Lab 3: Add a System Call

Instructor: Yaoqing Liu

TA: Garegin Grigoryan

Notes

- Students in lab session 1 (1:00-1:50)

  Due to concerns that students in the second lab session (2:00-2:50) may not have enough seats, please save or clean up your work, and leave the lab before session 2 (2:00). You can come back at 3:00, our TA will help you further till 5:00.

  Thanks for your understanding and cooperation!

Retrieve xv6

- Login to odin

```
$ssh YourName@odin.cslabs.clarkson.edu
$cd ~/cs444-s18/Lab3
```

- Download xv6.tar.gz file to your work directory Lab3

```
$wget http://people.clarkson.edu/~liu/CS444/Spring18/xv6.tar.gz
```

- Unzip it

```
$tar –xzvf xv6.tar.gz
```

System Call

- A function that a userspace application will use, so as to ask for a specific service to be provided by the operating system
- Making a System Call
  - User code pushes some parameters onto the user stack
  - Puts the system call number into the *eax* register
  - Generate software interrupt via "int $T_SYSCALL"

Handling System Call

- The kernel checks the system call number and jumps to the appropriate function
- System call handlers are named ***sys_{syscall name}***, eg, **sys_open**, and have a system call number of **SYS_{syscall name}**, eg, **SYS_open**
- These use **argint**, **argptr**, and **argstr** to retrieve the system call parameters from the user stack

Show an Example (1)

- Return memory size of a process
- syscall.h: register a system call number

```
#define SYS_getmz  24
```

- syscall.c: index the system call function. When a system call occurred with the system call number 24, the function pointed by the function pointer sys_getmz will be called.

```
extern int sys_getmz(void);
[SYS_getmz]   sys_getmz,
```

Show an Example (2)

- Sysproc.c: Implement the system call

```
int sys_getmz(void){
    int mz = (int) proc->sz;
    return mz;
}
```

- /userspace/usys.S: Add interface for our user program to access the system call

```
SYSCALL(getmz)
```

- /userspace/user.h: Add function declaration that our user program will call. Now a call to the below function from a user program will be mapped to the system call number 24 which is defined as SYS_getmz preprocessor directive. The system knows this system call and how to handle it.

```
int getmz(void);
```

Show an Example (3)

- userspace/getmz.c: create a new file with the following content

```
#include "types.h"
#include "stat.h"
#include "user.h"
int main(void) {
    printf(1, ”Current running process memory size: %d\n", getmz());
    exit();
}
```

- userspace/Makefile: extend **UPROGS** section as follow for compiling

```
_getmz\
```

Note: leave an empty line after this

- Compile and run

```
make
make qemu-nox
```

Your Tasks

- Create your own system call that returns the number of system calls already called by the OS
  - Declare a global variable ***scnum*** to store the number of system calls (Hint: Use **extern int** to refer to the variable in a system call implementation file, review C programming if you're not familiar with this)
  - Register the system call with a unique number in **YourName, e.g., sys_liu**
  - Declare the system call and add an entry to syscall table
  - Implement the system call (Hint: change syscall() function in syscall.c)
  - Write a user program to call the system call and return the number of system calls already called by the OS

Submission

- Take screenshots of your modified files and your running results
- Combine them into one PDF file and submit it on moodle

```
convert file1.png file2.png … YourName.pdf
```

- Due: Monday (Feb 5) 11:55pm

- DO NOT change the source code in the Lab3 folder once you finish your lab, our TA will check it
- Please DELETE all of your local files before leaving!!

Thank You!