

# System Calls

Philex Lin, 6 Mayr 2002.  
NCTU CIS Operating System lab.  
2002 Linux kernel trace seminar

# Agenda

- | What is the system call ?
- | How system calls are invoked ?
- | Flow control for system call
- | System calls
  - system\_call
  - lcall7 / lcall27

# System calls

- | Application point of view : just a function call
- | Kernel point of view : service provider

# What is a System Call?

- | Interface between user process and kernel
- | Occurs when a user process requests a service the kernel provides by calling a special function
- | System calls guard access to resources that the kernel manages
  - ◆ **I/O** (open,close,read,write and poll....)
  - ◆ **Process**(fork,execve,kill .....
  - ◆ **Time**(time, settimeofday....)
  - ◆ **Memory**(mmap,brk.....)

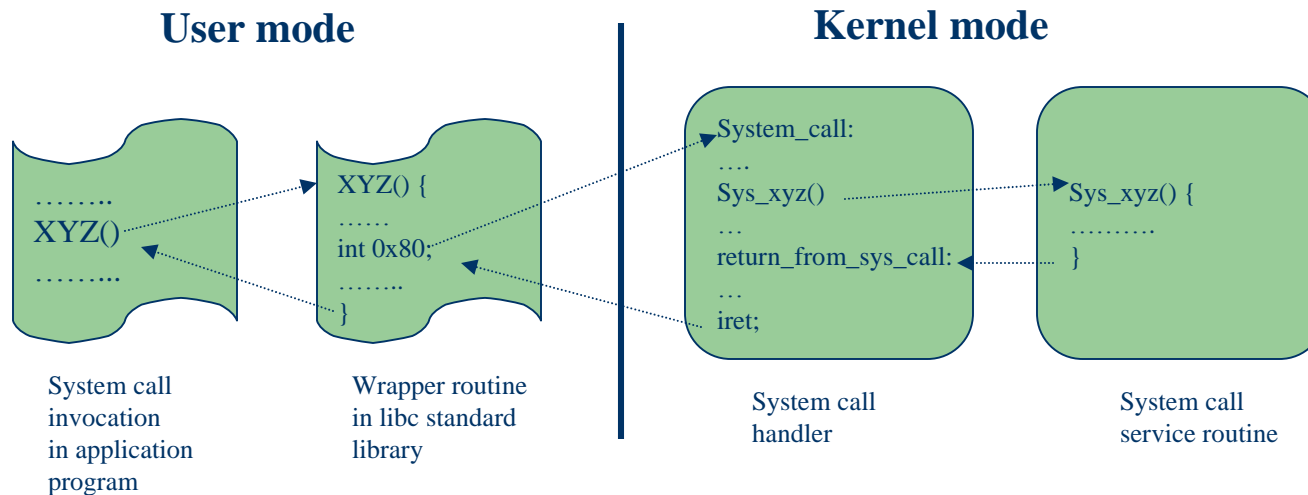
# What is a System Call?(Cont)

## I Return value

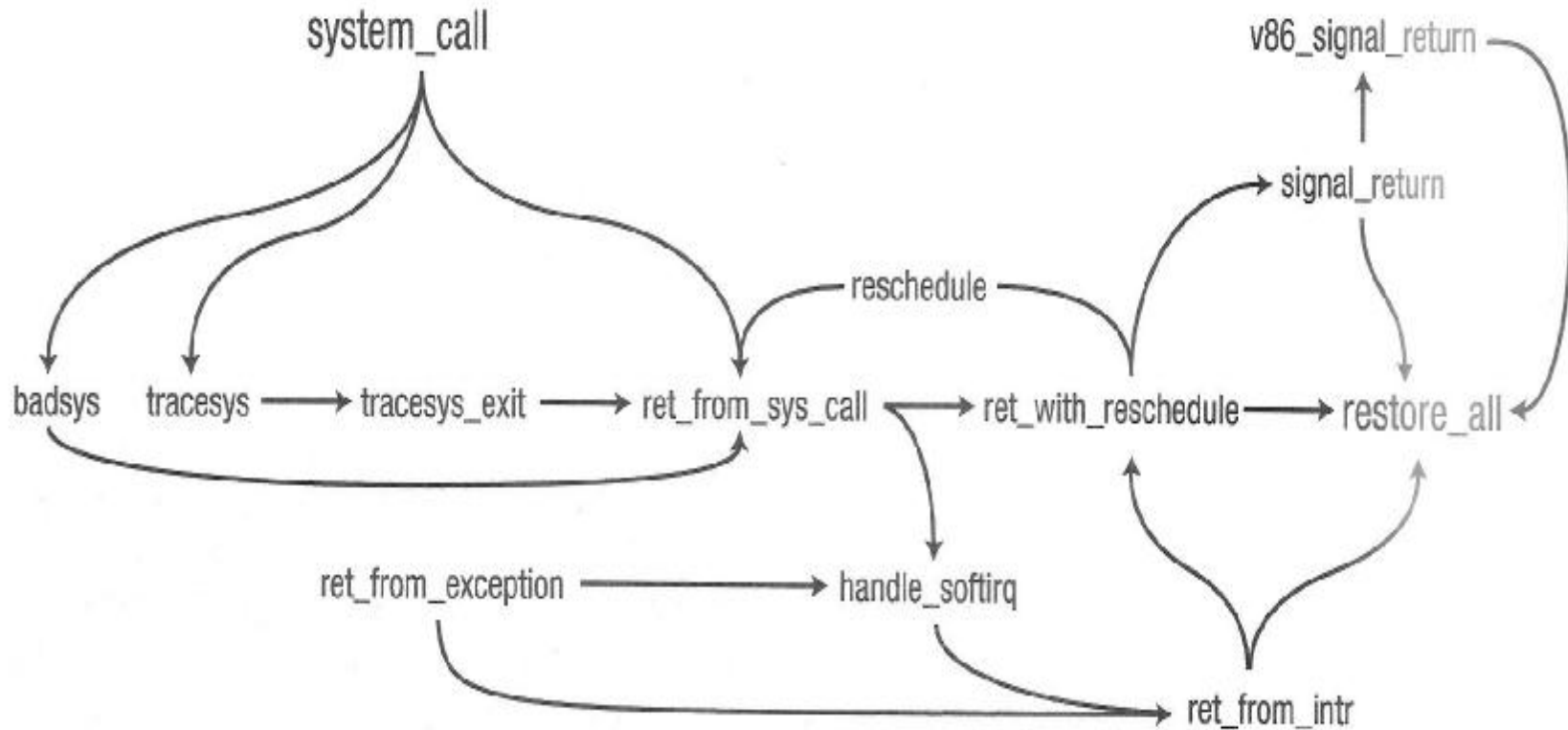
- ◆ By conventional, zero or positive to indicate success; negative return value to indicate an error
- ◆ A few system calls can return large negative values even on success(such as `lseek`)

# How System Calls Are Invoked?

- | `system_call` : entry point for all system calls
- | `lcall7` : used for iBCS2
- | `lcall27` : used for Solaris/x86 support



# Flow control for system\_call



# Stack layout

0(%esp) - %ebx  
4(%esp) - %ecx  
8(%esp) - %edx  
C(%esp) - %esi  
10(%esp) - %edi  
14(%esp) - %ebp  
18(%esp) - %eax  
1C(%esp) - %ds  
20(%esp) - %es  
24(%esp) - orig\_eax  
28(%esp) - %eip  
2C(%esp) - %cs  
30(%esp) - %eflags  
34(%esp) - %oldesp  
38(%esp) - %oldss

EBX = 0x00  
ECX = 0x04  
EDX = 0x08  
ESI = 0x0C  
EDI = 0x10  
EBP = 0x14  
EAX = 0x18  
DS = 0x1C  
ES = 0x20  
ORIG\_EAX = 0x24  
EIP = 0x28  
CS = 0x2C  
EFLAGS = 0x30  
OLDESP = 0x34  
OLDSS = 0x38

# Sys\_call\_table

```
ENTRY(sys_call_table)
    .long SYMBOL_NAME(sys_ni_syscall) /* 0 - old "setup()"
system call*/
    .long SYMBOL_NAME(sys_exit)
    .long SYMBOL_NAME(sys_fork)
    .long SYMBOL_NAME(sys_read)
    .long SYMBOL_NAME(sys_write)
    .long SYMBOL_NAME(sys_open)           /* 5 */
    .long SYMBOL_NAME(sys_close)
    .long SYMBOL_NAME(sys_waitpid)
    .long SYMBOL_NAME(sys_creat)
    .long SYMBOL_NAME(sys_link)
    .....
    .....
    .....
```

# System\_call(1)

```
ENTRY(system_call)
    pushl %eax                # save orig_eax
    SAVE_ALL
    GET_CURRENT(%ebx)
    cmpl $(NR_syscalls),%eax
    jae badsys
    testb $0x02,tsk_ptrace(%ebx) # PT_TRACESYS
    jne tracesys
    call *SYMBOL_NAME(sys_call_table)(,%eax,4)
    movl %eax,EAX(%esp)      # save the return value
ENTRY(ret_from_sys_call)
    cli                      # need_resched and signals atomic test
    cmpl $0,need_resched(%ebx)
    jne reschedule
    cmpl $0,sigpending(%ebx)
    jne signal_return
restore_all:
    RESTORE_ALL
```

```
#define SAVE_ALL
    cld;
    pushl %es;
    pushl %ds;
    pushl %eax;
    pushl %ebp;
    pushl %edi;
    pushl %esi;
    pushl %edx;
    pushl %ecx;
    pushl %ebx;
    movl $(__KERNEL_DS),%edx;
    movl %edx,%ds;
    movl %edx,%es;
```

# System\_call(2)

```
ENTRY(system_call)
    pushl %eax                # save orig_eax
    SAVE_ALL
    GET_CURRENT(%ebx)
    cmpl $(NR_syscalls),%eax
    jae badsys
    testb $0x02,tsk_ptrace(%ebx) # PT_TRACESYS
    jne tracesys
    call *SYMBOL_NAME(sys_call_table)(,%eax,4)
    movl %eax,EAX(%esp)      # save the return value
ENTRY(ret_from_sys_call)
    cli                      # need_resched and signals atomic test
    cmpl $0,need_resched(%ebx)
    jne reschedule
    cmpl $0,sigpending(%ebx)
    jne signal_return
restore_all:
    RESTORE_ALL
```

#define GET\_CURRENT(reg)  
 movl \$-8192, reg;  
 andl %esp, reg

# System\_call(3)

```
ENTRY(system_call)
    pushl %eax                # save orig_eax
    SAVE_ALL
    GET_CURRENT(%ebx)
    cmpl $(NR_syscalls),%eax
    jae badsys
    testb $0x02,tsk_ptrace(%ebx) # PT_TRACESYS
    jne tracesys
    call *SYMBOL_NAME(sys_call_table)(,%eax,4)
    movl %eax,EAX(%esp)      # save the return value
ENTRY(ret_from_sys_call)
    cli                      # need_resched and signals atomic test
    cmpl $0,need_resched(%ebx)
    jne reschedule
    cmpl $0,sigpending(%ebx)
    jne signal_return
restore_all:
    RESTORE_ALL
```

movl \$-ENOSYS,EAX(%esp)  
jmp ret\_from\_sys\_call

# System\_call(4)

```
ENTRY(system_call)
    pushl %eax                # save orig_eax
    SAVE_ALL
    GET_CURRENT(%ebx)
    cmpl $(NR_syscalls),%eax
    jae badsys
    testb $0x02,tsk_ptrace(%ebx)
    jne tracesys
    call *SYMBOL_NAME(sys_call_table)(,%eax,4)
    movl %eax,EAX(%esp)      # save the return value
ENTRY(ret_from_sys_call)
    cli                      # need_resched and signals atomic test
    cmpl $0,need_resched(%ebx)
    jne reschedule
    cmpl $0,sigpending(%ebx)
    jne signal_return
restore_all:
    RESTORE_ALL
```

tracesys:

```
    movl $-ENOSYS,EAX(%esp)
    call SYMBOL_NAME(syscall_trace)
    movl ORIG_EAX(%esp),%eax
    cmpl $(NR_syscalls),%eax
    jae tracesys_exit
    call *SYMBOL_NAME(sys_call_table)(,%eax,4)
    movl %eax,EAX(%esp)
```

tracesys\_exit:

```
    call SYMBOL_NAME(syscall_trace)
    jmp ret_from_sys_call
```

# System\_call(5)

```
ENTRY(system_call)
    pushl %eax          # save orig_eax
    SAVE_ALL
    GET_CURRENT(%ebx)
    cmpl $(NR_syscalls),%eax
    jae badsys
    testb $0x02,tsk_ptrace(%ebx) # PT_TRACESYS
    jne tracesys
    call *SYMBOL_NAME(sys_call_table)(,%eax,4)
    movl %eax,EAX(%esp) # save the return value
ENTRY(ret_from_sys_call)
    cli                # need_resched and signals atomic test
    cmpl $0,need_resched(%ebx)
    jne reschedule
    cmpl $0,sigpending(%ebx)
    jne signal_return
restore_all:
    RESTORE_ALL
```

call SYMBOL\_NAME(schedule)  
jmp ret\_from\_sys\_call

sti  
testl \$(VM\_MASK),EFLAGS(%esp)  
movl %esp,%eax  
jne v86\_signal\_return  
xorl %edx,%edx  
call SYMBOL\_NAME(do\_signal)  
jmp restore\_all

# System\_call(6)

```
ENTRY(system_call)
    pushl %eax                # save orig_eax
    SAVE_ALL
    GET_CURRENT(%ebx)
    cmpl $(NR_syscalls),%eax
    jae badsys
    testb $0x02,tsk_ptrace(%ebx) # PT_TRACESYS
    jne tracesys
    call *SYMBOL_NAME(sys_call_table)(,%eax,4)
    movl %eax,EAX(%esp)      # save the return value
ENTRY(ret_from_sys_call)
    cli                      # need_resched and signals atomic test
    cmpl $0,need_resched(%ebx)
    jne reschedule
    cmpl $0,sigpending(%ebx)
    jne signal_return
restore_all:
    RESTORE_ALL
```

```
#define RESTORE_ALL
    popl %ebx;
    popl %ecx;
    popl %edx;
    popl %esi;
    popl %edi;
    popl %ebp;
    popl %eax;
1:   popl %ds;
2:   popl %es;
    addl $4,%esp;
3:   iret;
```

# Icall7

## ENTRY(Icall7)

```
pushfl    # We get a different stack layout with call
pushl %eax # gates, which has to be cleaned up later..
SAVE_ALL    # due to call gates,
movl EIP(%esp),%eax    # this is eflags, not eip..
movl CS(%esp),%edx    # this is eip..
movl EFLAGS(%esp),%ecx # and this is cs..
movl %eax,EFLAGS(%esp) #
movl %edx,EIP(%esp)    # Now we move them to their
movl %ecx,CS(%esp)    # "normal" places
movl %esp,%ebx
pushl %ebx
andl $-8192,%ebx    # GET_CURRENT
movl exec_domain(%ebx),%edx # Get the execution domain
movl 4(%edx),%edx    # Get Icall7 handler for domain
pushl $0x7
call *%edx
addl $4, %esp
popl %eax
jmp ret_from_sys_call
```

# Icall27

## ENTRY(Icall27)

```
pushfl    # We get a different stack layout with call
pushl %eax # gates, which has to be cleaned up later..
SAVE_ALL  # due to call gates,
movl EIP(%esp),%eax # this is eflags, not eip..
movl CS(%esp),%edx  # this is eip..
movl EFLAGS(%esp),%ecx # and this is cs..
movl %eax,EFLAGS(%esp) #
movl %edx,EIP(%esp)   # Now we move them to their
movl %ecx,CS(%esp)   # "normal" places
movl %esp,%ebx
pushl %ebx
andl $-8192,%ebx     # GET_CURRENT
movl exec_domain(%ebx),%edx # Get the execution domain
movl 4(%edx),%edx    # Get Icall7 handler for domain
pushl $0x27
call *%edx
addl $4, %esp
popl %eax
jmp ret_from_sys_call
```

## Reference

- | **Linux Core Kernel Commentary**  
second edition
- | **Understanding the LINUX KERNEL**  
O'reilly
- | **Cross-Referencing Linux**  
– <http://lxr.linux.no/>