Virtual Memory in xv6

Satoru Yamamoto

05/11/2022

Contents

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

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

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

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

```
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

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

Therefore, I changed sz=0 to sz = PGSIZE to skip the fist page.
In fork function, it calls copyuvm function to create copy process state from a parent process's page table.

vm.c

```
for(i = 0; i < sz; i += PGSIZE){
    if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
      panic("copyuvm: pte should exist");
    if(!(*pte & PTE_P))
      panic("copyuvm: page not present");
    pa = PTE_ADDR(*pte);
    flags = PTE_FLAGS(*pte);
    if((mem = kalloc()) == 0)
      goto bad;
    memmove(mem, (char*)P2V(pa), PGSIZE);
    if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags) < 0) {
      kfree(mem);
      goto bad;
    }
 }
```

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

```
int
passPointerHelper(uint addr)
{
  struct proc *curproc = myproc();
  // this is init process
  if(curproc->pid == 1){
    return 1;
  }

  if(addr < PGSIZE){
    return 0;
  }else{
    return 1;
  }

}
```

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

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

3 Result

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

Figure 1: null test linux

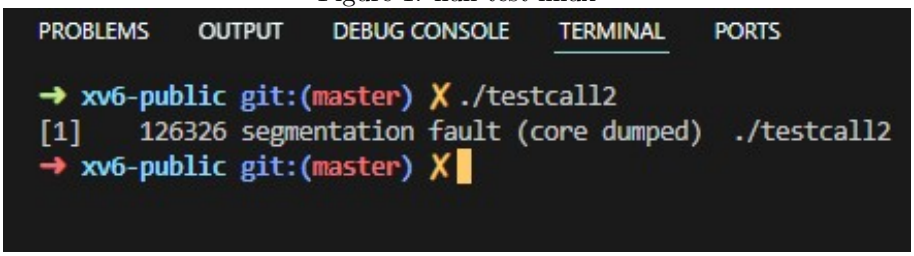

Figure 2: null test xv6

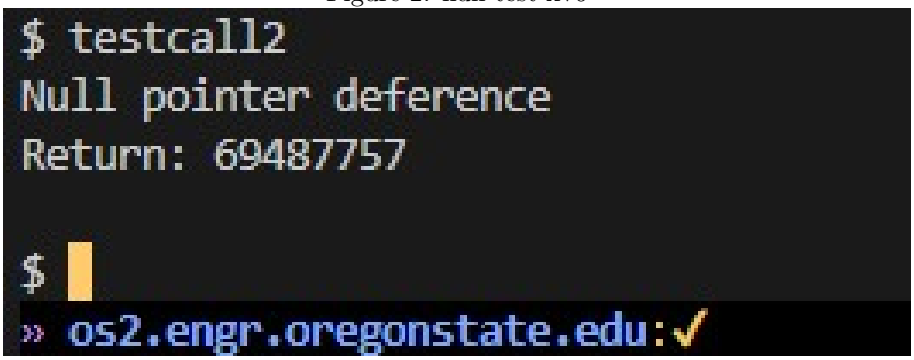

Figure 3: page size in xv6

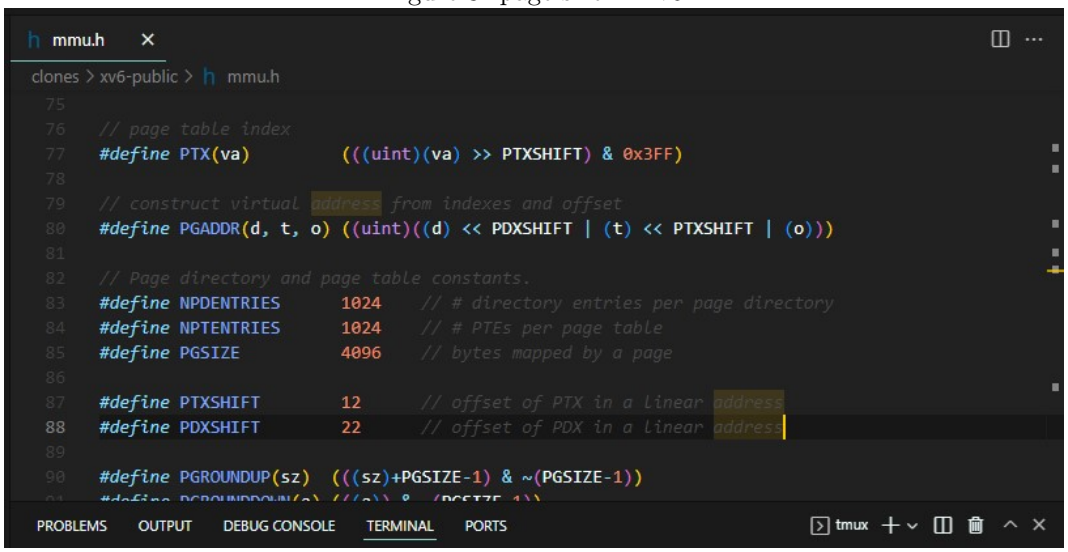

Figure 4: test run

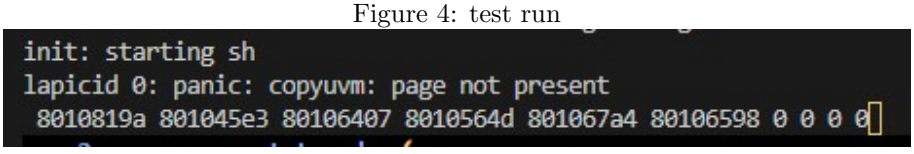

Figure 5: result

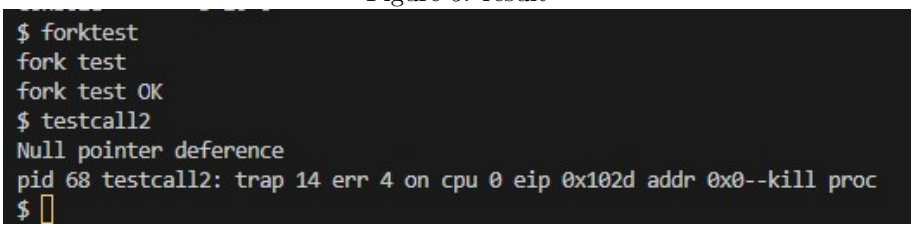

Figure 6: result usertests