과제**1.** 시스템 콜 과정 이해하기

CPU
커널모드

사용자 뿐만 아니라 커널 영역 메모리도 접근 가능

```
main()
{
    read()
}
```

1.C언어로 작성된 프로그램 호출

```
read()
{
    movl 0, %eax
    int $0x80
}
```

(0을 %eax에 전달, sys_read의 고유번호)

2.시스템 콜 호출 전, GNU C Library 에서 함수 호출

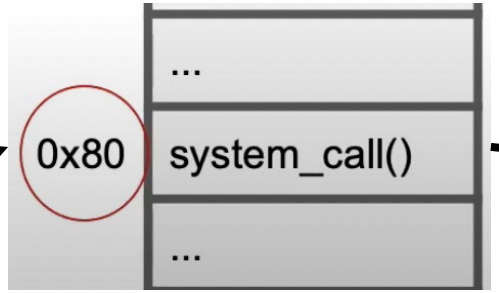
```
$ ldd --version
ldd (Ubuntu GLIBC 2.31-0ubuntu9.2) 2.31
Copyright (C) 2020 Free Software Foundation, Inc.
이 프로그램은 자유 소프트웨어입니다. 복사조건은 소스를 참조하십시오.  상품성
이나 특정 목적에 대한 적합성을 비롯하여 어떠한 보증도 하지 않습니다.
만든 사람: Roland McGrath 및 Ulrich Drepper.
```

CPU
사용자모드

사용자 영역 메모리에만 접근 가능

3. 128번 인터럽트 발생 (시스템콜 호출)

IDT(Interrupt Descriptor Table)



4.시스템 콜 테이블에서 실제 처리하는 sys_read 호출 (포인터 역할)

```
# The format is:
# <number> <abi> <name> <entry point>
#
# The abi is "common", "64" or "x32" for this file.
#
0       common  read                    sys_read
1       common  write                   sys_write
2       common  open                    sys_open
```

5. 리눅스 소스코드 확인
./arch/x86/include/linux/asm/irq_vectors.h

```
50 #define IA32_SYSCALL_VECTOR                0x80
```

```
101 #if defined(CONFIG_IA32_EMULATION)
102         SYSG(IA32_SYSCALL_VECTOR,       entry_INT80_compat),
103 #elif defined(CONFIG_X86_32)
104         SYSG(IA32_SYSCALL_VECTOR,       entry_INT80_32),
```

```
* Arguments:
* eax  system call number
* ebx  arg1
* ecx  arg2
* edx  arg3
* esi  arg4
* edi  arg5
* ebp  user stack
* 0(%ebp) arg6
*/
```

6. 0x80이 발생하면 entry_INT80 관련 내용을 수행해라

7. 엔트리에는 인터럽트에 따라 처리할 일이 어셈블리 언어로 작성되어 있음
(eax system call number 확인하기, 여기서 0번)

```
0       common  read                    sys_read
```

9. 헤더 확인(선언된 함수)
./include/linux/syscalls.h

```
598 asmlinkage long sys_read(unsigned int fd, char __user *buf, size_t count);
599 asmlinkage long sys_readahead(int fd, loff_t offset, size_t count);
```

과제**2.** 시스템 콜 실습

과제**2-1.** 새로운 시스템 콜 추가하기

(정답은 마지막 장에 2-2 정답과 함께 표기)

1.시스템콜번호할당

2. 시스템콜함수구현

```
파일  머신  보기  입력  장치  도움말
317     common  seccomp                 sys_seccomp
318     common  getrandom               sys_getrandom
319     common  memfd_create            sys_memfd_create
320     common  kexec_file_load         sys_kexec_file_load
321     common  bpf                     sys_bpf
322     64      execveat                sys_execveat/ptregs
323     common  userfaultfd             sys_userfaultfd
324     common  membarrier              sys_membarrier
325     common  mlock2                  sys_mlock2
326     common  copy_file_range         sys_copy_file_range
327     64      preadv2                 sys_preadv2
328     64      pwritev2                sys_pwritev2
329     common  pkey_mprotect           sys_pkey_mprotect
330     common  pkey_alloc              sys_pkey_alloc
331     common  pkey_free               sys_pkey_free
332     common  statx                   sys_statx
333     common  print_student_id        sys_print_student_id
334     common  print_student_info      sys_print_student_info
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
512     x32     rt_sigaction            compat_sys_rt_sigaction
513     x32     rt_sigreturn            sys32_x32_rt_sigreturn
514     x32     ioctl                   compat_sys_ioctl
515     x32     readv                   compat_sys_readv
516     x32     writev                  compat_sys_writev
517     x32     recvfrom                compat_sys_recvfrom
518     x32     sendmsg                 compat_sys_sendmsg
519     x32     recvmsg                 compat_sys_recvmsg
520     x32     execve                  compat_sys_execve/ptregs
521     x32     ptrace                  compat_sys_ptrace
522     x32     rt_sigpending           compat_sys_rt_sigpending
523     x32     rt_sigtimedwait         compat_sys_rt_sigtimedwait
524     x32     rt_sigqueueinfo         compat_sys_rt_sigqueueinfo
                                                ltstack
"arch/x86/entry/syscalls/syscall_64.tbl" 383L, 13360C                343,52-62     93%
```

```c
#include <linux/unistd.h>
#include <linux/errno.h>
#include <linux/kernel.h>
#include <linux/sched.h>

asmlinkage void sys_print_student_id(void)
{

        // TODO
        printk("My Student ID is 2013130890");

}
EXPORT_SYMBOL_GPL(sys_print_student_id);

asmlinkage void sys_print_student_info(char *name, char *major)
{
        // TODO
        printk("My Name is %s \n", name);
        printk("I major in %s", major);

}
EXPORT_SYMBOL_GPL(sys_print_student_info);
```

과제**2.** 시스템 콜 실습

과제**2-2.** 시스템 콜에 매개변수 전달하기

시스템콜함수선언➡️사용자영역프로그램작성➡️작성한 프로그램 실행 ➡️**dmesg** 명령어로 결과 확인
(2-1, 2-2 정답 함께 출력)

```
/*
 *_my code
 */
asmlinkage void sys_print_student_id(void);
asmlinkage void sys_print_student_info(char *name, char *major);

#endif
"include/linux/syscalls.h" 949L, 40939C                    944,3         Bot
```

```
int main(void)
{
        // TODO: Write your code here
        const char* arr[2];
        arr[0] = "SEUNG TAE KIM";
        arr[1] = "English Language and Literature";
        syscall(333);
        syscall(334,arr[0],arr[1]);
        return 0;
}
```

```
[    9.734565] RAPL PMU: hw unit of domain pp1-gpu 2^-0 Joules
[    9.734565] RAPL PMU: hw unit of domain psys 2^-0 Joules
[   39.908108] snd_intel8x0 0000:00:05.0: white list rate for 1028:0177 is 48000
[   39.969959] EXT4-fs (sda2): mounted filesystem with ordered data mode. Opts: (null)
[   40.210385] audit: type=1400 audit(1617337001.156:2): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/lib/snapd/snap-confine" pid=779 comm="apparmor_parser"
[   40.210392] audit: type=1400 audit(1617337001.156:3): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/lib/snapd/snap-confine//mount-namespace-capture-helper" pid=779 comm
="apparmor_parser"
[   40.214499] audit: type=1400 audit(1617337001.160:4): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/bin/lxc-start" pid=777 comm="apparmor_parser"
[   40.235583] audit: type=1400 audit(1617337001.180:5): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/bin/man" pid=778 comm="apparmor_parser"
[   40.235588] audit: type=1400 audit(1617337001.180:6): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="man_filter" pid=778 comm="apparmor_parser"
[   40.235590] audit: type=1400 audit(1617337001.180:7): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="man_groff" pid=778 comm="apparmor_parser"
[   40.239525] audit: type=1400 audit(1617337001.184:8): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="lxc-container-default" pid=775 comm="apparmor_parser"
[   40.239533] audit: type=1400 audit(1617337001.184:9): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="lxc-container-default-cgns" pid=775 comm="apparmor_parser"
[   40.239536] audit: type=1400 audit(1617337001.184:10): apparmor="STATUS" operation="profile_load"
 profile="unconfined" name="lxc-container-default-with-mounting" pid=775 comm="apparmor_parser"
[   40.239540] audit: type=1400 audit(1617337001.184:11): apparmor="STATUS" operation="profile_load"
 profile="unconfined" name="lxc-container-default-with-nesting" pid=775 comm="apparmor_parser"
[   42.377593] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   42.384692] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[   42.385212] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   45.141756] new mount options do not match the existing superblock, will be ignored
[   84.679806] My Student ID is 2013130890
[   84.679807] My Name is SEUNG TAE KIM
[   84.679808] I major in English Language and Literature
```

```
os-practice:  ~
→ gcc -o assignment assignment.c
assignment.c: In function ‘main’:
assignment.c:9:2: warning: implicit declaration of function ‘syscall’ [-Wimplicit-function-declarati
on]
  syscall(333);
  ^~~~~~~


os-practice:  ~
→ ./assignment

os-practice:  ~
→ dmesg_
```

```
[    84.679806] My Student ID is 2013130890
[    84.679807] My Name is SEUNG TAE KIM
[    84.679808] I major in English Language and Literature
```