IKO31204
Pemrograman Sistem
Jilid 8: System Call

Fakultas Ilmu Komputer - Universitas Indonesia

Sep 2012

topik

System call

referensi

HOWTO Add a system call to the 2.6 Kernel
http://www.csee.umbc.edu/~chettri/421/projects/hello_syscall.html

Implementing System Call on Linux 2.6 for i386
http://linuxkernel51.blogspot.com/2011/01/implementing-system-call-on-linux-26.html

System Call
http://en.wikipedia.org/wiki/System_call

referensi

Compiling Kernel Source (SysCall)
http://linuxkernel51.blogspot.com/2011/01/compiling-kernel-source.html

How to Invoke System Call in Application
http://linuxkernel51.blogspot.com/2011/02/how-to-invoke-system-call-in.html

SYSCALLS(2) Linux Programmer's Manual
http://www.kernel.org/doc/man-pages/online/pages/man2/syscalls.2.html

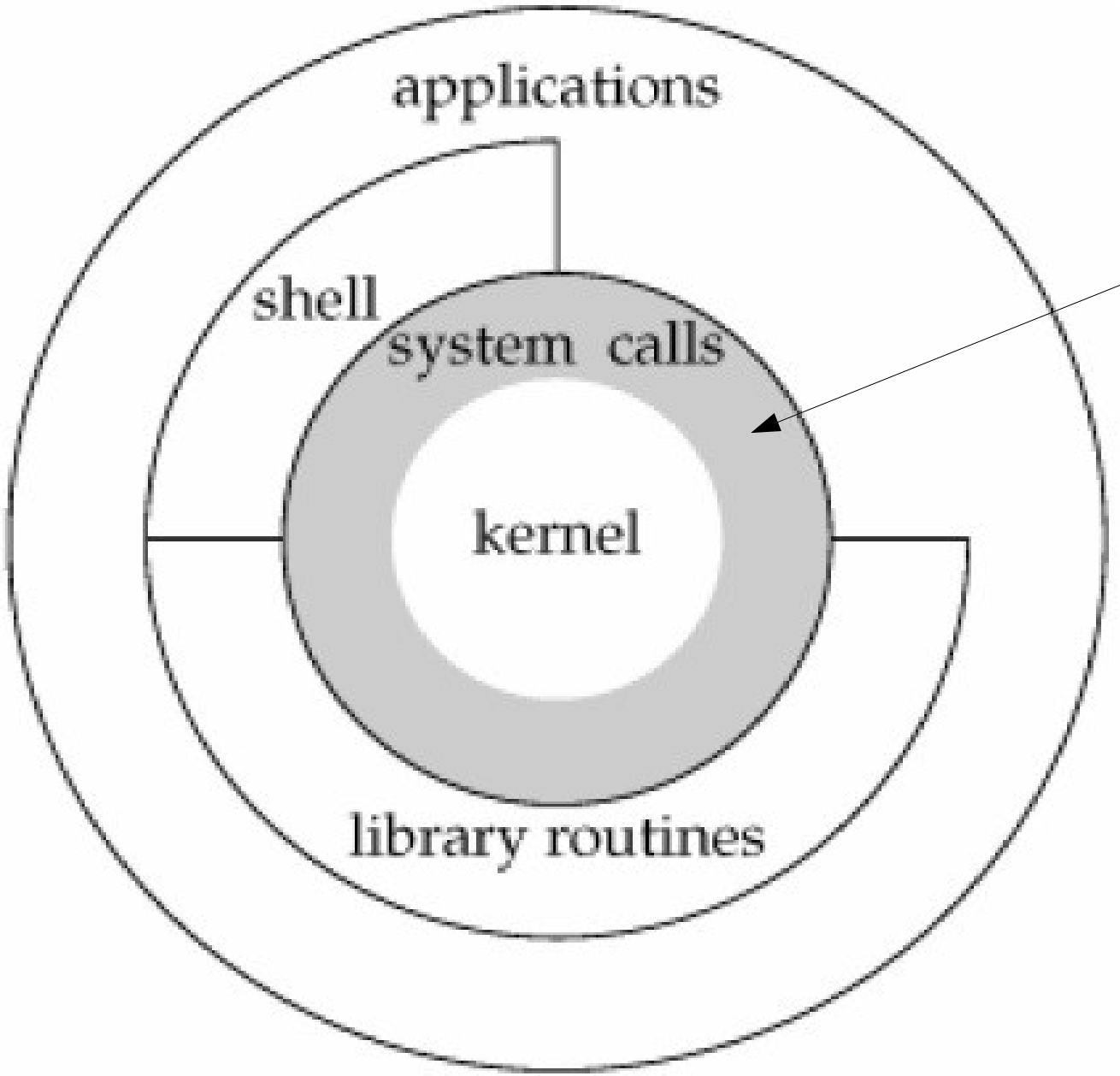
referensi

SysCalls

http://www.gnu.org/software/libc/manual/html_node/System-Calls.html

Apa itu Syscall ?

Syscall adalah adalah mekanisme yang digunakan oleh aplikasi (user level) untuk meminta layanan dari sistem operasi (kernel).



Lokasi
Syscall
beroperasi

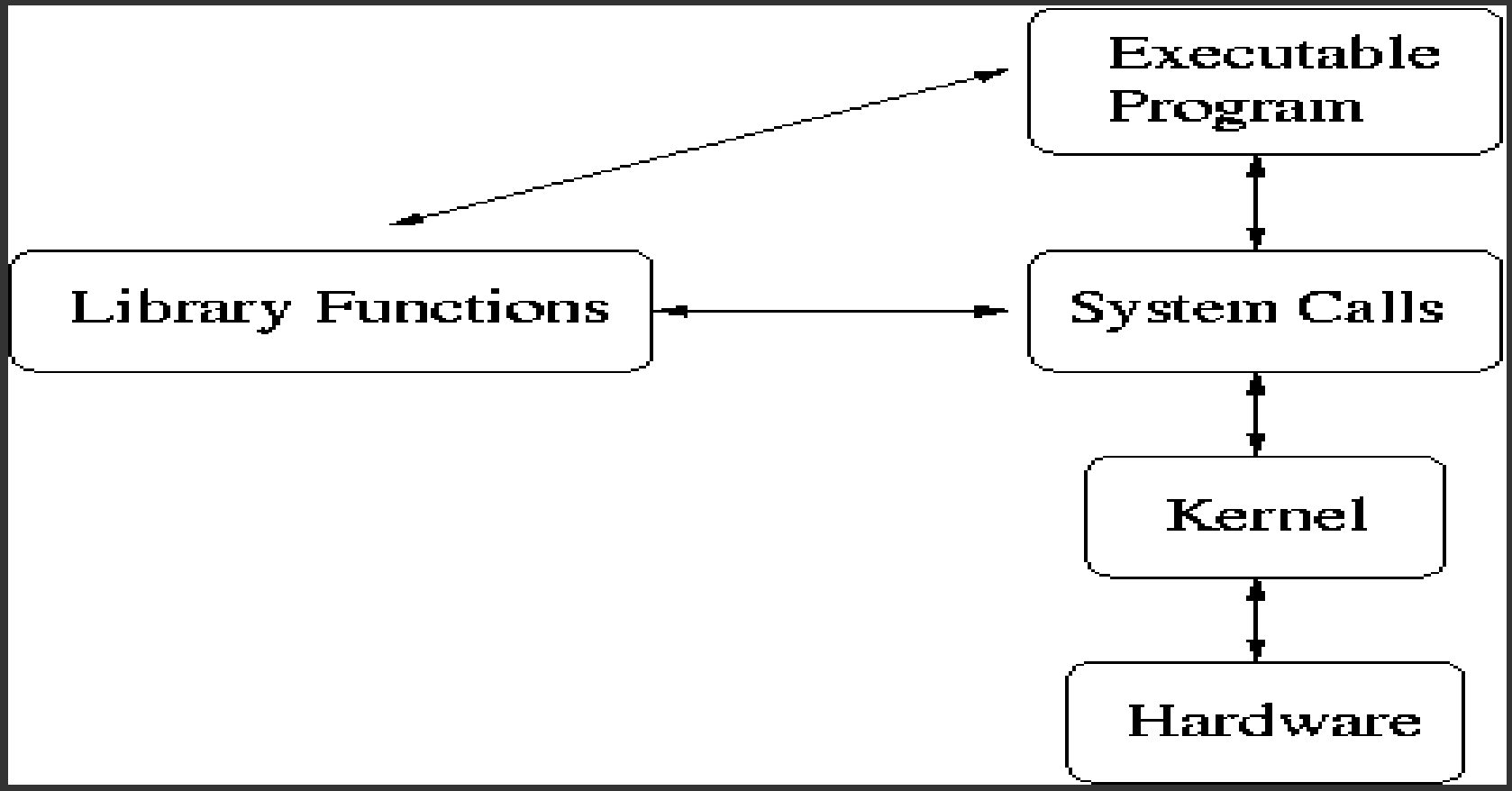
Wujud dari Syscall

Syscall berwujud sebuah method yang direpresentasikan sebagai nomor panggil unik yang harus didaftarkan di dalam berkas **syscall_table_32.**S pada source code kernel

Setiap nomor panggil dilayani oleh sebuah fungsi di dalam kernel

Memanggil Syscall

1. dapat langsung dipanggil
2. menggunakan perantara library



Cth pemanggilan langsung

int syscall(int number, arg1, arg2, arg3, dst …..);

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

int main() {
        int rc;
        rc = syscall(SYS_chmod, "tes", 0644);
        if (rc == -1)
           fprintf(stderr, "chmod failed, errno = %d\n", errno);
}
```

Memanggil via Library

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

int main(void) {
      long ID;
      /*--------------------------*/
      /* "libc" wrapped system call */
      /* SYS_getpid (Func No. is 20) */
      /*--------------------------*/
      ID = getpid();
      printf ("getpid()=%ld\n", ID);
      return(0);
}
```

Peta Syscall

Mapping nomor Syscall:
<kernel source>/arch/x86/kernel/
syscall_table_32.S (utk versi 2.6.26)

Lokasi fungsi "pelayan" bagi setiap syscall dapat bervariasi berdasarkan versi Kernelnya.
NB: Googling keyword
"*System Call Quick Reference*"

Cara membuat Syscall

1. Pilih dan berikan ID unik dan meletakkan alamat syscall baru pada table struktur data Syscall
2. Mendeklarasikan nama fungsi, struktur argumen input dan return type untuk syscall baru
3. Implementasi fungsi yang akan melayani syscall tersebu

Contoh membuat Syscall hello_world

1. ID unik & nama syscall baru (I)

vi arch/x86/kernel/syscall_table_32.S

```
ENTRY(sys_call_table)
        .long sys_restart_syscall       /* 0 - old "setup()" system call, used for restarting */
        .long sys_exit
.........
.........
        .long sys_timerfd_settime       /* 325 */
        .long sys_timerfd_gettime
```

Baris terakhir adalah **326**, maka ID unik yang bisa digunakan adalah **327.** Dengan demikian isi file syscall_table_32.S menjadi:

```
ENTRY(sys_call_table)
        .long sys_restart_syscall       /* 0 - old "setup()" system call, used for restarting */
        .long sys_exit
.........
.........
        .long sys_timerfd_settime       /* 325 */
        .long sys_timerfd_gettime
        .long sys_hello_world
```

1. ID unik & nama syscall baru (II)

 # vi include/asm-x86/unistd_32.h

```
#ifndef _ASM_I386_UNISTD_H_
#define _ASM_I386_UNISTD_H_

#define __NR_restart_syscall     0
#define __NR_exit                1
……..
#define __NR_timerfd_settime    325
#define __NR_timerfd_gettime    326
```

Kita tambahkan definisi Syscall terbaru dibawah Syscall nomor 326. Dengan demikian isi file unistd_32.h menjadi:

```
#ifndef _ASM_I386_UNISTD_H_
#define _ASM_I386_UNISTD_H_

#define __NR_restart_syscall     0
#define __NR_exit                1
……..
#define __NR_timerfd_settime    325
#define __NR_timerfd_gettime    326
#define __NR_hello_world    327
```

2. Deklarasikan fungsi

 # vi include/linux/syscalls.h

.....
.....
int kernel_execve(const char *filename, char *const argv(), char *const envp());

#endif

Kita tambahkan definisi Syscall **hello_world di atas** fungsi **kernel_execve.** Dengan demikian isi file **syscalls.h** menjadi:

.....
.....
asmlinkage long sys_hello_world(void);
int kernel_execve(const char *filename, char *const argv(), char *const envp());

#endif

3. Implemetasikan fungsi (I)

Fungsi dapat diletakkan di mana saja pada source code kernel. Cth: diletakkan pada **<source code kernel>/hello/**

 # cd /usr/src/linux-2.6.26
 # mkdir hello
 # cd hello
 # vi hello_world.c

```
/**
 * hello_world.c - The hello world system call, the best system call ever.
 **/
#include <linux/syscalls.h>

asmlinkage long sys_hello_world()
{
  printk("Hello World!\n");
  return 0;
}
```

 # vi Makefile

```
obj-y := hello_world.o
```

3. Implemetasikan fungsi (II)

Jangan lupa untuk memasukkan direktori **hello** ke dalam script otomatisasi kompilasi kernel, yaitu dalam Makefile utama kernel

 # cd /usr/src/linux-2.6.26
 # vi Makefile

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

Tambahkan direktori hello/ ke dalam script kompilasi Core kernel, menjadi:

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

Lakukan kompilasi seperti pada Modul B1

Pemanggilan langsung dgn Syscall Number 327

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

int main() {
        int rc;
        rc = syscall(327);
}
```

tanya jawab