Part1

Christina Pavlopoulou
`cpavl001@ucr.edu`

Andres Calderon
`acald013@ucr.edu`

Niloufar Hosseini Pour
`nhoss003@ucr.edu`

January 30, 2016

1 What files we changed to solve each part?

we changed the following files:
1- proc.h
2- proc.c
3- syscall.h
4- syscall.c
5- sysproc.c 6- user.h
7- defs.h
8- usys.S
9- Makefile

1- In proc.h file we added an integer variable inside the proc structure.

```
struct proc {

    int counter;
   .
   .
   .
}
```

2- In proc.c file, inside the fork() function, after allocating process, we initialized counter variable of proc structure to zero.

```
int fork(void){

    struct proc *np;
   \ \ Allocate process
    if((np = allocproc()) == 0)
    return -1;
```

```
        np -> counter=0;
}
```

3- In syscall.h file, we define the position of the system call vector that connect to our implementation.

```
    # define SYS_ counter 22
```

4- In syscall.c file,we define externally the function that connect the shell and the kernel, use the position defined in syscall.h to add the function to the system call vector.

```
    extern int sys_ counter(void);

  static int (*syscalls[])(void) = {
      .
      .
      .
     [SYS_ counter] sys_ counter, };

  void
     syscall(void)
     {
        sys_ counter();
     }
```

5- In sysproc.c, we added the real implementation of our system call method.

```
int
sys_ counter(void){
     proc-> counter++;
     return proc->counter;
}
```

6- In user.h file, we defined the function that can be called through the shell. Our system call function prototype.

```
        int counter(void);
```

7- In defs.h file, we added a forward declaration for our new system call under proc.c section.

\ \ proc.c
int   counter(void);

8- In USYS.S, we used the macro to connect the call of user to the system call function.

```
    SYSCALL(counter)
```

9- In Makefile file, we told make how to compile and link the program. Under UPROGS=\ section we added our program.

```
UPROGS=\
    _count\
```

2 What files we added?

We added count.c file as follows:

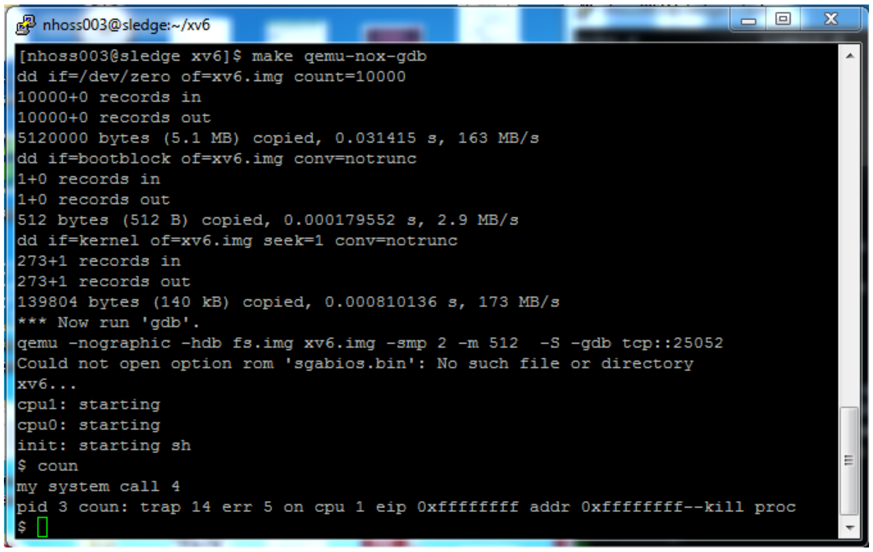
```
#include "types.h"
#include "user.h"
#include "syscall.h"

    int main()
    printf(1,"my system call %d \ n", counter());
    return 0;
     }
```

3 Output

For compiling our program and seeing the output we opened two terminals,
in the first terminal we wrote: qemu-nox-gdb
In the second terminal we wrote gdb -q -iex "set auto-load safe-path /home/csgrads/nhoss003/xv6/"
then we entered continue
then we came back to the first terminal and wrote the name of our program without .c extension.
Our output is:

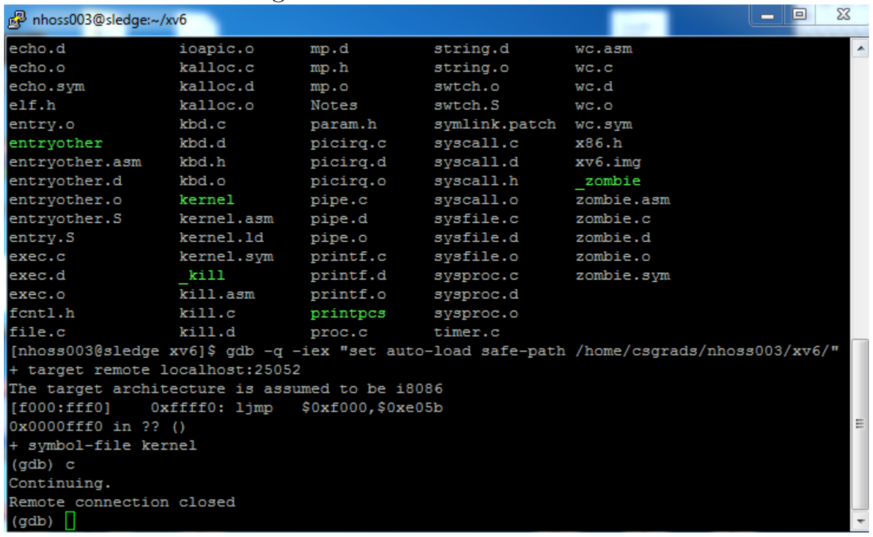
```
my system call 4
```

Here is a screen shot of our output:



```
[nhoss003@sledge xv6]$ make qemu-nox-gdb
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB) copied, 0.031415 s, 163 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000179552 s, 2.9 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
273+1 records in
273+1 records out
139804 bytes (140 kB) copied, 0.000810136 s, 173 MB/s
*** Now run 'gdb'.
qemu -nographic -hdb fs.img xv6.img -smp 2 -m 512  -S -gdb tcp::25052
Could not open option rom 'sgabios.bin': No such file or directory
xv6...
cpu1: starting
cpu0: starting
init: starting sh
$ coun
my system call 4
pid 3 coun: trap 14 err 5 on cpu 1 eip 0xffffffff addr 0xffffffff--kill proc
$ 
```

Figure 1: First Terminal Results



```
echo.d            ioapic.o      mp.d          string.d        wc.asm
echo.o            kalloc.c      mp.h          string.o        wc.c
echo.sym          kalloc.d      mp.o          swtch.o         wc.d
elf.h             kalloc.o      Notes         swtch.S         wc.o
entry.o           kbd.c         param.h       symlink.patch   wc.sym
entryother        kbd.d         picirq.c      syscall.c       x86.h
entryother.asm    kbd.h         picirq.d      syscall.d       xv6.img
entryother.d      kbd.o         picirq.o      syscall.h       _zombie
entryother.o      kernel        pipe.c        syscall.o       zombie.asm
entryother.S      kernel.asm    pipe.d        sysfile.c       zombie.c
entry.S           kernel.ld     pipe.o        sysfile.d       zombie.d
exec.c            kernel.sym    printf.c      sysfile.o       zombie.o
exec.d            _kill         printf.d      sysproc.c       zombie.sym
exec.o            kill.asm      printf.o      sysproc.d
fcntl.h           kill.c        printpcs      sysproc.o
file.c            kill.d        proc.c        timer.c
[nhoss003@sledge xv6]$ gdb -q -iex "set auto-load safe-path /home/csgrads/nhoss003/xv6/"
+ target remote localhost:25052
The target architecture is assumed to be i8086
[f000:fff0]    0xffff0: ljmp   $0xf000,$0xe05b
0x0000fff0 in ?? ()
+ symbol-file kernel
(gdb) c
Continuing.
Remote connection closed
(gdb) 
```

Figure 2: Second Terminal