

Lab 2 Report

Christina Pavlopoulou

cpavl001@ucr.edu

Niloufar Hosseini Pour

nhoss003@ucr.edu

Andres Calderon

acald013@ucr.edu

February 23, 2016

1 Null pointer at command line.

In xv6, user code is loaded into the first part of the address space. Accordingly, when we dereference a null pointer, because we start from the first page, first page is accessible for us so we will not get an exception, instead, we see some garbage value as the content of the dereferenced variable.

For resolving this problem, we changed 3 files (`exec.c`, `vm.c` and `Makefile`) in order to start from second page in the page table instead of first page.

1. `exec.c`: In `exec.c` file, we changed size instead of 0 to 4096 (`PGSIZE`), so address space will get filled from second page. We make the change at line 39 of listing 1 (`sz=PGSIZE` instead of `sz=0`).
2. `vm.c`: In `copyvm` function, where a copy of the parent process's page table is given to the child, we start the `for` loop from second page (`PGSIZE`) instead of 0. See line 319 at listing 2.
3. `Makefile`: In `Makefile`, entry point of user programs is set. So, we have to make the first page invalid, meaning where the first instruction is set to 0, and change the entry point to the next page at "0x1000". So we set `init` and `forktest` to start from second page (0x1000). See lines 140 and 147 at listing 3.

After changing all these, we tested our user code (a code that tries to access a null pointer) shown in listing 4, and now we see that the process has been trapped and killed (figure 1).

```
38 // Load program into memory.
39 sz = PGSIZE;
40 for(i=0, off=elf.phoff; i<elf.phnum; i++, off+=sizeof(ph)){
41     if(readi(ip, (char*)&ph, off, sizeof(ph)) != sizeof(ph))
42         goto bad;
43     if(ph.type != ELF_PROG_LOAD)
44         continue;
45     if(ph.memsz < ph.filesz)
46         goto bad;
47     if((sz = allocvm(pgdir, sz, ph.vaddr + ph.memsz)) == 0)
48         goto bad;
49     if(loadvm(pgdir, (char*)ph.vaddr, ip, ph.off, ph.filesz) < 0)
50         goto bad;
51 }
```

Listing 1: Changes in `exec.c` file.

```

307 // Given a parent process's page table, create a copy
308 // of it for a child.
309 pde_t*
310 copyuvm(pde_t *pgdir, uint sz)
311 {
312     pde_t *d;
313     pte_t *pte;
314     uint pa, i, flags;
315     char *mem;
316
317     if((d = setupkvm()) == 0)
318         return 0;
319     for(i = PGSIZE; i < sz; i += PGSIZE){
320         if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
321             panic("copyuvm: pte should exist");
322         if(!(*pte & PTE_P))
323             panic("copyuvm: page not present");
324         pa = PTE_ADDR(*pte);
325         flags = PTE_FLAGS(*pte);
326         if((mem = kalloc()) == 0)
327             goto bad;
328         memmove(mem, (char*)p2v(pa), PGSIZE);
329         if(mappages(d, (void*)i, PGSIZE, v2p(mem), flags) < 0)
330             goto bad;
331     }
332     return d;
333
334 bad:
335     freevm(d);
336     return 0;
337 }

```

Listing 2: Changes in vm.c file.

```

139 _%: %.o $(ULIB)
140 $(LD) $(LDFLAGS) -N -e main -Ttext 0x1000 -o $$ $~
141 $(OBJDUMP) -S $$ > $.asm
142 $(OBJDUMP) -t $$ | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$$/d' > $.sym
143
144 _forktest: forktest.o $(ULIB)
145 # forktest has less library code linked in - needs to be small
146 # in order to be able to max out the proc table.
147 $(LD) $(LDFLAGS) -N -e main -Ttext 0x1000 -o _forktest forktest.o ulib.o usys.o
148 $(OBJDUMP) -S _forktest > forktest.asm

```

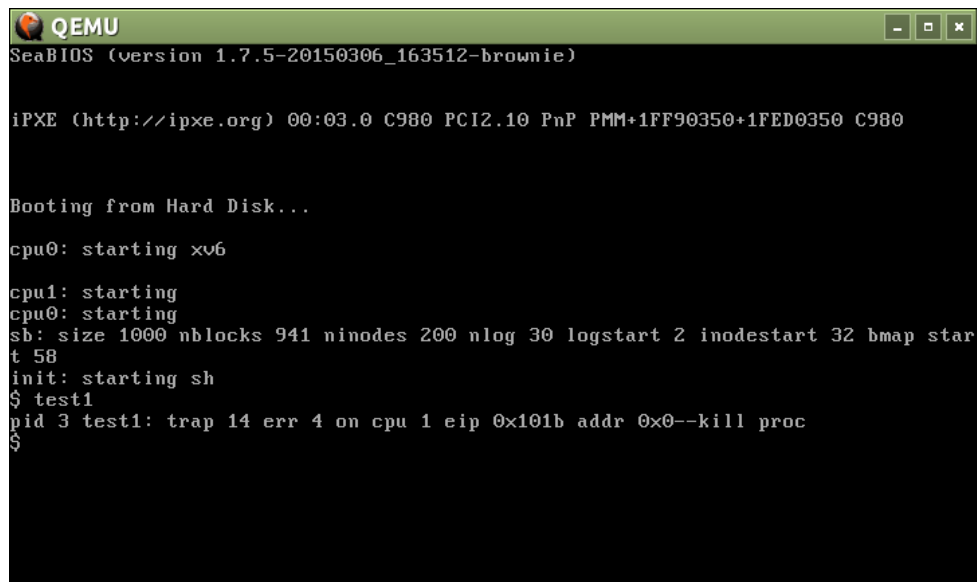
Listing 3: Changes in Makefile.

```

1 #include "types.h"
2 #include "user.h"
3 #include "syscall.h"
4
5 int main(){
6     int *p = 0;
7
8     printf(1, "%d\n", *p);
9     exit();
10 }

```

Listing 4: Test for null pointer catching at command line (test1.c file).

A screenshot of a QEMU terminal window. The title bar is green with the QEMU logo and the text "QEMU". The terminal content shows the SeaBIOS boot process. It starts with "SeaBIOS (version 1.7.5-20150306_163512-brownie)", followed by "iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF90350+1FED0350 C980". Then it says "Booting from Hard Disk...". The boot process continues with "cpu0: starting xv6", "cpu1: starting", "cpu0: starting", "sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58", "init: starting sh", "\$ test1", and finally "pid 3 test1: trap 14 err 4 on cpu 1 eip 0x101b addr 0x0--kill proc" followed by a prompt "\$".

```
SeaBIOS (version 1.7.5-20150306_163512-brownie)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF90350+1FED0350 C980

Booting from Hard Disk...

cpu0: starting xv6
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ test1
pid 3 test1: trap 14 err 4 on cpu 1 eip 0x101b addr 0x0--kill proc
$
```

Figure 1: Output of test 1.

2 Null pointer at system call.

For the second part of lab 2, we made some changes in `syscall.c` file. Specifically, we added an `if` clause in function `argptr` to check if we point at 0. If this is true, then we have an exception. The change is shown between lines 64 and 67 at listing 5. The reason that we make these changes is that now that we changed `xv6` so that each process does not start from the zero address, we, also, need to ensure that we pass to kernel a correct pointer via system calls. Without the changes, `xv6` checks only if the pointer lies between the process address space. However, we should add a check that returns an exception if the pointer points to 0.

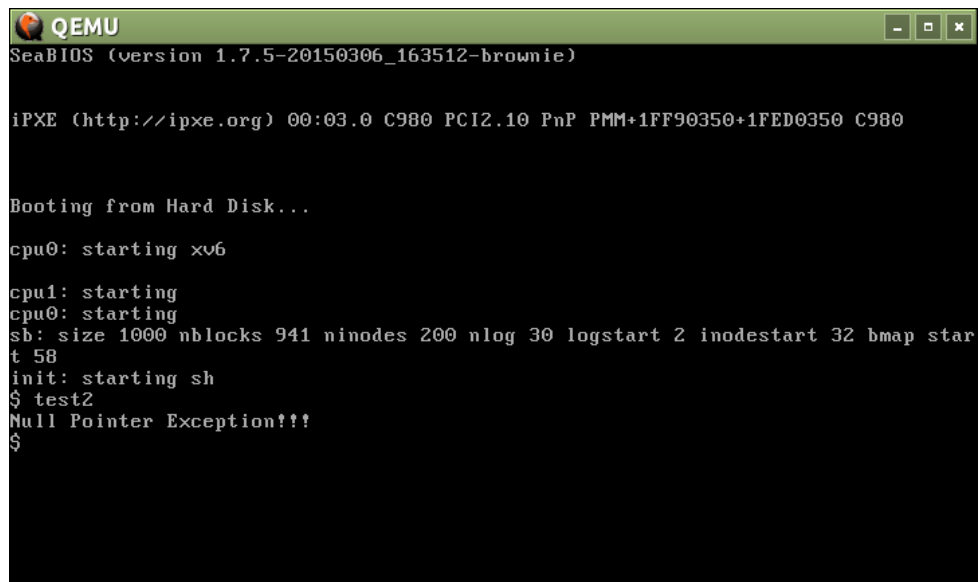
We tested our implementation by executing a system call that passes a pointer to 0, using the code in listing 6, and we got an exception. The output can be seen in figure 2. However, when we pass pointer to somewhere else (inside the process address space) the system call is executed normally.

```
51 // Fetch the nth word-sized system call argument as a pointer
52 // to a block of memory of size n bytes. Check that the pointer
53 // lies within the process address space.
54 int
55 argptr(int n, char **pp, int size)
56 {
57     int i;
58
59     if(argint(n, &i) < 0)
60         return -1;
61     if((uint)i >= proc->sz || (uint)i+size > proc->sz)
62         return -1;
63     *pp = (char*)i;
64     if(*pp == 0){
65         cprintf("Null Pointer Exception!!!\n");
66         return -1;
67     }
68     return 0;
69 }
```

Listing 5: Changes in `syscall.c` file.

```
1  #include "types.h"
2  #include "user.h"
3  #include "syscall.h"
4
5  int main(){
6      int *p = 0;
7
8      null(p);
9      exit();
10 }
```

Listing 6: Test for null pointer catching at system call (`test2.c` file).

A screenshot of a QEMU terminal window. The title bar is green with the QEMU logo and the text "QEMU". The terminal content is as follows:

```
SeaBIOS (version 1.7.5-20150306_163512-brownie)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF90350+1FED0350 C980

Booting from Hard Disk...

cpu0: starting xv6
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
$ test2
Null Pointer Exception!!!
$
```

Figure 2: Output of test 2.