

MIT COW

实现功能

只拷贝页表不拷贝内存

修改vm.c/uvmcopy(),实现只拷贝页表不拷贝内存。

注意要实现写保护。

```
15 int
16 uvmcopy(pagetable_t old, pagetable_t new, uint64 sz)
17 {
18     pte_t *pte;
19     uint64 pa, i;
20     uint flags;
21     //char *mem;
22
23     for(i = 0; i < sz; i += PGSIZE){
24         if((pte = walk(old, i, 0)) == 0)
25             panic("uvmcopy: pte should exist");
26         if((*pte & PTE_V) == 0)
27             panic("uvmcopy: page not present");
28         pa = PTE2PA(*pte);
29         *pte &= ~PTE_W; //add for lab cow
30         flags = PTE_FLAGS(*pte);
31         //if((mem = kalloc()) == 0)
32         //    goto err;
33         //memmove(mem, (char*)pa, PGSIZE);
34         if(mappages(new, i, PGSIZE, pa, flags) != 0){
35             //kfree(mem);
36             goto err;
37         }
38     }
39     return 0;
40 }
```

结果如图所示：

```
hart 1 starting
hart 2 starting
init: starting sh
$ cowtest
usertrap(): unexpected scause 0x000000000000000f pid=3
      sepc=0x000000000000009dc stval=0x0000000000004f88
usertrap(): unexpected scause 0x0000000000000002 pid=2
      sepc=0x00000000000001000 stval=0x0000000000000000
usertrap(): unexpected scause 0x000000000000000c pid=1
      sepc=0x00000000000000f96 stval=0x00000000000001000
panic: init exiting
```

错因查找

根据报错信息，scause=15，pid=3. 考虑到应该是写保护导致的page fault，应该是在执行cowtest或者是shell代码时候报错。根据sepc分别在 `user/cowtest.asm` 和 `user/sh.asm` 中查看 9dc 的指令，结果分别如下：

```
break;
9d8: bf7d j 996 <nulterminate+0x54>

00000000000009da <parsecmd>:
{
    9da: 7179 addi sp,sp,-48
    9dc: f406 sd ra,40(sp)
    9de: f022 sd s0,32(sp)
    9e0: ec26 sd s1,24(sp)
    9e2: e84a sd s2,16(sp)
    9e4: 1800 addi s0,sp,48
    9e6: fca43c23 sd a0,-40(s0)
    9e8: 00000000 ret

    9d4: e56080e7 jalr -426(ra) # 826 <printint>
    9d8: 8b4a mv s6,s2
    state = 0;
    9da: 4981 li s3,0
    9dc: bf91 j 930 <vprintf+0x60>
    printptr(fd, va_arg(ap, uint64));
    9de: 008b0793 addi a5,s6,8
    9e2: f8f43423 sd a5,-120(s0)
    9e6: 000b3983 ld s3,0(s6)
    putc(fd, '0');
    9ea: 03000593 li a1,48
    9ee: 8556 mv a0,s5
```

在shell中是sd指令，可见page fault是在shell fork之后调用exec()之前发生的。

trap中处理写保护page fault情况

在 `trap.c/usertrap()` 中增加写保护情况时的处理，scause=15. 单独写一个函数 `cowfault()` 来处理 cow fork 的写保护page fault问题。

```
00  syscall();
01  } else if(r_scause() == 15){
02      if(cowfault(p->pagetable,r_stval()) < 0){
03          p->killed = 1;
04      }
05  }else if((which_dev = devintr()) != 0){
06      // ok
07  } else {
```

```

-
2 //add for lab6 cow
3 int
4 cowfault(pagetable_t pagetable, uint64 va)
5 {
6     if(va >= MAXVA)
7         return -1;
8     pte_t *pte = walk(pagetable, va, 0);
9     if(pte == 0)
10        return -1;
11    if((*pte & PTE_U) == 0 || (*pte & PTE_V) == 0)
12        return -1;
13
14    uint64 pa1 = PTE2PA(*pte);
15    uint64 pa2 = (uint64)kalloc();
16    if(pa2 == 0){
17        printf("cow kalloc failed\n");
18        return -1;
19    }
20    memmove((void*)pa2, (void*)pa1, 4096);
21    *pte = PA2PTE(pa2) | PTE_V | PTE_U | PTE_W | PTE_X;
22    return 0;
23 }
24

```

结果

```

: init: starting sh
$ cowtest
: cow kalloc failed
: usertrap(): unexpected scause 0x0000000000000002 pid=2
:      sepc=0x0000000000000100 stval=0x0000000000000000
: usertrap(): unexpected scause 0x0000000000000002 pid=1
:      sepc=0x0000000000000396 stval=0x0000000000000000
: panic: init exiting
:

```

错因查找

scause=2表示非法指令。因为在该处理page fault的时候我是直接分配了新的内存和页表，父进程shell也一样。推测是因为重新分配内存指向了垃圾指令。

释放不用的页面

定义一个页面引用refcnt来count一个物理页面的使用进程数，便于释放不用的页面。

在 kernel.c 中定义即可，同时需要满足加操作，便于cow fork的时候调用（因为没有用到kalloc）。

```

25
26 int refcnt[PHYSTOP/PGSIZE];
27 void
28 increfcnt(uint64 pa)
29 {
30     int pn = pa/PGSIZE;
31     acquire(&kmem.lock);
32     if(pa >= PHYSTOP || refcnt[pn] < 1)
33         panic("panic incref");
34     refcnt[pn] += 1;
35     release(&kmem.lock);
36 }
37

```

在uvmcopy(), kalloc(), kfree()中添加refcnt相关操作。注意并发。

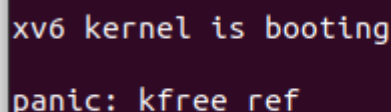
```
304 // free any allocated pages on return.
305 int
306 uvmcopy(pagetable_t old, pagetable_t new, uint64 sz)
307 {
308     pte_t *pte;
309     uint64 pa, i;
310     uint flags;
311     //char *mem;
312
313     for(i = 0; i < sz; i += PGSIZE){
314         if((pte = walk(old, i, 0)) == 0)
315             panic("uvmcopy: pte should exist");
316         if((*pte & PTE_V) == 0)
317             panic("uvmcopy: page not present");
318         pa = PTE2PA(*pte);
319         *pte &= ~PTE_W; //add for lab cow
320         flags = PTE_FLAGS(*pte);
321         //if((mem = kalloc()) == 0)
322             // goto err;
323         //memmove(mem, (char*)pa, PGSIZE);
324         increfcnt(pa); //add for cow
325         if(mappages(new, i, PGSIZE, pa, flags) != 0){
326             //kfree(mem);
327             goto err;
328         }
329     }
330     return 0;
331
332 void ~
333 kalloc(void)
334 {
335     struct run *r;
336
337     acquire(&kmem.lock);
338     r = kmem.freelist;
339     if(r){
340         kmem.freelist = r->next;
341         int pn = (uint64)r/PGSIZE;
342         if(refcnt[pn] != 0)
343             panic("kalloc ref!");
344         refcnt[pn] = 1;
345     }
346     release(&kmem.lock);
347     if(r)
348         memset((char*)r, 5, PGSIZE); // fill with junk
349     return (void*)r;
350 }
```

```

60 // initializing the allocator; see kinit above.)
61 void
62 kfree(void *pa)
63 {
64     struct run *r;
65
66     if(((uint64)pa % PGSIZE) != 0 || (char*)pa < end || (uint64)pa >= PHYSTOP)
67         panic("kfree");
68
69     acquire(&kmem.lock);
70     int pn = (uint64) pa/PGSIZE;
71     if(refcnt[pn] < 1)
72         panic("kfree ref");
73     refcnt[pn] -= 1;
74     int tmp = refcnt[pn];
75     release(&kmem.lock);
76
77     if(tmp > 0)
78         return;
79     // Fill with junk to catch dangling refs.
80     memset(pa, 1, PGSIZE);
81
82     r = (struct run*)pa;
83
84     acquire(&kmem.lock);
85     r->next = kmem.freelist;
86     kmem.freelist = r;
87     release(&kmem.lock);
88 }

```

结果



```

xv6 kernel is booting
panic: kfree ref

```

错因查找

刚启动就报kfree ref错误，可见是kinit的问题。注意到初始化调用的freerange()中会调用kfree，但此时的ref都为0，所以前面kfree对ref的操作没考虑到这一点。

修正

```

45 void
46 freerange(void *pa_start, void *pa_end)
47 {
48     char *p;
49     p = (char*)PGROUNDUP((uint64)pa_start);
50
51     for(; p + PGSIZE <= (char*)pa_end; p += PGSIZE){
52         refcnt[(uint64)p/PGSIZE] = 1;
53         kfree(p);
54     }
55 }
56

```

。。。写一半 我先去复习。。