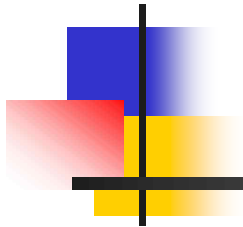


임베디드시스템 설계 강의자료 6

system call 2/2

(2014년도 1학기)



김 영 진

아주대학교
전자공학과

System call table and linkage

| Offset | Symbol | sys_call_table | System call location |
|--------|-----------------------------------|--|---------------------------------------|
| 0 | __NR_restart_syscall | .long sys_restart_syscall | --> ./linux/kernel/signal.c |
| 4 | __NR_exit | .long sys_exit | --> ./linux/kernel/exit.c |
| 8 | __NR_exit | .long sys_fork | --> ./linux/arch/386/kernel/process.c |
| 1272 | __NR_getcpu | .long sys_getcpu | --> ./linux/kernel/sys.c |
| 1276 | __NR_epoll_pwait | .long sys_epoll_pwait | --> ./linux/kernel/sys_ni.c |
| | __NR_syscalls | — — — — — | |
| | ↑ ./linux/include/asm/unistd.h | ↑ ./linux/arch/386/kernel/syscall_table.S | |

❖ Ref. <http://www.ibm.com/developerworks/linux/library/l-system-calls/>



SYSCALL_DEFINE 함수

❖ SYSCALL_DEFINEx

- Invoking a system call without library support (i.e., wrapper function)
- Macro 함수로 [\[kernel4412\] / include/linux/syscalls.h](#) 에 있음
- 문법은 다음과 같음
 - ✓ SYSCALL_DEFINE0(func-name)
 - ✓ SYSCALL_DEFINE1(func-name, arg1-type, arg1-name)
 - ✓ SYSCALL_DEFINE2(func-name, arg1-type, arg1-name, arg2-type, arg2-name)
- ...
- 매개변수(args)의 수에 따라서 SYSCALL_DEFINE0 ~ SYSCALL_DEFINE6 까지 존재
- Header file
 - ✓ #include <linux/syscalls.h>
- 리눅스 2.6.34 버전부터 `__syscall` 매크로 대신에 SYSCALL_DEFINE 매크로가 사용되고 있음
 - ✓ http://lxr.free-electrons.com/ident?v=3.0;i=SYSCALL_DEFINE



매개변수를 가진 시스템 호출

❖ 학습 내용

- 사용자 프로그램에서 `int` 1개를 입력 받아서 `system call` 호출시 매개변수로 하여 커널에 넘겨주고 커널의 `system call` 함수에서는 이 값을 5증가시켜 다시 사용자 응용으로 반환함
- 사용자 프로그램과 커널간의 데이터 이동시 문제 파악
- 사용자 프로그램과 커널간의 데이터 이동 문제 해결

매개변수를 가진 시스템 호출

❖ System call 번호 할당

- [kernel4412] / arch/arm/include/asm/unistd.h
- 새로 추가할 system call의 고유번호 정의 추가
#define __NR_newcall (__NR_SYSCALL_BASE+377)

```
#define __NR_fanotify_init      (__NR_SYSCALL_BASE+367)
#define __NR_fanotify_mark     (__NR_SYSCALL_BASE+368)
#define __NR_prlimit64        (__NR_SYSCALL_BASE+369)
#define __NR_name_to_handle_at (__NR_SYSCALL_BASE+370)
#define __NR_open_by_handle_at (__NR_SYSCALL_BASE+371)
#define __NR_clock_adjtime    (__NR_SYSCALL_BASE+372)
#define __NR_syncfs           (__NR_SYSCALL_BASE+373)
#define __NR_sendmmsg         (__NR_SYSCALL_BASE+374)
#define __NR_setns            (__NR_SYSCALL_BASE+375)
#define __NR_hellocall        (__NR_SYSCALL_BASE+376)
#define __NR_newcall          (__NR_SYSCALL_BASE+377)
/*
 * The following SWIs are ARM private.
 */
```

매개변수를 가진 시스템 호출

❖ System call 테이블에 system call 처리 함수 등록

- [kernel4412] / arch/arm/kernel/calls.S 수정
 - ➔ calls.S에 CALL(sys_newcall1) 등록
- unistd.h에서 정의한 호출번호와 일치

```
/* 370 */      CALL(sys_name_to_handle_at)
               CALL(sys_open_by_handle_at)
               CALL(sys_clock_adjtime)
               CALL(sys_syncfs)
               CALL(sys_sendmmsg)
/* 375 */      CALL(sys_setns)
               CALL(svs_hellocall)
               CALL(sys_newcall)
#ifdef syscalls_counted
.equ syscalls_padding, ((NR_syscalls + 3) & ~3) - NR_syscalls
#define syscalls_counted
#endif
.rept syscalls_padding
               CALL(sys_ni_syscall)
.endr
```

매개변수를 가진 시스템 호출

- ❖ **System call 처리 함수를 syscalls.h에 함수 선언 등록**
 - [kernel4412] / include/linux/syscalls.h 수정
 - ➔ Function prototype declaration

```
asmlinkage long sys_open_by_handle_at(int mountdirfd,
                                     struct file_handle __user *handle,
                                     int __user *mnt_id, int flag);
asmlinkage long sys_setns(int fd, int nstype);
asmlinkage long sys_hellocall(int a);
asmlinkage long sys_newcall(int a);
#endif
```

850,1



매개변수를 가진 시스템 호출

❖ System call 처리 함수 구현

- [kernel4412] / kernel/newcall.c 작성
- user 영역으로 부터 넘겨받은 값을 출력하고 5를 증가시켜서 return
- printk , a = a + 5; (kernel operation)

```
#include <linux/syscalls.h>
#include <asm/unistd.h>

SYSCALL_DEFINE1(newcall, int, a)
{
    printk("\n newcall : kernel a = %d \n", a);

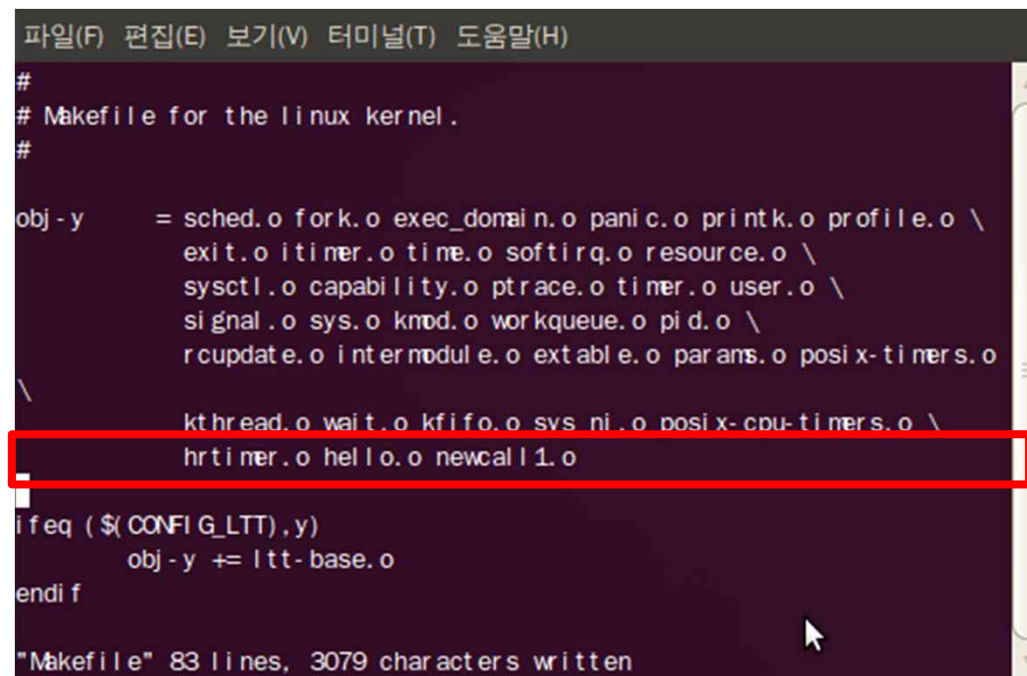
    a+=5; //kernel operation

    return a;
}
```


매개변수를 가진 시스템 호출

❖ Makefile 수정

➤ [kernel4412] /kernel/Makefile



```
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
#
# Makefile for the linux kernel.
#
obj-y      = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
             exit.o itimer.o time.o softirq.o resource.o \
             sysctl.o capability.o ptrace.o timer.o user.o \
             signal.o sys.o knod.o workqueue.o pid.o \
             rcupdate.o intermodule.o extable.o params.o posix-timers.o \
             \
             kthread.o wait.o kfifo.o svcs.o posix-cpu-timers.o \
             hrtimer.o hello.o newcall.o
ifeq ($(CONFIG_LTT),y)
    obj-y += ltt-base.o
endif
"Makefile" 83 lines, 3079 characters written
```

매개변수를 가진 시스템 호출

❖ Kernel Compile

- Ubuntu에서 다음을 수행
- # make zImage

```
root@ubuntu: /4412_Linux/kernel_4412
1      CREDITS      include lib      README      System.map
arch      crypto      init  MAINTAINERS  REPORTING-BUGS  tools
block     Documentation  ipc   Makefile     samples      usr
CHANGELOG.txt  drivers  Kbuild  mm      scripts      virt
config0625  firmware Kconfig Module.symvers security      vmlinux
COPYING     fs      kernel  net      sound        vmlinux.o

root@ubuntu: /4412_Linux/kernel_4412# make zImage
CHK      include/linux/version.h
CHK      include/generated/utsrelease.h
make[1]: `include/generated/mach-types.h' is up to date.
```

❖ Kernel fusing

- Windows에서 다음을 수행
- > fastboot.exe flash kernel zImage

```
관리자: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Wajou>cd ..

C:\Users>cd ..

C:\>cd platform-tools

C:\platform-tools>fastboot.exe flash kernel zImage
sending 'kernel' (3826 KB)...
OKAY [ 0.605s]
writing 'kernel'...
OKAY [ 0.502s]
finished. total time: 1.108s

C:\platform-tools>fastboot.exe reboot
rebooting...

finished. total time: 0.002s

C:\platform-tools>
```



매개변수를 가진 시스템 호출

❖ System Call을 호출하는 User Application 제작

- /usr/local/app/newcall_test.c 구현
- unistd.h에서 정의한 호출번호와 일치

```
#include <asm/unistd.h>
#include <errno.h>
#include <stdio.h>

int main()
{
    int a = 30;
    int i = 0;

    i = syscall(377, a);

    printf(" newcall1 : application a = %d \n", a);
    printf(" newcall1 : application function return %d \n", i);

    return 0;
}

~
~
~
```

매개변수를 가진 시스템 호출

❖ User Application 컴파일 (makefile사용)

- System call 호출 응용 프로그램의 컴파일용 Makefile을 작성
- /usr/local/app/Makefile

```
CC=arm-linux-gnueabihf-gcc

OBJS=newcall_test.o
TARGET=newcall_test

.SUFFIXES : .c .o

all:$(TARGET)

$(TARGET) : $(OBJS)
    $(CC) -o $@ $(OBJS)

clean:
    rm -f $(OBJS) $(TARGET)
```

```
root@ubuntu:/usr/local/app# make
arm-linux-gnueabihf-gcc -I/source/kernel/kernel_4412/arch/arm/include -c -o newcall_test.o newcall_test.c
arm-linux-gnueabihf-gcc -o newcall_test newcall_test.o
root@ubuntu:/usr/local/app# ls
hellocall_test.c  Makefile~      newcall_test.c
Makefile          newcall_test  newcall_test.o
root@ubuntu:/usr/local/app#
```

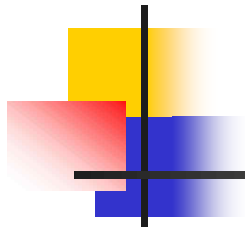
매개변수를 가진 시스템 호출

❖ User Application 전송

- 실행파일을 Tera Term의 zmodem을 이용하여 보드로 옮김
- Applicatoin File 실행 권한설정: `chmod 755 newcall_test`
- 시스템 호출 실행: `./newcall_test`

```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help
[root@linux /]# ls
bin/      lib/      mnt/      root/      tmp/
dev/      linuxrc?  newcall_test  run?      usr/
etc/      lost+found/  opt/     /sbin/      var/
home/     media/     proc/     sys/
[root@linux /]#
[root@linux /]#
[root@linux /]# chmod 777 newcall_test
[root@linux /]#
[root@linux /]#
[root@linux /]# ls
bin/      lib/      mnt/      root/      tmp/
dev/      linuxrc?  newcall_test*  run?      usr/
etc/      lost+found/  opt/     /sbin/      var/
home/     media/     proc/     sys/
[root@linux /]#
[root@linux /]#
[root@linux /]# ./newcall_test
newcall : kernel ab = 30
newcall1 : application a = 30
newcall1 : application function return 35
[root@linux /]#
```

사용자 프로그램에서 전달받은 값이 30으로 사용자 응용-커널간 데이터 전달에서 문제 발생



매개변수를 가진 시스템 호출

❖ 사용자 영역과 커널 영역 사이에서 값을 교환하는 커널 함수들

| 함수 | 설명 |
|---|---|
| put_user(x, ptr) | 커널 영역의 데이터를 사용자 영역으로 복사함. (ptr은 사용자 메모리영역의 선두주소, x는 커널변수) |
| get_user(x, ptr) | 사용자 영역의 데이터를 커널 영역으로 복사함. (ptr은 사용자 메모리영역의 선두주소, x는 커널변수) |
| copy_to_user(void __user *to, const void * from, unsigned long n) | 커널 영역의 n 크기만큼의 데이터를 사용자 영역으로 복사함. |
| copy_from_user(void * to, void __user *from, unsigned long n) | 사용자 영역의 n 크기만큼의 데이터를 커널 영역으로 복사함. |

Ref. <http://blog.naver.com/idthek?Redirect=Log&logNo=90119626977>



매개변수를 가진 시스템 호출

- ❖ 사용자 응용에서 버퍼에 대한 포인터로 system call stub(응용 측에서 시스템 호출을 위해서 호출하는 함수) 함수를 호출함.
- ❖ 호출되는 system call stub 함수에서 인자로 포인터를 넘겨 받음
- ❖ 넘겨받은 포인터를 이용하여 새로이 작성한 system call 함수 내에서 put_user(), get_user(), copy_to_user(), copy_from_user()를 호출하여 사용자-커널 영역 간의 데이터 교환이 일어나게 됨



매개변수를 가진 시스템 호출

- ❖ [kernel4412] / kernel/newcall.c 수정
- ❖ newcall.c에서 커널에서 사용자로의 데이터 전달을 위해 포인터 매개변수(int *b)와 put_user() 함수 사용
- ❖ #include <asm/uaccess.h> 필요

```
#include <linux/syscalls.h>
#include <asm/unistd.h>
#include <asm/uaccess.h>

SYSCALL_DEFINE2(newcall, int, a, int, *b)
{
    printk("\n newcall : kernel a = %d \n", a);

    a+=5; //kernel operation

    put_user(a,b); // copy a value to the user space

    return a;
}
```




매개변수를 가진 시스템 호출

- ❖ newcall_test.c에서 시스템 콜 함수에 대응하는 system stub function 수정
 - /usr/local/app/newcall_test.c 수정

```
#include <asm/unistd.h>
#include <errno.h>
#include <stdio.h>

int main()
{
    int a = 30;
    int b = 0;
    int i = 0;

    i = syscall(377,a ,&b);

    printf(" newcall1 : application a = %d b = %d \n", a ,b);
    printf(" newcall1 : application function return %d \n", i);

    return 0;
}
```



매개변수를 가진 시스템 호출

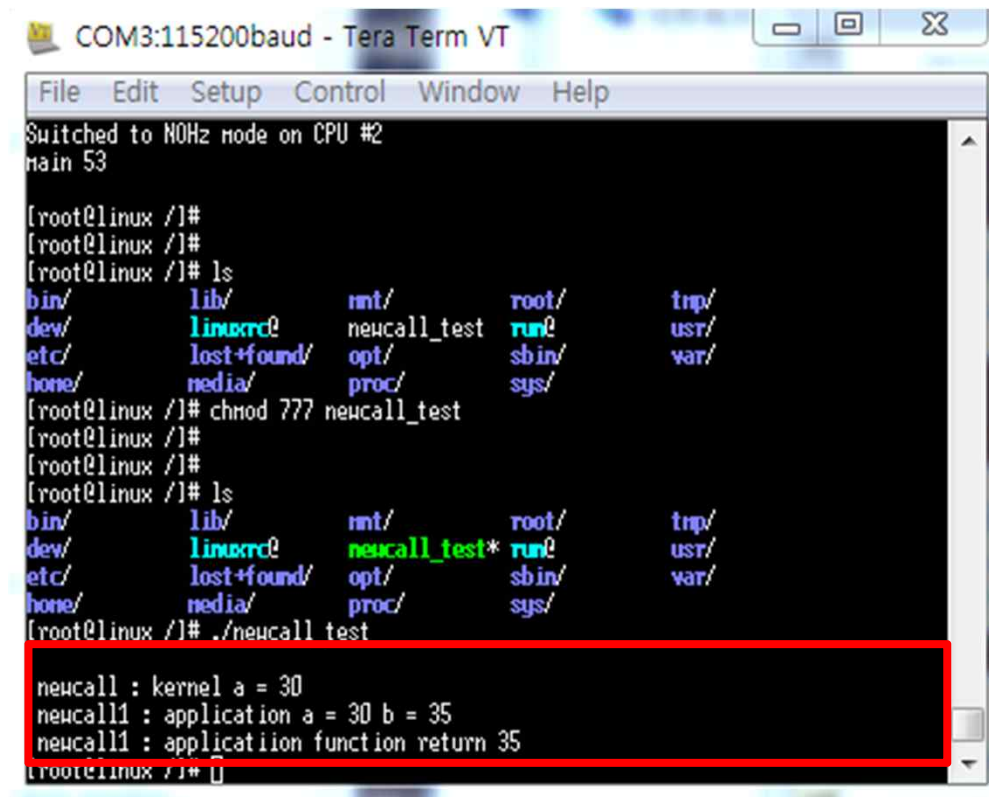
- ❖ **System call 처리 함수를 syscalls.h에 함수 선언 등록**
 - [kernel4412] / include/linux/syscalls.h 수정
 - Function prototype declaration

```
asmlinkage long sys_open_by_handle_at(int mountdirfd,  
                                     struct file_handle __user *handle,  
                                     int flags);  
asmlinkage long sys_setns(int fd, int nstype);  
asmlinkage long sys_hellocall(int a);  
asmlinkage long sys_newcall(int a, int *b);  
#endif
```

851.1 Bot

매개변수를 가진 시스템 호출

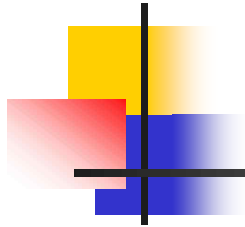
- ❖ 타겟 보드에서 실행파일 newcall_test 실행시 b의 값이 35이므로 제대로 동작함.
 - 함수 리턴 값은 35로 항상 제대로 동작함.



```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help
Switched to NOHz mode on CPU #2
main 53

[root@linux /]#
[root@linux /]#
[root@linux /]# ls
bin/      lib/      mt/       root/     tmp/
dev/      linuxrc? newcall_test run?      usr/
etc/      lost+found/ opt/     /sbin/    var/
home/     media/    proc/     sys/
[root@linux /]# chmod 777 newcall_test
[root@linux /]#
[root@linux /]#
[root@linux /]# ls
bin/      lib/      mt/       root/     tmp/
dev/      linuxrc? newcall_test* run?      usr/
etc/      lost+found/ opt/     /sbin/    var/
home/     media/    proc/     sys/
[root@linux /]# ./newcall test

newcall : kernel a = 30
newcall1 : application a = 30 b = 35
newcall1 : application function return 35
[root@linux /]#
```



매개변수를 가진 시스템 호출

❖ 질문: 전달해야 하는 데이터의 수가 많은 경우,
사용자-커널간 데이터 전달은?