운영체제 과제#2

20212908 이진

- 파일 경로

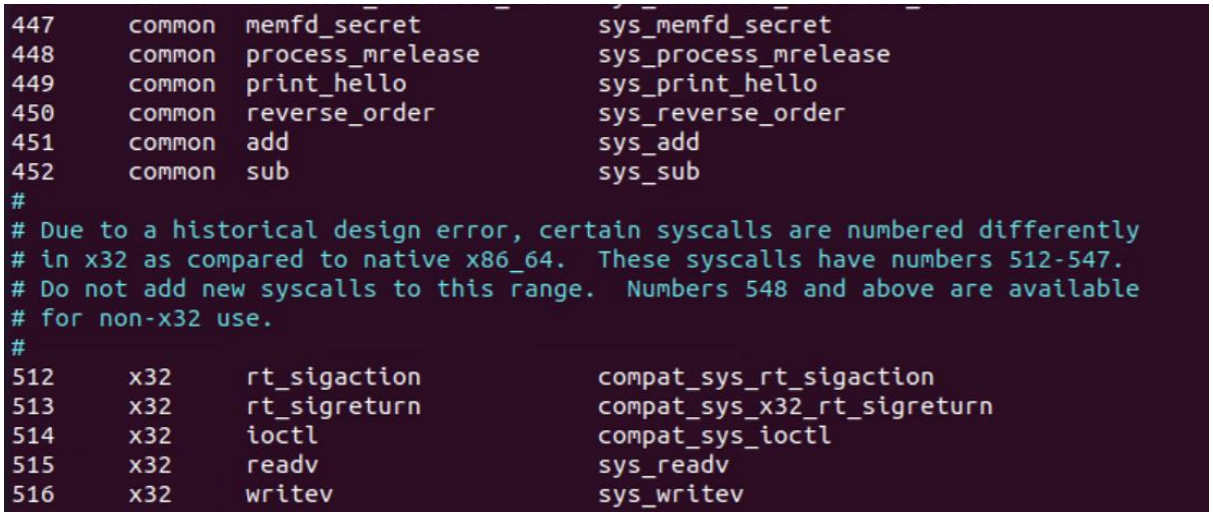
syscall_64.tbl : /usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls
syscalls.h : /usr/src/linux/linux-5.15.120/include/linux
sys_print_hello.c : /usr/src/linux/linux-5.15.120/kernel

실제 과제2에서는 쓰이지 않는 시스템콜 함수이지만, 과제하기 전에 테스트용을 위해 만들어뒀던 터라 함께 첨부하였습니다.

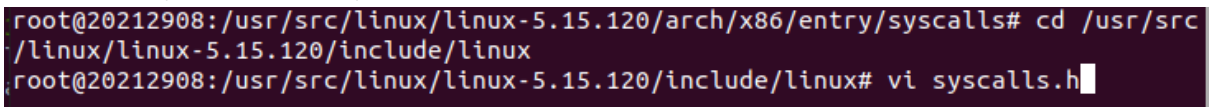
sys_reverse_order.c : /usr/src/linux/linux-5.15.120/kernel
sys_add.c : /usr/src/linux/linux-5.15.120/kernel
sys_sub.c : /usr/src/linux/linux-5.15.120/kernel
Makefile : /usr/src/linux/linux-5.15.120/kernel

- 시스템콜 추가

시스템콜 테이블 등록을 위해 /usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls 디렉토리로 이동한다. 64bit 운영체제 이므로 syscall_64.tbl 파일을 편집한다.



주석을 잘 읽어보고 사용 불가능한 구간을 유의해서 450번에 자릿수 역순 변경 시스템콜을 등록하고, 451번에 덧셈, 452번에 뺄셈하는 시스템콜을 등록한다.



/usr/src/linux/linux-5.15.120/include/linux 디렉토리로 이동 후 시스템콜 헤더 파일에 등록하기 위해 syscall.h 파일을 편집기로 열어준다.

```
long __do_semtimedop(int semid, struct sembuf *tsems, unsigned int nsops,
                     const struct timespec64 *timeout,
                     struct ipc_namespace *ns);

int __sys_getsockopt(int fd, int level, int optname, char __user *optval,
                int __user *optlen);
int __sys_setsockopt(int fd, int level, int optname, char __user *optval,
                int optlen);

asmlinkage long sys_print_hello(void);
asmlinkage long long sys_reverse_order(const char __user *, char __user *, size
_t);
asmlinkage long long sys_add(const long long, const long long);
asmlinkage long long sys_sub(const long long, const long long);

#endif
~
:wq
```

시스템콜 헤더 파일에 함수의 프로토타입을 정의한다. asmlinkage를 앞에 붙힘으로써 어셈블리 코드에서도 C 함수 호출이 가능해진다.

```
root@20212908:/usr/src/linux/linux-5.15.120/include/linux# cd /usr/src/linux/lin
ux-5.15.120/kernel
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi sys_reverse_order.c
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi sys_add.c
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi sys_sub.c
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi Makefile
```

/usr/src/linux/linux-5.15.120/kernel 디렉토리로 이동 후 추가할 시스템콜 구현 파일을 편집한다. 파일명은 상관없지만, 파일 내부의 함수는 sys_시스템콜 이름으로 작성해야 한다.

- 시스템콜 코드 설명

아래는 sys_reverse_order.c 코드의 설명이다.

```c
#include <linux/kernel.h>
#include <linux/syscalls.h>
#include <linux/uaccess.h>
// 사용자 공간의 메모리 접근을 위한 함수들이 포함된 헤더이다.
#include <linux/slab.h>
// kmalloc 과 kfree 와 같은 커널 메모리 할당 함수들이 포함된 헤더이다.

// 사용자 공간에서 전달된 문자열을 반전하여 다시 사용자 공간에 전달한다.
asmlinkage long long sys_reverse_order(const char __user *user_input, char __user *user_output, size_t len) {
    char *reversed_str;
    int i;

    // 커널 메모리에 문자열을 위한 공간을 할당한다.
    reversed_str = kmalloc(len + 1, GFP_KERNEL);
```

```
    // 사용자 공간에서 문자열을 읽어와 커널 공간으로 복사한다.
    if (copy_from_user(reversed_str, user_input, len)) {
        kfree(reversed_str);
        // copy_from_user 가 실패하면 -EFAULT 를 반환한다.
        return -EFAULT;
    }

    // 문자열의 끝을 나타내는 null 문자를 추가한다.
    reversed_str[len] = '\0';

    // 문자열을 역순으로 변경한다.
    for (i = 0; i < len / 2; i++) {
        char temp = reversed_str[i];
        reversed_str[i] = reversed_str[len - i - 1];
        reversed_str[len - i - 1] = temp;
    }

    // 반전된 문자열을 커널 공간에서 사용자 공간으로 복사한다.
    if (copy_to_user(user_output, reversed_str, len)) {
        kfree(reversed_str);
        return -EFAULT;
    }

    // 동적으로 할당한 메모리를 반환한다.
    kfree(reversed_str);
    return 0;
}

// 시스템 콜을 위한 매크로를 사용하여 함수를 정의한다.
// 매개변수가 3 개이므로 DEFINE3d 으로 정의한다.
SYSCALL_DEFINE3(reverse_order, const char __user *, user_input, char __user *, user_output, size_t, len) {
    return sys_reverse_order(user_input, user_output, len);
}
```

아래는 sys_add.c 코드이다. 간단하게 user mode에서 받은 2개의 인자를 더한 값을 리턴해 주는 코드이다.

```
root@20212908: /usr/src/linux/linux-5.15.120/kernel
#include <linux/kernel.h>
#include <linux/syscalls.h>

asmlinkage long long sys_add(const long long a, const long long b) {
        long long ret = a + b;
        printk("Add Result : %lld\n", ret);
        return ret;
}
SYSCALL_DEFINE2(sys_add, const long long, a, const long long, b) {
        return sys_add(a, b);
}
```

아래는 sys_sub.c 코드이다.마찬가지로 user mode에서 받은 2개의 인자를 뺀 값을 리턴해 주는 코드이다.

```
root@20212908: /usr/src/linux/linux-5.15.120/kernel
#include <linux/kernel.h>
#include <linux/syscalls.h>

asmlinkage long long sys_sub(const long long a, const long long b) {
        long long ret = a - b;
        printk("Sub Result : %lld\n", ret);
        return ret;
}
SYSCALL_DEFINE2(sys_sub, const long long, a, const long long, b) {
        return sys_sub(a, b);
}
~
```

```
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi sys_reverse_order.c
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi sys_add.c
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi sys_sub.c
root@20212908:/usr/src/linux/linux-5.15.120/kernel# vi Makefile
```

추가한 시스템콜이 다른 시스템콜과 함께 컴파일될 수 있도록 Makefile을 편집한다.

```
root@20212908: /usr/src/linux/linux-5.15.120/kernel

obj-y     = fork.o exec_domain.o panic.o \
            cpu.o exit.o softirq.o resource.o \
            sysctl.o capability.o ptrace.o user.o \
            signal.o sys.o umh.o workqueue.o pid.o task_work.o \
            extable.o params.o \
            kthread.o sys_ni.o nsproxy.o \
            notifier.o ksysfs.o cred.o reboot.o \
            async.o range.o smpboot.o ucount.o regset.o sys_print_hello.o \
            sys_reverse_order.o sys_add.o sys_sub.o

obj-$(CONFIG_USERMODE_DRIVER) += usermode_driver.o
obj-$(CONFIG_MODULES) += kmod.o
obj-$(CONFIG_MULTIUSER) += groups.o
```

추가해주었던 3개의 시스템콜 함수를 Makefile에 추가해주었다.

```
root@20212908:/usr/src/linux/linux-5.15.120/kernel# cd ..
root@20212908:/usr/src/linux/linux-5.15.120# make-kpkg --initrd --revision=3.0
kernel_image
```

커널 소스 디렉토리로 이동하여 새로 컴파일 후 재부팅을 실시한다. 기존의 revision값이 2.0이었으므로 그 값과 구분할 수 있도록 3.0이라는 다른 정수값을 입력하여 컴파일을 실시한다.

- **시스템콜 테스트**

```
root@20212908:/home/leejin/test1/hw2# cd /home/leejin/test1/hw2
root@20212908:/home/leejin/test1/hw2# vi assignment2.c
```

테스트 하고자 하는 폴더로 이동 후 새로운 c파일을 생성해 편집기로 연다.

아래는 assignment2.c 코드이다.

```c
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>

int main() {
    // getline 함수에 사용될 버퍼의 크기이다.
    size_t bufsize = 0;

    while (1) {
        printf("Input: ");
        // 입력 문자열을 저장할 포인터이다.
        char* input = NULL;

        // 표준 입력으로부터 한 줄을 읽는다.
        getline(&input, &bufsize, stdin);
        // 입력받은 문자열의 길이를 저장한다.
        size_t input_len = strlen(input);

        // 입력 길이가 0 이거나 개행 문자만 있는 경우 while 문을 빠져나온다. (프로그램 종료)
        if (input_len <= 0 || input[0] == '\n') break;

        // 개행 문자를 null 문자로 바꾼다.
        if (input[input_len - 1] == '\n') input[input_len - 1] = '\0';

        // 두 숫자 값을 저장할 변수이다.
```

```
        long long num1, num2;
        // 연산자(+, - 등)를 저장할 변수이다.
        char operator;
        // 출력 문자열 포인터이다. (현재는 입력과 동일)
        char* output = input;

        // 입력 문자열에서 두 숫자와 연산자를 추출한다.
        if (sscanf(input, "%lld%c%lld", &num1, &operator, &num2) == 3) {
            // 숫자가 자연수가 아닌 경우 예외처리한다.
            if (num1 <= 0 || num2 <= 0) {
                printf("Wrong Input!\n");
                continue;
            }
            // 연산자에 따라 적절한 시스템 콜을 호출한다.
            if (operator == '+') {
                // 시스템콜 452 번에 정의된 sys_sub 함수를 호출한다.
                // 인자는 입력받은 두 숫자를 넣어준다.
                long long return_value = syscall(452, num1, num2);
                printf("Output: %lld\n", return_value);
            } else if (operator == '-') {
                // 시스템콜 451 번에 정의된 sys_add 함수를 호출한다.
                // 인자는 입력받은 두 숫자를 넣어준다.
                long long return_value = syscall(451, num1, num2);
                printf("Output: %lld\n", return_value);
            } else {
                printf("Wrong Input!\n");
            }
        }
        // 입력 문자열에서 하나의 숫자를 추출한다.
        else if (sscanf(input, "%lld", &num1) == 1) {
            if (num1 <= 0) {
                printf("Wrong Input!\n");
                continue;
            }

            int flag = 0;
            // 입력 문자열이 숫자로만 이루어져있는지를 검사하기 위한 임시 포인터이다.
            char *tmp = input;

            // 입력이 숫자만으로 구성되어 있는지 확인한다.
```

```
                while (*tmp) {
                        if (!isdigit(*tmp)) {
                                flag = 1;
                                printf("Wrong Input!\n");
                                break;
                        }
                        tmp++;
                }

                // 잘못된 입력이면 다음 루프로 넘어간다.
                if (flag) continue;

                // 450 번으로 정의된 문자열을 반전시키는 시스템 콜을 호출한다.
                long long reverse_value = syscall(450, input, output, input_len);

                // 시스템 콜에 문제가 생긴 경우 오류 메시지를 출력한다.
                if (reverse_value == -1) {
                        perror("Error in syscall 450");
                }

                // 반전된 결과를 출력한다.
                printf("Output: %s\n", output + 1);
        }
        // 입력 문자열이 위 두가지 경우가 아닌 형태가 아닌 경우 예외 처리한다.
        else {
                printf("Wrong Input!\n");
        }

        // 할당된 메모리를 해제한다.
        free(input);
    }

    return 0;
}
```

- 출력 결과

```
root@20212908:/home/leejin/test1# ./assignment2.out
Input: 123
Output: 321
Input: 348957
Output: 759843
Input: 123+
Wrong Input!
Input:
root@20212908:/home/leejin/test1# ./assignment2.out
Input: 123+111
Output: 12
Input: 12350-13253
Output: 25603
Input: 123=123
Wrong Input!
Input: 123+-3
Wrong Input!
Input:
root@20212908:/home/leejin/test1#
```