

Part A

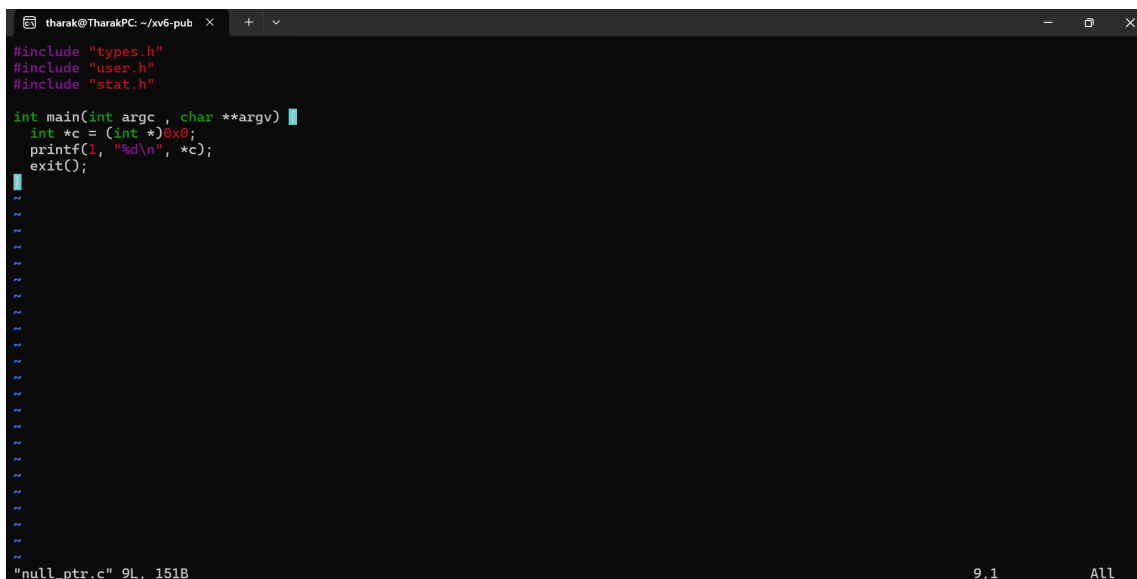
Null pointer dereference in Linux and xv6

- In both Linux and xv6, if we try to use a null pointer, it results in a segmentation fault. A null pointer that doesn't point to any valid memory location. When a program tries to read from or write to the memory at address 0, the operating system recognizes this as an attempt to access invalid memory and responds by triggering a segmentation fault.
- In XV6, when a null pointer is dereferenced, the operating system responds by generating a signal, and in this situation, it may lead to a kernel panic within the 0x3000 address space.

Dereferencing 3 pages

- Dereferencing a page is translating a virtual address to a physical address during memory access. a loop that iterates through three consecutive pages in the user space and dereferences the corresponding pointers.
- In the process of dereferencing three pages in xv6, the fundamental functionalities like cat, mkdir, and basic hello world programs should still work.

If we look at the code at null pointer dereference initially it looks like



```
tharak@TharakPC: ~/xv6-pub
#include "types.h"
#include "user.h"
#include "stat.h"

int main(int argc, char **argv)
{
    int *c = (int *)0x0;
    printf(1, "%d\n", *c);
    exit();
}

"null_ptr.c" 9L, 151B
```

This results in trap 6 error which is shown in output folder. It looks like this:

pid 3 null_ptr: trap 6 err 0 on cpu 0 eip 0x5 addr 0x0--kill proc

When we change the first three pages (0X3000). There need to be made changes in xv6 in order to implement.

- There needs to changes made in exec.c file. Here it is

```
// Load program into memory.
sz = PGSIZE;
for(i=0, off=elf.phoff; i<elf.phnum; i++, off+=sizeof(ph)){
    if(readi(ip, (char*)&ph, off, sizeof(ph)) != sizeof(ph))
        goto bad;
    if(ph.type != ELF_PROG_LOAD)
        continue;
    if(ph.memsz < ph.filesz)
```

In this size sz changed from 0 to PGSIZE

- Now in syscall.c i==0 needs to be added

```
// Fetch the nth word-sized system call argument as a pointer
// to a block of memory of size bytes. Check that the pointer
// lies within the process address space.
int
argptr(int n, char **pp, int size)
{
    int i;
    struct proc *curproc = myproc();

    if(argint(n, &i) < 0)
        return -1;
    if(size < 0 || (uint)i >= curproc->sz || (uint)i+size > curproc->sz || i == 0)
        return -1;
    *pp = (char*)i;
    return 0;
}

// Fetch the nth word-sized system call argument as a string pointer.
```

- We also need to change i value to PGSIZE for page reference in copyvm() in vm.c

```
pde_t*
copyvm(pde_t *pgdir, uint sz)
{
    pde_t *d;
    pte_t *pte;
    uint pa, i, flags;
    char *mem;

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

- Now change the pages in Makefile as required. Here we first 3 pages so 0X3000

```

tharak@TharakPC: ~/xv6-pub
tags: $(OBJJS) entryother.S _init
      etags *.S *.c

vectors.S: vectors.pl
| ./vectors.pl > vectors.S

ULIB = ulib.o usys.o printf.o umalloc.o

_%: %.o $(ULIB)
$(LD) $(LDFLAGS) -N -e main -Ttext 0X3000 -o $@ $^
$(OBJDUMP) -S $@ > $.asm
$(OBJDUMP) -t $@ | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$$/d' > $.sym

_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 0X3000 -o _forktest forktest.o ulib.o usys.o
$(OBJDUMP) -S _forktest > forktest.asm

mkfs: mkfs.c fs.h
      gcc -Werror -Wall -o mkfs mkfs.c

```

Now after implementing all these changes clean using 'make clean' and run qemu.

- We will get trap 14 error which is in output folder and looks like this:
pid 4 nullptr: trap 14 err 4 on cpu 0 eip 0x3000 addr 0x0--kill proc