man7.org > Linux > man-pages Linux/UNIX system programming training

NAME | SYNOPSIS | DESCRIPTION | RETURN VALUE |
NOTES | EXAMPLE | SEE ALSO | COLOPHON

Search online pages

SYSCALL(2) Linux Programmer's Manual SYSC

NAME

top

syscall - indirect system call

SYNOPSIS

top

```
#define _GNU_SOURCE         /* See feature_test_macros(7) */
#include <unistd.h>
#include <sys/syscall.h>   /* For SYS_xxx definitions */

long syscall(long number, ...);
```

DESCRIPTION

top

syscall() is a small library function that invokes the system ca whose assembly language interface has the specified *number* with specified arguments. Employing **syscall**() is useful, for exampl when invoking a system call that has no wrapper function in the library.

syscall() saves CPU registers before making the system call, re the registers upon return from the system call, and stores any code returned by the system call in errno(3) if an error occurs

Symbolic constants for system call numbers can be found in the file *<sys/syscall.h>*.

RETURN VALUE

top

The return value is defined by the system call being invoked. general, a 0 return value indicates success. A -1 return value indicates an error, and an error code is stored in *errno*.

NOTES

top

syscall() first appeared in 4BSD.

Architecture-specific requirements

Each architecture ABI has its own requirements on how system ca arguments are passed to the kernel. For system calls that have glibc wrapper (e.g., most system calls), glibc handles the deta copying arguments to the right registers in a manner suitable f architecture. However, when using **syscall**() to make a system c the caller might need to handle architecture-dependent details; requirement is most commonly encountered on certain 32-bit architectures.

For example, on the ARM architecture Embedded ABI (EABI), a 64- value (e.g., *long long*) must be aligned to an even register pai Thus, using **syscall**() instead of the wrapper provided by glibc, **readahead**() system call would be invoked as follows on the ARM architecture with the EABI:

```
    syscall(SYS_readahead, fd, 0,
            (unsigned int) (offset >> 32),
            (unsigned int) (offset & 0xFFFFFFFF),
            count);
```

Since the offset argument is 64 bits, and the first argument (*f* passed in *r0*, the caller must manually split and align the 64-b value so that it is passed in the *r2*/*r3* register pair. That mea inserting a dummy value into *r1* (the second argument of 0).

Similar issues can occur on MIPS with the O32 ABI, on PowerPC w the 32-bit ABI, and on Xtensa.

The affected system calls are fadvise64_64(2), ftruncate64(2), posix_fadvise(2), pread64(2), pwrite64(2), readahead(2), sync_file_range(2), and truncate64(2).

Architecture calling conventions

Every architecture has its own way of invoking and passing argu to the kernel. The details for various architectures are liste the two tables below.

The first table lists the instruction used to transition to ker mode, (which might not be the fastest or best way to transition the kernel, so you might have to refer to vdso(7)), the registe to indicate the system call number, and the register used to re the system call result.

arch/ABI	instruction	syscall #	retval	Notes
arm/OABI	swi NR	-	a1	NR is syscal
arm/EABI	swi 0x0	r7	r0	
blackfin	excpt 0x0	P0	R0	
i386	int $0x80	eax	eax	
ia64	break 0x100000	r15	r10/r8	bool error/ errno value
parisc	ble 0x100(%sr2, %r0)	r20	r28	
s390	svc 0	r1	r2	See below
s390x	svc 0	r1	r2	See below
sparc/32	t 0x10	g1	o0	

sparc/64	t 0x6d	g1	o0
x86_64	syscall	rax	rax

For s390 and s390x, NR (the system call number) may be passed directly with "svc NR" if it is less than 256.

The second table shows the registers used to pass the system ca arguments.

arch/ABI	arg1	arg2	arg3	arg4	arg5	arg6	arg7
arm/OABI	a1	a2	a3	a4	v1	v2	v3
arm/EABI	r0	r1	r2	r3	r4	r5	r6
blackfin	R0	R1	R2	R3	R4	R5	-
i386	ebx	ecx	edx	esi	edi	ebp	-
ia64	out0	out1	out2	out3	out4	out5	-
parisc	r26	r25	r24	r23	r22	r21	-
s390	r2	r3	r4	r5	r6	r7	-
s390x	r2	r3	r4	r5	r6	r7	-
sparc/32	o0	o1	o2	o3	o4	o5	-
sparc/64	o0	o1	o2	o3	o4	o5	-
x86_64	rdi	rsi	rdx	r10	r8	r9	-

Note that these tables don't cover the entire calling conventio architectures may indiscriminately clobber other registers not here.

EXAMPLE top

```
#define _GNU_SOURCE
#include <unistd.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <signal.h>

int
main(int argc, char *argv[])
{
    pid_t tid;

    tid = syscall(SYS_gettid);
    tid = syscall(SYS_tgkill, getpid(), tid, SIGHUP);
}
```

SEE ALSO top

_syscall(2), intro(2), syscalls(2), vdso(7)

COLOPHON top

```
       This page is part of release 3.70 of the Linux man-pages projec
       description of the project, information about reporting bugs, a
       latest version of this page, can be found at
       http://www.kernel.org/doc/man-pages/.
```

Linux **2014-05-10** **SYSC**

HTML rendering created 2014-07-09 by Michael Kerrisk, author of *The Linux Programming Interface*, maintainer of the Linux *man-pages* project.