NAME

syscalls − Linux system calls

SYNOPSIS

Linux system calls.

DESCRIPTION

The system call is the fundamental interface between an application and the Linux kernel.

System calls and library wrapper functions

System calls are generally not invoked directly, but rather via wrapper functions in glibc (or perhaps some other library). For details of direct invocation of a system call, see **intro**(2). Often, but not always, the name of the wrapper function is the same as the name of the system call that it invokes. For example, glibc contains a function **chdir**() which invokes the underlying "chdir" system call.

Often the glibc wrapper function is quite thin, doing little work other than copying arguments to the right registers before invoking the system call, and then setting *errno* appropriately after the system call has returned. (These are the same steps that are performed by**syscall**(2), which can be used to in voke system calls for which no wrapper function is provided.) Note: system calls indicate a failure by returning a negative error number to the caller on architectures without a separate error register/flag, as noted in **syscall**(2); when this happens, the wrapper function negates the returned error number (to make it positive), copies it to *errno*, and returns −1 to the caller of the wrapper.

Sometimes, however, the wrapper function does some extra work before invoking the system call. For example, nowadays there are (for reasons described below) two related system calls, **truncate**(2) and **truncate64**(2), and the glibc **truncate**() wrapper function checks which of those system calls are provided by the kernel and determines which should be employed.

System call list

Below is a list of the Linux system calls. In the list, the *Kernel* column indicates the kernel version for those system calls that were new in Linux 2.2, or have appeared since that kernel version. Note the following points:

- Where no kernel version is indicated, the system call appeared in Linux 1.0 or earlier.
- Where a system call is marked "1.2" this means the system call probably appeared in a Linux 1.1.x kernel version, and first appeared in a stable kernel with 1.2. (Development of the 1.2 kernel was initiated from a branch of kernel 1.0.6 via the 1.1.x unstable kernel series.)
- Where a system call is marked "2.0" this means the system call probably appeared in a Linux 1.3.x kernel version, and first appeared in a stable kernel with Linux 2.0. (Development of the Linux 2.0 kernel was initiated from a branch of Linux 1.2.x, somewhere around Linux 1.2.10, via the Linux 1.3.x unstable kernel series.)
- Where a system call is marked "2.2" this means the system call probably appeared in a Linux 2.1.x kernel version, and first appeared in a stable kernel with Linux 2.2.0. (Development of the Linux 2.2 kernel was initiated from a branch of kernel Linux 2.0.21 via the Linux 2.1.x unstable kernel series.)
- Where a system call is marked "2.4" this means the system call probably appeared in a Linux 2.3.x kernel version, and first appeared in a stable kernel with Linux 2.4.0. (Development of the Linux 2.4 kernel was initiated from a branch of Linux 2.2.8 via the Linux 2.3.x unstable kernel series.)
- Where a system call is marked "2.6" this means the system call probably appeared in a Linux 2.5.x kernel version, and first appeared in a stable kernel with Linux 2.6.0. (Development of Linux 2.6 was initiated from a branch of Linux 2.4.15 via the Linux 2.5.x unstable kernel series.)
- Starting with Linux 2.6.0, the development model changed, and new system calls may appear in each Linux 2.6.x release. In this case, the exact version number where the system call appeared is shown. This convention continues with the Linux 3.x kernel series, which followed on from Linux 2.6.39; and the Linux 4.x kernel series, which followed on from Linux 3.19; and the Linux 5.x kernel series, which followed on from Linux 4.20.

- In some cases, a system call was added to a stable kernel series after it branched from the previous stable kernel series, and then backported into the earlier stable kernel series. For example some system calls that appeared in Linux 2.6.x were also backported into a Linux 2.4.x release after Linux 2.4.15. When this is so, the version where the system call appeared in both of the major kernel series is listed.

The list of system calls that are available as at Linux 5.14 (or in a few cases only on older kernels) is as follows:

System call	**Kernel**	**Notes**
_llseek(2)	1.2	
_newselect(2)	2.0	
_sysctl(2)	2.0	Removed in 5.5
accept(2)	2.0	See notes on **socketcall**(2)
accept4(2)	2.6.28	
access(2)	1.0	
acct(2)	1.0	
add_key(2)	2.6.10	
adjtimex(2)	1.0	
alarm(2)	1.0	
alloc_hugepages(2)	2.5.36	Removed in 2.5.44
arc_gettls(2)	3.9	ARC only
arc_settls(2)	3.9	ARC only
arc_usr_cmpxchg(2)	4.9	ARC only
arch_prctl(2)	2.6	x86_64, x86 since 4.12
atomic_barrier(2)	2.6.34	m68k only
atomic_cmpxchg_32(2)	2.6.34	m68k only
bdflush(2)	1.2	Deprecated (does nothing) since 2.6
bind(2)	2.0	See notes on **socketcall**(2)
bpf(2)	3.18	
brk(2)	1.0	
breakpoint(2)	2.2	ARM OABI only, defined with **__ARM_NR** prefix
cacheflush(2)	1.2	Not on x86
capget(2)	2.2	
capset(2)	2.2	
chdir(2)	1.0	
chmod(2)	1.0	
chown(2)	2.2	See **chown**(2) for version details
chown32(2)	2.4	
chroot(2)	1.0	
clock_adjtime(2)	2.6.39	
clock_getres(2)	2.6	
clock_gettime(2)	2.6	
clock_nanosleep(2)	2.6	
clock_settime(2)	2.6	
clone2(2)	2.4	IA-64 only
clone(2)	1.0	
clone3(2)	5.3	
close(2)	1.0	
close_range(2)	5.9	
connect(2)	2.0	See notes on **socketcall**(2)
copy_file_range(2)	4.5	

creat(2)	1.0	
create_module(2)	1.0	Removed in 2.6
delete_module(2)	1.0	
dup(2)	1.0	
dup2(2)	1.0	
dup3(2)	2.6.27	
epoll_create(2)	2.6	
epoll_create1(2)	2.6.27	
epoll_ctl(2)	2.6	
epoll_pwait(2)	2.6.19	
epoll_pwait2(2)	5.11	
epoll_wait(2)	2.6	
eventfd(2)	2.6.22	
eventfd2(2)	2.6.27	
execv(2)	2.0	SPARC/SPARC64 only, for compatibility with SunOS
execve(2)	1.0	
execveat(2)	3.19	
exit(2)	1.0	
exit_group(2)	2.6	
faccessat(2)	2.6.16	
faccessat2(2)	5.8	
fadvise64(2)	2.6	
fadvise64_64(2)	2.6	
fallocate(2)	2.6.23	
fanotify_init(2)	2.6.37	
fanotify_mark(2)	2.6.37	
fchdir(2)	1.0	
fchmod(2)	1.0	
fchmodat(2)	2.6.16	
fchown(2)	1.0	
fchown32(2)	2.4	
fchownat(2)	2.6.16	
fcntl(2)	1.0	
fcntl64(2)	2.4	
fdatasync(2)	2.0	
fgetxattr(2)	2.6; 2.4.18	
finit_module(2)	3.8	
flistxattr(2)	2.6; 2.4.18	
flock(2)	2.0	
fork(2)	1.0	
free_hugepages(2)	2.5.36	Removed in 2.5.44
fremovexattr(2)	2.6; 2.4.18	
fsconfig(2)	5.2	
fsetxattr(2)	2.6; 2.4.18	
fsmount(2)	5.2	
fsopen(2)	5.2	
fspick(2)	5.2	
fstat(2)	1.0	
fstat64(2)	2.4	
fstatat64(2)	2.6.16	
fstatfs(2)	1.0	
fstatfs64(2)	2.6	

fsync(2)	1.0	
ftruncate(2)	1.0	
ftruncate64(2)	2.4	
futex(2)	2.6	
futimesat(2)	2.6.16	
get_kernel_syms(2)	1.0	Removed in 2.6
get_mempolicy(2)	2.6.6	
get_robust_list(2)	2.6.17	
get_thread_area(2)	2.6	
get_tls(2)	4.15	ARM OABI only, has **__ARM_NR** prefix
getcpu(2)	2.6.19	
getcwd(2)	2.2	
getdents(2)	2.0	
getdents64(2)	2.4	
getdomainname(2)	2.2	SPARC, SPARC64; available as **osf_getdomainname**(2) on Alpha since Linux 2.0
getdtablesize(2)	2.0	SPARC (removed in 2.6.26), available on Alpha as **osf_getdtablesize**(2)
getegid(2)	1.0	
getegid32(2)	2.4	
geteuid(2)	1.0	
geteuid32(2)	2.4	
getgid(2)	1.0	
getgid32(2)	2.4	
getgroups(2)	1.0	
getgroups32(2)	2.4	
gethostname(2)	2.0	Alpha, was available on SPARC up to Linux 2.6.26
getitimer(2)	1.0	
getpeername(2)	2.0	See notes on **socketcall**(2)
getpagesize(2)	2.0	Not on x86
getpgid(2)	1.0	
getpgrp(2)	1.0	
getpid(2)	1.0	
getppid(2)	1.0	
getpriority(2)	1.0	
getrandom(2)	3.17	
getresgid(2)	2.2	
getresgid32(2)	2.4	
getresuid(2)	2.2	
getresuid32(2)	2.4	
getrlimit(2)	1.0	
getrusage(2)	1.0	
getsid(2)	2.0	
getsockname(2)	2.0	See notes on **socketcall**(2)
getsockopt(2)	2.0	See notes on **socketcall**(2)
gettid(2)	2.4.11	
gettimeofday(2)	1.0	
getuid(2)	1.0	

getuid32(2)	2.4	
getunwind(2)	2.4.8	IA-64 only; deprecated
getxattr(2)	2.6; 2.4.18	
getxgid(2)	2.0	Alpha only; see NOTES
getxpid(2)	2.0	Alpha only; see NOTES
getxuid(2)	2.0	Alpha only; see NOTES
init_module(2)	1.0	
inotify_add_watch(2)	2.6.13	
inotify_init(2)	2.6.13	
inotify_init1(2)	2.6.27	
inotify_rm_watch(2)	2.6.13	
io_cancel(2)	2.6	
io_destroy(2)	2.6	
io_getevents(2)	2.6	
io_pgetevents(2)	4.18	
io_setup(2)	2.6	
io_submit(2)	2.6	
io_uring_enter(2)	5.1	
io_uring_register(2)	5.1	
io_uring_setup(2)	5.1	
ioctl(2)	1.0	
ioperm(2)	1.0	
iopl(2)	1.0	
ioprio_get(2)	2.6.13	
ioprio_set(2)	2.6.13	
ipc(2)	1.0	
kcmp(2)	3.5	
kern_features(2)	3.7	SPARC64 only
kexec_file_load(2)	3.17	
kexec_load(2)	2.6.13	
keyctl(2)	2.6.10	
kill(2)	1.0	
landlock_add_rule(2)	5.13	
landlock_create_ruleset(2)	5.13	
landlock_restrict_self(2)	5.13	
lchown(2)	1.0	See **chown**(2) for version details
lchown32(2)	2.4	
lgetxattr(2)	2.6; 2.4.18	
link(2)	1.0	
linkat(2)	2.6.16	
listen(2)	2.0	See notes on **socketcall**(2)
listxattr(2)	2.6; 2.4.18	
llistxattr(2)	2.6; 2.4.18	
lookup_dcookie(2)	2.6	
lremovexattr(2)	2.6; 2.4.18	
lseek(2)	1.0	
lsetxattr(2)	2.6; 2.4.18	
lstat(2)	1.0	
lstat64(2)	2.4	
madvise(2)	2.4	
mbind(2)	2.6.6	
memory_ordering(2)	2.2	SPARC64 only

membarrier(2)	3.17	
memfd_create(2)	3.17	
memfd_secret(2)	5.14	
migrate_pages(2)	2.6.16	
mincore(2)	2.4	
mkdir(2)	1.0	
mkdirat(2)	2.6.16	
mknod(2)	1.0	
mknodat(2)	2.6.16	
mlock(2)	2.0	
mlock2(2)	4.4	
mlockall(2)	2.0	
mmap(2)	1.0	
mmap2(2)	2.4	
modify_ldt(2)	1.0	
mount(2)	1.0	
move_mount(2)	5.2	
move_pages(2)	2.6.18	
mprotect(2)	1.0	
mq_getsetattr(2)	2.6.6	
mq_notify(2)	2.6.6	
mq_open(2)	2.6.6	
mq_timedreceive(2)	2.6.6	
mq_timedsend(2)	2.6.6	
mq_unlink(2)	2.6.6	
mremap(2)	2.0	
msgctl(2)	2.0	See notes on **ipc**(2)
msgget(2)	2.0	See notes on **ipc**(2)
msgrcv(2)	2.0	See notes on **ipc**(2)
msgsnd(2)	2.0	See notes on **ipc**(2)
msync(2)	2.0	
munlock(2)	2.0	
munlockall(2)	2.0	
munmap(2)	1.0	
name_to_handle_at(2)	2.6.39	
nanosleep(2)	2.0	
newfstatat(2)	2.6.16	See **stat**(2)
nfsservctl(2)	2.2	Removed in 3.1
nice(2)	1.0	
old_adjtimex(2)	2.0	Alpha only; see NOTES
old_getrlimit(2)	2.4	Old variant of **getrlimit**(2) that used a different value for **RLIM_INFINITY**
oldfstat(2)	1.0	
oldlstat(2)	1.0	
oldolduname(2)	1.0	
oldstat(2)	1.0	
oldumount(2)	2.4.116	Name of the old **umount**(2) syscall on Alpha
olduname(2)	1.0	
open(2)	1.0	
open_by_handle_at(2)	2.6.39	
open_tree(2)	5.2	

openat(2)	2.6.16	
openat2(2)	5.6	
or1k_atomic(2)	3.1	OpenRISC 1000 only
pause(2)	1.0	
pciconfig_iobase(2)	2.2.15; 2.4	Not on x86
pciconfig_read(2)	2.0.26; 2.2	Not on x86
pciconfig_write(2)	2.0.26; 2.2	Not on x86
perf_event_open(2)	2.6.31	Was perf_counter_open() in 2.6.31; renamed in 2.6.32
personality(2)	1.2	
perfctr(2)	2.2	SPARC only; removed in 2.6.34
perfmonctl(2)	2.4	IA-64 only; removed in 5.10
pidfd_getfd(2)	5.6	
pidfd_send_signal(2)	5.1	
pidfd_open(2)	5.3	
pipe(2)	1.0	
pipe2(2)	2.6.27	
pivot_root(2)	2.4	
pkey_alloc(2)	4.8	
pkey_free(2)	4.8	
pkey_mprotect(2)	4.8	
poll(2)	2.0.36; 2.2	
ppoll(2)	2.6.16	
prctl(2)	2.2	
pread64(2)		Added as "pread" in 2.2; renamed "pread64" in 2.6
preadv(2)	2.6.30	
preadv2(2)	4.6	
prlimit64(2)	2.6.36	
process_madvise(2)	5.10	
process_vm_readv(2)	3.2	
process_vm_writev(2)	3.2	
pselect6(2)	2.6.16	
ptrace(2)	1.0	
pwrite64(2)		Added as "pwrite" in 2.2; renamed "pwrite64" in 2.6
pwritev(2)	2.6.30	
pwritev2(2)	4.6	
query_module(2)	2.2	Removed in 2.6
quotactl(2)	1.0	
quotactl_fd(2)	5.14	
read(2)	1.0	
readahead(2)	2.4.13	
readdir(2)	1.0	
readlink(2)	1.0	
readlinkat(2)	2.6.16	
readv(2)	2.0	
reboot(2)	1.0	
recv(2)	2.0	See notes on **socketcall**(2)
recvfrom(2)	2.0	See notes on **socketcall**(2)
recvmsg(2)	2.0	See notes on **socketcall**(2)
recvmmsg(2)	2.6.33	

remap_file_pages(2)	2.6	Deprecated since 3.16
removexattr(2)	2.6; 2.4.18	
rename(2)	1.0	
renameat(2)	2.6.16	
renameat2(2)	3.15	
request_key(2)	2.6.10	
restart_syscall(2)	2.6	
riscv_flush_icache(2)	4.15	RISC-V only
rmdir(2)	1.0	
rseq(2)	4.18	
rt_sigaction(2)	2.2	
rt_sigpending(2)	2.2	
rt_sigprocmask(2)	2.2	
rt_sigqueueinfo(2)	2.2	
rt_sigreturn(2)	2.2	
rt_sigsuspend(2)	2.2	
rt_sigtimedwait(2)	2.2	
rt_tgsigqueueinfo(2)	2.6.31	
rtas(2)	2.6.2	PowerPC/PowerPC64 only
s390_runtime_instr(2)	3.7	s390 only
s390_pci_mmio_read(2)	3.19	s390 only
s390_pci_mmio_write(2)	3.19	s390 only
s390_sthyi(2)	4.15	s390 only
s390_guarded_storage(2)	4.12	s390 only
sched_get_affinity(2)	2.6	Name of **sched_getaffinity**(2) on SPARC and SPARC64
sched_get_priority_max(2)	2.0	
sched_get_priority_min(2)	2.0	
sched_getaffinity(2)	2.6	
sched_getattr(2)	3.14	
sched_getparam(2)	2.0	
sched_getscheduler(2)	2.0	
sched_rr_get_interval(2)	2.0	
sched_set_affinity(2)	2.6	Name of **sched_setaffinity**(2) on SPARC and SPARC64
sched_setaffinity(2)	2.6	
sched_setattr(2)	3.14	
sched_setparam(2)	2.0	
sched_setscheduler(2)	2.0	
sched_yield(2)	2.0	
seccomp(2)	3.17	
select(2)	1.0	
semctl(2)	2.0	See notes on **ipc**(2)
semget(2)	2.0	See notes on **ipc**(2)
semop(2)	2.0	See notes on **ipc**(2)
semtimedop(2)	2.6; 2.4.22	
send(2)	2.0	See notes on **socketcall**(2)
sendfile(2)	2.2	
sendfile64(2)	2.6; 2.4.19	
sendmmsg(2)	3.0	
sendmsg(2)	2.0	See notes on **socketcall**(2)

sendto(2)	2.0	See notes on **socketcall**(2)
set_mempolicy(2)	2.6.6	
set_robust_list(2)	2.6.17	
set_thread_area(2)	2.6	
set_tid_address(2)	2.6	
set_tls(2)	2.6.11	ARM OABI/EABI only (constant has **__ARM_NR** prefix)
setdomainname(2)	1.0	
setfsgid(2)	1.2	
setfsgid32(2)	2.4	
setfsuid(2)	1.2	
setfsuid32(2)	2.4	
setgid(2)	1.0	
setgid32(2)	2.4	
setgroups(2)	1.0	
setgroups32(2)	2.4	
sethae(2)	2.0	Alpha only; see NOTES
sethostname(2)	1.0	
setitimer(2)	1.0	
setns(2)	3.0	
setpgid(2)	1.0	
setpgrp(2)	2.0	Alternative name for **setpgid**(2) on Alpha
setpriority(2)	1.0	
setregid(2)	1.0	
setregid32(2)	2.4	
setresgid(2)	2.2	
setresgid32(2)	2.4	
setresuid(2)	2.2	
setresuid32(2)	2.4	
setreuid(2)	1.0	
setreuid32(2)	2.4	
setrlimit(2)	1.0	
setsid(2)	1.0	
setsockopt(2)	2.0	See notes on **socketcall**(2)
settimeofday(2)	1.0	
setuid(2)	1.0	
setuid32(2)	2.4	
setup(2)	1.0	Removed in 2.2
setxattr(2)	2.6; 2.4.18	
sgetmask(2)	1.0	
shmat(2)	2.0	See notes on **ipc**(2)
shmctl(2)	2.0	See notes on **ipc**(2)
shmdt(2)	2.0	See notes on **ipc**(2)
shmget(2)	2.0	See notes on **ipc**(2)
shutdown(2)	2.0	See notes on **socketcall**(2)
sigaction(2)	1.0	
sigaltstack(2)	2.2	
signal(2)	1.0	
signalfd(2)	2.6.22	
signalfd4(2)	2.6.27	
sigpending(2)	1.0	

sigprocmask(2)	1.0	
sigreturn(2)	1.0	
sigsuspend(2)	1.0	
socket(2)	2.0	See notes on **socketcall**(2)
socketcall(2)	1.0	
socketpair(2)	2.0	See notes on **socketcall**(2)
spill(2)	2.6.13	Xtensa only
splice(2)	2.6.17	
spu_create(2)	2.6.16	PowerPC/PowerPC64 only
spu_run(2)	2.6.16	PowerPC/PowerPC64 only
ssetmask(2)	1.0	
stat(2)	1.0	
stat64(2)	2.4	
statfs(2)	1.0	
statfs64(2)	2.6	
statx(2)	4.11	
stime(2)	1.0	
subpage_prot(2)	2.6.25	PowerPC/PowerPC64 only
swapcontext(2)	2.6.3	PowerPC/PowerPC64 only
switch_endian(2)	4.1	PowerPC64 only
swapoff(2)	1.0	
swapon(2)	1.0	
symlink(2)	1.0	
symlinkat(2)	2.6.16	
sync(2)	1.0	
sync_file_range(2)	2.6.17	
sync_file_range2(2)	2.6.22	
syncfs(2)	2.6.39	
sys_debug_setcontext(2)	2.6.11	PowerPC only
syscall(2)	1.0	Still available on ARM OABI and MIPS O32 ABI
sysfs(2)	1.2	
sysinfo(2)	1.0	
syslog(2)	1.0	
sysmips(2)	2.6.0	MIPS only
tee(2)	2.6.17	
tgkill(2)	2.6	
time(2)	1.0	
timer_create(2)	2.6	
timer_delete(2)	2.6	
timer_getoverrun(2)	2.6	
timer_gettime(2)	2.6	
timer_settime(2)	2.6	
timerfd_create(2)	2.6.25	
timerfd_gettime(2)	2.6.25	
timerfd_settime(2)	2.6.25	
times(2)	1.0	
tkill(2)	2.6; 2.4.22	
truncate(2)	1.0	
truncate64(2)	2.4	
ugetrlimit(2)	2.4	
umask(2)	1.0	
umount(2)	1.0	

umount2(2)	2.2	
uname(2)	1.0	
unlink(2)	1.0	
unlinkat(2)	2.6.16	
unshare(2)	2.6.16	
uselib(2)	1.0	
ustat(2)	1.0	
userfaultfd(2)	4.3	
usr26(2)	2.4.8.1	ARM OABI only
usr32(2)	2.4.8.1	ARM OABI only
utime(2)	1.0	
utimensat(2)	2.6.22	
utimes(2)	2.2	
utrap_install(2)	2.2	SPARC64 only
vfork(2)	2.2	
vhangup(2)	1.0	
vm86old(2)	1.0	Was "vm86"; renamed in 2.0.28/2.2
vm86(2)	2.0.28; 2.2	
vmsplice(2)	2.6.17	
wait4(2)	1.0	
waitid(2)	2.6.10	
waitpid(2)	1.0	
write(2)	1.0	
writev(2)	2.0	
xtensa(2)	2.6.13	Xtensa only

On many platforms, including x86-32, socket calls are all multiplexed (via glibc wrapper functions) through **socketcall**(2) and similarly System V IPC calls are multiplexed through **ipc**(2).

Although slots are reserved for them in the system call table, the following system calls are not implemented in the standard kernel: **afs_syscall**(2), **break**(2), **ftime**(2), **getpmsg**(2), **gtty**(2), **idle**(2), **lock**(2), **madvise1**(2), **mpx**(2), **phys**(2), **prof**(2), **profil**(2), **putpmsg**(2), **security**(2), **stty**(2), **tuxcall**(2), **ulimit**(2), and **vserver**(2) (see also **unimplemented**(2)). However, **ftime**(3), **profil**(3), and **ulimit**(3) exist as library routines. The slot for**ph ys**(2) is in use since Linux 2.1.116 for **umount**(2); **phys**(2) will never be implemented. The**getpmsg**(2) and **putpmsg**(2) calls are for k ernels patched to support STREAMS, and may never be in the standard kernel.

There was briefly **set_zone_reclaim**(2), added in Linux 2.6.13, and removed in Linux 2.6.16; this system call was never available to user space.

System calls on removed ports

Some system calls only ever existed on Linux architectures that have since been removed from the kernel:

AVR32 (port removed in Linux 4.12)

- **pread**(2)
- **pwrite**(2)

Blackfin (port removed in Linux 4.17)

- **bfin_spinlock**(2) (added in Linux 2.6.22)
- **dma_memcpy**(2) (added in Linux 2.6.22)
- **pread**(2) (added in Linux 2.6.22)
- **pwrite**(2) (added in Linux 2.6.22)
- **sram_alloc**(2) (added in Linux 2.6.22)
- **sram_free**(2) (added in Linux 2.6.22)

Metag (port removed in Linux 4.17)

- **metag_get_tls**(2) (add in Linux 3.9)

- **metag_set_fpu_flags**(2) (add in Linux 3.9)
- **metag_set_tls**(2) (add in Linux 3.9)
- **metag_setglobalbit**(2) (add in Linux 3.9)

Tile (port removed in Linux 4.17)

- **cmpxchg_badaddr**(2) (added in Linux 2.6.36)

NOTES

Roughly speaking, the code belonging to the system call with number __NR_xxx defined in */usr/include/asm/unistd.h* can be found in the Linux kernel source in the routine *sys_xxx*(). There are many exceptions, however, mostly because older system calls were superseded by newer ones, and this has been treated somewhat unsystematically. On platforms with proprietary operating-system emulation, such as sparc, sparc64, and alpha, there are many additional system calls; mips64 also contains a full set of 32-bit system calls.

Over time, changes to the interfaces of some system calls have been necessary. One reason for such changes was the need to increase the size of structures or scalar values passed to the system call. Because of these changes, certain architectures (notably, longstanding 32-bit architectures such as i386) now have various groups of related system calls (e.g., **truncate**(2) and **truncate64**(2)) which perform similar tasks, but which vary in details such as the size of their arguments. (As noted earlier, applications are generally unaware of this: the glibc wrapper functions do some work to ensure that the right system call is invoked, and that ABI compatibility is preserved for old binaries.) Examples of systems calls that exist in multiple versions are the following:

- By now there are three different versions of **stat**(2): *sys_stat*() (slot *__NR_oldstat*), *sys_newstat*() (slot *__NR_stat*), and *sys_stat64*() (slot *__NR_stat64*), with the last being the most current. A similar story applies for **lstat**(2) and **fstat**(2).

- Similarly, the defines *__NR_oldolduname*, *__NR_olduname*, and *__NR_uname* refer to the routines *sys_olduname*(), *sys_uname*(), and *sys_newuname*().

- In Linux 2.0, a new version of **vm86**(2) appeared, with the old and the new kernel routines being named *sys_vm86old*() and *sys_vm86*().

- In Linux 2.4, a new version of **getrlimit**(2) appeared, with the old and the new kernel routines being named *sys_old_getrlimit*() (slot *__NR_getrlimit*) and *sys_getrlimit*() (slot *__NR_ugetrlimit*).

- Linux 2.4 increased the size of user and group IDs from 16 to 32 bits. To support this change, a range of system calls were added (e.g., **chown32**(2), **getuid32**(2), **getgroups32**(2), **setresuid32**(2)), superseding earlier calls of the same name without the "32" suffix.

- Linux 2.4 added support for applications on 32-bit architectures to access large files (i.e., files for which the sizes and file offsets can't be represented in 32 bits.) To support this change, replacements were required for system calls that deal with file offsets and sizes. Thus the following system calls were added: **fcntl64**(2), **getdents64**(2), **stat64**(2), **statfs64**(2), **truncate64**(2), and their analogs that work with file descriptors or symbolic links. These system calls supersede the older system calls which, except in the case of the "stat" calls, have the same name without the "64" suffix.

  On newer platforms that only have 64-bit file access and 32-bit UIDs/GIDs (e.g., alpha, ia64, s390x, x86-64), there is just a single version of the UID/GID and file access system calls. On platforms (typically, 32-bit platforms) where the *64 and *32 calls exist, the other versions are obsolete.

- The *rt_sig** calls were added in Linux 2.2 to support the addition of real-time signals (see **signal**(7)). These system calls supersede the older system calls of the same name without the "rt_" prefix.

- The **select**(2) and **mmap**(2) system calls use five or more arguments, which caused problems in the way argument passing on the i386 used to be set up. Thus, while other architectures have *sys_select*() and *sys_mmap*() corresponding to *__NR_select* and *__NR_mmap*, on i386 one finds *old_select*() and *old_mmap*() (routines that use a pointer to an argument block) instead. These days passing five arguments is not a problem any more, and there is a *__NR__newselect* that corresponds directly to *sys_select*() and similarly *__NR_mmap2*. s390x is the only 64-bit architecture that has*old_mmap*().

Architecture-specific details: Alpha

getxgid(2)
returns a pair of GID and effective GID via registers **r0** and **r20**; it is provided instead of **getgid**(2) and **getegid**(2).

getxpid(2)
returns a pair of PID and parent PID via registers **r0** and **r20**; it is provided instead of **getpid**(2) and **getppid**(2).

old_adjtimex(2)
is a variant of **adjtimex**(2) that uses *struct timeval32*, for compatibility with OSF/1.

getxuid(2)
returns a pair of GID and effective GID via registers **r0** and **r20**; it is provided instead of **getuid**(2) and **geteuid**(2).

sethae(2)
is used for configuring the Host Address Extension register on low-cost Alphas in order to access address space beyond first 27 bits.

SEE ALSO

ausyscall(1), **intro**(2), **syscall**(2), **unimplemented**(2), **errno**(3), **libc**(7), **vdso**(7)