四类系统调用

# 进程管理

## pid = fork()

创建一个与父进程相同的子进程（精确副本，包括描述符、寄存器等内容）；

唯一可以在 POSIX 中创建进程的途径；

在父进程中返回进程标识符，在子进程中返回0，创建失败返回负数

## pid = waitpid(pid, &statloc, options)

等待pid子进程结束，将进程状态写入 stateloc，有可选的 options 选项。

若传入的 pid 为 -1，则返回任意一个第一个结束的子进程的 pid.

## s = execve(name, argv, envp)

执行 name 字符串指向的文件，以 argv 为命令行参数表，以 envp 为环境变量参数表；

引起整个核心映像被该文件所替代。

进程内存

正文段（如程序代码）

数据段（变量等）

堆栈段

堆区向上增长，栈区向上增长。

malloc 内部实现没有被标准化。数据段增长的系统调用是 brk，但不属于 POSIX 标准。

# 文件管理

open 系统调用，调用方式由第二个参数指定。

O\_RDONLY 只读

O\_WRONLY 只写

O\_RDWR 读写

O\_CREATE 创建文件

返回文件描述符

lseek 设置一个打开文件的描述符的偏移值。

fstat 获取文件信息。Linux 系统保存文件的最后修改时间、类型、权限、大小等信息。

## 目录和文件系统管理

## link

创建文件的链接

每个文件由一个文件描述符(fildes)，称之为 i-编号，与 i-节点表格（文件的属性表格）一一对应。一个目录就是 (i-编号，文件名）对表。

link 用于在目录中添加新对

unlink 则用于移除该对。

当一个文件描述符不存在所有目录中时，UNIX 00 会从磁盘中移除该文件

## mount

将两个文件系统合并为一个。

mount(source, dest, type)

type 指示只读还是可读写

## mkdir

创建空目录

rmdir 用于移除空目录

# 其他系统调用

## chdir

改变当前工作目录

chdir(path)

## chmod

chmod(file, mode)

改变文件模式

# 抵制Windows系统

## Win32 API

微软定义的一套过程，程序员通过这些过程获得操作系统的服务。

并非所有过程都是系统调用

多达上千个过程

不同的版本之间，是否为系统调用也使不确定的

微软保证以后会使接口稳定（。。。）

## 不同处

Windows 应用程序采用时间驱动的设计风格，而Unix 采用各种各样的系统调用。

Windows 不存在文件的链接（有链接文件的概念吧）、不存在文件系统的安装，不支持kill 信号，不支持文件模式（NT支持）