

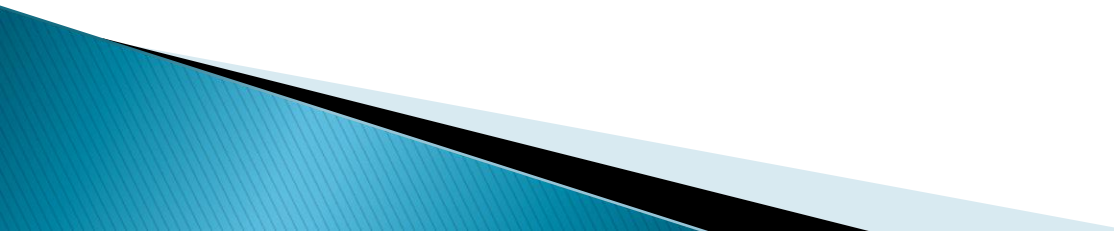
ECS1 50 Discussion 2

(4/9/07)

Adding System call

Header file needed:

```
#include <sys/types.h>
#include <sys/param.h>
#include <sys/proc.h>
#include <sys/module.h>
#include <sys/systm.h>
#include <sys/kernel.h>
#include <sys/systm.h>
```



Define system call arguments

- ▶ Simple struct:

```
struct mul2_args {  
    int x;  
};
```

Implement the function

```
static int mul2(struct thread* td, void* arg) {  
    struct mul2_args* a = (struct mul2_args*)arg;  
  
    /* Get param */  
    int x = a->x;  
  
    /* Set return value */  
    td->td_retval[0] = x * 10;  
  
    /* errno = 0 */  
    return 0;  
}
```

- ▶ Note 2 return values. `td->td_retval[0]` is the actual value returned from system call.
- ▶ Return 0 is returning error code to the kernel process.
- ▶ thread `*td` is the data structure containing process information of the process that called the system call.
 - For example: `td->td_proc->p_pid` returns the process id of that process.

Record for system call table

- ▶ Structure that gets put on system call table.

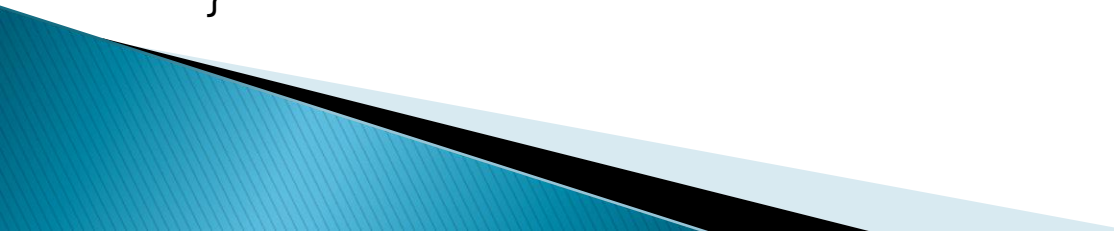
```
static struct sysent mul2_sysent = {  
    1,      // one arg  
    mul2    // function pointer  
};
```

- ▶ Define place on system call table to place this syscall at

```
// No set offset in syscall table;  
// just find the first available  
static int mul2_offset = NO_SYSCALL;
```

Initializing system call (Load/unload)

```
static int mul2_load(struct module* mod, int cmd, void* arg) {  
    int error = 0;  
  
    switch (cmd) {  
        case MOD_LOAD:  
            // initialize code goes here.  
            printf("mul2() loaded at %d\n", mul2_offset);  
            break;  
  
        case MOD_UNLOAD:  
            // do things when this module gets unloaded.  
            printf("mul2() unloaded from %d\n", mul2_offset);  
            break;  
  
        default:  
            error = EINVAL;  
            break;  
    }  
  
    return error;  
}
```



Macro to load the module

```
SYSCALL_MODULE(mu12, &mu12_offset, &mu12_sysent, mu12_load, NULL);
```

Makefile

```
SRCS = sample2.c
```

```
KMOD = sample2
```

```
.include <bsd.kmod.mk>
```

- ▶ Save that as Makefile, then type make
- ▶ You should get sample2.ko after the compilation is done.

Load/unload Module

- ▶ To Load
 - `kldload ./sample2.ko`
- ▶ To unload
 - `kldunload sample2`

Test program

```
#include <stdio.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <sys/module.h>

int main(void) {
    int syscall_num;
    struct module_stat stat;

    stat.version = sizeof(stat);
    modstat(modfind(" mul2 "), &stat);
    syscall_num = stat.data.intval;
    printf("mul2() is syscall #d\n", syscall_num);

    printf("mul2(0) = %d\n", syscall(syscall_num, 0));
    printf(" mul2(1) = %d\n", syscall(syscall_num, 1));
    printf("mul2(2) = %d\n", syscall(syscall_num, 2));
    printf("mul2(10) = %d\n", syscall(syscall_num, 10));
    printf("mul2(-1) = %d\n", syscall(syscall_num, -1));
}
```