Warmup

Satoru Yamamoto

04/09/2022

Contents

1 What I changed

1.1 proc.c/.h

I first added a global kernel variable for to count system calls.
proc.h and proc.c prepare for the first process, so I added my global kernel variable here.[1]

proc.c

```c
int syscallCount;                    //initialize variable
```

proc.h

```c
extern int syscallCount;             // counter for systemcall
```

1.2 main.c

Then I made counter variable to 0 in main.c.
The entry page is defined in main.c, so I made counter variable to 0 in main.c.

main.c

```c
int
main(void)
{
    syscallCount = 0;
    // other codes
}
```

1.3 sysproc.c

I created a required function in sysproc.c
sysproc.c contains the implementations of process related system calls, so I added an implementation of required system call.

sysproc.c

```c
// return the number of calls made since the system booted on success,
// and -1 on failure
int
sys_getsyscallinfo(void)
{
  // myproc return proc
  return syscallCount;
}
```

1.4 syscall.c/.h

Modified syscall.c and syscall.h to increment counter when system call is executed correctly and to be able to execute a required function.
syscall.h contains system call names and assigned number, so I added required function name. By following the invocation instruction in usys.S, I added SYS_ before my function name.
syscall.c contains helper functions to parse system call arguments for actual system call functions, so I modified syscall.c to parse required system call function. sysclal function is executed when system call is made. I increment my counter in the first if statement block because that is executed when the call is accessed.

syscall.c

```c
// add function
extern int sys_getsyscallinfo(void);
static int (*syscalls[])(void) = {
    //other system calls,
    [SYS_getsyscallinfo]   sys_getsyscallinfo
};
void
syscall(void)
{
  int num;
  struct proc *curproc = myproc();

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
    // increment sysCounter
    syscallCount++;
    curproc->tf->eax = syscalls[num]();
  } else {
    cprintf("%d %s: unknown sys call %d\n",
            curproc->pid, curproc->name, num);
    curproc->tf->eax = -1;
  }
}
```

syscall.h

```c
//other system calls
#define SYS_getsyscallinfo 22   // add system call number
```

1.5 user.h/usys.S

Modified user.h and usys.S to be able to call a required system call function.
user.h contains system call definitions, so I added a definition of required function.
usys.S contains a list of system calls that is exported by the kernel, so I added a required function by following invocation instruction.

user.h

```c
// system calls
// ohter system calls
int getsyscallinfo(void);
```

usys.S

```
// other system calls
SYSCALL(getsyscallinfo)
```

2 How I tested it

I created c program to test my system call in xv6-public directly.

testcall.c

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

int main(int argc, char *argv[])
{
    int cnt = getsyscallinfo();
    printf(1,"count %d\n", cnt);
    cnt = getsyscallinfo();
    printf(1,"count %d\n", cnt);
    exit();
}
```

Then I added my c program to UPROGS and and EXTRA in Makefile. After I modified Makefile, I typed these commands.

```
make clear
make
```

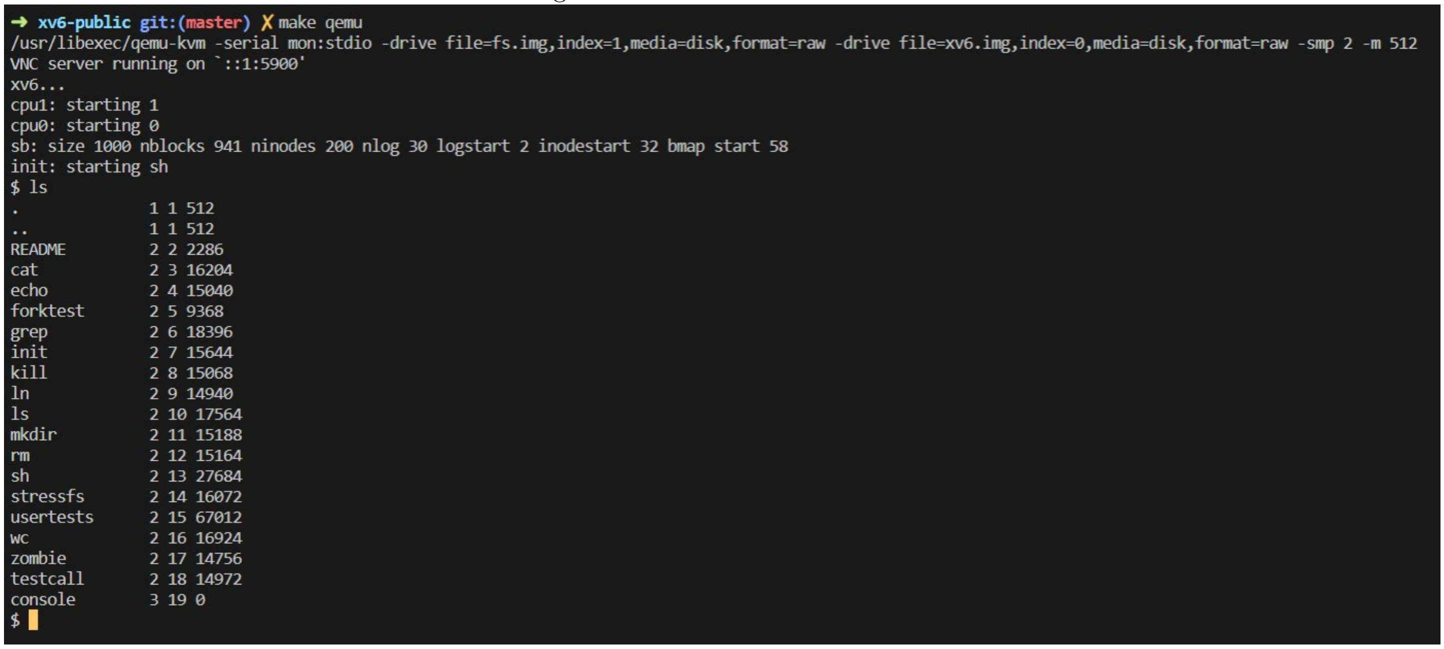
Now I can see that my testcall is added.*1
My test result is here.*2
There is 10 difference between my system call because the system prints character by character. In my case, it prints "*count %d*".

References

[1] https://ppan-brian.medium.com/first-process-from-xv6-76f63ab96f46

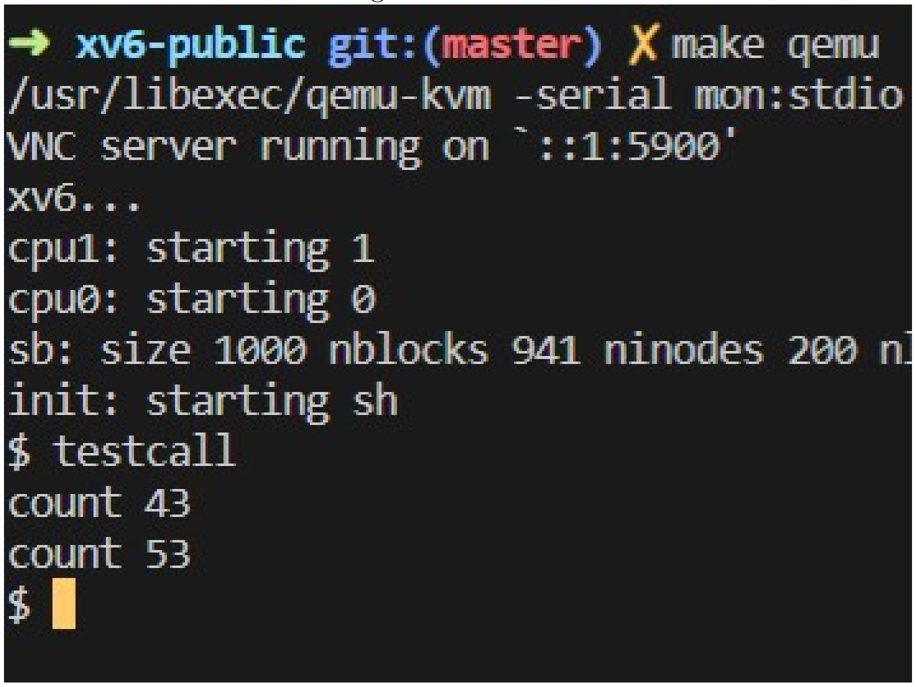
[2] https://www.cse.iitb.ac.in/~mythili/os/labs/lab-xv6-proc/xv6-proc.pdf

Figure 1: testcall



```
➜ xv6-public git:(master) ✗ make qemu
/usr/libexec/qemu-kvm -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
VNC server running on `::1:5900'
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 16204
echo           2 4 15040
forktest       2 5 9368
grep           2 6 18396
init           2 7 15644
kill           2 8 15068
ln             2 9 14940
ls             2 10 17564
mkdir          2 11 15188
rm             2 12 15164
sh             2 13 27684
stressfs       2 14 16072
usertests      2 15 67012
wc             2 16 16924
zombie         2 17 14756
testcall       2 18 14972
console        3 19 0
$
```

Figure 2: testcount



```
➜ xv6-public git:(master) ✗ make qemu
/usr/libexec/qemu-kvm -serial mon:stdio
VNC server running on `::1:5900'
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nl
init: starting sh
$ testcall
count 43
count 53
$
```