

Homework (lec4. hw7~11)

7) sys_call_table[] is in arch/x86/kernel/syscall_table_32.S. How many system calls does Linux 2.6 support? What are the system call numbers for exit, fork, execve, wait4, read, write, and mkdir? Find system call numbers for sys_ni_syscall, which is defined at kernel/sys_ni.c. What is the role of sys_ni_syscall?

```
.long sys_linkat
.long sys_symlinkat
.long sys_readlinkat      /* 305 */
.long sys_fchmodat
.long sys_faccessat
.long sys_pselect6
.long sys_ppoll
.long sys_unshare          /* 310 */
.long sys_set_robust_list
.long sys_get_robust_list
.long sys_splice
.long sys_sync_file_range
.long sys_tee              /* 315 */
.long sys_vmsplice
.long sys_move_pages
.long sys_getcpu
.long sys_epoll_pwait
.long sys_utimensat        /* 320 */
.long sys_signalfd
.long sys_timerfd_create
.long sys_eventfd
.long sys_fallocate
.long sys_timerfd_settime   /* 325 */
.long sys_timerfd_gettime
```

328,2-9 Bot

326번 까지 있으므로 327개의 syscall 종류가 있다.

```
ENTRY(sys_call_table)
.long sys_restart_syscall    /* 0 - old "setup()" system call, used f
or restarting */
.long sys_exit
.long sys_fork
.long sys_read
.long sys_write
.long sys_open              /* 5 */
.long sys_close
.long sys_waitpid
.long sys_creat
.long sys_link
.long sys_unlink            /* 10 */
.long sys_execve
.long sys_chdir
.long sys_time
.long sys_mknod
.long sys_chmod             /* 15 */
.long sys_lchown16
.long sys_ni_syscall        /* old break syscall holder */
.long sys_stat
.long sys_lseek
.long sys_getpid            /* 20 */
.long sys_mount
```

21,2-9 Top

exit = 1, fork = 2, read = 3, write = 4, execve = 11

```

.long sys_fstat
.long sys_pause
.long sys_utime      /* 30 */
.long sys_ni_syscall /* old stty syscall holder */
.long sys_ni_syscall /* old gtty syscall holder */
.long sys_access
.long sys_nice
.long sys_ni_syscall /* 35 - old ftime syscall holder */
.long sys_sync
.long sys_kill
.long sys_rename
.long sys_mkdir
.long sys_rmdir      /* 40 */
.long sys_dup
.long sys_pipe
.long sys_times
.long sys_ni_syscall /* old prof syscall holder */
.long sys_brk        /* 45 */
.long sys_setgid16
.long sys_getgid16
.long sys_signal
.long sys_geteuid16
.long sys_getegid16  /* 50 */
.long sys_acct

```

41,2-9

9%

mkdir = 39

```

.long sys_newfstat
.long sys_uname
.long sys_iopl      /* 110 */
.long sys_vhangup
.long sys_ni_syscall /* old "idle" system call */
.long sys_vm86old
.long sys_wait4
.long sys_swapoff   /* 115 */
.long sys_sysinfo
.long sys_ipc
.long sys_fsync
.long sys_sigreturn
.long sys_clone     /* 120 */
.long sys_setdomainname
.long sys_newuname
.long sys_modify_ldt
.long sys_adjtimex
.long sys_mprotect  /* 125 */
.long sys_sigprocmask
.long sys_ni_syscall /* old "create_module" */
.long sys_init_module
.long sys_delete_module
.long sys_ni_syscall /* 130: old "get_kernel_syms" */
.long sys_quotactl

```

110,2-9

35%

wait4 = 114

```

.long sys_ni_syscall /* 35 - old ftime syscall holder */
.long sys_sync
.long sys_kill
.long sys_rename
.long sys_mkdir
.long sys_rmdir      /* 40 */
.long sys_dup
.long sys_pipe
.long sys_times
.long sys_ni_syscall /* old prof syscall holder */
.long sys_brk        /* 45 */
.long sys_setgid16
.long sys_getgid16
.long sys_signal
.long sys_geteuid16
.long sys_getegid16  /* 50 */
.long sys_acct
.long sys_unmount    /* recycled never used phys() */
.long sys_ni_syscall /* old lock syscall holder */
.long sys_ioctl
.long sys_fcntl      /* 55 */
.long sys_ni_syscall /* old mpx syscall holder */
.long sys_setpgid
.long sys_ni_syscall /* old ulimit syscall holder */

```

49,2-9

11%

sys_ni_syscall을 많이 찾아볼 수 있다.

```

/*
 * Non-implemented system calls get redirected here.
 */
asm linkage long sys_ni_syscall(void)
{
    return -ENOSYS;
}

cond_syscall(sys_nfsservctl);
cond_syscall(sys_quotactl);
cond_syscall(sys32_quotactl);
cond_syscall(sys_acct);
cond_syscall(sys_lookup_dcookie);
cond_syscall(sys_swapon);
cond_syscall(sys_swapon);
cond_syscall(sys_kexec_load);
cond_syscall(compat_sys_kexec_load);
cond_syscall(sys_init_module);
cond_syscall(sys_delete_module);
cond_syscall(sys_socketpair);
cond_syscall(sys_bind);
cond_syscall(sys_listen);
cond_syscall(sys_accept);
cond_syscall(sys_connect);

```

sys_ni_syscall을 찾아가보니 별다른 기능이 없는 것을 확인할 수 있다.
더 이상 사용되지 않는 syscall의 번호들 인 것 같다.

8) Change the kernel such that it prints "length 17 string found" for each printf(s) when the length of s is 17. Run a program that contains a printf() statement to see the effect. printf(s) calls write(1, s, strlen(s)) system call which in turn runs

```

mov eax, 4 ; eax--4. 4 is system call number for "write"
int 128

```

INT 128 will make the cpu stop running current process and jump to the location written in IDT[128]. IDT[128] contains the address of system_call (located in arch/x86/kernel/entry_32.S).

Finally, system_call will execute

```
call *sys_call_table(,%eax,4)
```

which eventually calls sys_write() since eax=4 for write() system call.

* Sometimes the compiler generate "sysenter" instead of "int 128". In this case the cpu jumps to ia32_sysenter_target (also in entry_32.S) instead of system_call.

```

#include <stdio.h>
int main(){
    printf("12141755 hyunho!\n");
}

```

"test.c" [converted] 4L, 64C 3,2-9 All

17글자를 출력하는 "test" 명령이다.

```
#include <stdio.h>
int main()
{
    printf("no 17 word\n");
}

"test1.c" [converted] 4L, 58C 4,1 All
17글자가 아닌 문자열을 출력하는 "test1" 명령이다.
```

```
asmlinkage ssize_t sys_write(unsigned int fd, const char __user * buf, size_t count)
{
    struct file *file;
    ssize_t ret = -EBADF;
    int fput_needed;

    if(count==17)
        printk("length 17 string found\n");

    file = fget_light(fd, &fput_needed);
    if (file) {
        loff_t pos = file_pos_read(file);
        ret = vfs_write(file, buf, count, &pos);
        file_pos_write(file, pos);
        fput_light(file, fput_needed);
    }

    return ret;
}

asmlinkage ssize_t sys_pread64(unsigned int fd, char __user *buf,
                                size_t count, loff_t pos)
{
    392,1 45%
```

count가 17일 때 출력문을 추가하였다.

```
localhost linux-2.6.25.10 # echo 8 > /proc/sys/kernel/printk
localhost linux-2.6.25.10 # ./test
length 17 string found
12141755 hyunho?
localhost linux-2.6.25.10 # ./test1
no 17 word
localhost linux-2.6.25.10 # _
```

kernel에서 printk를 출력하기 위해 우선순위를 바꿔준 후 실행하였다.
17글자인 문자열을 출력하는 “test”를 실행했을 때 잘 출력되는 것을 확인할 수 있다.

9) You can call a system call indirectly with “syscall()”.

write(1, “hi”, 2);

can be written as

syscall(4, 1, “hi”, 2); // 4 is the system call number for “write” system call

Write a program that prints “hello” in the screen using syscall.

```
#include <stdio.h>
int main(){
    write(1, "hi", 2);
}
```

"test.c" [converted] 4L, 53C 3,2-9 All

write를 이용해 “hi”를 출력하는 코드이다.

```
#include <stdio.h>
int main(){
    syscall(4, 1, "hi", 2);
}
```

"test1.c" [converted] 4L, 58C 3,2-9 All

syscall을 이용하여 “hi”를 출력하는 코드이다.

```
localhost linux-2.6.25.10 # ./test
hilocalhost linux-2.6.25.10 # ./test1
hilocalhost linux-2.6.25.10 #
```

write, syscall(4번) 모두 같이 "hi"가 출력되는 것을 확인할 수 있다.

10) Create a new system call, my_sys_call with system call number 17 (system call number 17 is one that is not being used currently). Define my_sys_call() just before sys_write() in read_write.c. Write a program that uses this system call:

```
void main(){
    syscall(17); // calls a system call with syscall number 17
}
```

When the above program runs, the kernel should display

hello from my_sys_call

To define a new system call with syscall number x

- insert the new system call name in arch/x86/kernel/syscall_table_32.S at index x
- define the function in appropriate file (such as "read_write.c")
- recompile and reboot

To use this system call in a user program

```
- void main(){
    syscall(x);
}
```

```
ENTRY(sys_call_table)
    .long sys_restart_syscall      /* 0 - old "setup()" system call, used f
or restarting */
    .long sys_exit
    .long sys_fork
    .long sys_read
    .long sys_write
    .long sys_open                /* 5 */
    .long sys_close
    .long sys_waitpid
    .long sys_creat
    .long sys_link
    .long sys_unlink              /* 10 */
    .long sys_execve
    .long sys_chdir
    .long sys_time
    .long sys_mknod
    .long sys_chmod              /* 15 */
    .long sys_lchown16
    .long my_sys_call
//    .long sys_ni_syscall          /* old break syscall holder */
    .long sys_stat
```

17번 syscall에 my_sys_call을 추가한 모습이다.

```

        if (file) {
            loff_t pos = file_pos_read(file);
            ret = vfs_read(file, buf, count, &pos);
            file_pos_write(file, pos);
            fput_light(file, fput_needed);
        }

        return ret;
    }

    asmlinkage void my_sys_call(){
        printk("hello from my_sys_call\n");
    }

    asmlinkage ssize_t sys_write(unsigned int fd, const char __user * buf, size_t count)
    {
        struct file *file;
        ssize_t ret = -EBADF;
        int fput_needed;

        file = fget_light(fd, &fput_needed);
        if (file) {

```

sys_write 위에 asmlinkage를 이용하여 my_sys_call 함수를 추가한 모습이다.

```

#include <stdio.h>
int main(){
    syscall(17);
}

"test.c" [converted] 4L, 47C

```

syscall에서 방금 내가 추가한 17번을 호출하는 코드이다.

```

localhost linux-2.6.25.10 # echo 8 > /proc/sys/kernel/printk
localhost linux-2.6.25.10 # ./test
hello from my_sys_call
localhost linux-2.6.25.10 # _

```

kernel에서 printk를 출력하기 위해 우선순위를 먼저 바꿔준다.
“test” 실행 시 정상적으로 my_sys_call이 실행되는 것을 확인할 수 있다.

11) Modify the kernel such that it displays the system call number for all system calls. Run a simple program that displays "hello" in the screen and find out what system calls have been called.

```
ENTRY(sys_call_table)
    .long sys_restart_syscall      /* 0 - old "setup()" system call, used f
or restarting */
    .long sys_exit
    .long sys_fork
    .long sys_read
    .long sys_write
    .long sys_open                /* 5 */
    .long sys_close
    .long sys_waitpid
    .long sys_creat
    .long sys_link
    .long sys_unlink              /* 10 */
    .long sys_execve
    .long sys_chdir
    .long sys_time
    .long sys_mknod
    .long sys_chmod               /* 15 */
    .long sys_lchown16
    .long sys_my_print            // 17
//    .long sys_ni_syscall          /* old break syscall holder */
    .long sys_stat

```

20,14-21 Top

syscall table의 17번에 sys_my_print를 추가한 모습이다.

```
    }
    return ret;
}

asmlinkage void sys_my_print(int sys_num){
    printk("sys_num : %d\n", sys_num);
}

asmlinkage ssize_t sys_write(unsigned int fd, const char __user * buf, size_t co
unt)
{
    struct file *file;
    ssize_t ret = -EBADF;
    int fput_needed;

    file = fget_light(fd, &fput_needed);
    if (file) {
        loff_t pos = file_pos_read(file);
        ret = vfs_write(file, buf, count, &pos);
        file_pos_write(file, pos);
        fput_light(file, fput_needed);
    }
}

```

369,0-1 44%

sys_write 위에 asmlinkage를 이용한 sys_my_print 함수를 정의한 모습이다.
매개변수로 sys_num을 가져와서 printk를 이용하여 출력하게 된다.

kernel에서 printk를 출력하기 위해 우선순위를 바꿔준다.
“hello”를 출력하는 동안 syscall 되는 번호들이 정상적으로 출력되는 것을 확인할 수 있다.