

OS lab3 report

牛庆源 PB21111733

1. 实验目的与内容

- **part1:**使用显式空闲链表实现一个64位堆内存分配器（使用两种搜索算法，实现堆内存的动态扩容和实时的分配器内存使用情况统计），学会以动态链接库的形式制作库并使用（已给出）。
- **part2:**了解Linux系统虚拟内存的管理方式，了解VMA并遍历统计VMA，了解多级页表机制，完成一个从虚拟地址到物理地址的转换函数，了解Linux页面置换算法，统计页面冷热并输出图表观察，了解用户内存地址空间的分部并将用户进程中的数据转储至外存。

2. 实验平台

- VMware
- linux-4.9.263
- vscode

3.实验过程

part1 代码补全

1. void mem_init(void)

```
void mem_init(void)
{
    mem_start_brk = sbrk(MAX_HEAP); // 调用sbrk初始化mem_start_brk
    mem_brk = mem_start_brk;
    mem_max_addr = mem_start_brk + MAX_HEAP; // 堆增长空间大小为MAX_HEAP
}
```

2. void *mem_sbrk(int incr)

```
void *mem_sbrk(int incr)
{
    char *old_brk = mem_brk;
    if(old_brk + incr <= mem_max_addr) // 增长后没有超过最大值
        mem_brk += incr; // 直接增长
    else
    {
        sbrk(MAX_HEAP); // 用sbrk扩容MAX_HEAP大小
        mem_max_addr += MAX_HEAP; // 最大值也相应扩容MAX_HEAP
        mem_brk += incr; // 增长
    }
    return (void *)old_brk;
}
```

3. static void *find_fit_first(size_t asize)

```
static void *find_fit_first(size_t asize)
{
    size_t bp = free_listp;
    while(bp != 0) // 遍历
    {
        if(GET_SIZE(HDRP(bp)) > asize) // 找到
            return (void *)bp;
        bp = GET_SUCC(bp);
    }
    return NULL; // 没找到
}
```

4. static void* find_fit_best(size_t asize)

```
static void* find_fit_best(size_t asize)
{
    size_t bp = free_listp;
    size_t bestp = 0;
    int flag = 0;
    while(bp != 0) //遍历
    {
        if (GET_SIZE(HDRP(bp)) > asize)
            if (flag == 0)
            { // 首次赋值
                bestp = bp;
                flag = 1;
            }
            else
            { // 更新bestp
                if (GET_SIZE(HDRP(bp)) <= GET_SIZE(HDRP(bestp)))
                    bestp = bp;
            }
        bp = GET_SUCC(bp);
    }
    if(bestp == 0) // best一直没变
        return NULL;
    else // best被更新过
        return (void *)bestp;
}
```

5. static void place(void *bp, size_t asize)

```
static void place(void *bp, size_t asize)
{
    size_t bsize = GET_SIZE(HDRP(bp));
```

```

size_t prev_alloc = GET_PREV_ALLOC(HDRP(bp));
delete_from_free_list(bp);
if(bsize - asize < MIN_BLK_SIZE)
{
    // 修改当前块和下一个块的头部信息
    PUT(HDRP(bp), PACK(bsize, prev_alloc, 1)); // 当前块分配
    PUT(HDRP(NEXT_BLKP(bp)), GET(HDRP(NEXT_BLKP(bp))) | 0x2); // 下一个块的前邻
居块分配
}
else
{
    PUT(HDRP(bp), PACK(asize, prev_alloc, 1)); // 当前块分配
    char *new_bp = NEXT_BLKP(bp);
    size_t new_size = bsize - asize; // 分隔出的空闲块大小
    PUT(HDRP(new_bp), PACK(new_size, 1, 0)); // 当前块空闲 (头指针)
    PUT(FTRP(new_bp), PACK(new_size, 1, 0)); // 当前块空闲 (尾指针)
    coalesce(new_bp);
}
}

```

6. static void *coalesce(void *bp)

```

static void *coalesce(void *bp)
{
    /*add_to_free_list(bp);*/
    size_t prev_alloc = GET_PREV_ALLOC(HDRP(bp));
    size_t next_alloc = GET_ALLOC(HDRP(NEXT_BLKP(bp)));
    size_t size = GET_SIZE(HDRP(bp));
    if (prev_alloc && next_alloc) /* 前后都是已分配的块 */
    {
        add_to_free_list(bp); // 没有可以合并的块
    }
    else if (prev_alloc && !next_alloc) /*前块已分配, 后块空闲*/
    {
        char *next_bp = NEXT_BLKP(bp); // 和后块合并
        size_t next_size = GET_SIZE(HDRP(next_bp));
        delete_from_free_list(next_bp);
        PUT(HDRP(bp), PACK(size + next_size, 1, 0)); // 当前块空闲 (头)
        PUT(FTRP(bp), PACK(size + next_size, 1, 0)); // (尾)
        add_to_free_list(bp);
    }
    else if (!prev_alloc && next_alloc) /*前块空闲, 后块已分配*/
    {
        char *prev_bp = PREV_BLKP(bp); // 和前块合并
        size_t prev_size = GET_SIZE(HDRP(prev_bp));
        delete_from_free_list(prev_bp);
        PUT(FTRP(bp), PACK(size + prev_size, GET_PREV_ALLOC(HDRP(prev_bp)), 0));
        PUT(HDRP(prev_bp), PACK(size + prev_size, GET_PREV_ALLOC(HDRP(prev_bp)),
0));
        add_to_free_list(prev_bp);
        bp = prev_bp;
    }
}

```

```
    else /*前后都是空闲块*/
    {    // 参考前两个else if内容
        char *prev_bp = PREV_BLKP(bp);
        char *next_bp = NEXT_BLKP(bp);
        size_t prev_size = GET_SIZE(HDRP(prev_bp));
        size_t next_size = GET_SIZE(HDRP(next_bp));
        delete_from_free_list(prev_bp);
        delete_from_free_list(next_bp);
        // 扩容块占有前块和后块，所以头部尾部分别修改前块和后块的
        PUT(FTRP(next_bp), PACK((prev_size + size + next_size),
GET_PREV_ALLOC(HDRP(prev_bp)), 0));
        PUT(HDRP(prev_bp), PACK((prev_size + size + next_size),
GET_PREV_ALLOC(HDRP(prev_bp)), 0));
        add_to_free_list(prev_bp);
        bp = prev_bp;
    }
    return bp;
}
```

- 通过修改mm_malloc中的分配策略，分别调用workload测试得到结果：

find_fit_first:

```
before free: 0.999868; after free: 0.223753
time of loop 0 : 399ms
before free: 0.970494; after free: 0.215708
time of loop 1 : 549ms
before free: 0.967134; after free: 0.219046
time of loop 2 : 595ms
before free: 0.963973; after free: 0.217173
time of loop 3 : 601ms
before free: 0.961011; after free: 0.212307
time of loop 4 : 581ms
before free: 0.962702; after free: 0.215033
time of loop 5 : 542ms
before free: 0.960329; after free: 0.213856
time of loop 6 : 551ms
before free: 0.963557; after free: 0.214625
time of loop 7 : 580ms
before free: 0.957874; after free: 0.21242
time of loop 8 : 575ms
before free: 0.967218; after free: 0.21421
time of loop 9 : 562ms
before free: 0.96514; after free: 0.215154
time of loop 10 : 576ms
before free: 0.963841; after free: 0.218346
time of loop 11 : 572ms
before free: 0.966515; after free: 0.216837
time of loop 12 : 599ms
before free: 0.966461; after free: 0.214383
time of loop 13 : 570ms
before free: 0.967506; after free: 0.21579
time of loop 14 : 599ms
before free: 0.962781; after free: 0.215285
time of loop 15 : 587ms
before free: 0.959217; after free: 0.214106
time of loop 16 : 573ms
before free: 0.959343; after free: 0.214788
time of loop 17 : 633ms
before free: 0.959452; after free: 0.213562
time of loop 18 : 571ms
before free: 0.962971; after free: 0.215663
time of loop 19 : 562ms
```

find_fit_best:

```
before free: 0.999868; after free: 0.223753
time of loop 0 : 391ms
before free: 0.996245; after free: 0.22137
time of loop 1 : 2087ms
before free: 0.995636; after free: 0.225444
time of loop 2 : 2195ms
before free: 0.99278; after free: 0.22357
time of loop 3 : 2126ms
before free: 0.989666; after free: 0.218595
time of loop 4 : 2201ms
before free: 0.991216; after free: 0.22129
time of loop 5 : 2153ms
before free: 0.98877; after free: 0.220014
time of loop 6 : 2113ms
before free: 0.991977; after free: 0.220932
time of loop 7 : 2278ms
before free: 0.986383; after free: 0.218687
time of loop 8 : 2161ms
before free: 0.99582; after free: 0.220525
time of loop 9 : 2059ms
before free: 0.993767; after free: 0.221431
time of loop 10 : 2102ms
before free: 0.992171; after free: 0.224709
time of loop 11 : 2133ms
before free: 0.995274; after free: 0.223149
time of loop 12 : 2102ms
before free: 0.99519; after free: 0.220696
time of loop 13 : 2111ms
before free: 0.996343; after free: 0.222161
time of loop 14 : 2117ms
before free: 0.991545; after free: 0.22168
time of loop 15 : 2076ms
before free: 0.987915; after free: 0.220382
time of loop 16 : 2091ms
before free: 0.987803; after free: 0.221085
time of loop 17 : 2137ms
before free: 0.988287; after free: 0.219883
time of loop 18 : 2183ms
before free: 0.99167; after free: 0.222017
time of loop 19 : 2064ms
```

part2 func1~5

1. func1

```
static void scan_vma(void)
{
    printk("func == 1, %s\n", __func__);
    struct mm_struct *mm = get_task_mm(my_task_info.task);
    if (mm)
    {
        // 遍历VMA将VMA的个数记录到my_task_info的vma_cnt变量中
        struct vm_area_struct *vm_tmp = mm->mmap;
        int cnt = 0;
```

```

        while(vm_tmp)    // 遍历vma
        {
            vm_tmp = vm_tmp->vm_next;
            cnt ++;
        }
        my_task_info.vma_cnt = cnt;
        mmput(mm);
    }
}

```

result:

```

qyniu@ubuntu:~/code/lab3/lab3.2$ cat /sys/kernel/mm/ktest/vma
0, 39

```

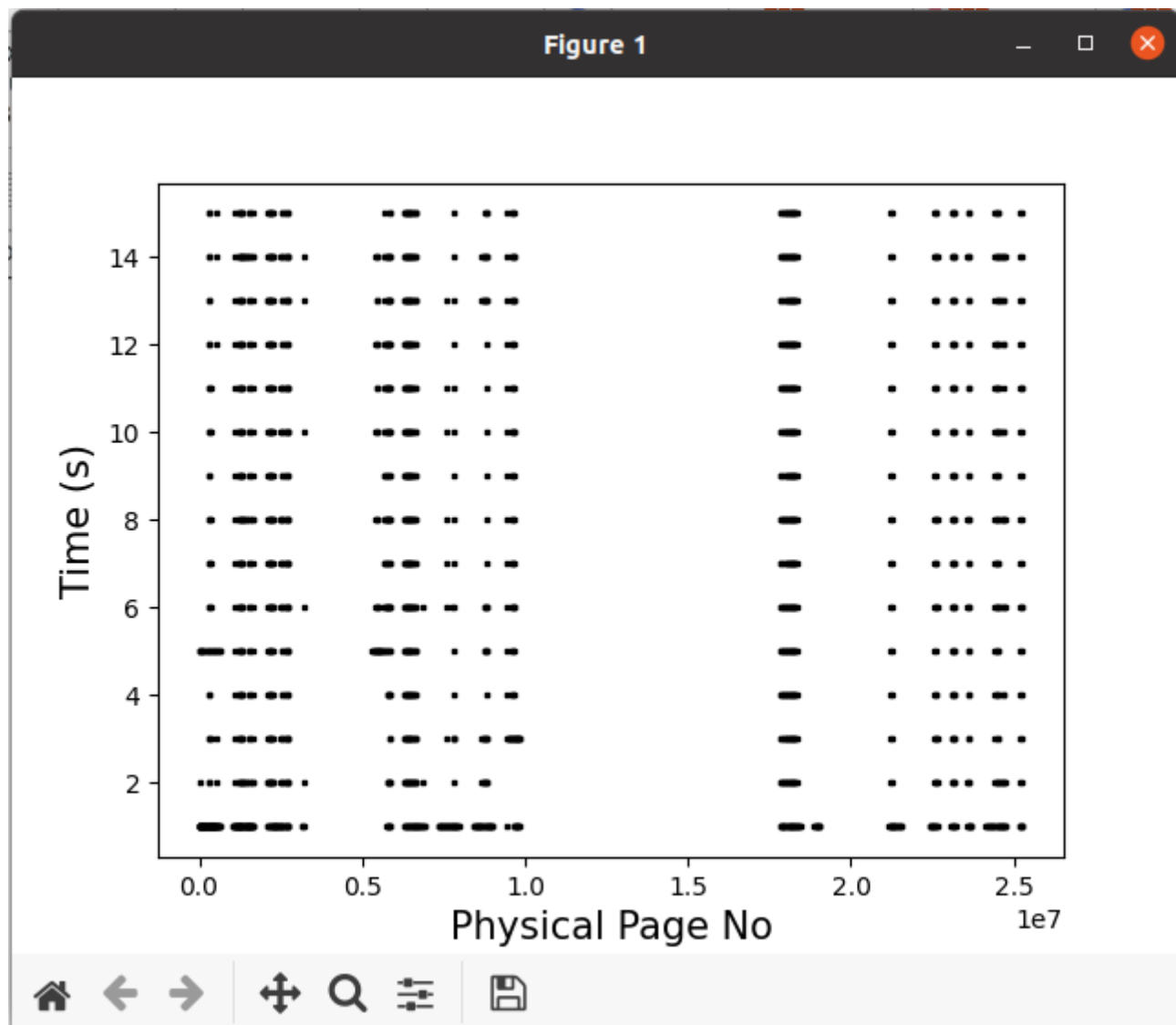
2. func2

```

static void print_mm_active_info(void)
{
    printk("func == 2, %s\n", __func__);
    struct mm_struct *mm = get_task_mm(my_task_info.task);
    if (mm)
    {
        struct vm_area_struct *vm_tmp = mm->mmap;
        while(vm_tmp)    // 遍历vma
        {
            unsigned long virt_addr = vm_tmp->vm_start;
            for ( ; virt_addr <= vm_tmp->vm_end ; virt_addr += PAGE_SIZE) // 遍历
                虚拟地址
                {
                    struct page *page = mfollow_page(vm_tmp, virt_addr, FOLL_GET);
                    if (!IS_ERR_OR_NULL(page)) // 检查page是否非空且有效
                    {
                        unsigned long vm_flags;
                        int freq = mpage_referenced(page, 0, (struct mem_cgroup *)
                        (page->memcg_data), &vm_flags);
                        好辨认)
                        if(freq > 1)    // 若被访问到(这里不为0即可, 大于1的时候得到的结果
                        {
                            // 地址写入
                            record_one_data(((page->index)*PAGE_SIZE));
                            // page->index是区域内的页索引或是页的线性地址除以PAGE_SIZE
                        }
                    }
                }
            vm_tmp = vm_tmp->vm_next;
        }
        flush_buf(1);    // 写入文件
        mmput(mm);
    }
}

```

result:



3. 多级页表遍历

```
static unsigned long virt2phys(struct mm_struct *mm, unsigned long virt)
{
    struct page *page = NULL;
    // TODO: 多级页表遍历: pgd->pud->pmd->pte, 然后从pte到page, 最后得到pfn
    pgd_t *pgd_tmp = pgd_offset(mm, virt);
    // 每次下一级页表读之前检查这一级是否有效且非空
    if (pgd_none(*pgd_tmp) || pgd_bad(*pgd_tmp))
        return NULL;
    pud_t *pud_tmp = pud_offset((p4d_t *)pgd_tmp, virt); // 强制转换
    if (pud_none(*pud_tmp) || pud_bad(*pud_tmp))
        return NULL;
    pmd_t *pmd_tmp = pmd_offset(pud_tmp, virt);
    if (pmd_none(*pmd_tmp) || pmd_bad(*pmd_tmp))
        return NULL;
    pte_t *pte_tmp = pte_offset_kernel(pmd_tmp, virt);
    if (pte_none(*pte_tmp) || pte_bad(*pte_tmp))
        return NULL;
    page = pte_page(*pte_tmp);
    if (page)
```



```

    {
        return page_to_pfn(page);
    }
    else
    {
        pr_err("func: %s page is NULL\n", __func__);
        return NULL;
    }
}

```

4. func3

```

static void traverse_page_table(struct task_struct *task)
{
    printk("func == 3, %s\n", __func__);
    struct mm_struct *mm = get_task_mm(my_task_info.task);
    if (mm)
    {
        struct vm_area_struct *vm_tmp = mm->mmap;
        while (vm_tmp) // 遍历vma
        {
            unsigned long virt_addr; // 遍历vma中的虚拟地址
            for (virt_addr = vm_tmp->vm_start; virt_addr <= vm_tmp->vm_end ;
virt_addr += PAGE_SIZE)
            { // 页表遍历
                record_two_data(virt_addr, virt2phys(task->mm, virt_addr));
            }
            vm_tmp = vm_tmp->vm_next;
        }
        flush_buf(1);
        mmput(mm);
    }
    else
    {
        pr_err("func: %s mm_struct is NULL\n", __func__);
    }
}

```

```

27 0x55b30f681000--0x0
28 0x55b30f682000--0x0
29 0x55b30f683000--0x0
30 0x55b30f684000--0x0
31 0x55b30f685000--0x133e7a
32 0x55b30f686000--0x1102f9
33 0x55b30f687000--0x118daa
34 0x55b30f688000--0x118633
35 0x55b30f689000--0x10efdd
36 0x55b30f68a000--0x126f18
37 0x55b30f68b000--0x12d9db
38 0x55b30f68c000--0x11020b
39 0x55b30f68d000--0x110219
40 0x55b30f68e000--0x10f6d7
41 0x55b30f68f000--0x11021c
42 0x55b30f690000--0x10f6df
43 0x55b30f691000--0x11da8c
44 0x55b30f692000--0x138d68
45 0x55b30f693000--0x11034d
46 0x55b30f694000--0x123ae9
47 0x55b30f695000--0x13213c
48 0x55b30f696000--0x13590a
49 0x55b30f697000--0x12b2ee
50 0x55b30f698000--0x13370c
51 0x55b30f699000--0x138b05
52 0x55b30f69a000--0x118637
53 0x55b30f69b000--0x126b32
54 0x55b30f69c000--0x117f9c
55 0x55b30f69d000--0x12ed33
56 0x55b30f69e000--0x13651b
57 0x55b30f69f000--0x135bed
58 0x55b30f6a0000--0x10f6af
59 0x55b30f6a1000--0x13043e
60 0x55b30f6a2000--0x13b17f

```

result:

5. func4、5

```

static void print_seg_info(void)
{
    struct mm_struct *mm;
    unsigned long addr;
    printk("func == 4 or func == 5, %s\n", __func__);
    mm = get_task_mm(my_task_info.task);
    if (mm == NULL)
    {
        pr_err("mm_struct is NULL\n");
        return;
    }
    // func4
    for (addr = mm->start_data; addr < mm->end_data; addr += PAGE_SIZE) // 遍历所有地址
    {
        struct vm_area_struct *vm_tmp = find_vma(mm, addr);
        unsigned long virt_addr;
    }
}

```

```
        for (virt_addr = vm_tmp->vm_start; virt_addr <= vm_tmp->vm_end ; virt_addr
+= PAGE_SIZE) // 遍历所有虚拟地址
        {
            struct page *page = mfollow_page(vm_tmp, virt_addr, FOLL_GET); // 得到
虚拟地址对应的page
            if (!IS_ERR_OR_NULL(page)) // page是否可用
            {
                char* virt_addr1 = kmap_atomic(page); // 得到可以直接访问的虚拟地址
                memcpy(buf,virt_addr1,PAGE_SIZE); // 拷贝到buf
                curr_buf_length += PAGE_SIZE;
                flush_buf(0); // buf数据写入文件
                kunmap_atomic(virt_addr1); // 释放
            }
        }
    }
    // func5 修改了addr的首尾地址
    for (addr = mm->start_code; addr < mm->end_code;addr += PAGE_SIZE)
    {
        struct vm_area_struct *vm_tmp = find_vma(mm,addr);
        unsigned long virt_addr;
        for (virt_addr = vm_tmp->vm_start; virt_addr <= vm_tmp->vm_end ; virt_addr
+= PAGE_SIZE)
        {
            struct page *page = mfollow_page(vm_tmp, virt_addr, FOLL_GET);
            if (!IS_ERR_OR_NULL(page))
            {
                char* virt_addr1 = kmap_atomic(page);
                memcpy(buf,virt_addr1,PAGE_SIZE);
                curr_buf_length += PAGE_SIZE;
                flush_buf(0);
                kunmap_atomic(virt_addr1);
            }
        }
    }
    flush_buf(1);
    mmput(mm);
}
```

```
expr_result_4_20230529_1859.txt workload.cc
1=====Scene 1=====[Enter Barnardo and Francisco, two sentinels.]BARNARDO Who's there?-
FRANCISCO Nay, answer me. Stand and unfold yourself.BARNARDO Long live the King!FRANCISCO
Barnardo?BARNARDO He.FRANCISCO You come most carefully upon your hour.BARNARDO 'Tis now struck
twelve. Get thee to bed, Francisco.FRANCISCO For this relief much thanks. 'Tis bitter cold, And
I am sick at heart.BARNARDO Have you had quiet guard?FRANCISCO Not a mouse
stirring.BARNARDO Well, good night.If you do meet Horatio and Marcellus, The rivals of my
watch, bid them make haste.[Enter Horatio and Marcellus.]FRANCISCO I think I hear them.--Stand
ho! Who is there?HORATIO Friends to this ground.MARCELLUS And liegemen to the
Dane.FRANCISCO Give you good night.MARCELLUS O farewell, honest soldier. Who hath relieved you?
FRANCISCO Barnardo hath my place. Give you good night.[Francisco exits.]MARCELLUS Holla,
Barnardo.BARNARDO Say, what, is Horatio there?HORATIO A piece of him.BARNARDO Welcome,
Horatio.--Welcome, good Marcellus.HORATIO What, has this thing appeared again tonight?BARNARDO
I have seen nothing.MARCELLUS Horatio says 'tis but our fantasy And will not let belief take
hold of him Touching this dreaded sight twice seen of us. Therefore I have entreated him
along with us to watch the minutes of this night, That, if again this apparition come, He may
approve our eyes and speak to it.HORATIO Tush, tush, 'twill not appear.BARNARDO Sit down
awhile, And let us once again assail your ears, That are so fortified against our story, What we
have two nights seen.HORATIO Well, sit we down, And let us hear Barnardo speak of
this.BARNARDO Last night of all, When yond same star that's westward from the pole Had made his
course t' illumine that part of heaven Where now it burns, Marcellus and myself, The bell then
beating one--[Enter Ghost.]MARCELLUS Peace, break thee off! Look where it comes
again.BARNARDO In the same figure like the King that's dead.MARCELLUS, [to Horatio] Thou art a
scholar. Speak to it, Horatio.BARNARDO Looks he not like the King? Mark it, Horatio.HORATIO Most
like. It harrows me with fear and wonder.BARNARDO It would be spoke to.MARCELLUS Speak to it,
Horatio.HORATIO What art thou that usurp'st this time of night, Together with that fair and
warlike form In which the majesty of buried Denmark Did sometimes march? By heaven, I charge
thee, speak.MARCELLUS It is offended.BARNARDO See, it stalks away.HORATIO Stay! speak! speak! I
charge thee, speak! [Ghost exits.]MARCELLUS 'Tis gone and will not answer.BARNARDO How now,
Horatio, you tremble and look pale. Is not this something more than fantasy? What think you on
't?HORATIO Before my God, I might not this believe Without the sensible and true avouch Of mine
own eyes.MARCELLUS Is it not like the King?HORATIO As thou art to thyself. Such was the very
armor he had on When he the ambitious Norway combated. So frowned he once when, in an angry
parle, He smote the sledded Polacks on the ice. 'Tis strange.MARCELLUS Thus twice before, and
jump at this dead hour, With martial stalk hath he gone by our watch.HORATIO In what particular
thought to work I know not, But in the gross and scope of mine opinion This bodes some strange
eruption to our state.MARCELLUS Good now, sit down, and tell me, he that knows, Why this same
strict and most observant watch So nightly toils the subject of the land, And why such daily
cast of brazen cannon And foreign mart for implements of war, Why such impress of shipwrights,
whose sore task Does not divide the Sunday from the week. What might be toward that this sweaty
haste Doth make the night joint laborer with the day? Who is 't that can inform me?HORATIO That
can I. At least the whisper goes so: our last king, Whose image even but now appeared to us, Was,
as you know, by Fortinbras of Norway, Thereto prick'd on by a most emulate pride, Dared to the
combat; in which our valiant Hamlet (For so
thi1=====Scene 1=====[Enter Barnardo and Francisco, two sentinels.]BARNARDO Who's there?-
FRANCISCO Nay, answer me. Stand and unfold yourself.BARNARDO Long live the King!FRANCISCO
Barnardo?BARNARDO He.FRANCISCO You come most carefully upon your hour.BARNARDO 'Tis now struck
twelve. Get thee to bed, Francisco.FRANCISCO For this relief much thanks. 'Tis bitter cold, And
I am sick at heart.BARNARDO Have you had quiet guard?FRANCISCO Not a mouse
stirring.BARNARDO Well, good night.If you do meet Horatio and Marcellus, The rivals of my
watch, bid them make haste.[Enter Horatio and Marcellus.]FRANCISCO I think I hear them.--Stand
ho! Who is there?HORATIO Friends to this ground.MARCELLUS And liegemen to the
Dane.FRANCISCO Give you good night.MARCELLUS O farewell, honest soldier. Who hath relieved you?
FRANCISCO Barnardo hath my place. Give you good night.[Francisco exits.]MARCELLUS Holla,
Barnardo.BARNARDO Say, what, is Horatio there?HORATIO A piece of him.BARNARDO Welcome,
Horatio.--Welcome, good Marcellus.HORATIO What, has this thing appeared again tonight?BARNARDO
I have seen nothing.MARCELLUS Horatio says 'tis but our fantasy And will not let belief take
hold of him Touching this dreaded sight twice seen of us. Therefore I have entreated him
along with us to watch the minutes of this night, That, if again this apparition come, He may
approve our eyes and speak to it.HORATIO Tush, tush, 'twill not appear.BARNARDO Sit down
awhile, And let us once again assail your ears, That are so fortified against our story, What we
have two nights seen.HORATIO Well, sit we down, And let us hear Barnardo speak of
this.BARNARDO Last night of all, When yond same star that's westward from the pole Had made his
course t' illumine that part of heaven Where now it burns, Marcellus and myself, The bell then
beating one--[Enter Ghost.]MARCELLUS Peace, break thee off! Look where it comes
again.BARNARDO In the same figure like the King that's dead.MARCELLUS, [to Horatio] Thou art a
scholar. Speak to it, Horatio.BARNARDO Looks he not like the King? Mark it, Horatio.HORATIO Most
like. It harrows me with fear and wonder.BARNARDO It would be spoke to.MARCELLUS Speak to it,
Horatio.HORATIO What art thou that usurp'st this time of night, Together with that fair and
warlike form In which the majesty of buried Denmark Did sometimes march? By heaven, I charge
thee, speak.MARCELLUS It is offended.BARNARDO See, it stalks away.HORATIO Stay! speak! speak! I
charge thee, speak! [Ghost exits.]MARCELLUS 'Tis gone and will not answer.BARNARDO How now,
Horatio, you tremble and look pale. Is not this something more than fantasy? What think you on
't?HORATIO Before my God, I might not this believe Without the sensible and true avouch Of mine
own eyes.MARCELLUS Is it not like the King?HORATIO As thou art to thyself. Such was the very
armor he had on When he the ambitious Norway combated. So frowned he once when, in an angry
parle, He smote the sledded Polacks on the ice. 'Tis strange.MARCELLUS Thus twice before, and
jump at this dead hour, With martial stalk hath he gone by our watch.HORATIO In what particular
thought to work I know not, But in the gross and scope of mine opinion This bodes some strange
eruption to our state.MARCELLUS Good now, sit down, and tell me, he that knows, Why this same
strict and most observant watch So nightly toils the subject of the land, And why such daily
cast of brazen cannon And foreign mart for implements of war, Why such impress of shipwrights,
whose sore task Does not divide the Sunday from the week. What might be toward that this sweaty
haste Doth make the night joint laborer with the day? Who is 't that can inform me?HORATIO That
can I. At least the whisper goes so: our last king, Whose image even but now appeared to us, Was,
as you know, by Fortinbras of Norway, Thereto prick'd on by a most emulate pride, Dared to the
combat; in which our valiant Hamlet (For so
thi
```

4.实验总结

本次实验用两种算法实现了内存分配器，并了解了VMA和页表机制，过程中对Linux系统的内核有了更多的理解。实验内容较多相应时间也比较充足。