**《操作系统》**

**实 验 指 导 手 册**

**授课教师： 刘知青**

**实验指导教师： 刘知青**

**教学对象： 二年级本科生**

**开课时间： 春季学期**

**学生姓名： 裴仪瑶**

**学号： 2014212052**

**成绩：**

**北京邮电大学软件学院**

**2016年5月25日**

1. 实验目的

理解应用内存分配系统和存储结构。

1. 实验内容

编写c语言标准的内存分配函数。

1. 实验环境

Linux；codeblocks

1. 实验要求

利用页表分配内存并保存信息数据。利用系统函数mmap编写malloc, calloc, realloc, 和free函数。

1. 实验代码

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<sys/mman.h>

#define BLOCK\_SIZE 24

void \*first\_block=NULL;

typedef struct s\_block \*t\_block;

struct s\_block{

size\_t size; /\* 数据区大小 \*/

t\_block prev; /\* 指向上个块的指针 \*/

t\_block next; /\* 指向下个块的指针 \*/

int free; /\* 是否是空闲块 \*/

void \*ptr; /\* Magic pointer，指向data \*/

char data[1] /\* 这是一个虚拟字段，表示数据块的第一个字节，长度不应计入meta \*/

};

/\* 找block\*/

t\_block find\_block(t\_block \*last, size\_t size) {

t\_block b = first\_block;

while(b && !(b->free && b->size >= size)) {

\*last = b;

b = b->next;

}

return b;

}

t\_block extend\_heap(t\_block last, size\_t s) {

t\_block b;

b=mmap(first\_block,0,PROT\_READ,MAP\_FIXED,0,0);

if(mmap(first\_block,BLOCK\_SIZE + s,PROT\_READ,MAP\_FIXED,0,0) == (void \*)-1)////////////////////////

return NULL;

b->size = s;

b->next = NULL;

if(last)

last->next = b;

b->free = 0;

return b;

}

void\* mymalloc(size\_t s) {

t\_block b, last;

if(first\_block) {

/\* 查找合适的block \*/

last = first\_block;

b = find\_block(&last, s);

if(b)

b->free = 0;

else/\* 没有合适的block，开辟一个新的 \*/

b = extend\_heap(last, s);

}

else {

b = extend\_heap(NULL, s);

first\_block = b;

}

}

void\* mycalloc(size\_t number, size\_t size) {

size\_t \*new;

size\_t s8, i;

new = mymalloc(number \* size);

if(new) {

s8 = number \* size;

for(i = 0; i < s8; i++)

new[i] = 0;

}

return new;

}

t\_block get\_block(void \*p) {

char \*tmp;

tmp = p;

return (p = tmp -= BLOCK\_SIZE);

}

int valid\_addr(void \*p) {

if(first\_block) {

if(p > first\_block && p < mmap(first\_block,0,PROT\_READ,MAP\_FIXED,0,0)) {

return p == (get\_block(p))->ptr;

}

}

return 0;

}

t\_block fusion(t\_block b) {

if (b->next && b->next->free) {

b->size += BLOCK\_SIZE + b->next->size;

b->next = b->next->next;

if(b->next)

b->next->prev = b;

}

return b;

}

void myfree(void \*p) {

t\_block b;

if(valid\_addr(p)) {

b = get\_block(p);

b->free = 1;

if(b->prev && b->prev->free)

b = fusion(b->prev);

if(b->next)

fusion(b);

else {

if(b->prev)

b->prev->prev = NULL;

else

first\_block = NULL;

brk(b);

}

}

}

void copy\_block(t\_block src, t\_block dst) {

size\_t \*sdata, \*ddata;

size\_t i;

sdata = src->ptr;

ddata = dst->ptr;

for(i = 0; (i \* 8) < src->size && (i \* 8) < dst->size; i++)

ddata[i] = sdata[i];

}

int \*myrealloc(void \*p, size\_t size) {

size\_t s;

t\_block b, new;

int \*newp;

if(valid\_addr(p)) {

s = size;

b = get\_block(p);

/\* 看是否可进行合并 \*/

if(b->next && b->next->free&& (b->size + BLOCK\_SIZE + b->next->size) >= s)

fusion(b);

else {

newp = mymalloc (s);

if (!newp)

return NULL;

new = get\_block(newp);

copy\_block(b, new);

myfree(p);

return(newp);

}

return (p);

}

return NULL;

}

void main(){

void\* p = mymalloc(1024);

myrealloc(p, 100);

myfree(p);

char\* a = mymalloc(1);

a[0]='1';

myfree(a);

}