

API.

源代码层的接口.

ABI

二进制接口.

通过工具链: 编译器, 连接器.

~~POSIX~~

POSIX 和 SUS 系统接口标准 UNIX

C语言标准. ANSI 的 ISO C...

gcc C语言

g++ C++语言.

LSB Linux 标准规范

一切皆文件.

fd 文件描述符 int

文件编号

支持多个进程共享同一个 fd, 没有限制文件的并发访问

索引节点. inode 数值唯一 与文件关联

目录: 可读如索引 inode 的映射.

链接 link.

目录也有索引节点 ✓

对目录的操作: 添加链接, 删除链接.

硬链接: 多个名称映射到一个 inode.

符号链接: 快捷方式. (没有自己的 inode 和数据)



特殊文件 (以文件表示的内核对象)

块设备文件 作为字节数组访问 (硬盘, 等 随机访问)

字符设备文件 线性字节序列如 keyboard EOT

命名管道 FIFO 的 IPC (在文件中) 普通管道在内存中

UNIX 域套接字 使用 sockets 文件交换

文件命名空间 (统一)

挂载, 删除 (挂载卸载)

挂载点: 挂载到命名空间的特定位置

分区 (分成多个文件系统)

sector 扇区

block 块

page 页

进程

ELF 可执行可链接文件 (可运行的代码)

文本段

数据段

bss 段

线程有存储局部变量的栈

pid

进程树 根: init init 为 init 的继承

进程终止并不被删除, 等待父进程查账状态才删除



用户和组.

UID PID有UID关联 (通过uid)

root uid = 0

effective uid. saved uid. fs uid.
每个用户属于一个或多个组:
gid

权限.

文件权限

所有者/组/权限 (读, 写, 执行) 其他人的权限
三个 3个位. 共 9位. 保存在 mode 中

rwX 者组其他

信号.

单向异步通知机制.

数值常量 + 文本名.

IPC.

管道, 命名管道, 信号量, 消息队列, 共享内存, 快速用户空间互斥
futex

文件.

错误处理.

1. 特殊变量 errno 在 <errno.h> 中.

extern int errno;



<stdio.h> 中

perror()

strerror() strerror_r()

文件 I/O

文件表 (表)

fd. int

cookies 打开文件返回 fd

进程 至少 3 个 fd 0: stdin 1: stdout 2: stderr
链接控制: keyboard. screen.

可重定向

fd 可访问 设备文件, 管道, futexes 7170 socket
锁

打开文件

open (name, flag, mode)

读写

qrent

read()

0 - EOF

< len

返回多种结果



~~标准 I/O~~

标准 I/O 是 C 标准库提供的一个用户缓冲库

4.1. 分散/聚集 I/O

一次调用, 对多个缓冲区输入输出 到单流
(支持原子性) 或 向量 I/O (其他是线性 I/O)

`readv()` `writelv()`

`iovec` 结构体 描述一个缓冲区, (段) 组及指向量
count 较大: 内核动态分配较内存
较小: 在栈上创建一个段数组

4.2. Event Poll. (Linux 特有)

`poll` 和 `select` 需要遍历 fd fd 很多时性能差.

Event poll 三步: 1. 初始化 `epoll`, 2. 添加/删除 fd, 3. 事件等待.

1. `epoll_create()` 返回 fd. 监视完后要用 `close()` 关闭
2. `epoll_ctl()` 加入/删除 fd. `epoll_event` 结构体.
3. `epoll_wait()`. (timeout)

边缘触发, 条件触发.

边缘触发: 每满足条件的操作结束后返回.

条件触发: 一旦满足条件的操作一发生就返回 (默认)

4.3. 存储映射.

(进程地址空间)

将文件映射到内存中 (存到内存), 直接通过内存访问文件.

`mmap()`



页是内存管理单元(MMU)的粒度单位.

```
long page_size = sysconf(_SC_PAGESIZE); 更新  
(asm/pages.h) PAGE_SIZE.
```

信号: `fprintf`.

`mmap()` 取消 `mmap()` 映射.

~~`mmap()` 不会带来~~

`mremap()` 调整映射大小.

通过 `mremap`, `mmap` 等实现内存分配

`mprotect` 改变访问权限.

通过映射同步文件.

`msync`. (等同 `fsync`) 将映射写回磁盘.

`madvise`

预读、请求某块内容: 提前读取下一块

4.4. 普通文件 I/O 提示.

`posix_fadvise()` 和 `readahead()`

4.5. `synchronous` 读写操作结束后返回. `asynchronous`

`synchronized`. 读写操作时都是磁盘. `nonsynchronous`

`Linux` 支持 I/O.

4.6. I/O 调度器和 I/O 性能

`seek`, 磁头移动 8ms 以上

I/O 调度器, 通过控制 I/O 请求的顺序和时间点, 提高性能
(减少 `seek` 代价)



~~柱面~~ 磁头寻址:

柱面, 磁头, 扇区.

磁道 磁片.

三元组 \leftrightarrow

映射 (物理块/设备块)
块号 (LBA)

逻辑块寻址

文件系统是逻辑块: (物理块的整数倍)

I/O 调度:

合并: 多个相邻的 I/O 请求合为一个

排序, 对块号递增排序.

读请求: (必须返回最新数据).

若数据不在缓存中, 从磁盘读取会有延迟 (delay)

避免排序带来请求延迟

① Deadline I/O 调度.

三个队列

标准队列

读 I/O

写 I/O

排序

50ms

5s

② Anticipatory I/O 调度. (期望)

对读请求响应完成后等待 6ms 期望连续的读

③ CFS I/O (完全公平队列)

每个进程都排队. 采用时间片轮询

读请求优先级更高

④ Noop I/O

只合并, 不排序.

SSD 用 Noop 或 CFS

优化 I/O 性能.



用户空间 I/O 调度

对内核提交的 I/O 请求进行排序提交

1 按路径排序

stdin

2 按 inode 排序

stdout

3 按物理块排序

stderr

ioctl 获取物理块号

