

# Operating System Concepts

COP4610.02

## Mini Project 1

Noah Baldwin

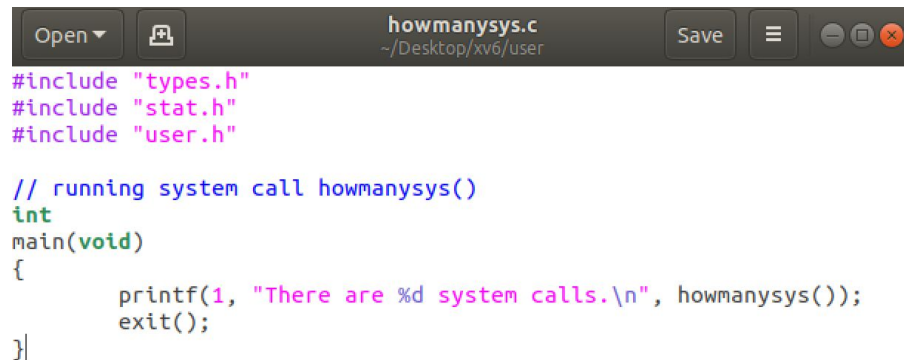
Cody Carroll

Paul Teleweck

Participants name	Code section	Report Section	Documentation Sec	Presentation Sec
Noah Baldwin	33.33%	33.34%	33.33%	33.3%
Cody Carroll	33.33%	33.33%	33.34%	33.3%
Paul Teleweck	33.34%	33.33%	33.33%	33.3%

1. Created a user program within “user” called howmanysys.c

- Added function to print number of system calls.



```

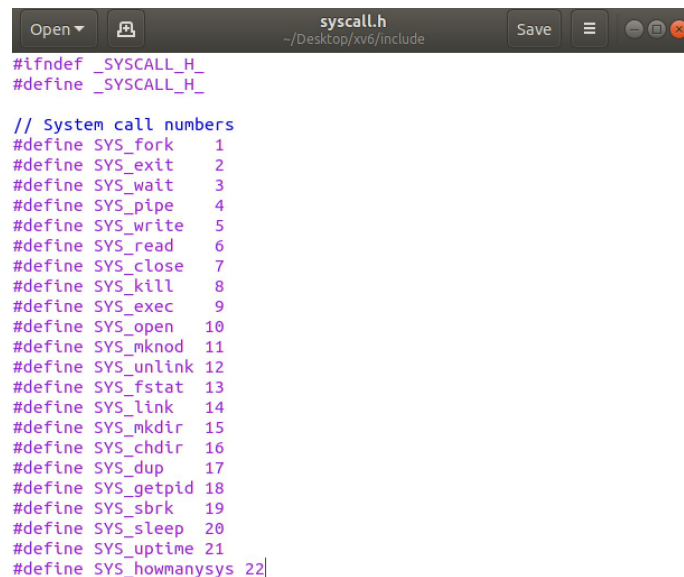
#include "types.h"
#include "stat.h"
#include "user.h"

// running system call howmanysys()
int
howmanysys()
{
    printf(1, "There are %d system calls.\n", howmanysys());
    exit();
}
main(void)
{
    howmanysys();
}

```

2. Add #define statement in syscall.h to add howmanysys into system calls.

- Create a system call number for howmanysys



```

#ifndef _SYSCALL_H_
#define _SYSCALL_H_

// 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

```

3. Add [sys\_howmanysys] sys\_howmanysys, inside the file syscall.c.

```
Open ▾ syscall.c ~/Desktop/xv6/kernel Save ≡
// can catch definitions that don't match

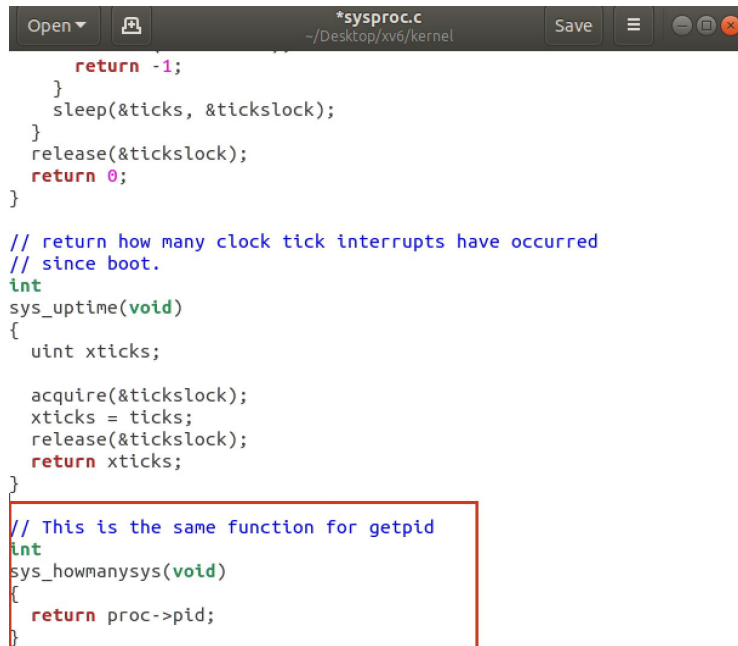
// array of function pointers to handlers for all the syscalls
static int (*syscalls[])(void) = {
[SYS_chdir] sys_chdir,
[SYS_close] sys_close,
[SYS_dup] sys_dup,
[SYS_exec] sys_exec,
[SYS_exit] sys_exit,
[SYS_fork] sys_fork,
[SYS_fstat] sys_fstat,
[SYS_getpid] sys_getpid,
[SYS_kill] sys_kill,
[SYS_link] 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,
};
```

4. Add int howmanysys(void); to the file, sysfunc.h.

```
Open ▾ sysfunc.h ~/Desktop/xv6/kernel Save ≡
#ifndef _SYSFUNC_H_
#define _SYSFUNC_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);
```

5. Add `sys_howmanysys` function inside `sysproc.c`
- Checks for valid system call number, calls appropriate handler and returns system call number.



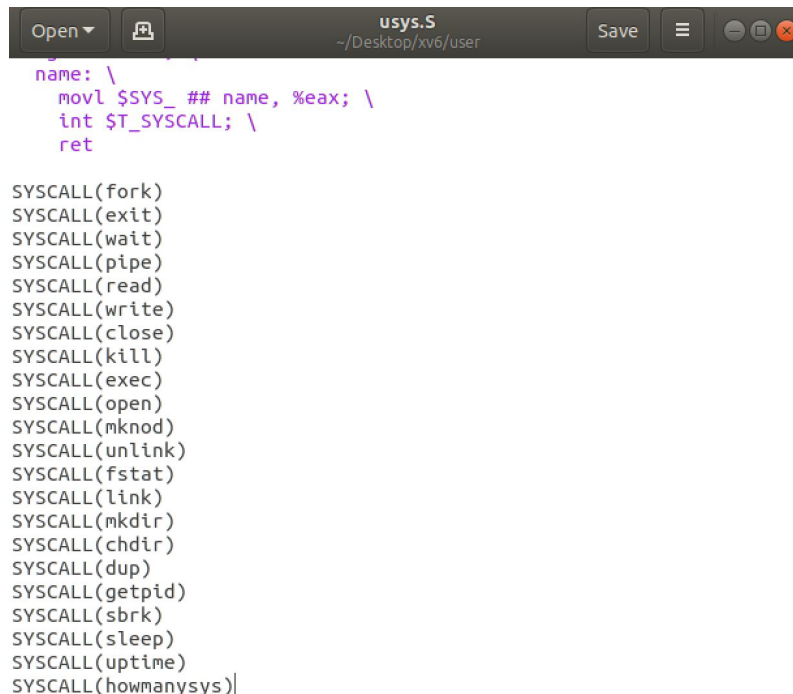
```
return -1;
}
sleep(&ticks, &tickslock);
}
release(&tickslock);
return 0;
}

// return how many clock tick interrupts have occurred
// since boot.
int
sys_uptime(void)
{
    uint xticks;

    acquire(&tickslock);
    xticks = ticks;
    release(&tickslock);
    return xticks;
}

// This is the same function for getpid
int
sys_howmanysys(void)
{
    return proc->pid;
}
```

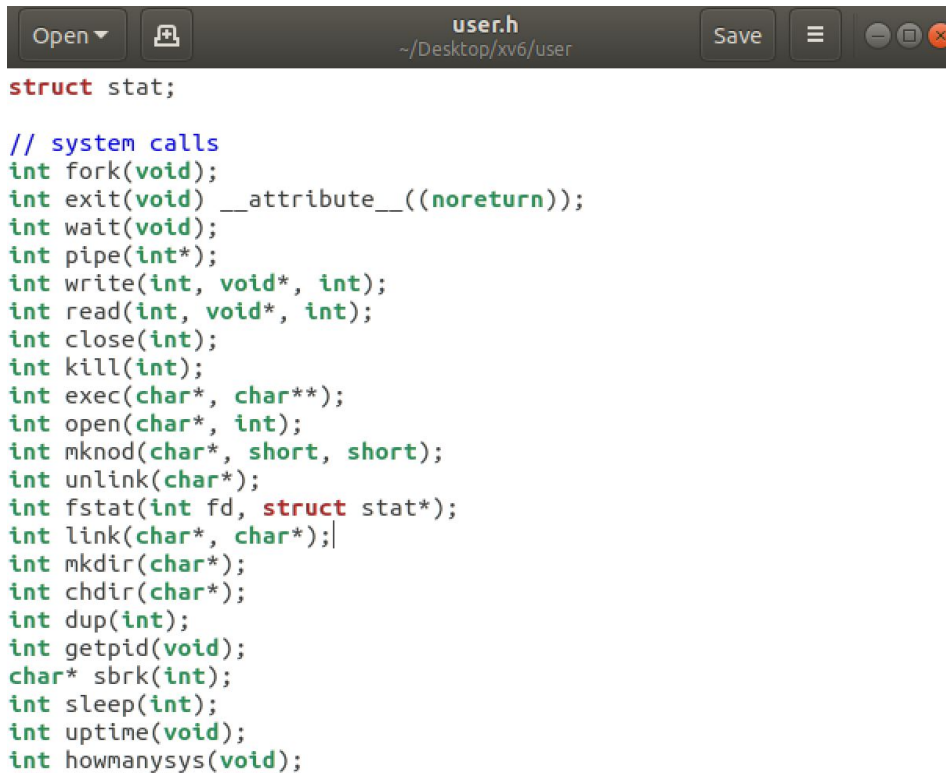
6. In the file, `usys.s`, we added the system call, `SYSCALL(howmanysys)`.



```
name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret

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)|
```

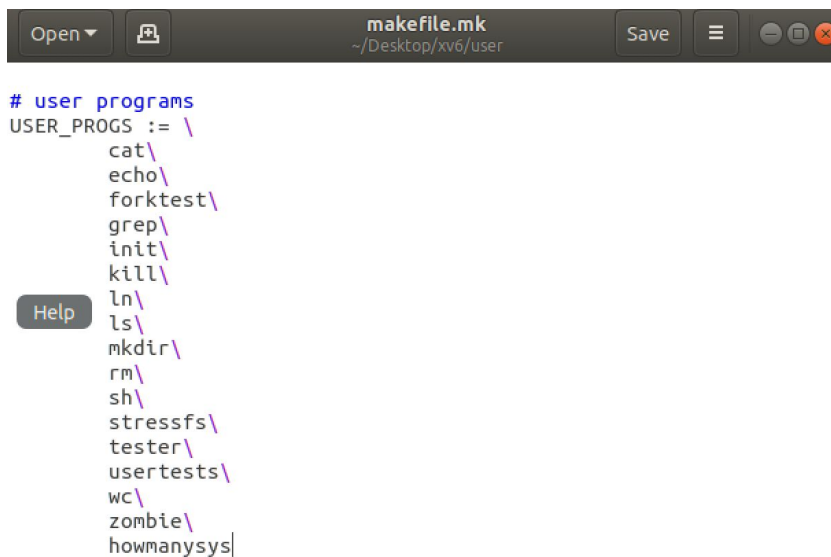
7. In the file, user.h we add the value, int howmanysys.



```
struct stat;

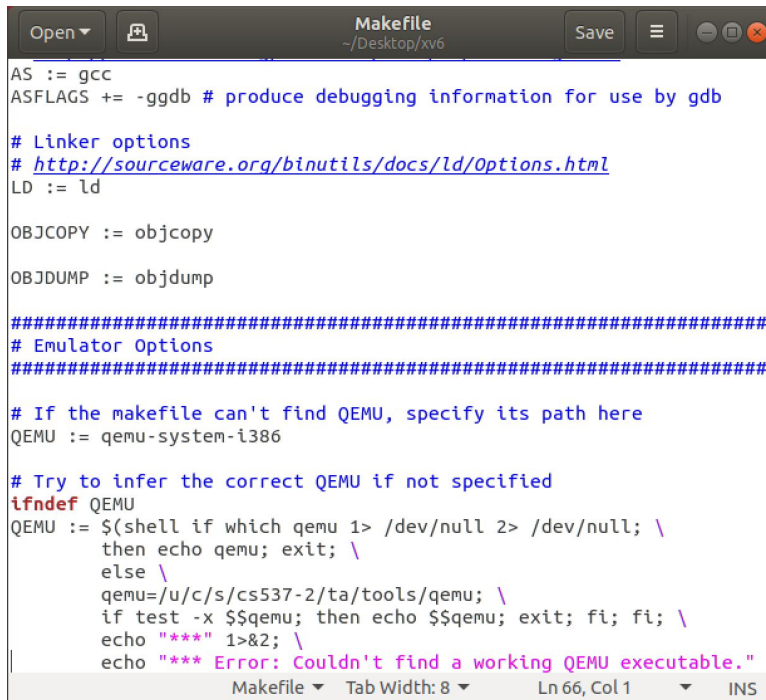
// system calls
int fork(void);
int exit(void) __attribute__((noreturn));
int wait(void);
int pipe(int*);
int write(int, void*, int);
int read(int, void*, int);
int close(int);
int kill(int);
int exec(char*, char**);
int open(char*, int);
int mknod(char*, short, short);
int unlink(char*);
int fstat(int 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);
```

8. Add the user file howmanysys to the file, makefile.mk.



```
# user programs
USER_PROGS := \
    cat\
    echo\
    forktest\
    grep\
    init\
    kill\
    ln\
    ls\
    mkdir\
    rm\
    sh\
    stressfs\
    tester\
    usertests\
    wc\
    zombie\
    howmanysys
```

## 9. Makefile



```
Open ▾ Makefile ~/Desktop/xv6 Save ≡
AS := gcc
ASFLAGS += -ggdb # produce debugging information for use by gdb

# Linker options
# http://sourceware.org/binutils/docs/ld/Options.html
LD := ld

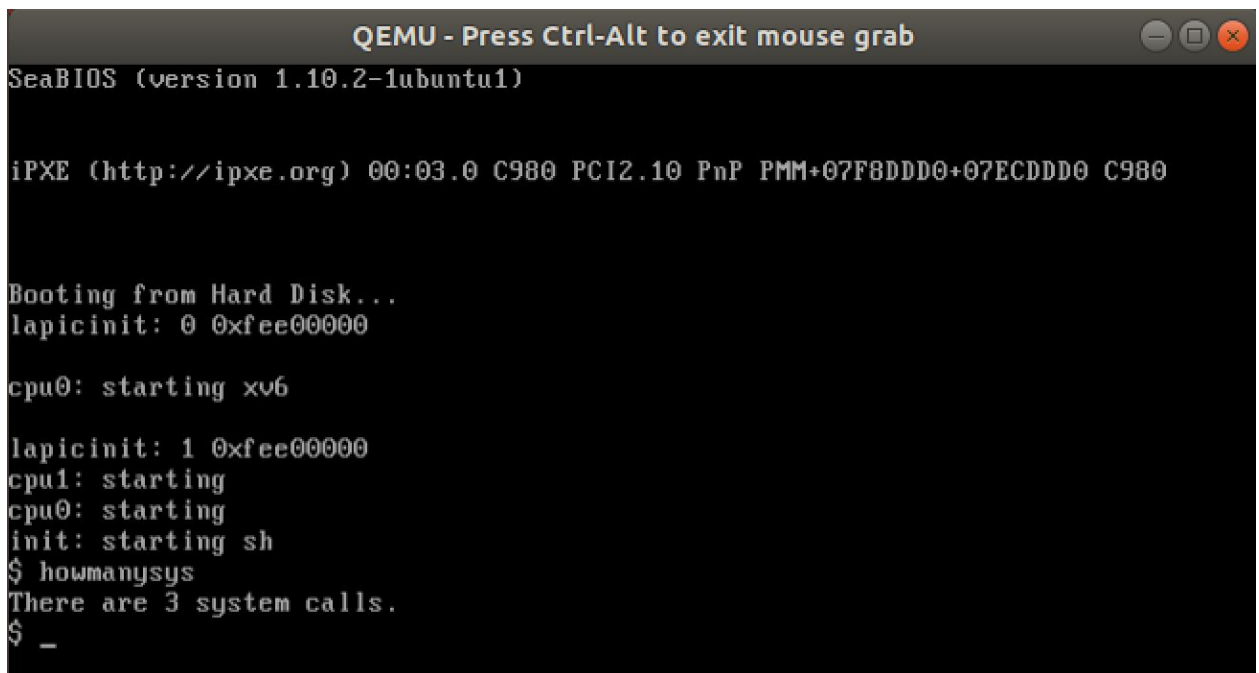
OBJCOPY := objcopy
OBJDUMP := objdump

#####
# 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.")
Makefile ▾ Tab Width: 8 ▾ Ln 66, Col 1 ▾ INS
```

## 10. Output



```
QEMU - Press Ctrl-Alt to exit mouse grab
SeaBIOS (version 1.10.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DDDD+07ECDDDD C980

Booting from Hard Disk...
lapicinit: 0 0xfec00000

cpu0: starting xv6

lapicinit: 1 0xfec00000
cpu1: starting
cpu0: starting
init: starting sh
$ howmanysys
There are 3 system calls.
$ _
```