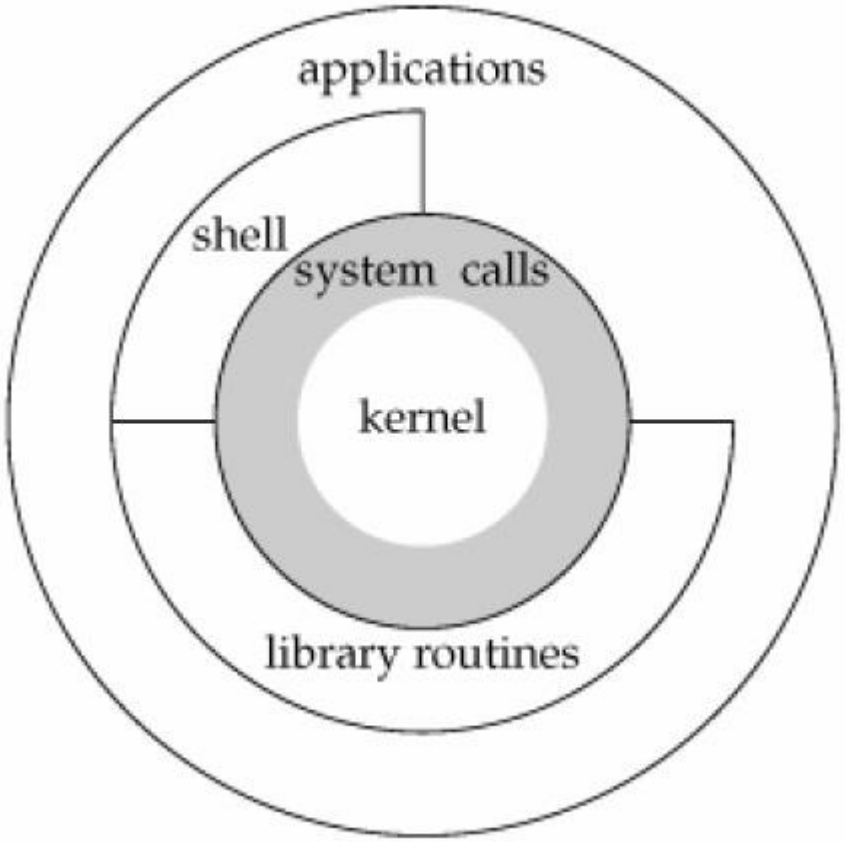
Unix System Calls

An Advanced Introduction to Unix/C Programming

John Dempsey
COMP-232 Programming Languages
California State University, Channel Islands

Unix Kernel & System Calls



Unix System Calls

- Unix System Calls provide an interface into the Unix Kernel.
- Application programs and library calls can call system calls.
- To learn more and view a list of system calls, you can type:

    % **man syscalls**

System Calls

- To support Unix/Linux on a new hardware architecture, you would need to support system calls.
- There are hundreds of system calls.
- As such, we'll highlight the system calls you are more likely to use.
- To learn how to use the system call, use man:

% man stat.2

(You can copy the example code in stat for your programming assignment.)

System Calls

Function	Function Prototype	Description
access	int access(const char *pathname, int mode);	Checks whether the calling process can access the file pathname. mode can be R_OK, W_OK, X_OK.
alarm	unsigned int alarm(unsigned int seconds);	Set an alarm clock for delivery of a SIGALRM signal in seconds seconds. alarm(0) turns off the alarm.
chdir	int chdir(const char *path);	Changes the current working directory to path.
chmod	int chmod(const char *pathname, mode_t mode);	Changes the file's mode bits to specified mode.
chown	int chown(const char *pathname, uid_t owner, gid_t group);	Changes the owner and group of a file.
chroot	int chroot(const char *path);	Changes the root directory of the calling process to that specified in path.
close	int close(int fd);	Closes a file descriptor fd.
exit	void _exit(int status);	Terminates the calling process.

System Calls

Function	Function Prototype	Description
flock	int flock(int fd, int operating);	Apply or remove an advisory lock on an open file specified by fd.
fork	pid_t fork(void);	Creates a new child process by duplicating the calling process. The child process returns a 0 pid.
fstat	int fstat(int fd, struct stat *statbuf);	Returns information about a file based on the file descriptor fd.
fsync	int fsync(int fd);	Flushes all modified buffer memory pages for file referenced by file descriptor fd.
getcwd	char *getcwd(char *buf, size_t size);	Copies pathname of the current working directory into buf of length size.
getdomainname	int getdomainname(char *name, size_t len);	Returns the domain name in the character array name of size len.
gethostname	int gethostname(char *name, size_t len);	Returns hostname in the character array name of length len.

System Calls

Function	Function Prototype	Description
getpid	pid_t getpid(void);	Get process id (PID) of the calling process.
getppid	pid_t getppid(void);	Get process id of the parent of the calling process.
getsid	pid_t getsid(pid_t pid);	Returns session id of the process with process id pid. If pid=0, returns session id of calling process.
gettimeofday	int gettimeofday(struct timeval *tv, struct timezone *tz);	Returns the number of seconds since Epoch and timezone.
getuid	uid_t getuid(void);	Returns the real user id of calling process.
geteuid	uid_t geteuid(void);	Returns the effective user of calling process.
kill	int kill(pid_t pid, int sig);	Sends any signal sig to process pid.
lseek	off_t lseek(int fd, off_t offset, int whence);	Repositions the file offset of fd to the offset based on whence, e.g., SEEK_SET, SEEK_CUR+offset, SEEK_END+offset.

System Calls

Function	Function Prototype	Description
mkdir	int mkdir(const char *pathname, mode_t mode);	Creates a directory named pathname.
mknod	int mknod(const char *pathname, mode_t mode, dev_t dev);	Creates a filesystem node (file, device special file, or named pipe) named pathname with mode and dev.
msgget	int msgget(key_t key, int msgflg);	Returns message queue id identified by key.
msgrcv	ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflag);	Receives a message from message queue msqid.
msgsnd	int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);	Sends a message to the message queue msqid.
open	int open(const char *pathname, int flags);	Open and possibly create a file.
pause	int pause(void);	Causes the calling process to sleep until a signal is delivered.
pipe	int pipe(int pipefd);	Create pipe.

System Calls

Function	Function Prototype	Description
read	ssize_t read(int fd, void *buf, size_t count);	Read from a file descriptor fd.
rmdir	int rmdir(const char *pathname);	Delete a directory.
shmat	void *shmat(int shmid, const void *shmaddr, in shmflg);	Attach to shared memory segment identified by shmid.
shmctl	int shmctl(int shmid, int cmd, struct shmid_ds *buf);	Performs cmd on shared memory segment specified by shmid.
shmdt	int shmdt(const void *shmaddr);	Detaches from shared memory located at shmaddr.
shmget	int shmget(key_t key, size_t size, int shmflg);	Allocates shared memory segment based on key.
signal	sighandler_t signal(int signum, sighandler_t handler);	Calls function handler when signal signum is generated.
stat	int stat(const char *pathname, struct stat *statbuf);	Get file status for pathname.

System Calls

Function	Function Prototype	Description
sysinfo	int sysinfo(struct sysinfo *info);	Returns system information on uptime, load averages, memory usage, swap space, processes.
time	time_t time(time_t *tloc);	Get time in seconds.
unlink	int unlink(const char *pathname);	Delete pathname.
write	ssize_t write(int fd, const void *buf, size_t count);	Write to a file descriptor fd.

strace (truss) – Shows system calls used

```
john@oho:~$ cat hello.c
#include <stdio.h>

int main()
{
        printf("Hello World!\n");
}

john@oho:~$ gcc hello.c -o hello

john@oho:~$ hello
Hello World!
```

```
john@oho:~$ strace hello
execve("./hello", ["hello"], 0x7fffc0fa2c30 /* 20 vars */) = 0
brk(NULL)                               = 0x7fffd3c3e000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffdc2df360) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=33844, ...}) = 0
mmap(NULL, 33844, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2d21297000
... and on and on ...
fstat(1, {st_mode=S_IFCHR|0660, st_rdev=makedev(0x4, 0x1), ...}) = 0
ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
brk(NULL)                               = 0x7fffd3c3e000
brk(0x7fffd3c5f000)                     = 0x7fffd3c5f000
write(1, "Hello World!\n", 13Hello World!)        = 13
exit_group(0)                           = ?
+++ exited with 0 +++
```

strings

```
john@oho:~$ cat hello.c
#include <stdio.h>
char *COPYRIGHT="(C) Copyright 2022 John Dempsey - All Rights Reserved.";
int main() {
	printf("Hello World!\n");
}
john@oho:~$ gcc hello.c -o hello
john@oho:~$ strings hello | head -20
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
__cxa_finalize
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
(C) Copyright 2022 John Dempsey - All Rights Reserved.
Hello World!
:*3$"
GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
crtstuff.c
```

strip

```
% gcc -g stocks.c -o stocks
%ls -l stocks
-rwxr-xr-x 1 root root 25792 Mar  1 22:18 stocks

% gcc stocks.c -o stocks
% ls -l stocks
-rwxr-xr-x 1 root root 21416 Mar  1 22:17 stocks

% strip stocks
%ls -l stocks
-rwxr-xr-x 1 root root 18568 Mar  1 22:17 stocks

% gcc -O4 stocks.c -o stocks
%  ls -l stocks
-rwxr-xr-x 1 root root 17376 Mar  1 22:21 stocks

% strip stocks
% ls -l stocks
-rwxr-xr-x 1 root root 14472 Mar  1 22:22 stocks
```

alarm

```
john@oho:~/LEC$ cat alarm.c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>

void timeout_reached(int sig)
{
  printf("Timeout reached. Signal number=%d\n", sig);
}

int main()
{
    signal(SIGALRM, timeout_reached);

    printf("Test case 1: Sleep 3 seconds.\n");
    alarm(5);
    printf("Do something 1 ...\n");
    sleep(3);
    alarm(0);

    printf("Test case 2: Sleep 20 seconds.\n");
    alarm(5);
    printf("Do something 2 ...\n");
    sleep(20);
    alarm(0);

    printf("All done!\n");
}
```

```
john@oho:~/LEC$ gcc alarm.c; a.out
Test case 1: Sleep 3 seconds.
Do something 1 ...
Test case 2: Sleep 20 seconds.
Do something 2 ...
Timeout reached. Signal number=14
All done!
```

access, chdir, getcwd, mkdir, rmdir Example

```
john@oho:~/LEC$ cat sys.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>

int main()
{
    char  command[150];
    char  directory[50];
    int   i;
    getcwd(directory, 50);
    strcat(directory, "/test.dir");
    if ((i = access(directory, R_OK & W_OK & X_OK)) == 0) {
      printf("Directory %s exists already.\n", directory);
      printf("Let's remove it and run mkdir again.\n");
      rmdir(directory);
    }
    mkdir(directory, 0755);
    chdir(directory);
    getcwd(directory, 50);
    printf("Current directory is now %s.\n", directory);
    sprintf(command, "ls -ld %s; ls -lR %s", directory, directory);
    system(command);
}
```

```
john@oho:~/LEC$ ls
a.out  alarm.c  sys.c

john@oho:~/LEC$ gcc sys.c; a.out
Current directory is now /home/john/LEC/test.dir.
drwxr-xr-x 1 john john 4096 Sep 27 23:16 /home/john/LEC/test.dir
/home/john/LEC/test.dir:
total 0

john@oho:~/LEC$ ls
a.out  alarm.c  sys.c  test.dir

john@oho:~/LEC$ a.out
Directory /home/john/LEC/test.dir exists already.
Let's remove it and run mkdir again.
Current directory is now /home/john/LEC/test.dir.
drwxr-xr-x 1 john john 4096 Sep 27 23:16 /home/john/LEC/test.dir
/home/john/LEC/test.dir:
total 0

john@oho:~/LEC$
```