

Virtual Memory in xv6

Satoru Yamamoto

05/11/2022

Contents

1	Null pointer dereference on xv6 and Linux	1
2	Implementation	1
3	Result	3

1 Null pointer dereference on xv6 and Linux

The first thing I did is to create a program that dereferences a null pointer.

testcall2.c

```
1 int main(int argc, char *argv[])
2 {
3     int* null_p = (int*)0;
4     printf(1, "Null pointer deference\n");
5     printf(1, "Return: %d\n\n", *null_p);
6     exit();
7 }
```

The result of running program on Linux and xv6 is here.*1 *2

We can see that the result on xv6 displays does not return an exception for null pointer dereference.

2 Implementation

To make sure each page size is 4096 (0x1000) bytes, I checked mmu.h file.*3

To change the user programs' entry point to make the first page invalid for Null pointer dereference, I edited Makefile.

Makefile

```
1  # change line 149
2  _%: %.o $(ULIB)
3  $(LD) $(LDFLAGS) -N -e main -Ttext 0x1000 -o $$@ $^
4  $(OBJDUMP) -S $$@ > $.asm
5  $(OBJDUMP) -t $$@ | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$$/d' > $.sym
6  # change line 156
7  _forktest: forktest.o $(ULIB)
8  # forktest has less library code linked in - needs to be small
9  # in order to be able to max out the proc table.
10 $(LD) $(LDFLAGS) -N -e main -Ttext 0x1000 -o _forktest forktest.o ulib.o usys.o
11 $(OBJDUMP) -S _forktest > forktest.asm
```

In the part of loading program into memory in exec function, sz is used to call allocuvm function.

exec.c

```
1 if((sz = allocuvm(pgdir, sz, ph.vaddr + ph.memsz)) == 0)
```

allocuvm function allocates page tables and physical memory to grow process from oldsz to newsz.

vm.c

```
1 int allocuvm(pde_t *pgdir, uint oldsz, uint newsz)
```

Therefore, I changed sz=0 to sz = PGSIZE to skip the first page.

In fork function, it calls copyuvm function to create copy process state from a parent process's page table.

vm.c

```
1 for(i = 0; i < sz; i += PGSIZE){
2     if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
3         panic("copyuvm: pte should exist");
4     if(!(*pte & PTE_P))
5         panic("copyuvm: page not present");
6     pa = PTE_ADDR(*pte);
7     flags = PTE_FLAGS(*pte);
8     if((mem = kalloc()) == 0)
9         goto bad;
10    memmove(mem, (char*)P2V(pa), PGSIZE);
11    if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags) < 0) {
12        kfree(mem);
13        goto bad;
14    }
15 }
```

I run xv6 to see what error message the program return. It displays "copyuvm: page not present" because PTR_P flag is not set for the parent table.*4

The description of walkpgdir function says that "Return the address of the PTE in page table pgdir that corresponds to virtual address va," and it seems like copyuvm function passes the virtual address of first page (i=0) to call walkpgdir functoin. It is causing an issue, so I changed i=0 to i=PGSIZE to skip the first page.

After these changes, the system shows an exception for null pointer dereference program.5

However, I failed validatetest in usertests. This test tries to crash the kernel by passing in a badly placed integer, so I added passPointerHelper functoin that tests if pointer is bad.

syscall.c

```
1 int
2 passPointerHelper(uint addr)
3 {
4     struct proc *curproc = myproc();
5     // this is init process
6     if(curproc->pid == 1){
7         return 1;
8     }
9
10    if(addr < PGSIZE){
11        return 0;
12    }else{
13        return 1;
14    }
15
16 }
```

Then I added this condition to fetchint function and fetchstr function in syscall.c to ensure it does not pass a bad pointer.

```
1 if(!passPointerHelper(addr)){
2     return -1;
3 }
```

3 Result

Here is the result of null pointer dereference program and usertests.*5 *6

Figure 1: null test linux

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

→ xv6-public git:(master) X ./testcall2
[1] 126326 segmentation fault (core dumped) ./testcall2
→ xv6-public git:(master) X

```

Figure 2: null test xv6

```

$ testcall2
Null pointer deference
Return: 69487757

$
» os2.engr.oregonstate.edu:✓

```

Figure 3: page size in xv6

```

h mmu.h x
clones > xv6-public > h mmu.h
75
76 // page table index
77 #define PTX(va) (((uint)(va) >> PTXSHIFT) & 0x3FF)
78
79 // construct virtual address from indexes and offset
80 #define PGADDR(d, t, o) ((uint)((d) << PDXSHIFT | (t) << PTXSHIFT | (o)))
81
82 // Page directory and page table constants.
83 #define NPENTRIES 1024 // # directory entries per page directory
84 #define NPTENTRIES 1024 // # PTEs per page table
85 #define PGSIZE 4096 // bytes mapped by a page
86
87 #define PTXSHIFT 12 // offset of PTX in a linear address
88 #define PDXSHIFT 22 // offset of PDX in a linear address
89
90 #define PGROUNDUP(sz) (((sz)+PGSIZE-1) & ~(PGSIZE-1))
91 #define PGROUNDDOWN(addr) ((addr) & ~(PGSIZE-1))

```

Figure 4: test run

```

init: starting sh
lapicid 0: panic: copyvm: page not present
8010819a 801045e3 80106407 8010564d 801067a4 80106598 0 0 0 0

```

Figure 5: result

```
$ forktest
fork test
fork test OK
$ testcall2
Null pointer deference
pid 68 testcall2: trap 14 err 4 on cpu 0 eip 0x102d addr 0x0--kill proc
$
```

Figure 6: result usertests

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

read succeeded ok
small file test ok
big files test
big files ok
many creates, followed by unlink test
many creates, followed by unlink; ok
openiput test
openiput test ok
exitiput test
exitiput test ok
iput test
iput test ok
mem test
allocuvn out of memory
mem ok
pipe1 ok
preempt: kill... wait... preempt ok
exitwait ok
rmdot test
rmdot ok
fourteen test
fourteen ok
bigfile test
bigfile test ok
subdir test
subdir ok
linktest
linktest ok
unlinkread test
unlinkread ok
dir vs file
dir vs file OK
empty file name
empty file name OK
fork test
fork test OK
bigdir test
bigdir ok
uio test
pid 592 usertests: trap 13 err 0 on cpu 0 eip 0x4b79 addr 0x801dc130--kill proc
uio test done
exec test
ALL TESTS PASSED
$
» os2.engr.oregonstate.edu:✓
```