

Recitation Week #10

System Calls and Synchronization

Alejandro Cabrera
Operating Systems
COP4610 / CGS5765

Today's Recitation

- Implementing System Calls
- Concurrent Aspects of Project 3
- Concurrency vs. Parallelism
- Global Data vs. Local Data
 - What Should You Synchronize?
- Kernel Synchronization Primitives in Detail
- More Project 3 Hints

Implementing System Calls

```
int start_elevator(const int #);  
int issue_request(const int #1,  
                 const int #2);  
int stop_elevator(const int #);
```

- Need to implement the functions above. But how?

Adding a System Call to Kernel

- Files to modify:
 - LINUX_DIR/arch/x86/kernel/syscall_table_32.S
 - LINUX_DIR/include/asm-generic/unistd.h
 - LINUX_DIR/include/linux/syscalls.h
 - LINUX_DIR/Makefile
- Files to add:
 - LINUX_DIR/PROJECT_NAME/Makefile
 - LINUX_DIR/PROJECT_NAME/PROJECT_NAME.c
- User-space drivers:
 - SOME_DIR/driver.c
 - SOME_DIR/driver.h

A Concrete Example: kval Interface

```
int kval;  
int sys_getval();  
int sys_plus(int x);  
int sys_minus(int x);  
int sys_setval(int x);
```

- Goal: implement a kernel variable and system calls to manipulate it.

References for kval Interface

1. Implementing a System Call on Linux for i386
2. `linux-2.6.31.1/kernel/`
 - `time.c /* settimeofday, SYSCALL_DEFINEx */`
3. Many hours getting kval to work and solving inconsistencies in [1] above.

Modifying syscall_table_32.S

```
...  
    .long sys_preadv  
    .long sys_pwritev  
    .long sys_rt_tgsigqueueinfo  
    .long sys_perf_counter_open /* 335 */  
/* EOF */
```

Modifying syscall_table_32.S

```
...  
    .long sys_preadv  
    .long sys_pwritev  
    .long sys_rt_tgsigqueueinfo  
    .long sys_perf_counter_open /* 335 */  
    .long sys_getval  
/* EOF */
```

Modifying syscall_table_32.S

```
...  
    .long sys_preadv  
    .long sys_pwritev  
    .long sys_rt_tgsigqueueinfo  
    .long sys_perf_counter_open /* 335 */  
    .long sys_getval  
    .long sys_plus  
/* EOF */
```

Modifying syscall_table_32.S

```
...  
    .long sys_preadv  
    .long sys_pwritev  
    .long sys_rt_tgsigqueueinfo  
    .long sys_perf_counter_open /* 335 */  
    .long sys_getval  
    .long sys_plus  
    .long sys_minus  
/* EOF */
```

Modifying syscall_table_32.S

```
...  
    .long sys_preadv  
    .long sys_pwritev  
    .long sys_rt_tgsigqueueinfo  
    .long sys_perf_counter_open /* 335 */  
    .long sys_getval  
    .long sys_plus  
    .long sys_minus  
    .long sys_setval /* 340 */  
/* EOF */
```

Modifying syscall_table_32.S

```
...  
    .long sys_preadv  
    .long sys_pwritev  
    .long sys_rt_tgsigqueueinfo  
    .long sys_perf_counter_open /* 335 */  
    .long sys_getval  
    .long sys_plus  
    .long sys_minus  
    .long sys_setval /* 340 */  
    /* Remember these numbers- you'll need  
       them in user-space! */  
/* EOF */
```

Modifying unistd.h

```
/* midfile */  
#define __NR_perf_counter_open 241  
__SYSCALL(__NR_perf_counter_open, sys_perf_counter_open)  
  
#undef __NR_syscalls  
#define __NR_syscalls 242  
/* midfile */
```

Modifying unistd.h

```
/* midfile */
#define __NR_perf_counter_open 241
__SYSCALL(__NR_perf_counter_open, sys_perf_counter_open)

#define __NR_getval 242
__SYSCALL(__NR_getval, sys_getval)

#undef __NR_syscalls
#define __NR_syscalls 242
/* midfile */
```

Modifying unistd.h

```
/* midfile */
#define __NR_perf_counter_open 241
__SYSCALL(__NR_perf_counter_open, sys_perf_counter_open)

#define __NR_getval 242
__SYSCALL(__NR_getval, sys_getval)
#define __NR_plus 243
__SYSCALL(__NR_plus, sys_plus)

#undef __NR_syscalls
#define __NR_syscalls 242
/* midfile */
```

Modifying unistd.h

```
/* midfile */
#define __NR_perf_counter_open 241
__SYSCALL(__NR_perf_counter_open, sys_perf_counter_open)

#define __NR_getval 242
__SYSCALL(__NR_getval, sys_getval)
#define __NR_plus 243
__SYSCALL(__NR_plus, sys_plus)
#define __NR_minus 244
__SYSCALL(__NR_minus, sys_minus)

#undef __NR_syscalls
#define __NR_syscalls 242
/* midfile */
```

Modifying unistd.h

```
/* midfile */
#define __NR_perf_counter_open 241
__SYSCALL(__NR_perf_counter_open, sys_perf_counter_open)

#define __NR_getval 242
__SYSCALL(__NR_getval, sys_getval)
#define __NR_plus 243
__SYSCALL(__NR_plus, sys_plus)
#define __NR_minus 244
__SYSCALL(__NR_minus, sys_minus)
#define __NR_setval 245
__SYSCALL(__NR_setval, sys_setval)

#undef __NR_syscalls
#define __NR_syscalls 242
/* midfile */
```

Modifying unistd.h

```
/* midfile */
#define __NR_perf_counter_open 241
__SYSCALL(__NR_perf_counter_open, sys_perf_counter_open)

#define __NR_getval 242
__SYSCALL(__NR_getval, sys_getval)
#define __NR_plus 243
__SYSCALL(__NR_plus, sys_plus)
#define __NR_minus 244
__SYSCALL(__NR_minus, sys_minus)
#define __NR_setval 245
__SYSCALL(__NR_setval, sys_setval)

#undef __NR_syscalls
#define __NR_syscalls 246
/* midfile */
```

Modifying syscalls.h

```
asm linkage long sys_perf_counter_open(...);  
/* EOF */
```

Modifying syscalls.h

```
asmlinkage long sys_perf_counter_open(...);
```

```
asmlinkage long sys_getval(void);
```

```
/* EOF */
```

Modifying syscalls.h

```
asmlinkage long sys_perf_counter_open(...);  
  
asmlinkage long sys_getval(void);  
asmlinkage long sys_plus(int x);  
/* EOF */
```

Modifying syscalls.h

```
asmlinkage long sys_perf_counter_open(...);  
  
asmlinkage long sys_getval(void);  
asmlinkage long sys_plus(int x);  
asmlinkage long sys_minus(int x);  
/* EOF */
```

Modifying syscalls.h

```
asm linkage long sys_perf_counter_open(...);  
  
asm linkage long sys_getval(void);  
asm linkage long sys_plus(int x);  
asm linkage long sys_minus(int x);  
asm linkage long sys_setval(int x);  
/* EOF */
```

Modifying Makefile

```
# midfile
# Objects we will link into vmlinux /
# subdirs we need to visit
init-y := init/
drivers-y := drivers/ sound/ firmware/
net-y := net/
libs-y := lib/
core-y := usr/
endif # KBUILD_EXTMOD
```

```
# midfile
```

Modifying Makefile

```
# midfile
# Objects we will link into vmlinux /
# subdirs we need to visit
init-y := init/
drivers-y := drivers/ sound/ firmware/
net-y := net/
libs-y := lib/
core-y := usr/ kva1/
endif # KBUILD_EXTMOD
```

```
# midfile
```

Creating Project kval: Makefile

```
# @file LINUX_DIR/kval/Makefile  
obj-y := kval.o
```

Creating Project kval: kval.c

```
/* @file LINUX_DIR/kval/kval.c
 */
#include <linux/syscalls.h>
#include <linux/module.h>

int kval = 0;
```

Creating Project kval: kval.c

Helper Functions

```
int do_kval_getval(void)
{
    return kval;
}

int do_kval_setval(int x)
{
    kval = (x > 0) ? x : 0;
    return kval;
}
```

Creating Project kval: kval.c

Helper Functions

```
int do_kval_plus(int x)
{
    kval += (x > 0) ? x : 0;
    return kval;
}
```

```
int do_kval_minus(int x)
{
    kval -= (x > 0) ? x : 0;
    return kval;
}
```

Creating Project kval: kval.c

Exporting Symbols

```
EXPORT_SYMBOL_GPL(do_kval_getval);  
EXPORT_SYMBOL_GPL(do_kval_setval);  
EXPORT_SYMBOL_GPL(do_kval_plus);  
EXPORT_SYMBOL_GPL(do_kval_minus);
```

Creating Project kval: kval.c

Defining System Calls

```
SYSCALL_DEFINE0(getval)
{
    return do_kval_getval();
}

SYSCALL_DEFINE1(setval, int, x)
{
    return do_kval_setval(x);
}
```

Creating Project kval: kval.c

Defining System Calls

```
SYSCALL_DEFINE1(plus, int, x)
{
    return do_kval_plus(x);
}
```

```
SYSCALL_DEFINE1(minus, int, x)
{
    return do_kval_minus(x);
}
```

Creating User kval: kval_driver.h

```
#ifndef __KVAL_DRIVER_H
#define __KVAL_DRIVER_H

#define __NR_getval 337
#define __NR_plus 338
#define __NR_minus 339
#define __NR_setval 340

int kval_getval();
int kval_plus(int);
int kval_minus(int);
int kval_setval(int);
#endif
```

Creating User kval: kval_driver.c

```
#include <unistd.h>
#include <stdio.h>
#include "kval_driver.h"

int kval_getval()
{
    return syscall(__NR_getval);
}

int kval_setval(int x)
{
    return syscall(__NR_setval, x);
}
```

Creating User kval: kval_driver.c

```
int kval_plus(int x)
{
    return syscall(__NR_plus, x);
}

int kval_minus(int x)
{
    return syscall(__NR_minus, x);
}
```

Creating User kval: kval_driver.c

```
int main()
{
    printf("setval(0): %d\n", kval_setval(0));
    printf("getval(): %d\n", kval_getval());
    printf("plus(10): %d\n", kval_plus(10));
    printf("minus(10): %d\n", kval_minus(10));
    printf("getval(): %d\n", kval_getval());

    return 0;
}
```

Creating User kval: Makefile

```
CFLAGS = -Wall -Wextra -pedantic -g
```

```
all: kval
```

```
kval: kval.c kval.h
```

```
    gcc $(CFLAGS) kval.c -o kval
```

```
clean:
```

```
    rm -f *.o kval
```

Getting to a Working kval

1. Re-compile the kernel
 1. This will build kval interface into the kernel
2. Install kernel
3. Reboot
4. Compile user-space driver
5. Run user-space driver
6. Profit

Extending kval

- Adding a procfs/sysfs interface
 - Showing value of kval at runtime
 - Allowing writing to kval through procfs
- Adding synchronization
 - Multiple processes/processors can access kval at the same time – need to guarantee consistency
- Adding more complex structures
 - How to handle dynamically allocated memory?

Extending kval

- Adding a procfs/sysfs interface
 - Showing value of kval at runtime
 - Allowing writing to kval through procfs
- Adding synchronization
 - Multiple processes/processors can access kval at the same time – need to guarantee consistency
- Adding more complex structures
 - How to handle dynamically allocated memory?

Concurrency Aspects of Project 3

- Synchronizing access to request queue(s)
 - Multiple producers may access request queue(s) at the same time
 - Multiple consumers may access request queue(s) at the same time
- Synchronizing access to other global data

Concurrency Aspects of Project 3 In Terms of Elevators

- Passengers may appear on a floor at the same time as an elevator arrives at a floor
- The procs display may be read at the same time that you're updating the number of passengers that you've serviced, or the total number of passengers on a given floor
- More, depending on your implementation strategy
- How do you guarantee correctness?

A Detour: Concurrency vs. Parallelism

- Concurrency – Multiple processes may access the same resource at the same time
- Parallelism – A process using multiple threads
- In elevator speak:
 - (REQUIRED) A producer may **concurrently** add passengers to the request queue while the elevator services requests
 - (NOT REQUIRED) The elevators, in **parallel**, may service requests

Global Data vs. Local Data

- Global Data – Data that is declared at global scope, e.g. outside of any function body
 - Often necessary for kernel programming
- Local Data – Data that is declared within a function
- Global data is particularly sensitive to concurrency issues.
 - Be **extra** careful when handling globals
- Local data is sensitive to concurrency issues when it depends on global data or when parallel access is possible
 - Think carefully about whether it needs to be synchronized

Synchronization Primitives

- Semaphores
 - User space
 - Kernel space
- Mutexes
 - User space
 - Kernel space
- Spin locks
 - Kernel space
- Atomic Functions

Synchronization Primitives (We'll Only Cover These)

- Mutexes
 - User space
 - Kernel space

Mutexes

- Mutex – A construct to provide MUTual EXclusion
- Based on the concept of a semaphore
- Can be locked or unlocked
 - Only one thread may hold the lock at a time
- More efficient than kernel semaphores
- Stricter than kernel semaphores
 - Also, more debug-friendly!

Kernel-Space Mutex - Initialization

```
DEFINE_MUTEX(mutex_name);
```

- Statically initializes a variable of type struct mutex with name mutex_name.

Kernel-Space Mutex - Locking

```
void mutex_lock(struct mutex *);  
int  mutex_lock_interruptible(struct mutex *);
```

- `mutex_lock()` guarantees that the mutex is locked – waits indefinitely
- `mutex_lock_interruptible()` locks a mutex as long as it is not interrupted
 - _ returns 0 if locking succeeded
 - _ returns < 0 if interrupted
 - In this case, just return `-ERESTARTSYS` to try again
- Use of the interruptible version is typically preferred

Kernel-Space Mutex - Unlocking

```
void mutex_unlock(struct mutex *);
```

- `mutex_unlock()` guarantees that the mutex is unlocked

Typical Kernel-Space Mutex Code

```
if (mutex_lock_interruptible(&gspca_dev->queue_lock))
    return -ERESTARTSYS;
gspca_dev->users--;

/* if the file did the capture, free the streaming resources */
if (gspca_dev->capt_file == file) {
    if (gspca_dev->streaming) {
        mutex_lock(&gspca_dev->usb_lock);
        gspca_stream_off(gspca_dev);
        mutex_unlock(&gspca_dev->usb_lock);
    }
    frame_free(gspca_dev);
    gspca_dev->capt_file = NULL;
    gspca_dev->memory = GSPCA_MEMORY_NO;
}
file->private_data = NULL;
module_put(gspca_dev->module);
mutex_unlock(&gspca_dev->queue_lock);
```

Typical Kernel-Space Mutex Code

```
if (mutex_lock_interruptible(&gspca_dev->queue_lock))
    return -ERESTARTSYS;
gspca_dev->users--;

/* if the file did the capture, free the streaming resources */
if (gspca_dev->capt_file == file) {
    if (gspca_dev->streaming) {
        mutex_lock(&gspca_dev->usb_lock);
        gspca_stream_off(gspca_dev);
        mutex_unlock(&gspca_dev->usb_lock);
    }
    frame_free(gspca_dev);
    gspca_dev->capt_file = NULL;
    gspca_dev->memory = GSPCA_MEMORY_NO;
}
file->private_data = NULL;
module_put(gspca_dev->module);
mutex_unlock(&gspca_dev->queue_lock);
```

User-Space Mutex - Initialization

```
int pthread_mutex_init(pthread_mutex_t *, NULL);  
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;  
int pthread_mutex_destroy(pthread_mutex_t *);
```

- `PTHREAD_MUTEX_INITIALIZER` statically allocates a mutex.
- `pthread_mutex_init()` dynamically allocates a mutex
- `pthread_mutex_destroy()` frees a mutex

User-Space Mutex - Locking

```
int pthread_mutex_lock(pthread_mutex_t *);
```

- Returns 0 on locking, < 0 otherwise

User-Space Mutex - Unlocking

```
int pthread_mutex_unlock(pthread_mutex_t *);
```

- Returns 0 on unlocking, < 0 otherwise

Project 3 Hints

- Refer to `linux/kernel/time.c`
- Implement elevator structures along with system calls
- Separate implementation from interface with the use of helper functions
- Review my sample code

Next Time:

- More Project 3 Hints
- Project 3 Demonstration Registration

Any Questions?