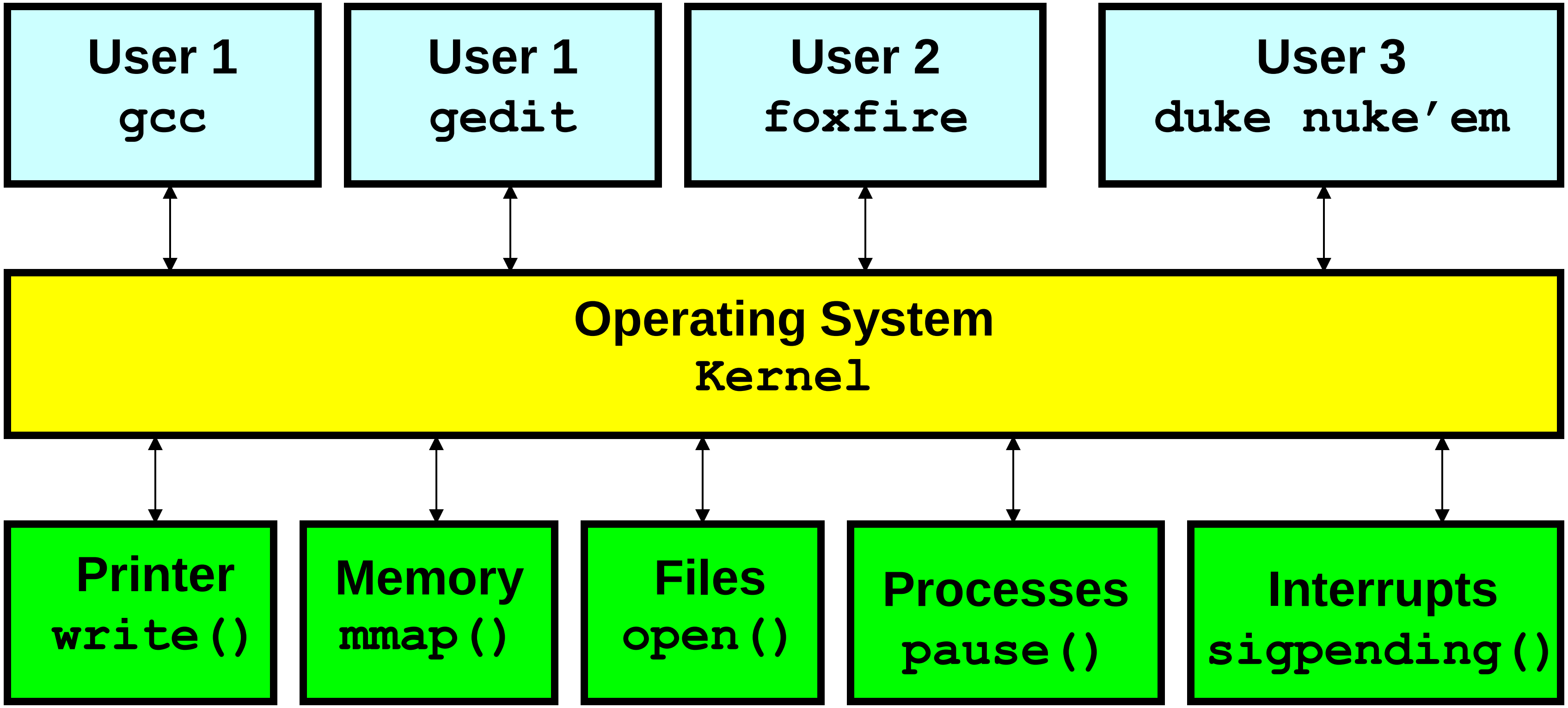
Chapter 7

System Calls

Operating Systems

An Operating System is a “middle-man” between the hardware of the computer and the user applications running on the computer.



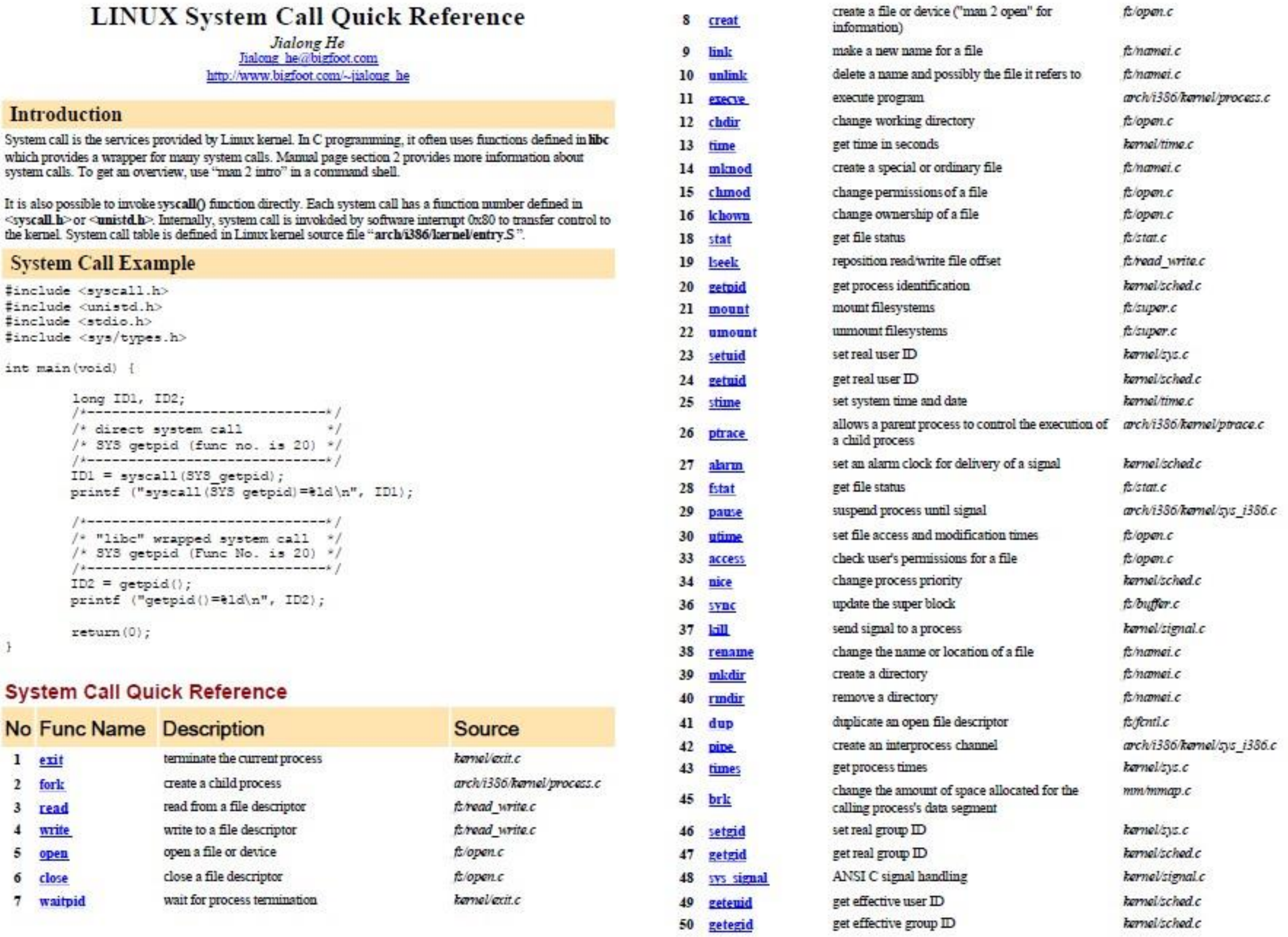
7.1 Families of Operations

The Operating System protects the computer's resources by supplying functions calls which are simply requests from the applications to the O/S for access to resources.

This set of function calls is generally called an Application Program Interface, or API.

Modern Unix operating systems provide hundreds of different *system calls*.

http://www.cheat-sheets.org/saved-copy/Linux_Syscall_quickref.pdf



LINUX System Call Quick Reference

Jialong He
Jialong_he@bigfoot.com
http://www.bigfoot.com/~jialong_he

Introduction

System call is the services provided by Linux kernel. In C programming, it often uses functions defined in **libc** which provides a wrapper for many system calls. Manual page section 2 provides more information about system calls. To get an overview, use "man 2 intro" in a command shell.

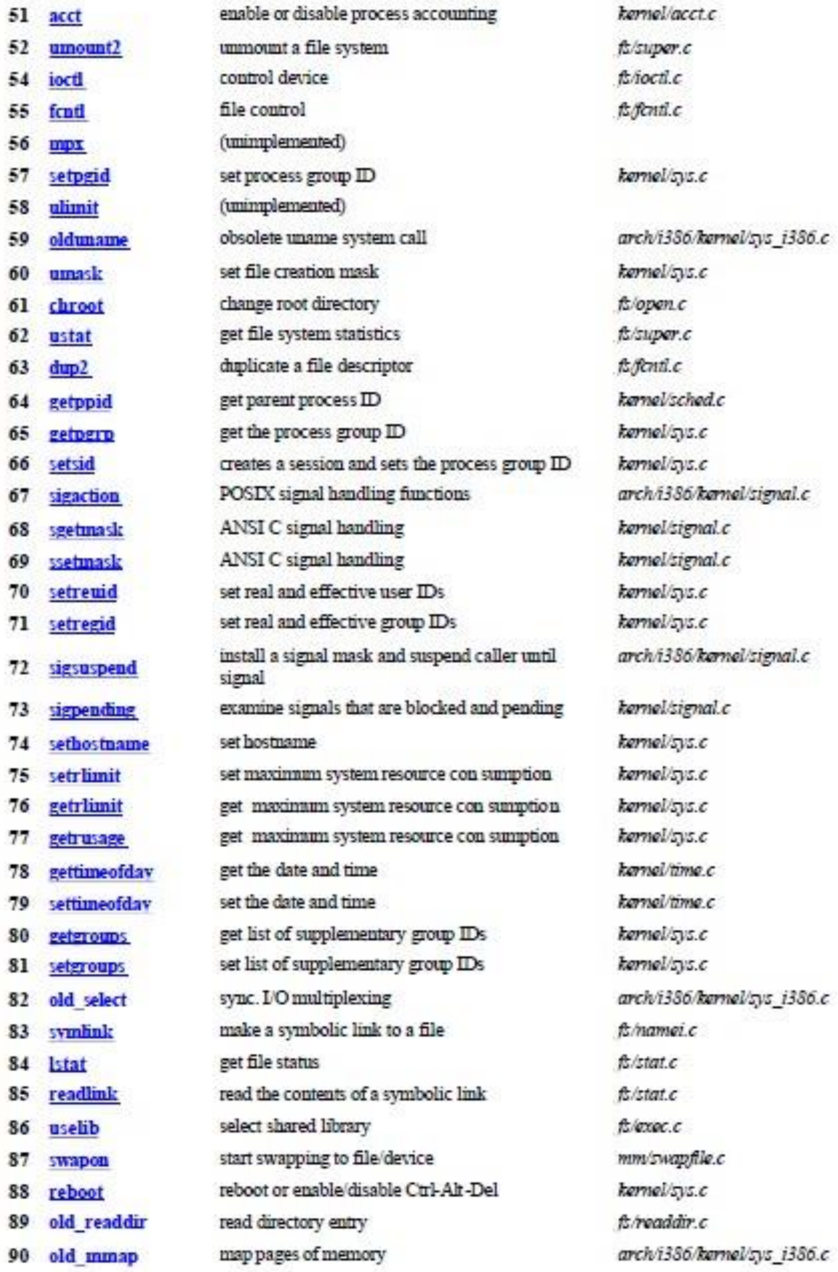
It is also possible to invoke sys**call()** function directly. Each system call has a function number defined in <**syscall.h**> or <**unistd.h**>. Internally, system call is invokded by software interrupt 0x80 to transfer control to the kernel. System call table is defined in Linux kernel source file "**arch/i386/kernel/entry.S**".

System Call Example

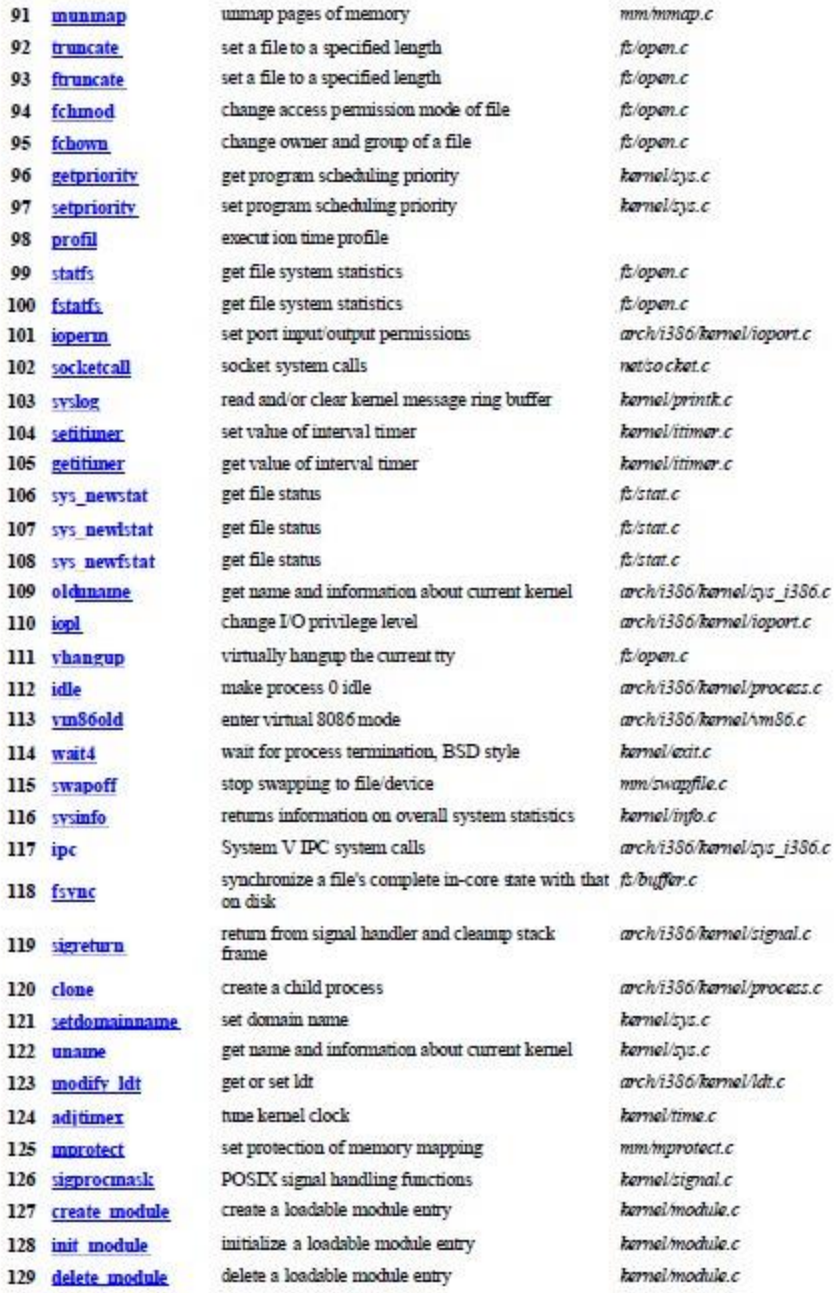
```
#include <syscall.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>

int main(void) {

        long ID1, ID2;
        /*-----------------------------*/
        /* direct system call          */
        /* SYS getpid (func no. is 20) */
        /*-----------------------------*/
        ID1 = syscall(SYS_getpid);
        printf ("syscall(SYS getpid)=%ld\n", ID1);

        /*-----------------------------*/
        /* "libc" wrapped system call  */
        /* SYS getpid (Func No. is 20) */
        /*-----------------------------*/
        ID2 = getpid();
        printf ("getpid()=%ld\n", ID2);

        return(0);
}
```

System Call Quick Reference

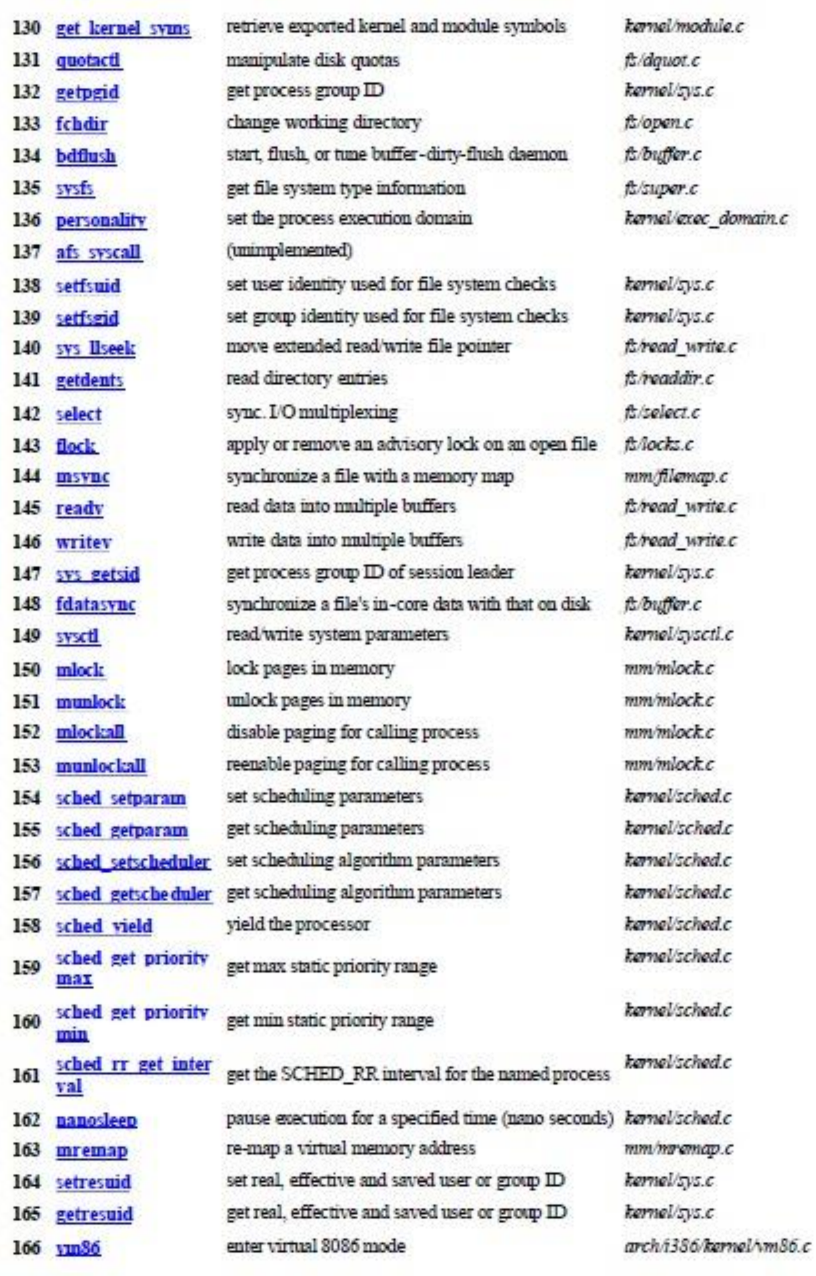
No	Func Name	Description	Source
1	exit	terminate the current process	*kernel/exit.c*
2	fork	create a child process	*arch/i386/kernel/process.c*
3	read	read from a file descriptor	*fs/read_write.c*
4	write	write to a file descriptor	*fs/read_write.c*
5	open	open a file or device	*fs/open.c*
6	close	close a file descriptor	*fs/open.c*
7	waitpid	wait for process termination	*kernel/exit.c*
8	creat	create a file or device ("man 2 open" for information)	*fs/open.c*
9	link	make a new name for a file	*fs/namei.c*
10	unlink	delete a name and possibly the file it refers to	*fs/namei.c*
11	execve	execute program	*arch/i386/kernel/process.c*
12	chdir	change working directory	*fs/open.c*
13	time	get time in seconds	*kernel/time.c*
14	mknod	create a special or ordinary file	*fs/namei.c*
15	chmod	change permissions of a file	*fs/open.c*
16	lchown	change ownership of a file	*fs/open.c*
18	stat	get file status	*fs/stat.c*
19	lseek	reposition read/write file offset	*fs/read_write.c*
20	getpid	get process identification	*kernel/sched.c*
21	mount	mount filesystems	*fs/super.c*
22	umount	unmount filesystems	*fs/super.c*
23	setuid	set real user ID	*kernel/sys.c*
24	getuid	get real user ID	*kernel/sched.c*
25	stime	set system time and date	*kernel/time.c*
26	ptrace	allows a parent process to control the execution of a child process	*arch/i386/kernel/ptrace.c*
27	alarm	set an alarm clock for delivery of a signal	*kernel/sched.c*
28	fstat	get file status	*fs/stat.c*
29	pause	suspend process until signal	*arch/i386/kernel/sys_i386.c*
30	utime	set file access and modification times	*fs/open.c*
33	access	check user's permissions for a file	*fs/open.c*
34	nice	change process priority	*kernel/sched.c*
36	sync	update the super block	*fs/buffer.c*
37	kill	send signal to a process	*kernel/signal.c*
38	rename	change the name or location of a file	*fs/namei.c*
39	mkdir	create a directory	*fs/namei.c*
40	rmdir	remove a directory	*fs/namei.c*
41	dup	duplicate an open file descriptor	*fs/fcntl.c*
42	pipe	create an interprocess channel	*arch/i386/kernel/sys_i386.c*
43	times	get process times	*kernel/sys.c*
45	brk	change the amount of space allocated for the calling process's data segment	*mm/mmap.c*
46	setgid	set real group ID	*kernel/sys.c*
47	getgid	get real group ID	*kernel/sched.c*
48	sys_signal	ANSI C signal handling	*kernel/signal.c*
49	geteuid	get effective user ID	*kernel/sched.c*
50	getegid	get effective group ID	*kernel/sched.c*



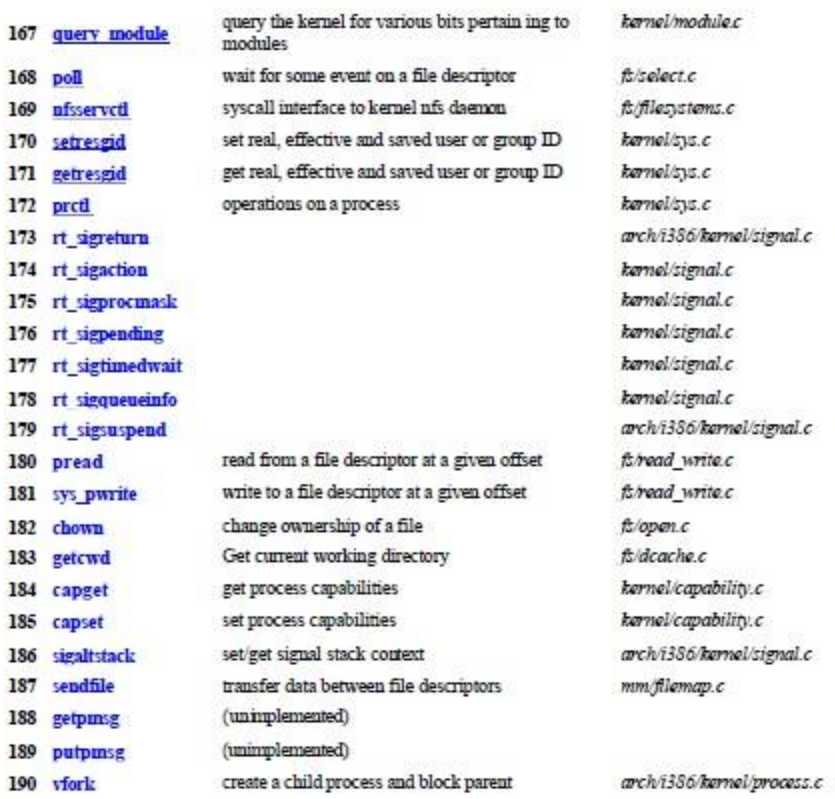
51	acct	enable or disable process accounting	kernel/acct.c
52	umount2	unmount a file system	fs/super.c
54	ioctl	control device	fs/ioctl.c
55	fcntl	file control	fs/fcntl.c
56	mpx	(unimplemented)	
57	setpgid	set process group ID	kernel/sys.c
58	ulimit	(unimplemented)	
59	olduname	obsolete uname system call	arch/i386/kernel/sys_i386.c
60	umask	set file creation mask	kernel/sys.c
61	chroot	change root directory	fs/open.c
62	ustat	get file system statistics	fs/super.c
63	dup2	duplicate a file descriptor	fs/fcntl.c
64	getppid	get parent process ID	kernel/sched.c
65	getpgrp	get the process group ID	kernel/sys.c
66	setsid	creates a session and sets the process group ID	kernel/sys.c
67	sigaction	POSIX signal handling functions	arch/i386/kernel/signal.c
68	sgetmask	ANSI C signal handling	kernel/signal.c
69	ssetmask	ANSI C signal handling	kernel/signal.c
70	setreuid	set real and effective user IDs	kernel/sys.c
71	setregid	set real and effective group IDs	kernel/sys.c
72	sigsuspend	install a signal mask and suspend caller until signal	arch/i386/kernel/signal.c
73	sigpending	examine signals that are blocked and pending	kernel/signal.c
74	sethostname	set hostname	kernel/sys.c
75	setrlimit	set maximum system resource con sumption	kernel/sys.c
76	getrlimit	get maximum system resource con sumption	kernel/sys.c
77	getrusage	get maximum system resource con sumption	kernel/sys.c
78	gettimeofday	get the date and time	kernel/time.c
79	settimeofday	set the date and time	kernel/time.c
80	getgroups	get list of supplementary group IDs	kernel/sys.c
81	setgroups	set list of supplementary group IDs	kernel/sys.c
82	old_select	sync. I/O multiplexing	arch/i386/kernel/sys_i386.c
83	symlink	make a symbolic link to a file	fs/namei.c
84	lstat	get file status	fs/stat.c
85	readlink	read the contents of a symbolic link	fs/stat.c
86	uselib	select shared library	fs/exec.c
87	swapon	start swapping to file/device	mm/swapfile.c
88	reboot	reboot or enable/disable Ctrl-Alt-Del	kernel/sys.c
89	old_readdir	read directory entry	fs/readdir.c
90	old_mmap	map pages of memory	arch/i386/kernel/sys_i386.c



91	munmap	unmap pages of memory	mm/mmap.c
92	truncate	set a file to a specified length	fs/open.c
93	ftruncate	set a file to a specified length	fs/open.c
94	fchmod	change access permission mode of file	fs/open.c
95	fchown	change owner and group of a file	fs/open.c
96	getpriority	get program scheduling priority	kernel/sys.c
97	setpriority	set program scheduling priority	kernel/sys.c
98	profil	execut ion time profile	
99	statfs	get file system statistics	fs/open.c
100	fstatfs	get file system statistics	fs/open.c
101	ioperm	set port input/output permissions	arch/i386/kernel/ioport.c
102	socketcall	socket system calls	net/socket.c
103	syslog	read and/or clear kernel message ring buffer	kernel/printk.c
104	setitimer	set value of interval timer	kernel/itimer.c
105	getitimer	get value of interval timer	kernel/itimer.c
106	sys_newstat	get file status	fs/stat.c
107	sys_newlstat	get file status	fs/stat.c
108	sys_newfstat	get file status	fs/stat.c
109	olduname	get name and information about current kernel	arch/i386/kernel/sys_i386.c
110	iopl	change I/O privilege level	arch/i386/kernel/ioport.c
111	vhangup	virtually hangup the current tty	fs/open.c
112	idle	make process 0 idle	arch/i386/kernel/process.c
113	vm86old	enter virtual 8086 mode	arch/i386/kernel/vm86.c
114	wait4	wait for process termination, BSD style	kernel/exit.c
115	swapoff	stop swapping to file/device	mm/swapfile.c
116	sysinfo	returns information on overall system statistics	kernel/info.c
117	ipc	System V IPC system calls	arch/i386/kernel/sys_i386.c
118	fsync	synchronize a file's complete in-core state with that on disk	fs/buffer.c
119	sigreturn	return from signal handler and cleanup stack frame	arch/i386/kernel/signal.c
120	clone	create a child process	arch/i386/kernel/process.c
121	setdomainname	set domain name	kernel/sys.c
122	uname	get name and information about current kernel	kernel/sys.c
123	modify_ldt	get or set ldt	arch/i386/kernel/ldt.c
124	adjtimex	tune kernel clock	kernel/time.c
125	mprotect	set protection of memory mapping	mm/mprotect.c
126	sigprocmask	POSIX signal handling functions	kernel/signal.c
127	create_module	create a loadable module entry	kernel/module.c
128	init_module	initialize a loadable module entry	kernel/module.c
129	delete_module	delete a loadable module entry	kernel/module.c



130	get_kernel_syms	retrieve exported kernel and module symbols	*kernel/module.c*
131	quotactl	manipulate disk quotas	*fs/dquot.c*
132	getpgid	get process group ID	*kernel/sys.c*
133	fchdir	change working directory	*fs/open.c*
134	bdflush	start, flush, or tune buffer-dirty-flush daemon	*fs/buffer.c*
135	sysfs	get file system type information	*fs/super.c*
136	personality	set the process execution domain	*kernel/exec_domain.c*
137	afs_syscall	(unimplemented)	
138	setfsuid	set user identity used for file system checks	*kernel/sys.c*
139	setfsgid	set group identity used for file system checks	*kernel/sys.c*
140	sys_llseek	move extended read/write file pointer	*fs/read_write.c*
141	getdents	read directory entries	*fs/readdir.c*
142	select	sync. I/O multiplexing	*fs/select.c*
143	flock	apply or remove an advisory lock on an open file	*fs/locks.c*
144	msync	synchronize a file with a memory map	*mm/filemap.c*
145	readv	read data into multiple buffers	*fs/read_write.c*
146	writev	write data into multiple buffers	*fs/read_write.c*
147	sys_getsid	get process group ID of session leader	*kernel/sys.c*
148	fdatasync	synchronize a file's in-core data with that on disk	*fs/buffer.c*
149	sysctl	read/write system parameters	*kernel/sysctl.c*
150	mlock	lock pages in memory	*mm/mlock.c*
151	munlock	unlock pages in memory	*mm/mlock.c*
152	mlockall	disable paging for calling process	*mm/mlock.c*
153	munlockall	reenable paging for calling process	*mm/mlock.c*
154	sched_setparam	set scheduling parameters	*kernel/sched.c*
155	sched_getparam	get scheduling parameters	*kernel/sched.c*
156	sched_setscheduler	set scheduling algorithm parameters	*kernel/sched.c*
157	sched_getscheduler	get scheduling algorithm parameters	*kernel/sched.c*
158	sched_yield	yield the processor	*kernel/sched.c*
159	sched_get_priority_max	get max static priority range	*kernel/sched.c*
160	sched_get_priority_min	get min static priority range	*kernel/sched.c*
161	sched_rr_get_interval	get the SCHED_RR interval for the named process	*kernel/sched.c*
162	nanosleep	pause execution for a specified time (nano seconds)	*kernel/sched.c*
163	mremap	re-map a virtual memory address	*mm/mremap.c*
164	setresuid	set real, effective and saved user or group ID	*kernel/sys.c*
165	getresuid	get real, effective and saved user or group ID	*kernel/sys.c*
166	vm86	enter virtual 8086 mode	*arch/i386/kernel/vm86.c*



167	query_module	query the kernel for various bits pertain ing to modules	*kernel/module.c*
168	poll	wait for some event on a file descriptor	*fs/select.c*
169	nfsservctl	syscall interface to kernel nfs daemon	*fs/filesystems.c*
170	setresgid	set real, effective and saved user or group ID	*kernel/sys.c*
171	getresgid	get real, effective and saved user or group ID	*kernel/sys.c*
172	prctl	operations on a process	*kernel/sys.c*
173	rt_sigreturn		*arch/i386/kernel/signal.c*
174	rt_sigaction		*kernel/signal.c*
175	rt_sigprocmask		*kernel/signal.c*
176	rt_sigpending		*kernel/signal.c*
177	rt_sigtimedwait		*kernel/signal.c*
178	rt_sigqueueinfo		*kernel/signal.c*
179	rt_sigsuspend		*arch/i386/kernel/signal.c*
180	pread	read from a file descriptor at a given offset	*fs/read_write.c*
181	sys_pwrite	write to a file descriptor at a given offset	*fs/read_write.c*
182	chown	change ownership of a file	*fs/open.c*
183	getcwd	Get current working directory	*fs/dcache.c*
184	capget	get process capabilities	*kernel/capability.c*
185	capset	set process capabilities	*kernel/capability.c*
186	sigaltstack	set/get signal stack context	*arch/i386/kernel/signal.c*
187	sendfile	transfer data between file descriptors	*mm/filemap.c*
188	getpmsg	(unimplemented)	
189	putpmsg	(unimplemented)	
190	vfork	create a child process and block parent	*arch/i386/kernel/process.c*

Families of Operations

These functions can be broken down into several families.

- Memory Management — Allocates, de-allocates and protects memory for programs.
- Time Management — Provides access to the system clock.
- File System Management — Allows programs to create, modify, and delete files.
- Process Management — Allows programs to create, maintain, and delete processes.

Families of Operations 2

- Signal Management — Provides for inter-process communication.
- Socket Management — Provides for inter-process communication between machines on a network.
- Thread Management — Allows programs to create, manage, and delete threads.
- Miscellaneous Functions — Allows user to access functionality of the O/S in a safe manner.

7.2 Libraries and System Calls

What is the difference between `open()` and `fopen()` or `read()` and `fread()`?

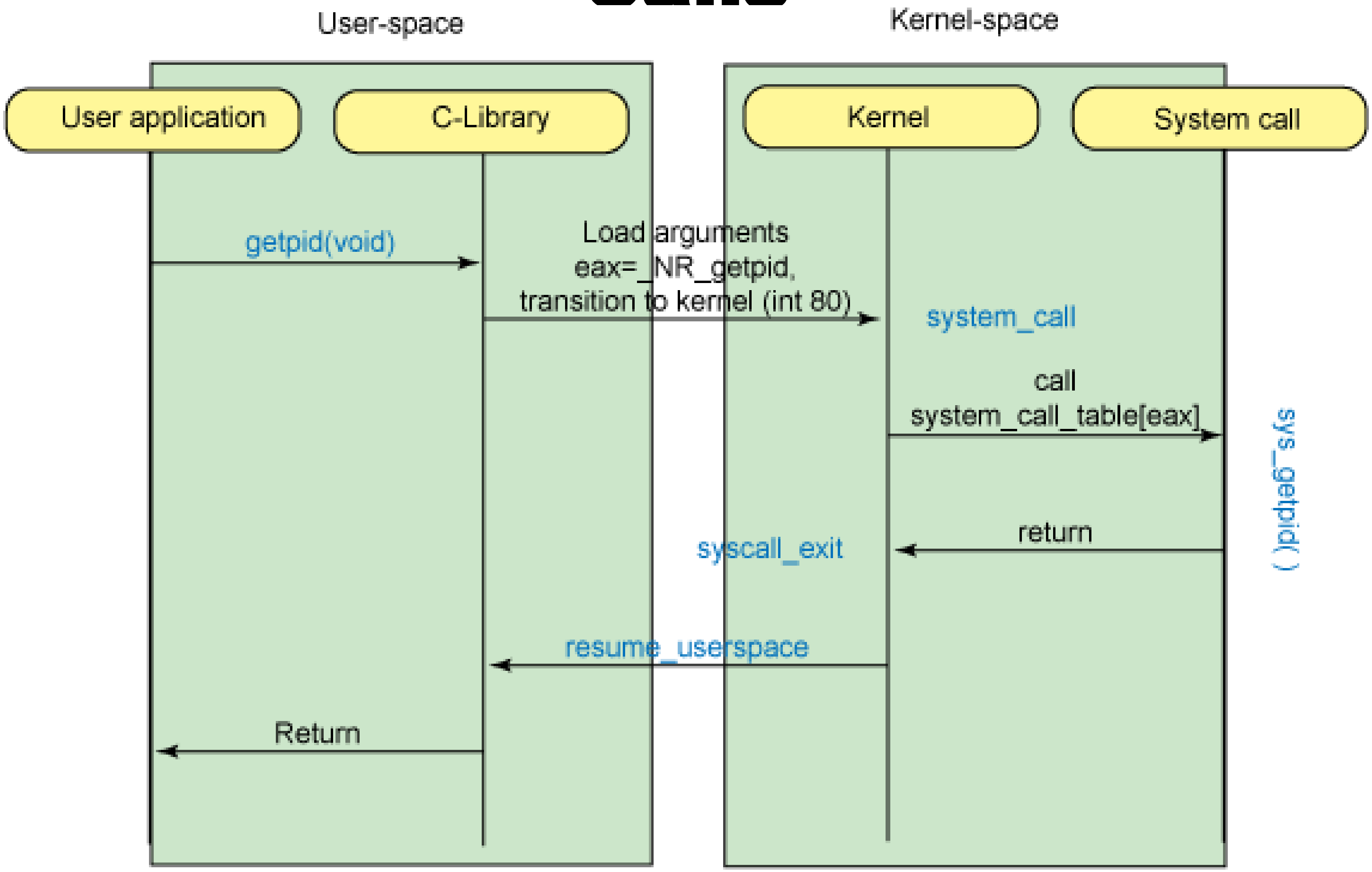
`open()` and `read()` are system calls.

`fopen()` and `fread()` are library functions which call system calls.

In most cases, the difference between library system functions and system calls is negligible.

The closer you get to devices (hardware) and their drivers, however, the greater their differences become.

Libraries accessing System Calls



http://www-128.ibm.com/developerworks/linux/library/l-system-calls/

Library Functions vs. System Calls

System calls are more powerful than library functions. That is, they give you more control over the hardware because they are more specific to the hardware.

Library functions are built on top of system calls. For example, `malloc()` is built on top of the `mmap()` and `brk()`. That is, the C standard library function `malloc()` calls the system function `mmap()` to get the job done.

Library Functions
vs.
System Calls

As the name might imply, standard library functions tend to be more standardized than system calls.

The ANSI group defines library standards and the POSIX group defines system call standards. However, since system calls provide direct access to the kernel, different O/S’s will have, at least, slightly different system calls.

Library Functions vs. System Calls 2

When you call a System Calls, the O/S performs a *context switch*. That is, your application program stops running while the kernel program runs for a while to perform the requested operation.

Therefore, system calls take longer to execute than a user function. Standard library calls often optimize operations to minimize the number of system calls saving execution time.

Library Functions vs. System Calls 3

As an example, consider the system calls `open()`, `read()`, `write()`, and `close()` versus the library functions `fopen()`, `fread()`, `fwrite()`, and `fclose()`.

The library calls can be more efficient because they use buffering. When `fread()` is called a block of data is read into a buffer using `read()` instead of only the amount that is requested. If `fread()` is called again, it may not be necessary to actually call `read()`.

System Call Manual

The man pages for system calls, library functions, and system programs are all stored separately from each other.

`man 1 ls` – Provides info on the system program `ls`.

`man 2 stat` – Provides info on the system call `stat()`.

`man 3 printf` – Provides info on the C library function `printf()`.

System Call Manual

It is important to know that man by default first looks in section 1 and then in higher sections for the argument.

Some entries can be found in multiple sections. For example there is a `stat` program in section 1 and a `stat()` system call in section 2.

```
man 1 stat
man 2 stat
```