|

text mode (Lynx, Printer) tables arranged to 110 chars/line.

[intro] [a:index] [s:index] [#:index] [sys_fcntl]
[1..64] [65..128] [129..192] [193..256] [257..] [ref] [struc] [fcntl] [ioctl] [pguide] [man2]
[next] [back] [linux] [main] [bot] [top]
Syscalls pages archive (110K->1M), updated (2.3.0, ver. 1.66) man 2 pages (170K->1,2M)

- **Fight Patents abuse by EC authorities - Gegen Patentmißbrauch durch die EU-Behörden.**

- syscall-no. in **eax**; args left to right in **ebx**,**ecx**,**edx**,**esi**,**edi**(,**ebp**), all preserved.
- returned 0xfffff000 < eax <= 0xffffffff may be an error code (-4096 < eax < 0).
- **Common reference**:
  - <unistd.h>, <linux/types.h>, <asm/types.h>, <asm/posix_types.h>, and <linux/kernel.h> for data structures.
  - The syscall interface is defined in arch/i386/kernel/entry.S, symbols for loadable modules kernel/ksyms.c.
  - Error numbers and messages in include/asm/errno.h.
  - Basic data types
    short: word (16bit), **int**: dword (32bit), **long**: dword (32bit), **pointer**: dword (32bit).

P	exit	int status		1
	terminate current process			
	arg	eax	1	
		ebx	exit code	
	return	eax	-/-	
	errors	eax	-/-	
	source		kernel/exit.c	

P	fork	struct pt_regs		2
	create a child process resource;			
	resource utilizations:=0, file locks & pending signals not inherited.			
	simplified "clone" w/ flags set to SIGCHLD and caller's ESP			
	arg	eax	2	
		ebx...	all register values passed to duplicate job	
	return	eax	clone : zero caller: pid_t pid (dword) or, -ve error code	
	errors	EAGAIN, ENOMEM		

ref	linux/sched.h, asm/ptrace.h.
source	arch/i386/kernel/process.c, kernel/fork.c

P read int fd, void *buf, size_t count 3

read up to count bytes from file fd into buffer *buf

arg	eax	3
	ebx	file descriptor
	ecx	ptr to input buffer
	edx	buffer size, max. count of bytes to receive
return	eax	no. of bytes received, file pointer advanced accordingly
errors	eax	EAGAIN, EBADF, EFAULT, EINTR, EINVAL, EIO, EISDIR
		note that "buffer overflow" is not immanent to the system call but, was introduced by ignorant "programming" rsp (lib)"C" useage/design...
source		fs/read_write.c

P write int fd, void *buf, size_t count, 4

write (up to) count bytes of data from *buf to file fd

arg	eax	4
	ebx	file descriptor
	ecx	ptr to output buffer
	edx	count of bytes to send
return	eax	no. of sent bytes (if POSIX conforming f.s.)
errors	eax	EAGAIN, EBADF, EFAULT, EINTR, EINVAL, EIO, ENOSPC, EPIPE
source	fs/read_write.c	

P open const char *pathname, int flags, mode_t mode 5

open a file or device, create/truncate if corresponding mode flag(s) set

arg	eax	5
	ebx	ptr to asciz abs or rel pathname
	ecx	file access bits
	edx	file permissions, mode
return	eax	int fd (16bit filedescriptor)
errors	eax	ACCESS, EXIST, FAULT, ISDIR, LOOP, MFILE, NAMETOOLONG, NFILE, NOENT, NODEV, NODIR, NOMEM, NOSPC, NXIO, ROFS, TXTBSY
ref		linux/types.h, linux/posix_types.h, linux/stat.h -re- struct stat, asm/fcntl.h, asm/types.h, asm/posix_types.h, fcntl.h, sys/stat.h. sys/types.h
source		fs/open.c

file access bits

O_ACCMODE 0003
O_RDONLY 00
O_WRONLY 01
O_RDWR 02
O_CREAT 0100
O_EXCL 0200
O_NOCTTY 0400
O_TRUNC 01000
O_APPEND 02000
O_NONBLOCK 04000
O_NDELAY O_NONBLOCK
O_SYNC 010000 specific to ext2 fs and block devices
FASYNC 020000 fcntl, for BSD compatibility
O_DIRECT 040000 direct disk access hint - currently ignored
O_LARGEFILE 0100000
O_DIRECTORY 0200000 must be a directory
O_NOFOLLOW 0400000 don't follow links

file permissions flags

S_ISUID 04000 set user ID on execution
S_ISGID 02000 set group ID on execution
S_ISVTX 01000 sticky bit
S_IRUSR 00400 read by owner (S_IREAD)
S_IWUSR 00200 write by owner (S_IWRITE)
S_IXUSR 00100 execute/search by owner (S_IEXEC)
S_IRGRP 00040 read by group
S_IWGRP 00020 write by group
S_IXGRP 00010 execute/search by group
S_IROTH 00004 read by others (R_OK)
S_IWOTH 00002 write by others (W_OK)
S_IXOTH 00001 execute/search by others (X_OK)

S_IRWXUGO (S_IRWXU|S_IRWXG|S_IRWXO) -re- umask
S_IALLUGO (S_ISUID|S_ISGID|S_ISVTX|S_IRWXUGO)
S_IRUGO (S_IRUSR|S_IRGRP|S_IROTH)
S_IWUGO (S_IWUSR|S_IWGRP|S_IWOTH)
S_IXUGO (S_IXUSR|S_IXGRP|S_IXOTH)

P	close	int fd		6
	close a file by fd reference			
	arg	eax	6	
		ebx	file descriptor	
	return	eax	-/-	
	errors	eax	EBADF (-1)	
	source	fs/open.c		

---

waitpid	pid_t pid,	int *status,	int options	7

wait for a specified child process termination; simplified call to wait4

arg	eax	7
	ebx	2nd-ary process id or specification
	ecx	NULL or, ptr to buf for exitted 2nd-ary job status
	edx	option flags, zero or WNOHANG, WUNTRACED
return	eax	pid_t, pid of 2ndary job which exitted
	ecx	if buffer supplied, exit state of 2nd-ary job
errors	eax	CHILD, INVAL, RESTARTSYS
ref	libc: sys/wait.h, wait.c &c, [types]	
source	kernel/exit.c	

---

creat const char *pathname mode_t mode 8

create a file

arg	eax	8
	ebx	ptr to asciz abs or rel pathname
	ecx	file permissions, mode
return	eax	int fd filedescriptor
errors	eax	access, exist, fault, isdir, loop, mfile, nametoolong nfile, noent, nodev, nodir, nomem, nospc, nxio rofs, txtbsy
ref	[types], sys/stat.h, fcntl.h	
source	fs/open.c	
note	sys_creat is identical to sys_open w. O_CREAT|O_WRONLY|O_TRUNC flags.	

P link const char *oldpath const char *newpath 9

create hard link to an existing file

arg	eax	9
	ebx	ptr to asciz abs or rel name of existing file
	ecx	ptr to asciz abs or rel name of new file-name
return	eax	-/-
errors	eax	acces, io, perm, exist, fault, loop mlink, nametoolong, noent, nomem, nospc, notdir, perm, rofs, xdev
ref	-/-	
source	fs/namei.c	

P unlink const char *pathname 10

delete a name and remove file when not busy

arg	eax	10
	ebx	ptr to asciz abs or rel file-name

return	eax	-/-
errors	eax	acces, fault, io, isdir, loop, nametoolong, noent, nomem, notdir, perm, rofs
ref	-/-	
source	fs/namei.c	

P execve const char *filename, char const argv[], char const envp[] 11

execute a program

arg	eax:	11
	ebx:	ptr to <nul> terminated string of program path&name
	ecx:	ptr to zero terminated list of ptrs to <nul> terminated program argument stgs
	edx:	ptr to zero terminated list of ptrs to <nul> terminated environment strings
return		no return, executed prog inherits resources and overwrites caller
errors	eax:	2big, acces, inval, io, isdir, libbad loop, nfile, noexec, noent, nomem, notdir, fault, nametoolong, perm, txtbusy
ref	arch/i386/kernel/process.c	
note		waiting caller of a "fork"ed or "clone"d job being released immediately after entry to execve.

P chdir const char *path 12

change working directory

arg	eax	12
	ebx	ptr to asciz abs or rel file-name
return	eax	-/-
errors	eax	acces, badf, fault, io, loop, nametoolong, noent, nomem, notdir
ref	-/-	
source	fs/open.c	

---

time time_t *t 13

get time in seconds since 1-jan-1970

arg	eax	13
	ebx	NULL or, ptr to buffer which receives a copy of the return value
return	eax	((time_t)-1) seconds
errors	eax	fault
ref	linux/time.h	
source	kernel/time.c	

P mknod const char *pathname mode_t mode dev_t dev 14

create a filesystem node (file, device, special file, named pipe)
return -/-
errors acces, exist, fault, inval, loop, nametoolong, noent, nomem, nospc, notdir, perm, rofs
ref sys/stat.h (mode flags), fcntl.h, [types]
source fs/namei.c; include/linux/major.h, Documentation/devices.txt

---

P chmod const char *path mode_t mode 15
change file permissions (attributes)
return -/-
errors acces, badf, fault, io, loop, nametoolong, noent, nomem, notdir, perm, rofs
ref [types], sys/stat.h, asm/stat.h
source fs/open.c

-re- **access mode flags**

---

lchown const char *path uid_t owner gid_t group 16
change owner of a file, don't follow symbolic links
return -/-
errors (many...)
source fs/open.c

---

break 17
n.i.

---

oldstat 18
(abandoned)

P lseek int fd off_t offset int whence 19
change file pointer of file ref-d by fd
arg eax 19
ebx file descriptor
ecx disp to the rsp. file position -re - "whence"
edx "whence" flag, either one of
SEEK_SET 0 add disp to beginning of file,
SEEK_CUR 1 add disp to current position,

		SEEK_END 2 add disp to end of file.
return	eax	off_t ptr posn. re beginning of file
errors	eax	badf, inval, ispipe
ref	unistd.h, linux/unistd.h, [types]	
source	fs/read_write.c	

---

getpid	void		20
get pid of current process			
return	pid_t pid		
errors	-/-		
source	kernel/sched.c		

P	mount	c ch *specialfile	c ch *dir	c ch *fstype	ul rwflag	c v *data	21

mount a filesystem

arg	eax	21		
	ebx	ptr to asciz name of device/specialfile		
	ecx	ptr to asciz name of mount-point		
	edx	ptr to asciz name of file system type		
flags	esi	MS_MGC_MSK	0xC0ED	in m.s. 16-bit; 'magic', required till 2.4.0-t9
		MS_RDONLY	1	Mount read-only
		MS_NOSUID	2	Ignore suid and sgid bits
		MS_NODEV	4	Disallow access to device special files
		MS_NOEXEC	8	Disallow program execution
		MS_SYNCHRONOUS	16	Writes are synced at once
		MS_REMOUNT	32	Alter flags of a mounted FS
		MS_MANDLOCK	64	Allow mandatory locks on an FS
		MS_NOATIME	1024	Do not update access times
		MS_NODIRATIME	2048	Do not update directory access times
		MS_BIND	4096	
		MS_REC	16384	
		MS_VERBOSE	32768	
		MS_ACTIVE	(1<<30)	
		MS_NOUSER	(1<<31)	
	edi	ptr to device independent, additional data (max. page-size)		
return	eax	-/-		
errors	eax	any which can occur in the particular f.s. or, in the kernel		
	K4+	multiple **identical mounts apparently not considered an error**, any more, which, could result to quite dangerous situations, e.g. after single user emergency startup, where the entire file system might become mounted on the root device!		

ref sys/mount.h linux/mount.h linux/fs.h
source fs/super.c, fs/namespace.c (2.4)

---

umount K4:oldumount const char *specialfile | *dir 22
unmount a filesystem
return -/-
errors any which can occur in the particular f.s. or, in the kernel
ref sys/mount.h, /proc/filesystems, umount2
source fs/super.c, fs/namespace.c (2.4)
NOTE inconsistently def'd wrt prior sources in kernel 2.4.18!
according to 2.4.18 explanation 'oldumount' executes (new) 'mount *device,0'
[hp] 2.4.18 variant found to not operating on (linked) device files, any more.

P setuid uid_t uid 23
set real user id
return -/-
errors eperm
source kernel/sys.c

P getuid void 24
get real user id
return uid_t uid
errors -/-
ref unistd.h, linux/unistd.h, [types]
source kernel/sched.c

---

stime time_t *t 25
set secinds since 1-jan-1970
return -/-
errors eperm
ref time.h
source kernel/time.c

---

ptrace long request, pid_t pid, long addr, long data 26
trace a child process
arg eax 26

	ebx	requested action	
	ecx	process id of the target job	"child"
	edx	address in target job	"address"
	esi	address in tracing job	"data"
return	eax	as specified by the request	
errors	io, fault, perm, srch		
ref	linux/ptrace.h. asm/ptrace.h, linux/sched.h. asm/user.h,		
source	arch/i386/kernel/ptrace.c		

requests PTRACE_.. ret

0. PEEKTEXT pid,addr,*data read word at location addr.
1. PEEKDATA pid,addr,*data read the word at location addr in the USER area.
2. PEEKUSR pid,addr,*data "
3. POKETEXT pid,addr,*data write the word at location addr.
4. POKEDATA pid,addr,*data "
5. POKEUSR pid,addr,*data write the word at location addr in the USER area
6. TRACEME -/- set the ptrace bit in the process flags.
7. CONT pid,,signo restart after signal.
8. KILL pid make the child exit, send it a sigkill.
9. SINGLESTEP pid sigtrap set the trap flag.
10. n.n.
11. n.n.
12. GETREGS pid,,*data get all gp regs from the child. re register.
13. SETREGS pid,,*data set all gp regs in the child.
14. GETFPREGS pid,,*data get the child FPU state.
15. SETFPREGS pid,,*data set the child FPU state.
16. ATTACH pid attach to already running process
17. DETACH pid detach a process that was attached.
18. K4 GETFPXREGS pid,,*data get the child extended FPU state.
19. K4 SETFPXREGS pid,,*data set the child extended FPU state.
20. n.n.
21. K4 SETOPTIONS pid,,data -no comment- (option 'PTRACE_O_TRACESYSGOOD' or none)
22. n.n.
23. n.n.
24. SYSCALL pid,,signo continue and stop at next (return from) syscall

registers

0. EBX
1. ECX
2. EDX
3. ESI
4. EDI
5. EBP
6. EAX
7. DS
8. ES
9. FS
10. GS
11. ORIG_EAX
12. EIP
13. CS
14. EFL

15. UESP
16. SS
17. FRAME_SIZE

P alarm u int seconds 27
send SIGALARM at a specified time
return 0 or unsigned int seconds until prev. alarms are due to execute
errors -/-
source kernel/sched.c
sys_alarm is a simplified call to sys_setitimer.

---

oldfstat 28
(abandoned)

---

pause void 29
sleep until signal
return EINTR if signal caught
errors restartnohand
source kernel/sched.c

P utime char *filename struct utimbuf *buf 30
change access and/or modification time of an inode
return -/-
errors acces, noent
ref sys/time.h, utime.h, [types]
source fs/open.c

---

stty 31
n.i.

---

gtty 32
n.i.

P	access	const char *pathname	int mode	33
	check user's permissions for a file			
	arg	eax	33	
		ebx	ptr to asciz string w. path name	
		ecx	mode flags, R_OK, W_OK, X_OK, F_OK = 0, re 'open'	
	return	eax	-/-	
	errors	eax	all f.s. and file access related errors can occur	
	source	fs/open.c, libc/gcc "unistd.h"		

---

nice	int inc		34
change process priority			
return	-/-		
errors	perm		
source	kernel/sched.c		

---

ftime		35
	n.i.	

P	sync	void	36
	commit buffer cache to disk		
	return	always 0	
	errors	-/-	
	source	fs/buffer.c	

P	kill	pid_t pid	int sig	37
	send a signal to a process			
	return	-/-		
	errors	inval, perm, srch		
	ref	kernel/sys.c		
	source	kernel/signal.c, [more]		

```
pid             signal sent to
-------- | ----------------------------------------
>  0       process {pid}
   0       every process in caller's process group
  -1       all processes but the very first
< -1       every process in group |{gid}| ("killpg", re getgid)

no action if sig = 0, but error checking occurs, i.e.
this can be used to check whether a process exists.
```

P rename c ch *oldpath c ch *newpath 38
move/rename a file
return -/-
errors busy, exist, isdir, notempty, xdev (and other f.s. errors)
ref stdio.h
source fs/namei.c

P mkdir const char *path mode_t mode 39
create a directory
return -/-
errors (f.s. & perm's related errors)
ref sys/stat.h, fcntl.h, unistd.h, linux/unistd.h, [types]
source fs/namei.c

P rmdir const char *pathname 40
delete an empty directory
return -/-
errors access, busy, fault, loop, nametoolong,
noent, nomem, notdir, notempty, perm, rofs
source fs/namei.c

P dup int oldfd 41
create a file descriptor duplicate
return int new fd, the lowest until then un-used number
errors badf, mfile (,inval)
source fs/fcntl.c
note dup as well as dup2 assign new fd-s but don't duplicate the channel's description,
any modification, e.g by an ioctl, would affect all instances of a resp. fd!

P pipe int fd[2] 42
create a pipe
arg eax 42
ebx ptr to writable array of two dwords
return [ebx] fd(0) at [ebx] for input from, fd(1) at [ebx+4] output to pipe
errors eax fault, mfile, nfile
source arch/i386/kernel/sys_i386.c

note: re mknod for named pipes.

---

	times	struct tms *buf	43
	get process times		
	return	-/-	
	errors	-/-	
	ref	sys/times.h	
	source	kernel/sys.c	

---

	prof		44
		n.i.	

P	brk	void *end_data_segment	45
	change data section size, top of .bss		
	arg	eax	45
		ebx	(virtual) address of new .bss top in range of data-space bottom and, below any linked library or 16K below .ss bottom, rsp, within the limits of the owner process' data size - re getrlimit.
		ebx := 0	can be used to finding the currently valid top address of .bbs.
	return eax currently, after call valid top address		
	errors	eax	nomem
	source		mm/mmap.c

P	setgid	gid_t gid	46
	set real group id of current process		
	return	-/-	
	errors	perm	
	source	kernel/sys.c	

P	getgid	void	47
	get real group id of current process		
	return	gid_t gid	
	errors	-/-	

ref	unistd.h, linux/unistd.h, [types]	
source	kernel/sched.c	

P	signal	*signal(int signum, void (*handler)(int))	48
	setup a signal handler		
	return	void previous value of signal hnd	
	errors	sig_err	
	source	kernel/signal.c	

---

	geteuid	void	49
	get effective user id of current process		
	return	uid_t uid	
	errors	-/-	
	ref	unistd.h, linux/unistd.h, [types]	
	source	kernel/sched.c	

---

	getegid	void	50
	get effective group id of current process		
	return	gid_t gid	
	errors	-/-	
	ref	unistd.h, linux/unistd.h, [types]	
	source	kernel/sched.c	

---

	acct	int acct(const char *filename)	51
	switch process accounting on/off		
	return	0 or -ve ernum	
	errors	nosys, nomem, perm, access, io, users	
	source	kernel/sys.c	

---

	umount2 K4:umount	const char *specialfile | *dir	int flag	52
	arg	*specialfile ptr to asciz mountpoint(?) or device/specialfile name flag 1 to forcedly un-mount, else 0.		
	return	zero after success, else ernum		
	errors	any f.s. error		
	source	fs/namespace.c		
	ref	oldumount (2.4, formerly named "umount")		

<manpages-1.50> include ref to "umount2" by 'man 2 umount'

NOTE: renamed to "umount" w. kernel 2.4.18(+?); from the source file:

"Now umount can handle mount points as well as block devices. This is important for filesystems which use unnamed block devices. We now support a flag for forced unmount like the other 'big iron' unixes. Our API is identical to OSF/1 to avoid making a mess of AMD.

---

lock 53

n.i.

P ioctl int fd int request char *argp 54

manipulate a character device

arg	eax	54
	ebx	file descriptor
	ecx	command code
	edx	ptr to writable data area or other arg structure
return	eax	zero on success
errors	eax	badf, fault, inval, notty
ref		asm/ioctl.h - **ioctl** page.
source		fs/ioctl.c
		for instance, console ioctls provide many nice features - most of which aren't overly helpful because, all(!) consoles would globally be affected!

---

fcntl int fd int cmd [struct flock *lock] 55

file/-descriptor control

arg	eax	55
	ebx	file descriptor
	ecx	command code
	edx	file locks: ptr to writable struct flock
return	eax	according to the rsp command
errors	eax	acces, again, badf, deadlk, fault, intr, inval, mfilem nolock, perm
ref		fcntl page.
source		fs/fcntl.c, unistd.h, linux/unistd.h, fcntl.h.

---

mpx 56

n.i.

P	setpgid	pid_t pid	pid_t pgid	57
	set process group id			
	return	-/-		
	errors	inval, perm, srch		
	source	kernel/sys.c		

ulimit	int cmd	(...)	58	libc
get/set resource limits - not a system call				
ref	syscalls setrlimit, getrlimit			
ref (glibc2)	man (3) ulimit, sysdeps/generic/ulimit.c, sysdeps/unix/sysv/linux/ulimit.c			
NOTE:	not a kernel syscall			

oldolduname		59
(syscall removed)		
source	utsname.h, struc oldoldutsname	

umask	mode_t mask		60
set file creation mask			
arg	eax	60	
	ebx	file creation permissions, masked w. S_IRWXUGO by kernel code	
return	eax	mode_t previously valid mask	
errors	-/-		
ref	sys/stat.h, [types]		
source	kernel/sys.c		

P	chroot	const char *path	61
	change root directory (tree)		
	return	-/-	
	errors	any, dependent on f.s. type	
	source	fs/open.c	

ustat	62
n.i. - libc call	

P	dup2	int oldfd	int newfd	63

duplicate file descriptor oldfd, on success replace newfd w/ duplicated.

arg	eax	63
	ebx	file descriptor to duplicate
	ecx	file descriptor which the duplicate should be assigned to
return	eax	duplicate fd
errors	eax	badf, mfile
source	fs/fcntl.c	
note	dup2 as well as dup assign new fd-s but don't duplicate the channel's description, any modification, e.g by an ioctl, would affect all instances of a resp. fd! 'newfd' will be closed prior to re-assignation, if applicable.	

P	getppid	void		64

get parent process id

arg	eax	64
return	eax	pid_t pid
errors	-/-	
source	kernel/sched.c	

[next] [back] [bot] [top]
[intro] [a:index] [#:index] [1..64] [65..128] [129..192] [193..] [ref] [struc] [ioctl] [pguide]

H.-Peter Recktenwald, Berlin, 18.Feb.2000 = .hpr.l0 = 12/20/2010 10:07:07 :