Changes

Before telling you about which files that I change, let take a look at how things working in the xv6. The first thing is forking a new process, as a memory perspective, the important function is "copyuvm" that you can find it in a vm.c. Also we should change the location that program loads in exec() from 0 to PGSIZE. To stop compiler from load into page 0 we should also change the make file.

1) **vm.c**: Instead of staring copying in copyuvm from page 0, i.e. 'i=0', we should start from i=PGSIZE which is the next page, i.e. page 1;

```
for(i = PGSIZE; i < sz; i += PGSIZE){
```

2) **exec.c**: Load program into page 1 of the memory (userspace);

```
sz = PGSIZE;
```

3) **MakeFile**: Change start execution address to 0x1000 (4096) [each page is 4k]

```
_%: %.o $(ULIB)
    $(LD) $(LDFLAGS) -N -e main -Ttext 0x1000 -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 0x1000 -o _forktest forktest.o ulib.o usys.o
    $(OBJDUMP) -S _forktest > forktest.asm
```

These changes are enough for Null pointer dereference without kernel, but when we need have a syscall, we should change the required code for add syscall (like lab 1), plus additional changes in the syscall.c for checking pointers before passing (argptr());

4) **Sysproc.c:**

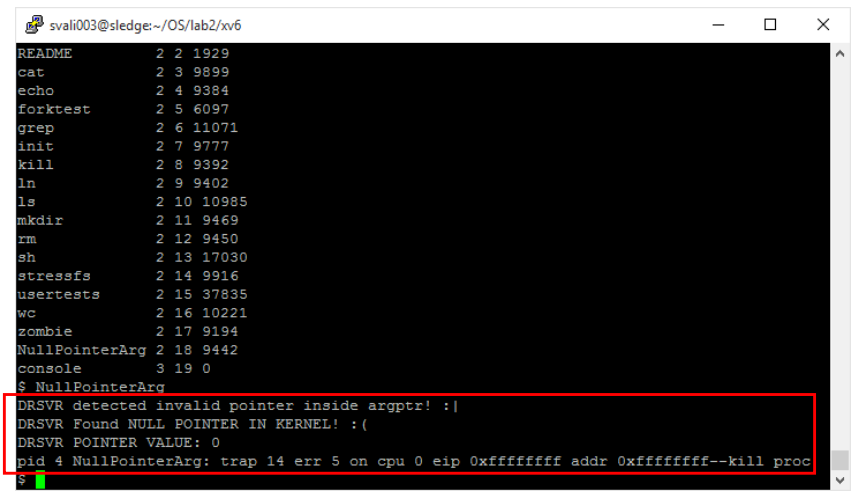
```
int
sys_getdrsvrptr(void)
{
    char  *pointer ;

    if(argptr(0, &pointer,1) < 0) {
            cprintf("DRSVR Found NULL POINTER IN KERNEL! :( \n");
            return -1;
            }
  else {
            cprintf("DRSVR Pointer has been set in kernel! :) \n");
     return 0;
    }
}
```

5) **Syscall.c**:

```
if((uint)i >= proc->sz || (uint)i+size > proc->sz || (uint)i == 0 ) {
    // DRSVR - CS202 - LAB2
    cprintf("DRSVR detected invalid pointer inside argptr! :| \n");
  return -1;
 }
```

6) **Proc.h, Syscall.h, User.h, Usys.s, Defs.h; [For adding Syscall]**
7) **NullPointerArg.c:** User code for testing my job ☺

```
#include "types.h"
#include "user.h"

int main(int argc, char* argv[]){

   int  * myPointer = 0; // for null dereference
   char * myPointer2 = 0; // for kernel test

   getdrsvrptr(myPointer2); // syscall kernel test

   printf(1,"DRSVR POINTER VALUE: %d\n",myPointer); // user test

   return 0;
                                        }
```



You can find my changes by searching for DRSVR or CS202 inside the code ☺ As you can see process has been trapped and killed because of Null pointer dereference.