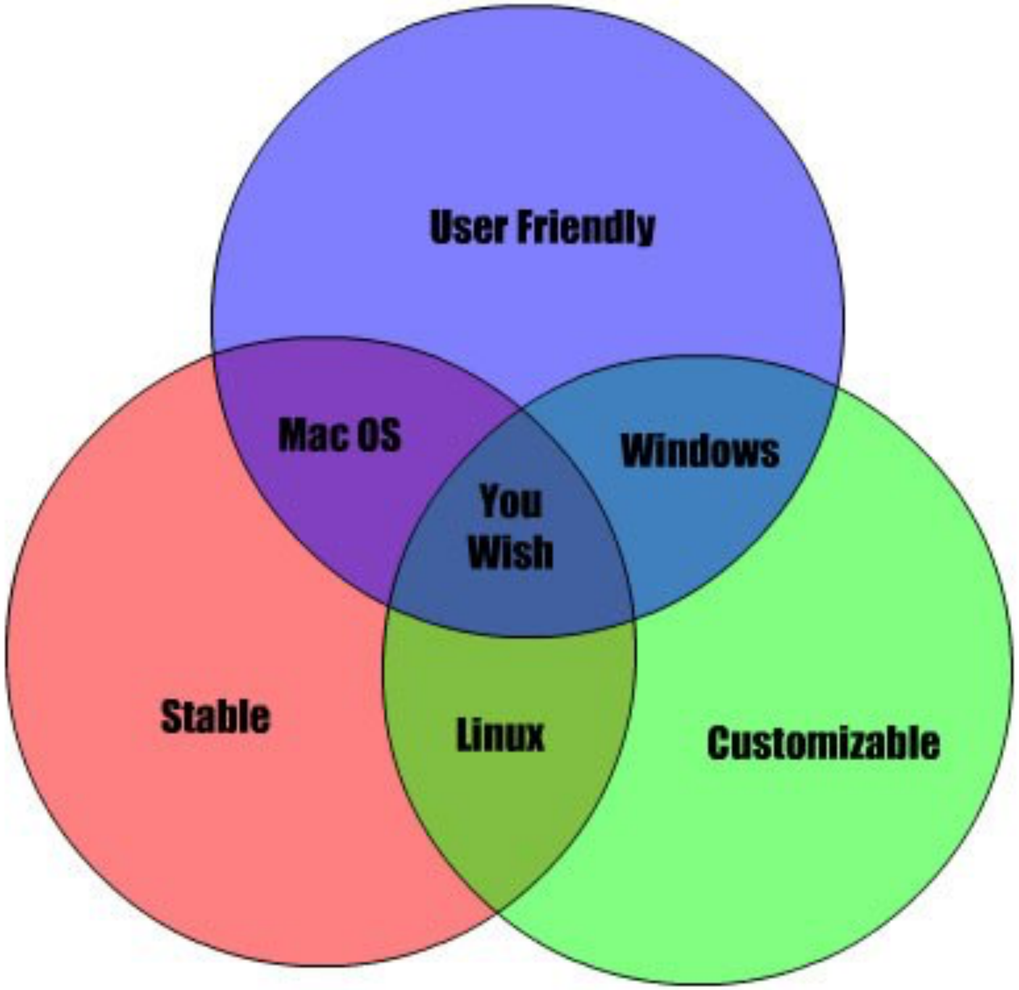
CS 492: Operating Systems

System Calls

Instructor: Iraklis Tsekourakis

Email: itsekour@stevens.edu

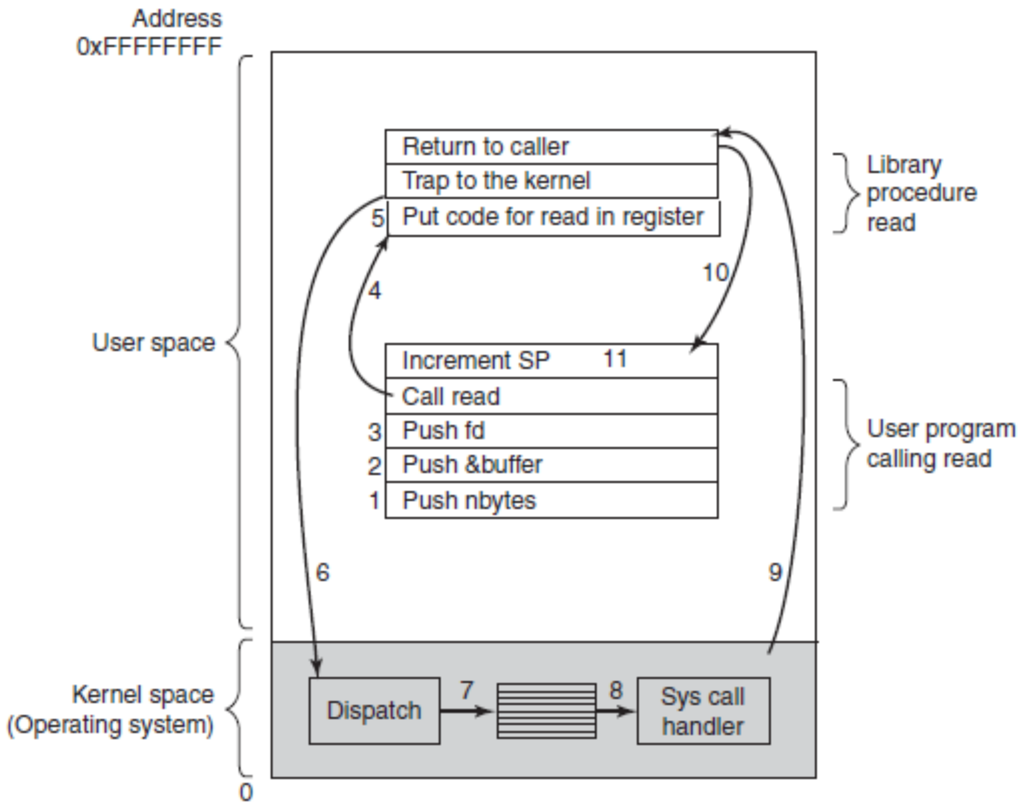
Operating Systems



Today’s lecture

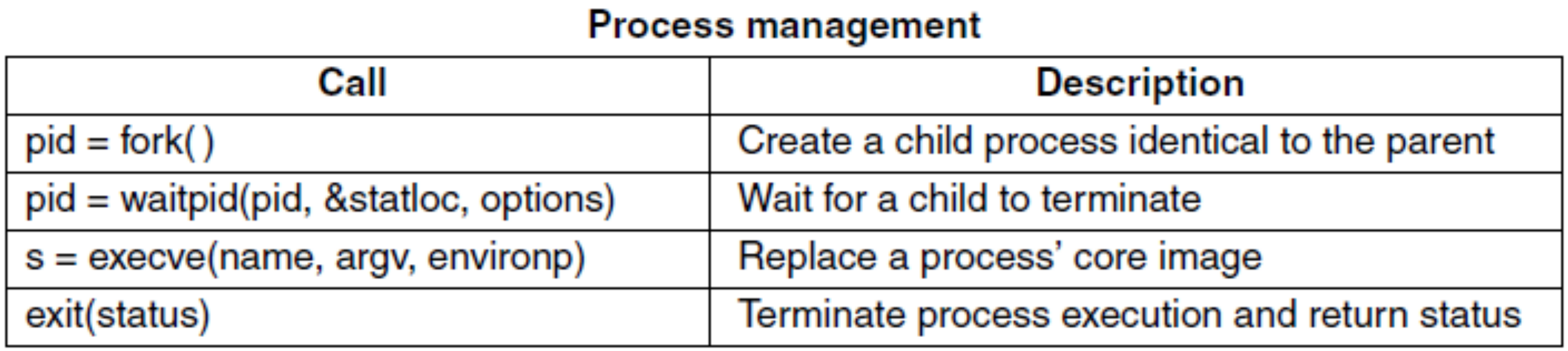
- **Process-to-OS communication** - Explain in detail how a system calls is made and returns, and describe the main UNIX system calls

System Calls (1)



The 11 steps in making the system call *read(fd, buffer, nbytes).*

System Calls (2)

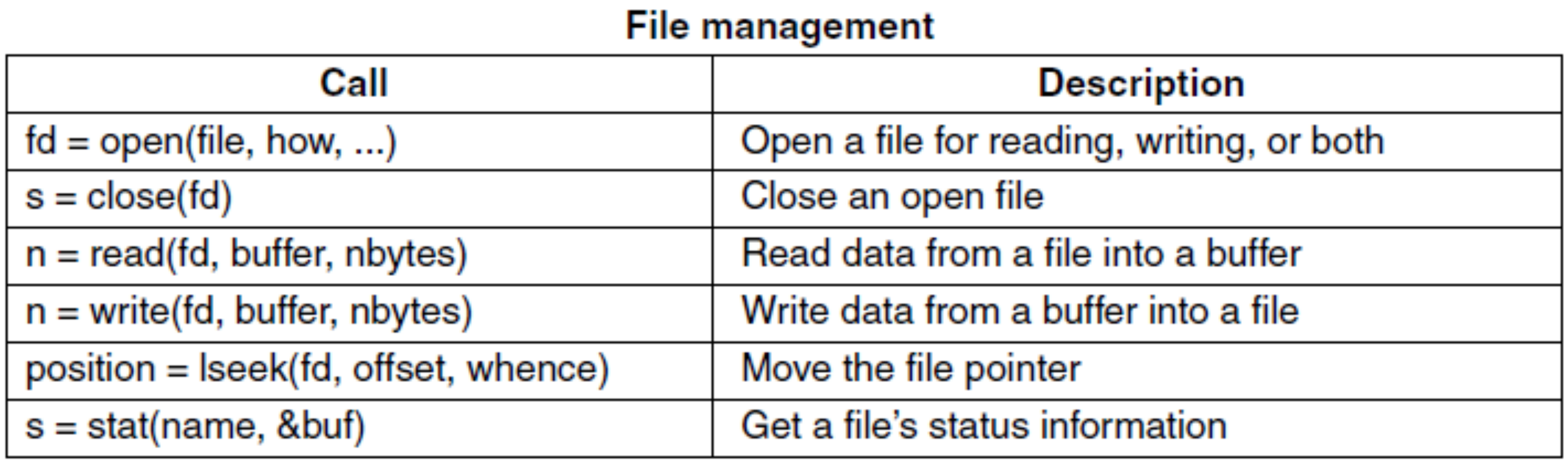


Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

Some of the major POSIX system calls. The return code *s* is −1 if an error has occurred. The return codes are as follows: *pid* is a process id

System Calls (3)

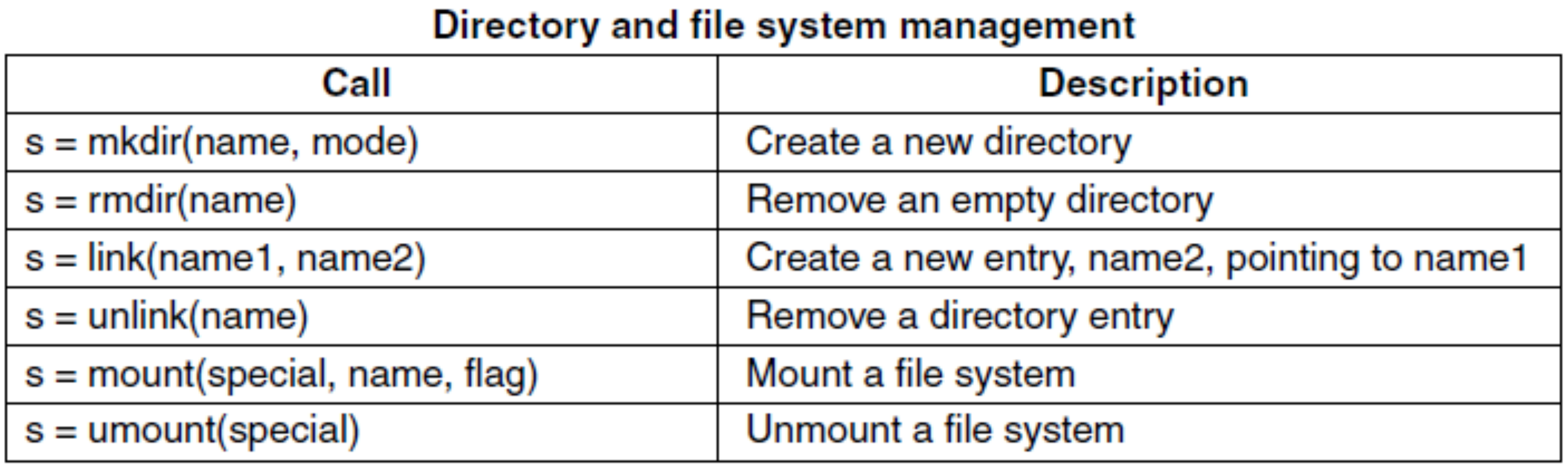


File management

Call	Description
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

Some of the major POSIX system calls. The return code *s* is −1 if an error has occurred. The return codes are as follows: *fd* is a file descriptor, *n* is a byte count, and *position* is an offset within the file.

System Calls (4)

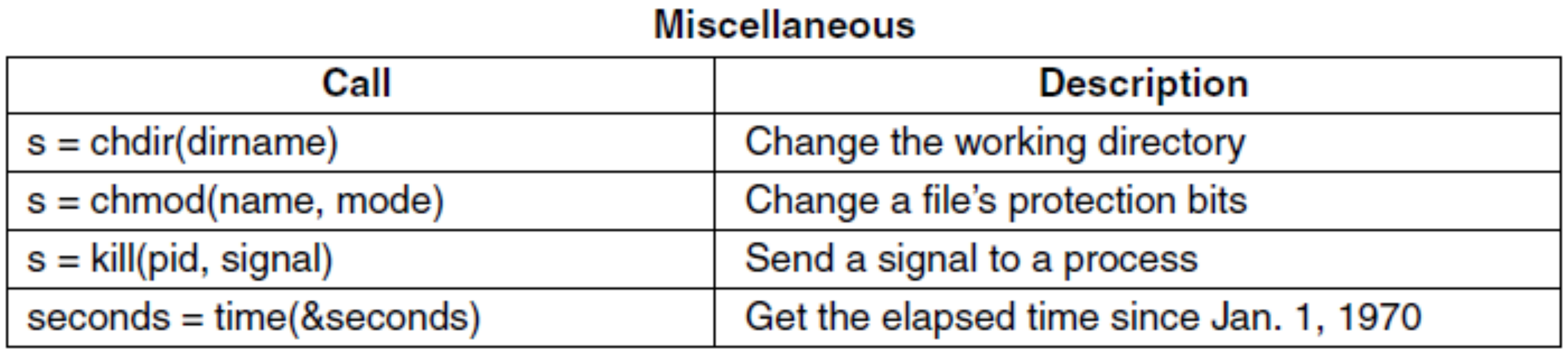


Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Some of the major POSIX system calls. The return code *s* is −1 if an error has occurred.

System Calls (5)



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

Some of the major POSIX system calls. The return code *s* is −1 if an error has occurred. The return codes are as follows: *seconds* is the elapsed time.

System Calls for Process Management

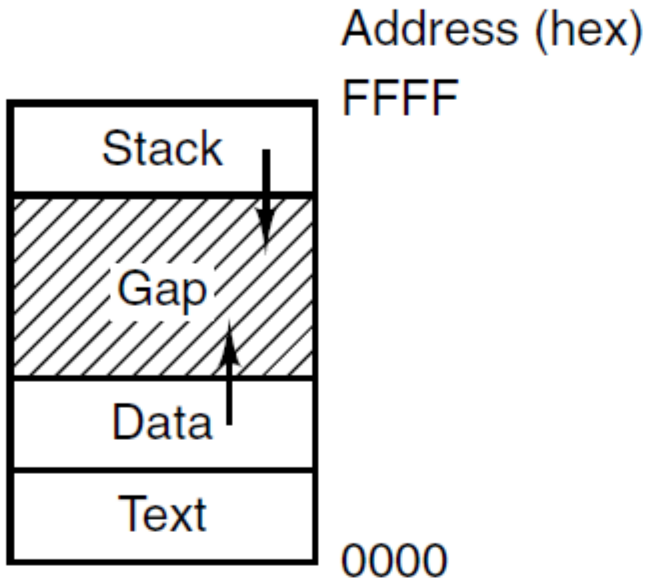
```
#define TRUE 1

while (TRUE) {                                  /* repeat forever */
      type_prompt( );                           /* display prompt on the screen */
      read_command(command, parameters);        /* read 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 */
      }
}
```

A stripped-down shell. *TRUE* is assumed to be defined as 1.

Question

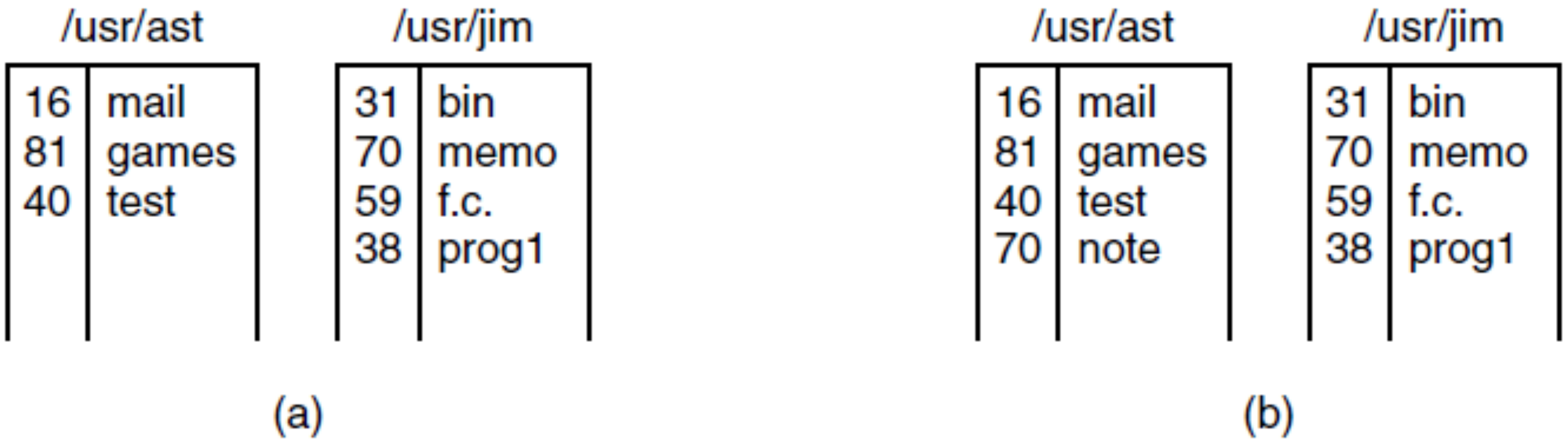
To a programmer, a system call looks like any other call to a library procedure. Is it important that a programmer know which library procedures result in system calls? Under what circumstances and why?

System Calls for File Management



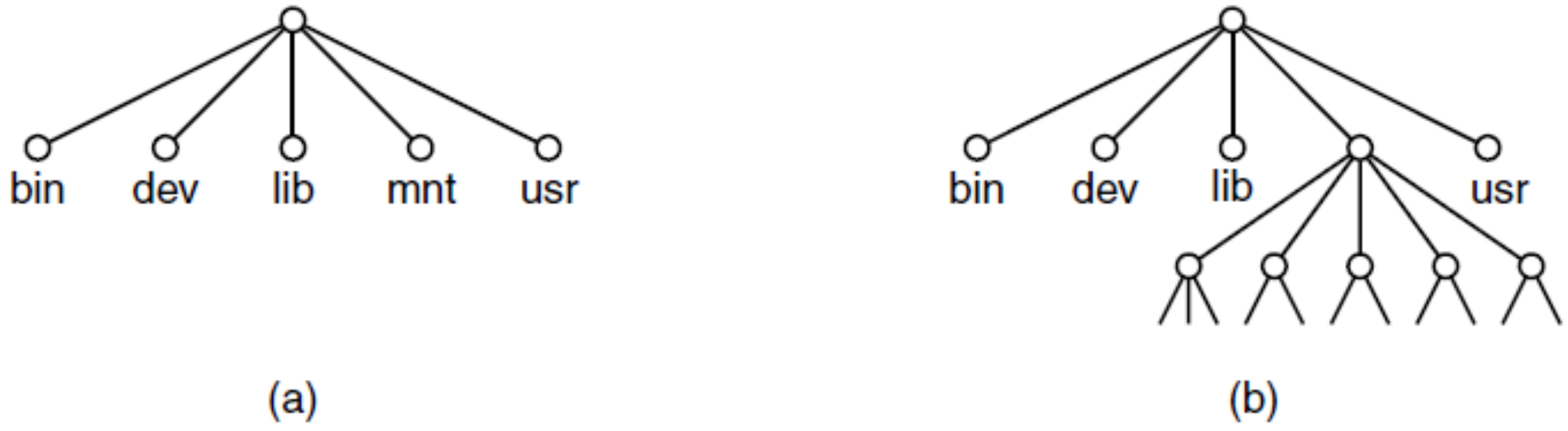
Processes have three segments:
text, data, and stack

System Calls for Directory Management (1)



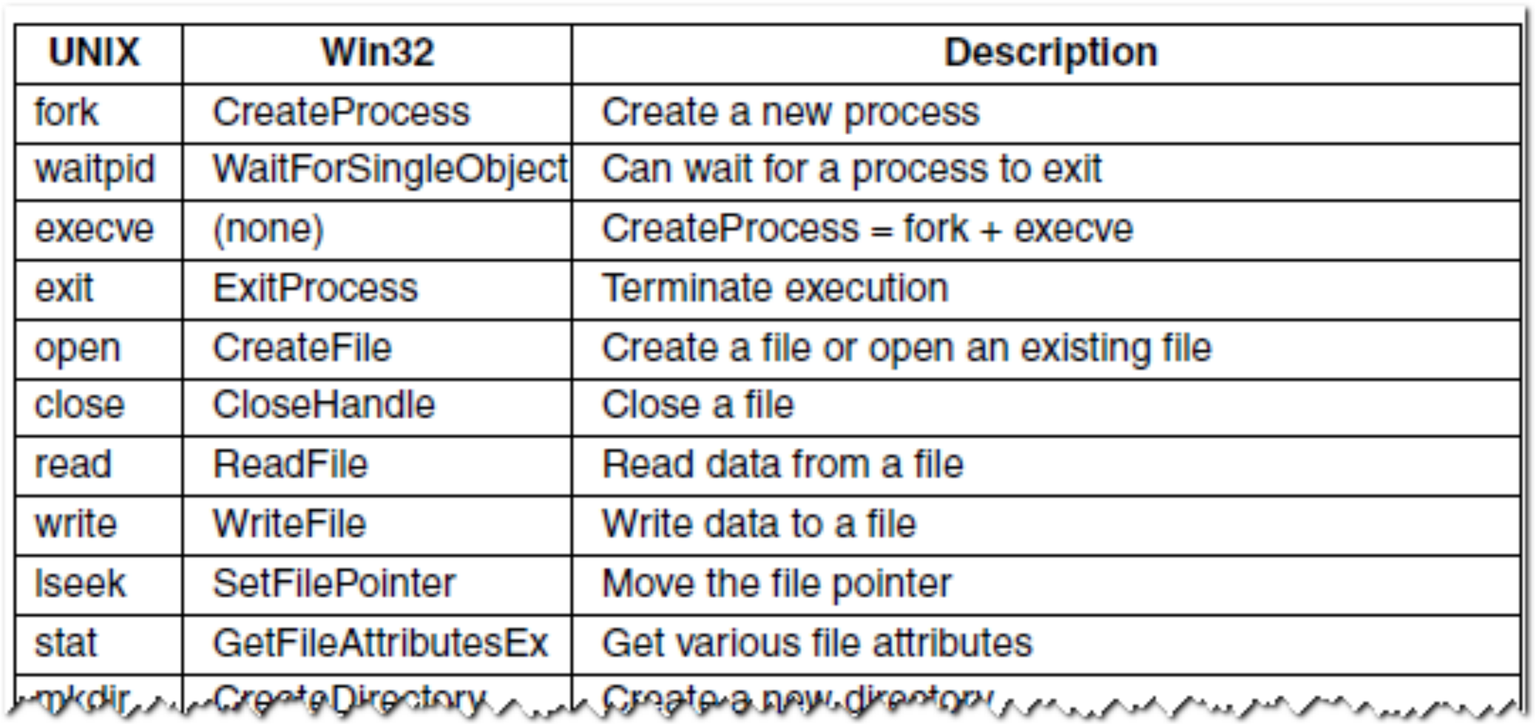
(a) Two directories before linking *usr/jim/memo* to *ast*'s directory. (b) The same directories after linking.

System Calls for Directory Management (2)



(a) File system before the mount.
(b) File system after the mount.

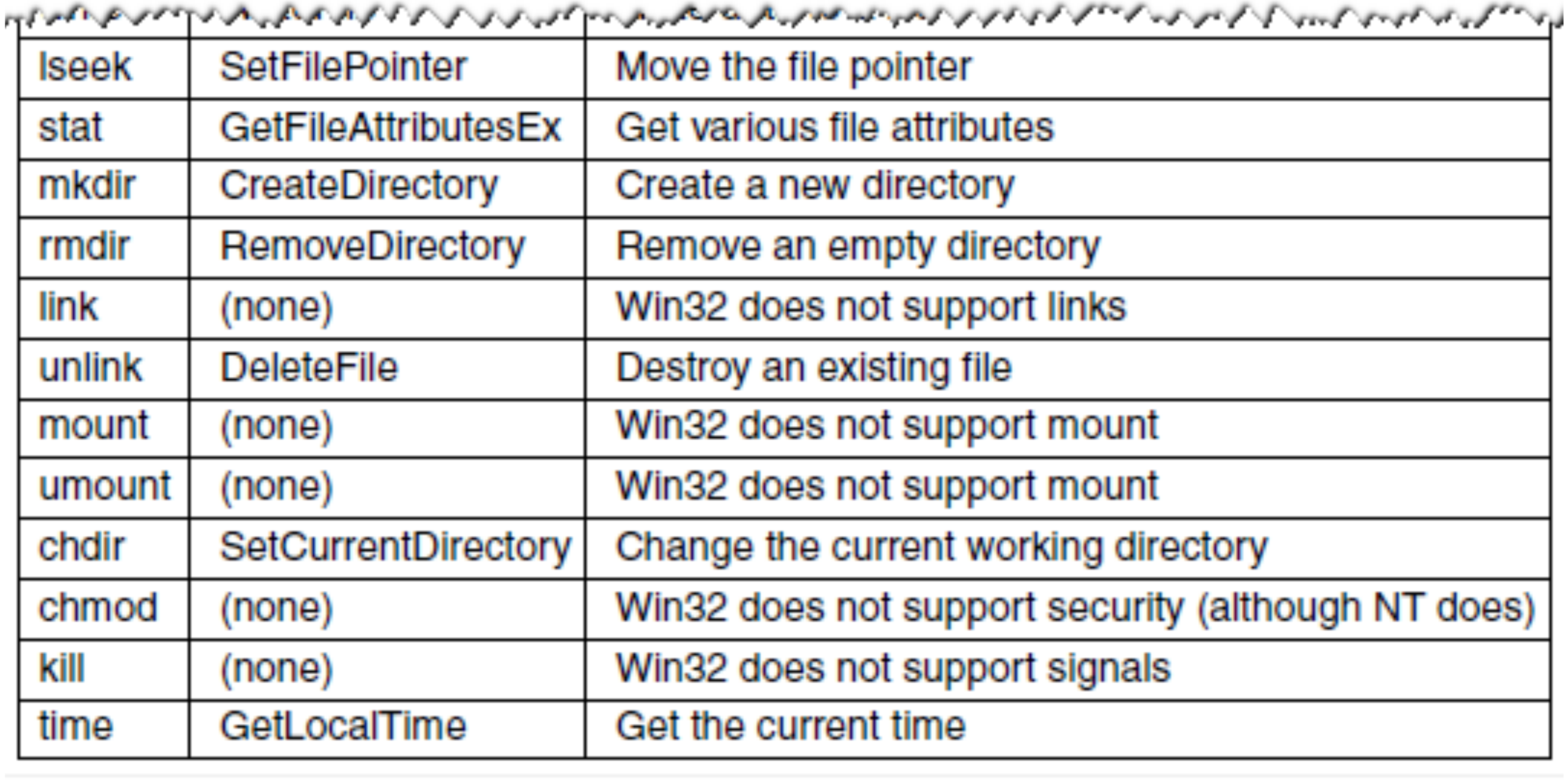
The Windows Win32 API (1)



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

The Win32 API calls that roughly correspond to the UNIX calls

The Windows Win32 API (2)



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

The Win32 API calls that roughly correspond to the UNIX calls