System Calls for File Management

Call	Description
fd = open(name,how)	Open a file for reading and/or writing
s = close(fd)	Close an open file
n = read(fd,buffer,size)	Read data from a file into a buffer
n = write(fd,buffer,size)	Write data from a buffer into a file
s = lseek(fd,offset,whence)	Move the "current" pointer for a file
s = stat(name,&buffer)	Get a file's status information (in *buffer*)
s = mkdir(name,mode)	Create a new directory
s = rmdir(name)	Remove a directory (must be empty)
s = link(name1,name2)	Create a new entry (*name2*) that points to the same object as *name1*
s = unlink(name)	Remove *name* as a link to an object (deletes the object if *name* was the only link to it)

System Calls for 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
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 1 Jan 1970

```
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 */
 }
}
```

- Fork returns a value (pid)
  - Zero in child
  - Child's PID in parent
  - Used to differentiate child from parent