

2022년 2학기 운영체제실습 4주차

System Call Programming

System Software Laboratory
School of Computer and Information Engineering
Kwangwoon Univ.

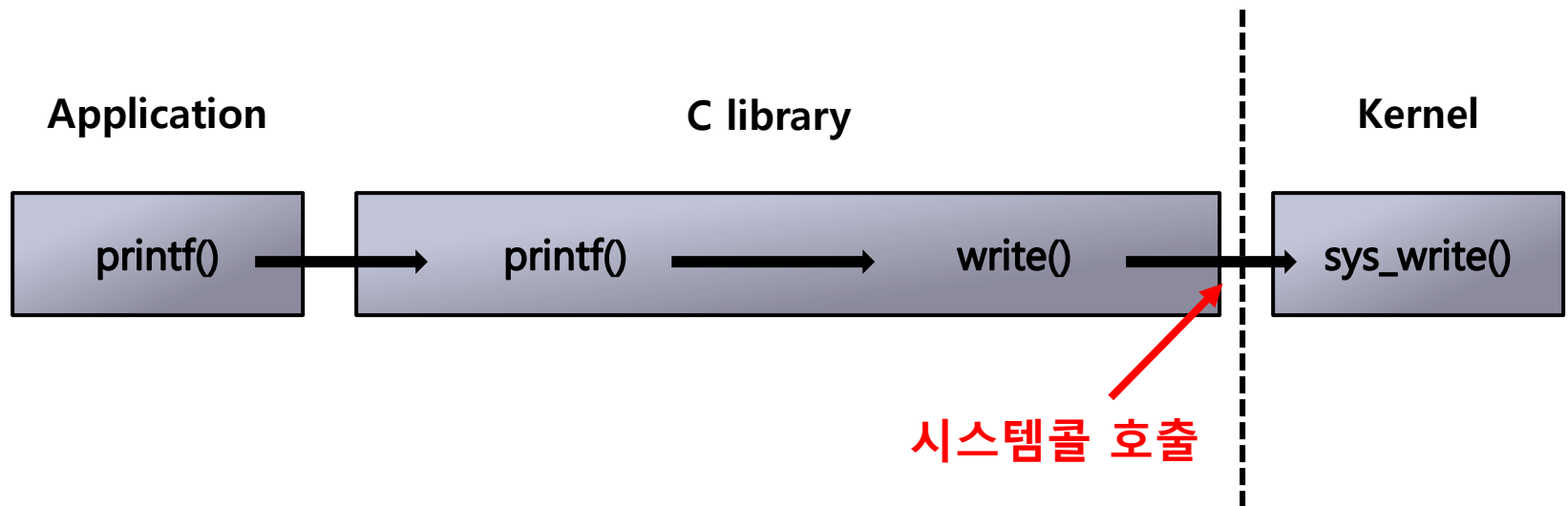
Contents

- **System Call**
 - 실습 1. 새로운 System Call 작성
 - 실습 2. 인자를 받는 System Call 작성
- **데이터 교환 함수**
- **Timer**
 - Jiffies
 - `current_kernel_time()`

System Call

■ System Call

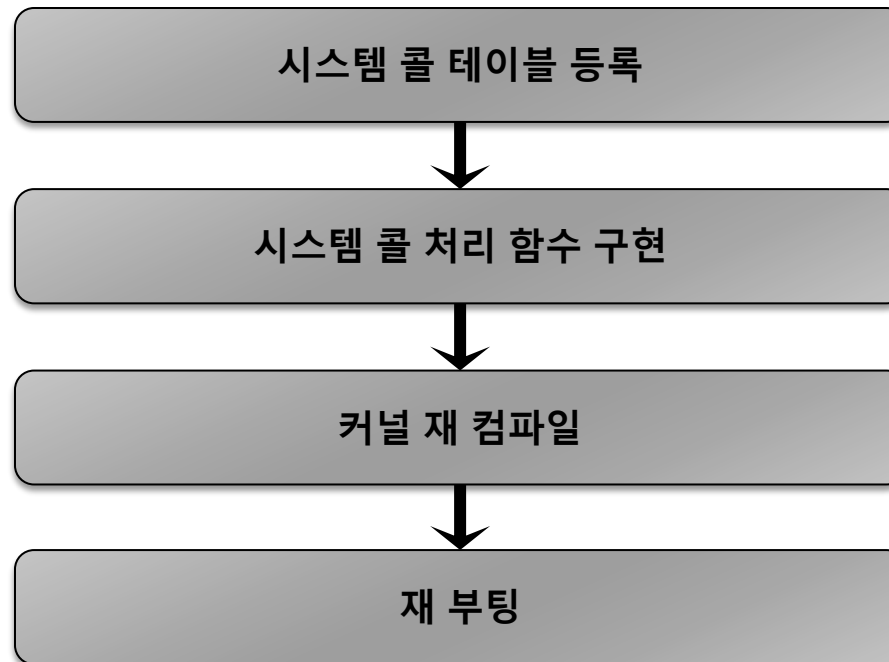
- 사용자 응용 프로그램에서 운영체제의 기능을 사용할 수 있게 해주는 통로
- 사용자 모드에 있는 프로세스가 CPU, disk, printer 등의 하드웨어 장치와 상호 작용할 수 있도록 기능을 제공하는 인터페이스
- 소프트웨어 인터럽트를 통해 사용자 프로그램에서 커널에게 보내는 서비스 요청



<printf() 호출 시 어플리케이션과 커널의 관계>

| System Call

- System Call 구현



실습 1. 새로운 System Call 작성

- System Call 테이블 등록

- \$ vi arch/x86/entry/syscalls/syscall_64.tbl

```
sslab@ubuntu:~/Downloads/linux-4.19.67$ vi arch/x86/entry/syscalls/syscall_64.tbl
```

- 맨 마지막으로 이동 후 입력
 - 548 common hello __x64_sys_hello

```
385 544 x32 io_submit      __x32_compat_sys_io_submit
386 545 x32 execveat      __x32_compat_sys_execveat/ptregs
387 546 x32 preadv2       __x32_compat_sys_preadv64v2
388 547 x32 pwritev2      __x32_compat_sys_pwritev64v2
389 548 common hello      __x64_sys_hello
```

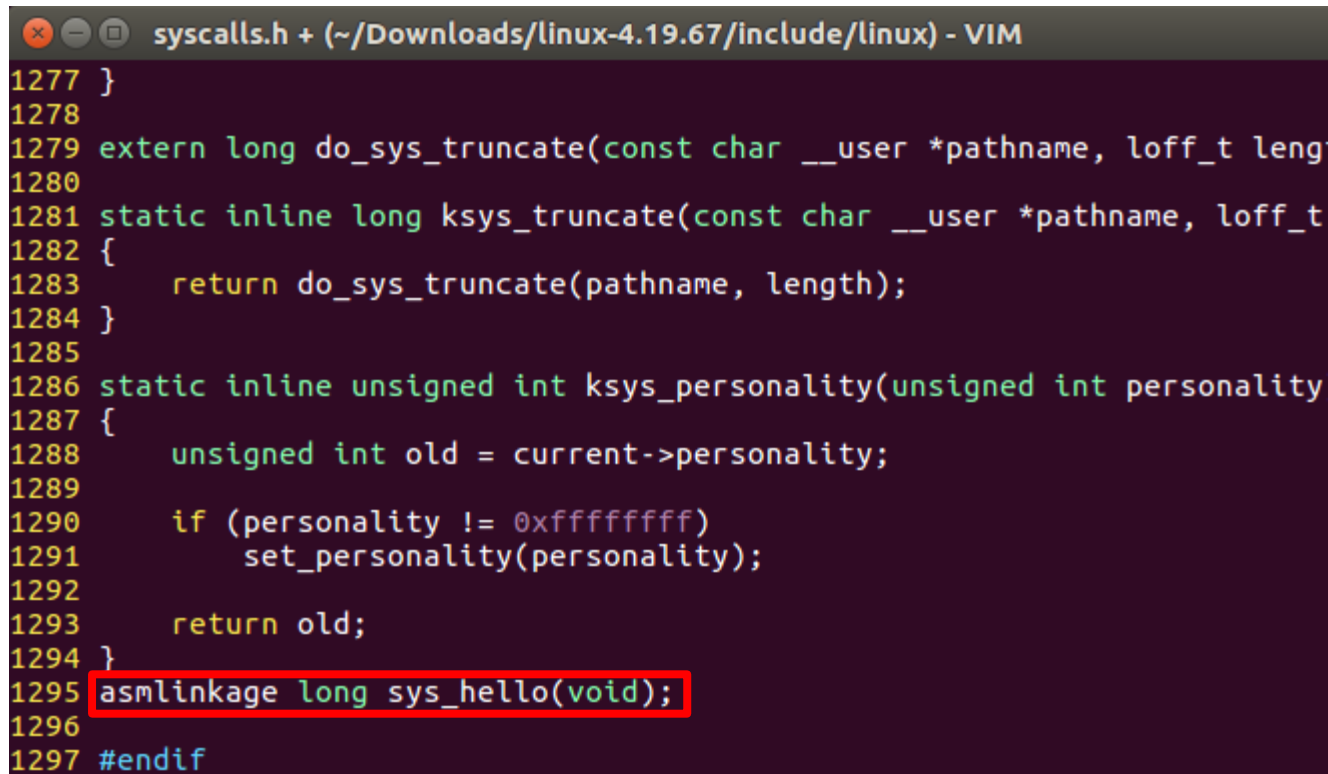
실습 1. 새로운 System Call 작성

System Call 테이블 등록

- \$ vi include/linux/syscalls.h

```
sslab@ubuntu:~/Downloads/linux-4.19.67$ vi include/linux/syscalls.h
```

- #endif 이전에 작성



```
syscalls.h + (~/Downloads/linux-4.19.67/include/linux) - VIM
1277 }
1278
1279 extern long do_sys_truncate(const char __user *pathname, loff_t leng
1280
1281 static inline long ksys_truncate(const char __user *pathname, loff_t
1282 {
1283     return do_sys_truncate(pathname, length);
1284 }
1285
1286 static inline unsigned int ksys_personality(unsigned int personality
1287 {
1288     unsigned int old = current->personality;
1289
1290     if (personality != 0xffffffff)
1291         set_personality(personality);
1292
1293     return old;
1294 }
1295 asmlinkage long sys_hello(void);
1296
1297 #endif
```

실습 1. 새로운 System Call 작성

System Call 함수 구현

- \$ mkdir hello
- \$ vi hello/hello.c

```
sslab@ubuntu:~/Downloads/linux-4.19.67$ mkdir hello
sslab@ubuntu:~/Downloads/linux-4.19.67$ vi hello/hello.c
```

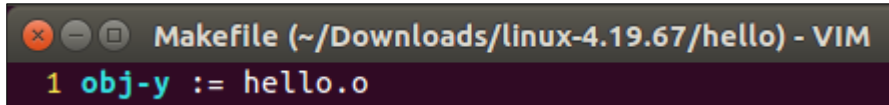
```
1 #include <linux/kernel.h>
2 #include <linux/syscalls.h>
3
4 SYSCALL_DEFINE0(hello)
5 {
6     printk("Hello world\n");
7
8     return 0;
9 }
```

- SYSCALL_DEFINE0(hello)
 - "hello" 라는 이름으로 시스템 콜을 생성하는 매크로
 - 이를 수행함으로써, sys_hello 뿐만 아니라 다양한 함수들이 자동적으로 생성
 - 강의 자료 5페이지의 __x64_sys_hello도 본 단계에서 자동으로 생성됨
 - 숫자 "0"은 해당 시스템 콜이 인자를 0개 받는 것을 의미
 - 인자 수에 따라 다음과 같은 것들이 존재
 - SYSCALL_DEFINE1(), SYSCALL_DEFINE2(), ... SYSCALL_DEFINE6()

실습 1. 새로운 System Call 작성

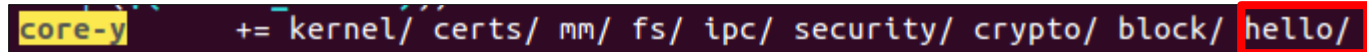
■ System Call 함수 구현

- \$ vi hello/Makefile



```
Makefile (~/Downloads/linux-4.19.67/hello) - VIM
1 obj-y := hello.o
```

- \$ vi Makefile
 - core-y 패턴 검색 후 2번 째 결과에서



```
core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hello/
```

■ 커널 컴파일 후 재부팅

- \$ make -jn
- \$ make modules_install
- \$ make install
- \$ reboot
 - 방금 컴파일 한 커널로 부팅

실습 1. 새로운 System Call 작성

- 새로운 System Call을 테스트 할 프로그램 작성 및 동작

- \$ mkdir ~/working
- \$ cd ~/working
- \$ vi hello_test.c

```
sslab@ubuntu:~$ mkdir ~/working
sslab@ubuntu:~$ cd ~/working
sslab@ubuntu:~/working$ vi hello_test.c
```

```
#include <stdio.h> /* printf() */
#include <unistd.h> /* syscall() */
#include <sys/syscall.h> /* syscall() */

int main(void)
{
    long ret;

    ret = syscall(548);
    printf("%ld\n", ret);

    return 0;
}
```

실습 1. 새로운 System Call 작성

- 새로운 System Call을 테스트 할 프로그램 작성 및 동작 (cont'd)

- `$./a.out`

```
sslab@ubuntu:~/working$ gcc hello_test.c
sslab@ubuntu:~/working$ ls
a.out  hello_test.c
sslab@ubuntu:~/working$ ./a.out
0
```

- `$ dmesg`

```
[ 8.708961] Bluetooth: BNEP socket layer initialized
[ 8.922665] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[ 8.928159] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[ 8.937714] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
None
[ 8.938406] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[ 9.728541] Bluetooth: RFCOMM TTY layer initialized
[ 9.728545] Bluetooth: RFCOMM socket layer initialized
[ 9.728549] Bluetooth: RFCOMM ver 1.11
[ 193.055285] Hello world
```

- 시스템 콜 호출을 실패한 경우, `syscall()` 함수에서 -1 반환

실습 2. 인자를 받는 System Call 작성

System Call 테이블 등록

- \$ vi arch/x86/entry/syscalls/syscall_64.tbl
 - 아래와 같이 작성

```
385 544 x32 io_submit      __x32_compat_sys_io_submit
386 545 x32 execveat      __x32_compat_sys_execveat/ptregs
387 546 x32 preadv2       __x32_compat_sys_preadv64v2
388 547 x32 pwritev2      __x32_compat_sys_pwritev64v2
389 548 common hello      x64 sys hello
390 549 common add        __x64_sys_add
```

System Call 테이블 등록

- \$ vi include/linux/syscalls.h
 - #endif 위에 작성

```
1286 static inline unsigned int ksys_personality(unsigned int personality)
1287 {
1288     unsigned int old = current->personality;
1289
1290     if (personality != 0xffffffff)
1291         set_personality(personality);
1292
1293     return old;
1294 }
1295
1296 asmlinkage long sys hello(void);
1297 asmlinkage long sys_add(int, int);
1298
1299 #endif
```

실습 2. 인자를 받는 System Call 작성

- System Call 함수 구현

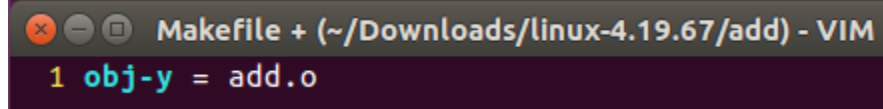
- \$ mkdir add
- \$ vi add/add.c

```
1 #include <linux/kernel.h>
2 #include <linux/syscalls.h>
3
4 SYSCALL_DEFINE2(add, int, a, int, b)
5 {
6     long ret;
7
8     ret = a+b;
9
10    return ret;
11 }
```

실습 2. 인자를 받는 System Call 작성

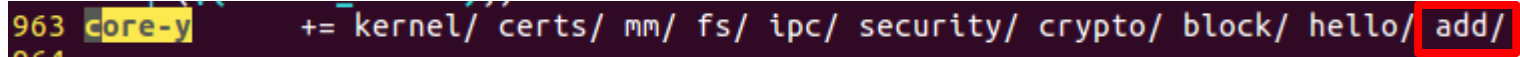
- System Call 함수 구현

- \$ vi add/Makefile



```
Makefile + (~/.Downloads/linux-4.19.67/add) - VIM
1 obj-y = add.o
```

- \$ vi Makefile
 - core-y 패턴 검색 후 2번 째 결과에서



```
963 core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hello/ add/
```

- 커널 컴파일 후 재부팅

실습 2. 인자를 받는 System Call 작성

- 새로운 System Call을 테스트 할 프로그램 작성 및 동작
 - \$ cd ~/working
 - \$ vi add_test.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>

int main(void)
{
    int a,b;
    long ret;

    a = 7;
    b = 4;

    ret = syscall(549, a,b);
    printf("%d op. %d = %ld\n", a, b, ret);

    return 0;
}
```

```
sslab@ubuntu:~/working$ gcc add_test.c
sslab@ubuntu:~/working$ ls
add_test.c  a.out  hello_test.c
sslab@ubuntu:~/working$ ./a.out
7 op. 4 = 11
sslab@ubuntu:~/working$
```

데이터 교환 함수

- 사용자 영역과 커널 영역 사이에서 값을 교환하는 커널 내 함수
 - included in <asm/uaccess.h>
 - A simple variable
 - put_user(void *x, void *ptr) → 커널 영역의 데이터를 사용자 영역으로 복사
 - get_user(void *x, void *ptr) → 사용자 영역의 데이터를 커널 영역으로 복사
 - x : variable to store
 - ptr : source address
 - A block of data
 - copy_to_user(void *to, void *from, unsigned long n)
 - 커널 영역의 블록 데이터를 사용자 영역으로 복사
 - copy_from_user(void *to, void *from, unsigned long n)
 - 사용자 영역의 블록 데이터를 커널 영역으로 복사
 - to : destination address
 - from : source address
 - n : # of bytes to copy

Timer

- 시간 정보에 관한 전역 변수
 - Jiffies, HZ, xtime
- Jiffies
 - 시스템에 내장된 타이머에서 주기적으로 발생시키는 인터럽트
 - 시스템이 시작된 이후 경과된 타이머 Tick의 수를 저장

Timer

■ HZ (진동 수)

- 초당 몇 번 tick이 발생하는가를 의미
- Tick: 인터럽트 사이의 시간 주기 ($\text{tick} = 1/\text{HZ}$)
 - i386의 경우 커널 2.4에서는 100, 2.6.10 이전 버전의 커널에서는 1000, 2.6.10 이후 커널에서는 250을 기본값으로 함
- 진동수가 높을 수록 시스템의 정확도가 높아지나, 타이머 인터럽트의 처리 비용이 높아짐

■ xtime

- 현재 시각(wall time)이 xtime변수로 정의되어 있음
- `xtime.tv_sec`은 1970년 1월 1일 이후부터 지금까지의 시간(초 단위)을 의미
- `xtime.tv_nsec`은 마지막 초 이후에 경과된 나노 초를 의미

| jiffies

- **jiffies**

- 시스템이 부팅된 이후 발생한 tick 수를 저장

- **사용법**

- 초를 jiffies로 변환하는 방법
 - $\text{Second} * \text{HZ}$
- Jiffies를 초로 변환하는 방법
 - $\text{Jiffies} / \text{HZ}$

jiffies

- **사용 예제**

- 현재 시각 : unsigned long time_stamp = jiffies;
- 현재로부터 1tick 후 : unsigned long next_tick = jiffies + 1;
- 현재로부터 5초 후 : unsigned long later = jiffies + 5 * HZ;

- **jiffies in Kernel 2.6**

- jiffies 값을 64비트로 변경
 - 64비트 jiffies값은 jiffies_64변수로 선언
- jiffies_64값을 참조하기 위한 함수
 - get_jiffies_64()

current_kernel_time()

- **current_kernel_time()**

- 현재 시각(wall time)

```
struct timespec current_kernel_time(void);  
  
struct timespec {  
    time_t      tv_sec;          /* seconds */  
    long        tv_nsec;        /* nanoseconds */  
};
```

- `current_kernel_time().tv_sec`
 - 1970년 1월 1일 이후 지금까지의 시간(epoch time)을 초단위로 저장
 - `current_kernel_time().tv_nsec`
 - 마지막 초 이후에 경과된 나노 초

Assignment 2

- 제출 기한: 2022. 09.22(목) ~ 2022.10.13(목) 23:59:59
- Delay 없음
- 업로드 양식에 어긋날 경우 감점 처리
- Hardcopy 제출하지 않음