Close()

First, try to run following code:

```
int main(int argc, char* argv[]) {
        close(10);
        return 0;
}
```

close() is defclared in user.h line 11, yet its definition is assembly code in usys.S. After macro replacement, it's like,

```
#define SYSCALL(close) \
   .globl close; \
   close: \
      movl $21, %eax; \    // #define SYS_close    21, in syscall.h
      int $0x40; \ //            #define T_SYSCALL    64          // system call, in traps.h 64=0x40
      ret
```

Here, int $0x40 is an interruption call with interrupt number 0x40. Interruption is called trap in xv6. Next, we need to check interrupt table to find want int 64 will do. The interrupt table is set when the kernel boots. As for syscall, we can see it in trap.c line 24,

```
SETGATE(idt[T_SYSCALL], 1, SEG_KCODE<<3, vectors[T_SYSCALL], DPL_USER);
```

SEG_KCODE<<3 means the code is in **kernel** mode.

vectors[64] is an assemble function defined in vector.S

```
.globl vector65
vector64:
   pushl $0
   pushl $64
   jmp alltraps // this is the function called when execute int $0x40
```

In trapasm.S, we can find the definition of alltraps,

```
…
# Call trap(tf), where tf=%esp
   pushl %esp
   call trap //    Next function will be called
   addl $4, %esp
…
```

In trap.c, we can find the definition of trap(tf)

```
void
trap(struct trapframe *tf){
   …
      syscall();
    … }
```

Then, we should go to syscall(). in syscall.c Line 141, we can find

```
curproc->tf->eax = syscalls[num](); //syscalls[21] is sys_close.
```

This line means call sys_close then store return value to, curproc->tf->eax. Currproc is

current running process, our program at the first place.

int sys_close(void) is defined in sysfile.c,line 93

```
int sys_close(void){
   …
   if(argfd(0, &fd, &f) < 0)// fetch the argument of the sys_close call
      return -1;
   myproc()->ofile[fd] = 0;
   fileclose(f);
   return 0;
}
```

Here, the first thing to do is fetch the argument we input when calling close(10). And we now go to **argfd**.

```
static int
argfd(int n, int *pfd, struct file **pf)
{
   int fd;
   struct file *f;

   if(argint(n, &fd) < 0)
      return -1;
   if(fd < 0 || fd >= NOFILE || (f=myproc()->ofile[fd]) == 0)      //ERROR!
      return -1;
   if(pfd)
      *pfd = fd;
   if(pf)
      *pf = f;
   return 0;
}
```

f=myproc()->ofile is a file* array, contanning all the open files in the current process.
The file we want close **ofile[10]** is 0 (NULL), thus the **argfd** will return -1, and **sys_close** return -1, **trap** will returns to **alltraps.**
Alltraps executes **iret** and restores registers from tf.
The program now returns to user_mode with return value -1.