbbb, next=0x0)
Now, release C.
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x10562c)
EEB(start=0x10562c, next=0xcccccccc)
EEB(start=0xcccccccc, next=0x0)
Now, release D.
eFPartition(start=0x1055e0, totalN=0x4, perSize=0x20, firstFree=0x10564c)
EEB(start=0x10564c, next=0xdddddddd)
EEB(start=0xdddddddd, next=0x0)
```

mytest:

自己设计的样例，输出自己的姓名和学号，调用 addNewCmd 添加到 cmd 中

```
Student >:mytest
mytest
allocated.
BUF1(addr=0x1056c0) filled with : lizhuo_PB19000064
```

实验中遇到的问题

1. 没理清文件结构，对全局变量重定义
2. 使用指针前,忘记判断是否为空指针
3. 对 size 理解出错，实际空间比 逻辑空间大