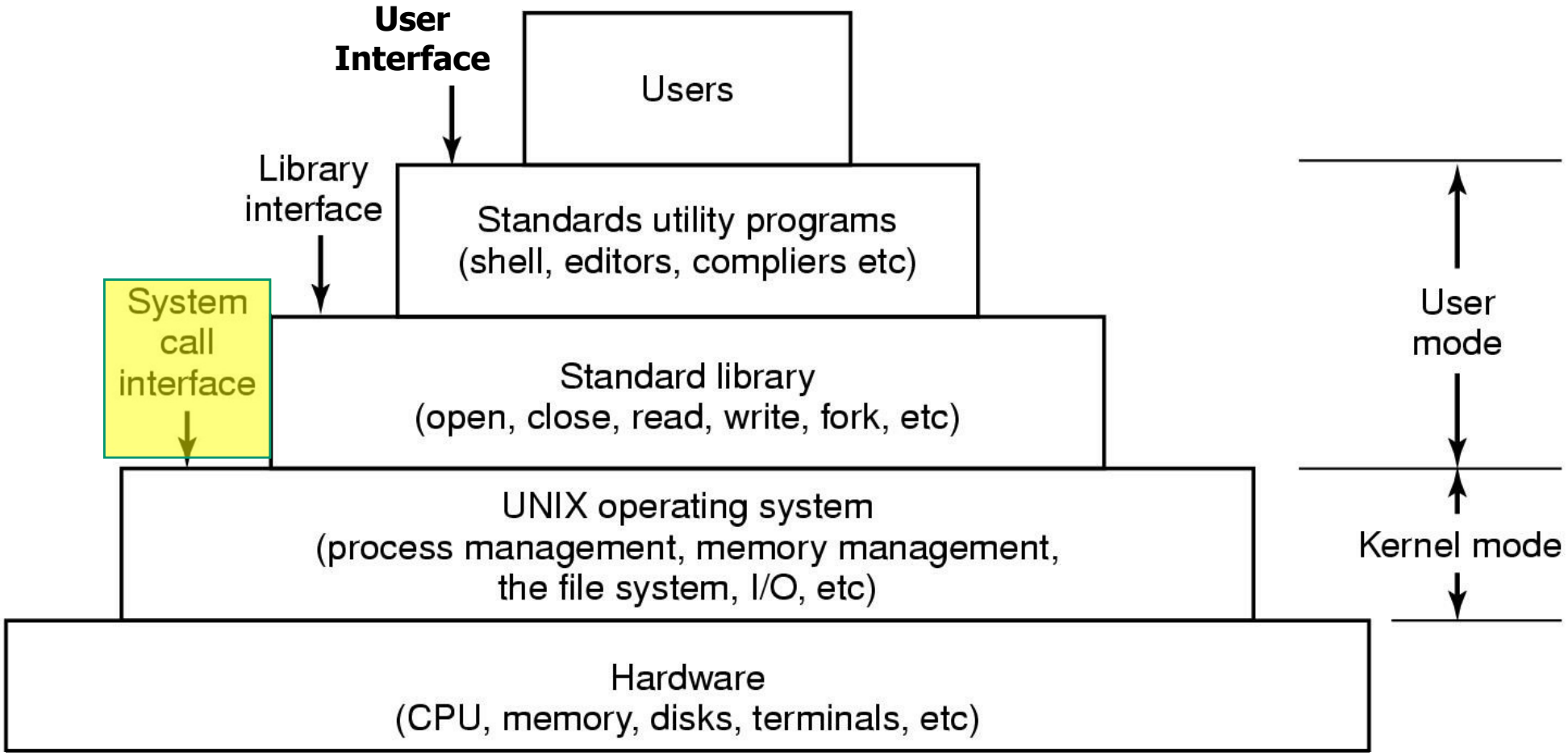
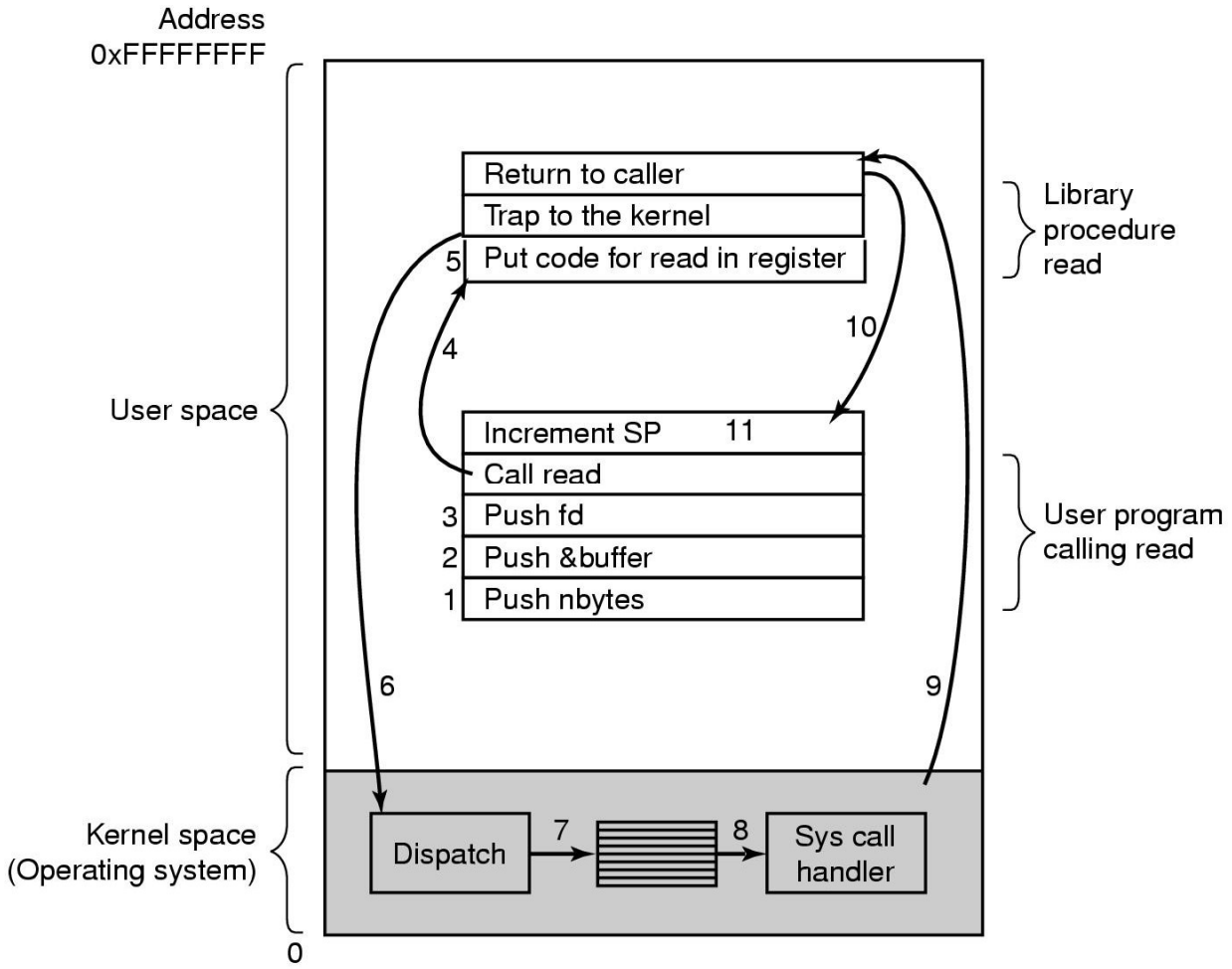
UNIX


User Interface
Users
Library interface
Standards utility programs
(shell, editors, compliers etc)
System call interface
Standard library
(open, close, read, write, fork, etc)
UNIX operating system
(process management, memory management,
the file system, I/O, etc)
Hardware
(CPU, memory, disks, terminals, etc)
User mode
Kernel mode

Steps in Making a System Call



There are 11 steps in making the system call read (fd, buffer, nbytes)

System Calls for Process Management

System call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, opts)	Wait for a child to terminate
s = execve(name, argv, envp)	Replace a process' core image
exit(status)	Terminate process execution and return status
s = sigaction(sig, &act, &oldact)	Define action to take on signals
s = sigreturn(&context)	Return from a signal
s = sigprocmask(how, &set, &old)	Examine or change the signal mask
s = sigpending(set)	Get the set of blocked signals
s = sigsuspend(sigmask)	Replace the signal mask and suspend the process
s = kill(pid, sig)	Send a signal to a process
residual = alarm(seconds)	Set the alarm clock
s = pause( )	Suspend the caller until the next signal

s is an error code

pid is a process ID

residual is the remaining time from the previous alarm

POSIX

Signal	Cause
SIGABRT	Sent to abort a process and force a core dump
SIGALRM	The alarm clock has gone off
SIGFPE	A floating-point error has occurred (e.g., division by 0)
SIGHUP	The phone line the process was using has been hung up
SIGILL	The user has hit the DEL key to interrupt the process
SIGQUIT	The user has hit the key requesting a core dump
SIGKILL	Sent to kill a process (cannot be caught or ignored)
SIGPIPE	The process has written to a pipe which has no readers
SIGSEGV	The process has referenced an invalid memory address
SIGTERM	Used to request that a process terminate gracefully
SIGUSR1	Available for application-defined purposes
SIGUSR2	Available for application-defined purposes

System Calls for File Management

System call	Description
fd = creat(name, mode)	One way to create a new file
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information
s = fstat(fd, &buf)	Get a file's status information
s = pipe(&fd[0])	Create a pipe
s = fcntl(fd, cmd, ...)	File locking and other operations

- **s** is an error code
- **fd** is a file descriptor
- **position** is a file offset

The lstat System Call

Device the file is on
I-node number (which file on the device)
File mode (includes protection information)
Number of links to the file
Identity of the file's owner
Group the file belongs to
File size (in bytes)
Creation time
Time of last access
Time of last modification

Fields returned by the lstat system call.

System Calls for Directory Management

System call	Description
s = mkdir(path, mode)	Create a new directory
s = rmdir(path)	Remove a directory
s = link(oldpath, newpath)	Create a link to an existing file
s = unlink(path)	Unlink a file
s = chdir(path)	Change the working directory
dir = opendir(path)	Open a directory for reading
s = closedir(dir)	Close a directory
dirent = readdir(dir)	Read one directory entry
rewinddir(dir)	Rewind a directory so it can be reread

(also mount/umount)

- **s** is an error code
- **dir** identifies a directory stream
- **dirent** is a directory entry

System Calls for File Protection

System call	Description
s = chmod(path, mode)	Change a file's protection mode
s = access(path, mode)	Check access using the real UID and GID
uid = getuid( )	Get the real UID
uid = geteuid( )	Get the effective UID
gid = getgid( )	Get the real GID
gid = getegid( )	Get the effective GID
s = chown(path, owner, group)	Change owner and group
s = setuid(uid)	Set the UID
s = setgid(gid)	Set the GID

- **s** is an error code
- **uid** and **gid** are the UID and GID, respectively

System Calls for Memory Management

System call	Description
s = brk(addr)	Change data segment size
a = mmap(addr, len, prot, flags, fd, offset)	Map a file in
s = unmap(addr, len)	Unmap a file

- **s** is an error code
- **b** and **addr** are memory addresses
- **len** is a length
- **prot** controls protection
- **flags** are miscellaneous bits
- **fd** is a file descriptor
- **offset** is a file offset

Some System Calls For Miscellaneous Tasks

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

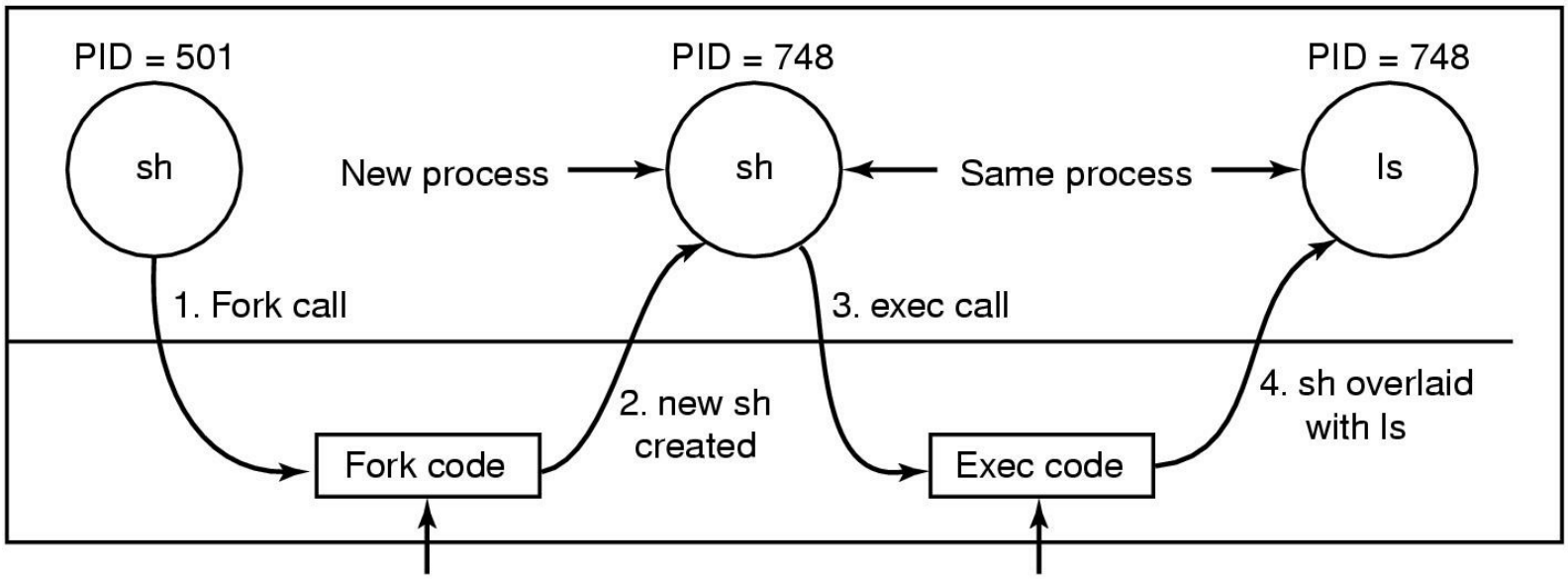
System Calls (1)

- A stripped down shell:

```
while (TRUE) {                                  /* repeat forever */
    type_prompt( );                             /* display prompt */
    read_command (command, parameters)          /* input from terminal */

if (fork() != 0) {                              /* fork off child process */
    /* Parent code */
    waitpid( -1, &status, 0);                   /* wait for child to exit */
} else {
    /* Child code */
    execve (command, parameters, 0);            /* execute command */
 }
}
```

The *ls* Command



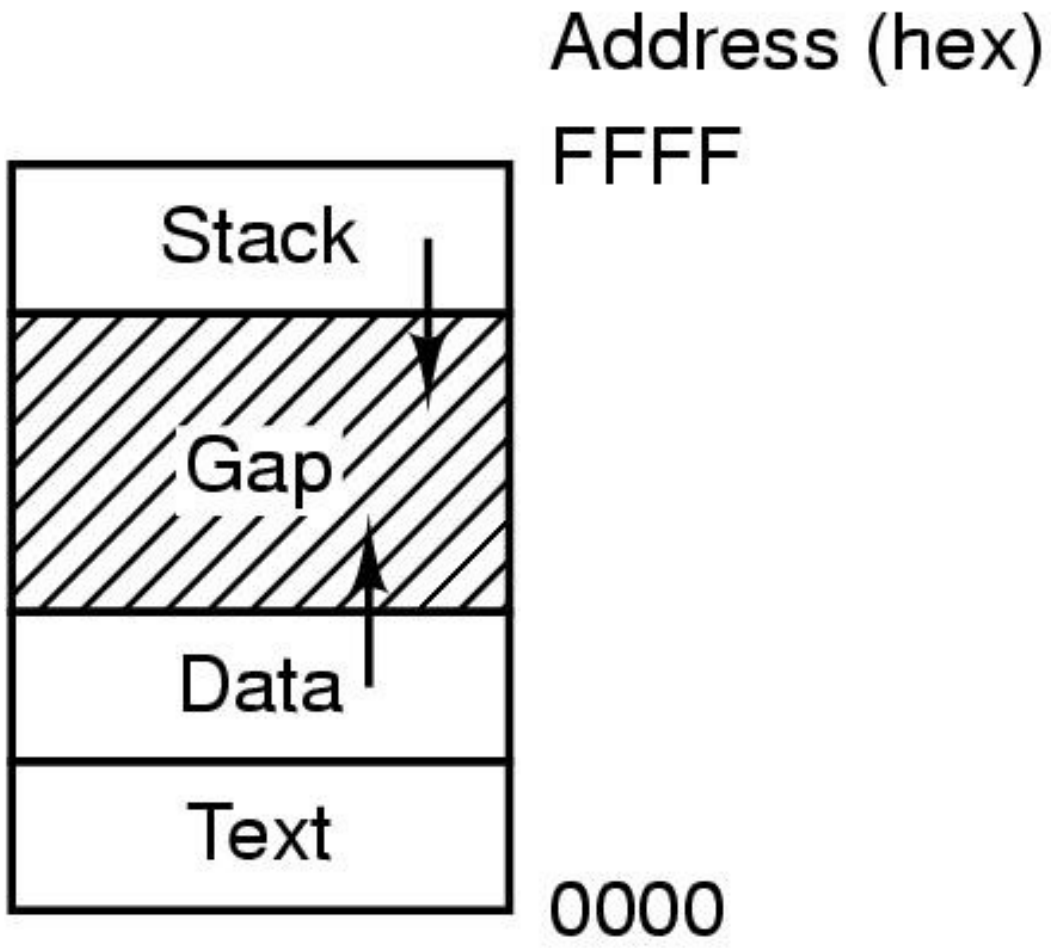
Steps in executing the command *ls* type to the shell

fork vs clone: Clone Flags

Flag	Meaning when set	Meaning when cleared
CLONE_VM	Create a new thread	Create a new process
CLONE_FS	Share umask, root, and working dirs	Do not share them
CLONE_FILES	Share the file descriptors	Copy the file descriptors
CLONE_SIGHAND	Share the signal handler table	Copy the table
CLONE_PID	New thread gets old PID	New thread gets own PID

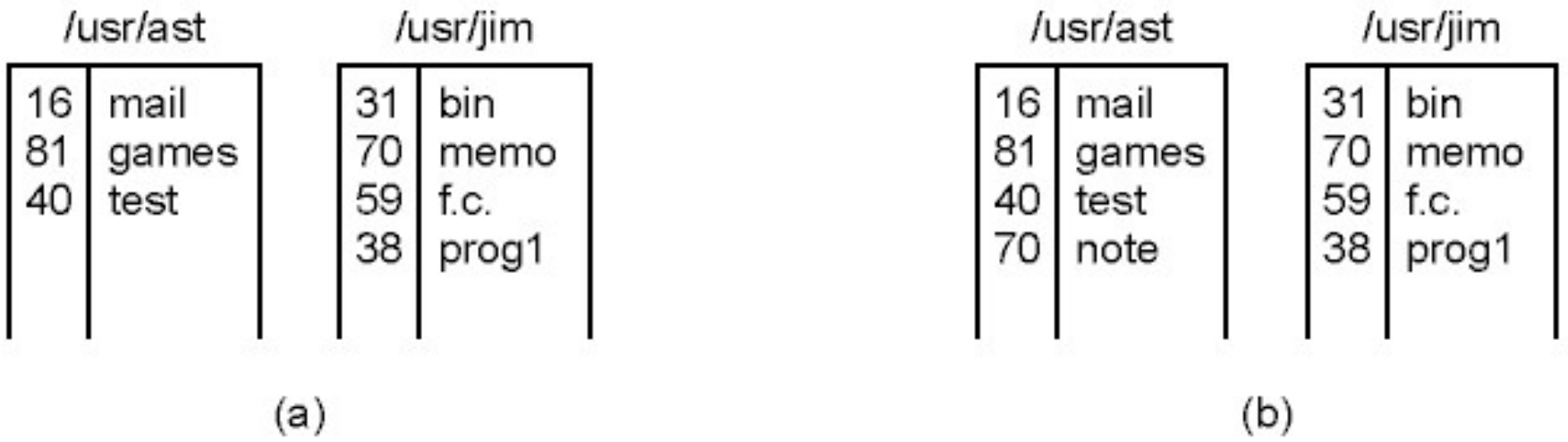
Bits in the sharing_flags bitmap

System Calls (2)



- Processes have three segments: text, data, stack

System Calls (3)



(a) Two directories before linking *usr/jim/memo* to ast's directory

(b) The same directories after linking

System Calls (4)

(a) File system before the mount

(b) File system after the mount

Unix/Win32 System Calls (5)

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time