Operating System Concepts

Mini Project 1

Andre Ripley

Participants name	Code section	Report Section	Documentation Sec	Presentation Sec
Andre Ripley	100%	100%	100%	100%

To track the amount of syscalls made while xv6 is running, changes were made to syscall.h, syscall.c, sysfunc.h, user.h, usys.S, sys.proc.c, makefile, and user.h. All these files were modified slightly by adding howmanysys to each file. This allowed the howmanysys.c to call howmanysys()(the system call). One file was created howmanysys.c to return and validate how many syscalls were made.

File created

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

int main(void) {
    printf(1, "There have been %d system calls made.\n", howmanysys());
    exit();
}
```

howmanysys.c

The first thing to implement was the driver function. This allowed to test if the syscall would work or not and if it did print out how many syscalls there are.

Files Modified

```c
static int (*syscalls[])(void) = {
[SYS_chdir]   sys_chdir,
[SYS_close]   sys_close,
[SYS_dup]     sys_dup,
[SYS_exec]    sys_exec,
[SYS_exit]    sys_exit,
[SYS_fork]    sys_fork,
[SYS_fstat]   sys_fstat,
[SYS_getpid]  sys_getpid,
[SYS_kill]    sys_kill,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_mknod]   sys_mknod,
[SYS_open]    sys_open,
[SYS_pipe]    sys_pipe,
[SYS_read]    sys_read,
[SYS_sbrk]    sys_sbrk,
[SYS_sleep]   sys_sleep,
[SYS_unlink]  sys_unlink,
[SYS_wait]    sys_wait,
[SYS_write]   sys_write,
[SYS_uptime]  sys_uptime,
[SYS_howmanysys] sys_howmanysys,
};
```

syscall.c

```
// System call numbers
#define SYS_fork         1
#define SYS_exit         2
#define SYS_wait         3
#define SYS_pipe         4
#define SYS_write        5
#define SYS_read         6
#define SYS_close        7
#define SYS_kill         8
#define SYS_exec         9
#define SYS_open        10
#define SYS_mknod       11
#define SYS_unlink      12
#define SYS_fstat       13
#define SYS_link        14
#define SYS_mkdir       15
#define SYS_chdir       16
#define SYS_dup         17
#define SYS_getpid      18
#define SYS_sbrk        19
#define SYS_sleep       20
#define SYS_uptime      21
#define SYS_howmanysys  22
```

syscall.h

```
// System call handlers
int sys_chdir(void);
int sys_close(void);
int sys_dup(void);
int sys_exec(void);
int sys_exit(void);
int sys_fork(void);
int sys_fstat(void);
int sys_getpid(void);
int sys_kill(void);
int sys_link(void);
int sys_mkdir(void);
int sys_mknod(void);
int sys_open(void);
int sys_pipe(void);
int sys_read(void);
int sys_sbrk(void);
int sys_sleep(void);
int sys_unlink(void);
int sys_wait(void);
int sys_write(void);
int sys_uptime(void);
int sys_howmanysys(void); // adds system call handler for howmanysys
```

sysfunc.h

```c
// system calls
int fork(void);
int exit(void) __attribute__((noreturn));
int wait(void);
int pipe(int*);
int write(int, void*, int);
int read(int, void*, int);
int close(int);
int kill(int);
int exec(char*, char**);
int open(char*, int);
int mknod(char*, short, short);
int unlink(char*);
int fstat(int fd, struct stat*);
int link(char*, char*);
int mkdir(char*);
int chdir(char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
int howmanysys(void);
```

user.h

```
SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
SYSCALL(exec)
SYSCALL(open)
SYSCALL(mknod)
SYSCALL(unlink)
SYSCALL(fstat)
SYSCALL(link)
SYSCALL(mkdir)
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(howmanysys)
```

usys.S

```
#####################################################################################
# Emulator Options
#####################################################################################

# If the makefile can't find QEMU, specify its path here
QEMU := qemu-system-i386

# Try to infer the correct QEMU if not specified
ifndef QEMU
QEMU := $(shell if which qemu 1> /dev/null 2> /dev/null; \
        then echo qemu; exit; \
        else \
        qemu=/u/c/s/cs537-2/ta/tools/qemu; \
        if test -x $$qemu; then echo $$qemu; exit; fi; fi; \
        echo "***" 1>&2; \
        echo "*** Error: Couldn't find a working QEMU executable." 1>&2; \
        echo "*** Is the directory containing the qemu binary in your " 1>&2; \
        echo "*** PATH or have you tried setting the QEMU variable in " 1>&2; \
        echo "*** Makefile?" 1>&2; \
        echo "***" 1>&2; exit 1)
endif
```

The makefile had to be updated by adding QEMU:= qemu-system-i1386

```
extern uint called;

void
syscall(void)
{
  int num;
  called++; // increment called every time a system call is made
  num = proc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num] != NULL) {
    proc->tf->eax = syscalls[num]();
  } else {
    cprintf("%d %s: unknown sys call %d\n",
            proc->pid, proc->name, num);
    proc->tf->eax = -1;
  }
}
```

syscall.c

Syscall.c was modified again to increment every time a system call was made. This will will give us the appropriate value of the system calls.

```
uint called = 0;
int
sys_howmanysys(void)
{
  return called;
}
```

sysproc.c

Sysproc.c initializes the called so every time the called is incremented from syscall.c it will return the value of called.

```
xv6...
lapicinit: 1 0xfee00000
cpu1: starting
cpu0: starting
init: starting sh
$ howmanysys
There have been 47 system calls made.
$ ls
.              1 1 512
..             1 1 512
cat            2 2 9484
init           2 3 9324
ln             2 4 9012
kill           2 5 9048
sh             2 6 16212
forktest       2 7 5812
echo           2 8 9032
howmanysys     2 9 8864
ls             2 10 10584
stressfs       2 11 9244
tester         2 12 8964
usertests      2 13 34868
mkdir          2 14 9104
rm             2 15 9088
zombie         2 16 8820
wc             2 17 9808
grep           2 18 10604
console        3 19 0
$ howmanysys
There have been 699 system calls made.
$
```

Output

As you can see after going into the xv6 folder and typing “make qemu” and then “howmanysys” it shows the number of system calls made.

Conclusion:

Working with xv6 was a bit of a task to create syscalls. It was a lot of trial and error to get it to compile. I had a hard time considering I was running it Ubuntu on a jetson

because performance on my VM was incredibly low. I took me some time to realize I didn’t install qemu and that was causing most of the problems as well as adding a handler in sysfunc.h. Not adding the handler returned compile problems for howmanysys(). In the end after those issues were resolved it compiled and ran smoothly outputting the number of system calls made.