# System Level I/O

李天驰 游震邦

2021/12/4

# Outline (Part I)

- 文件操作
  - open/close, read/write, seek…
  - stat
- 文件类型
  - 普通文件、目录文件……
- I/O包
  - UNIX I/O，标准I/O，RIO…

# Outline (Part II)

- 共享文件
  - 描述符表
  - 打开文件表
  - v-node表
- I/O重定向

# UNIX I/O – open/close

- Open
  - 路径名：绝对路径/相对路径
    - 为什么要用相对路径？方便、可移植
  - flags
    - 读写权限：只读、只写、可读可写
    - 写方式：create, trunc, append（在每次写操作前，都要设置文件位置到文件末尾）
  - mode
    - 对拥有者和其他人的权限，读/写/执行
    - umask

# UNIX I/O – open/close

- Close
  - 如果不close，会怎么样?
    - 内存泄漏、在stdio上有缓冲区会引起错误

# UNIX I/O – read/write

- 暴力复制
- 不足值:
  - 读时遇到EOF
  - 从终端读文本行
  - 读和写网络套接字

# UNIX I/O – 普通文件和目录文件

- 普通文件
  - open/close
  - read/write
  - 权限
    - 可读
    - 可写
    - 可执行
- 目录文件
  - opendir/closedir
  - readdir
  - 权限:
    - 可读
    - 可写
    - 可执行(cd)

# UNIX I/O - stat

- stat/fstat
  - 头部数据和文件内容
  - 依赖平台，所以stdio中没有

# UNIX I/O vs stdio

- UNIX I/O
  - open/close
  - read/write
  - lseek
  - dup/dup2

- stdio
  - fopen/fdopen/fclose
  - fscanf/fprintf, scanf/printf
    - fread/fwrite（较少用）
  - fseek
  - freopen

  - fflush

# stdio-mode（常用）

- “r”：只读
- “w”：只写，截断
- “a”：附加
- “r+”：打开文本文件并更新
- “w+”：创建文本文件并更新，截断
- “a+”：附加并更新
- “b”：在上面的选项后加“b”，打开二进制文件

# stdio

- 将文件模型化为流（stream）

- 缓冲区（buffer）
  - 全缓冲：磁盘文件
  - 行缓冲：stdin, stdout
  - 无缓冲

# stdio-缓冲区引起的问题

- 文件没有close:
  - 非正常退出时，在缓冲区中的数据不会被自动刷新
- 信号安全:
  - printf信号不安全，因为要获得缓冲区的锁
- 交替读写
  - 在读和写之间必须刷新缓冲区，因为读和写共享相同的缓冲区，并且读和写之间不会自动刷新
  - 取决于实现，C标准不保证
- 用一个文件不要用两个流去写，否则可能产生同步错误

# RIO

- 无缓冲
  - rio_readn
  - rio_written
- 带缓冲
  - rio_readinitb
  - rio_readlineb/rio_readnb
  - rio_readn/rio_writen

# 共享文件&I/O重定向

- 描述符表
- 文件表
- v-node表

- I/O重定向
  - dup
  - dup2

# 编程建议

- 用哪一套I/O包?
  - 只要有可能就使用标准I/O
  - 网络套接字：在读之前刷新缓冲区，在写之前读完缓冲区，或者使用RIO
- 对同一个文件
  - 尽可能不要共享文件位置
  - 当有写者时，不要有其他的读者/写者
  - 可以有多个读者（但是不要共享文件位置）