面经预热Day10（OS专题）

1、进程和线程的区别

进程是系统进行资源分配和调度的基本单位。

线程Thread是操作系统能够进行运算调度的最小单位，线程是进程的子任务，是进程内的执行单元，一个进程至少有一个线程，一个进程可以运行多个线程，这些线程共享同一块内存。

1. 资源开销：
    - 进程：由于每个进程都有独立的内存空间，创建和销毁进程的开销较大，进程间切换需要保存和回复整个进程的状态，因此上下文切换的开销较高
    - 线程：线程共享相同的内存空间，创建和销毁线程的开销较小。线程间切换只需要保存和恢复少量的线程上下文，因此上下文切换的开销较小
2. 通信与同步：
    - 进程：由于进程间相互隔离，进程之间的通信需要使用一些特殊机制，如管道、消息队列、共享内存等
    - 线程：由于线程共享相同的内存空间，它们之间可以直接访问共享数据，线程间通信更加方便
3. 安全性：
    - 进程：由于进程间相互隔离，一个进程的崩溃不会直接影响其他进程的稳定性
    - 线程：由于线程共享相同的内存空间，一个线程的错误可能会影响整个进程的稳定性

2、进程调度算法

进程调度算法是操作系统中用来管理和调度进程（也称为任务或作业）执行的方法，这些算法决定了在多任务环境下，如何为各个进程分配CPU事件，以实现公平性、高吞吐量、低延迟等不同的调度目标。

1. 先来先服务调度算法

按照进程到达的先后顺序进行调度，即最早到达的进程先执行，直到完成或阻塞

2. 最短作业优先调度算法

   优先选择运行时间最短的进程来运行

3. 高响应比优先调度算法

   综合考虑等待时间和服务时间的比率，选择具有最高响应比的进程来执行

4. 时间片轮转调度算法

   将CPU时间划分为时间片（时间量），每个进程在一个时间片内运行，然后切换到下一个进程

5. 最高优先级调度算法

   为每个进程分配一个优先级，优先级较高的进程先执行，这可能导致低优先级进程长时间等待，可能引发饥饿问题

6. 多级反馈队列调度算法

   将进程划分为多个队列，每个队列具有不同的优先级，进程在队列之间移动，具有更高优先级的队列的进程会更早执行，而长时间等待的进程会被提升到更高优先级队列

7. 最短剩余时间优先

   每次选择剩余执行时间最短的进程来执行

8. 最大吞吐量调度

   旨在最大化单位时间内完成的进程数量

3、进程间通信方式

1. 管道：是一种半双工的通信方式，数据只能单向流动而且只能在具有父子进程关系的进程间使用
2. 命名管道：也是半双工的通信方式，但是它允许无亲缘关系进程间的通信
3. 信号量：是一个计数器，可以用来控制多个进程对共享资源的访问，常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源，因此主要作为进程间以及同一进程内不同线程之间的同步手段

4. 消息队列：消息队列是消息的链表，存放在内核中并由消息队列标识符标识，消息队列克服了信号传递信息少、管道只能承载无格式字节流以及缓冲区大小受限等缺点
5. 信号：用于通知接收进程某个事件已经发生，从而迫使进程执行信号处理程序
6. 共享内存：就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问。共享内存是最快的进程通信方式，它是针对其他进程间通信方式运行效率低而专门设计的，它往往与其他通信机制，比如信号量配合 使用，来实现进程间的同步和通信。
7. Socket套接字：是支持TCP/IP的网络通信的基本操作单元，主要用于在客户端和服务器之间通过网络进行通信

进程间通信

管道

管道：实际上就是一个内存缓冲区

匿名管道

命令例子：`ps -ef | grep mysql`

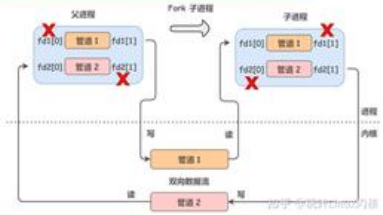
中间的|是管道命令 是指ps命令与grep同时执行

功能是将前一个命令的输出，作为后一个命令的输入

特点

匿名管道只能用于父子进程或兄弟进程之间，必须用于具有亲缘关系的进程间的通信

匿名管道只能用于父子进程或兄弟进程之间，必须用于具有亲缘关系的进程间的通信



相关接口

int pipe(int fd[2]);

fd[0]: 读的一端

fd[1]: 写的一端

通信双方的进程中写数据的一方需要把fd[0]先close掉，读的一方需要先把fd[1]给close掉

命名管道/FIFO

命令: `mkfifo mypipe`

在使用命名管道前，先需要通过 mkfifo 命令来创建，并且指定管道名字

特点

有名管道是FIFO文件，存在于文件系统中，可以通过文件路径名来指出

有名管道可以在不具有亲缘关系的进程间进行通信

相关接口 int mkfifo(const char *pathname, mode_t mode);

pathname：即将创建的FIFO文件路径，如果文件存在需要先删除

mode：和open()中的参数相同

优缺点

优点 使用简单

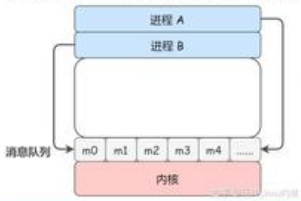
缺点

效率比较低，不适合进程间频繁地交换数据

管道只能传输无格式的字节流

消息队列

消息队列的本质：就是存放在内存中的消息的链表



优缺点

优点

对于交换较少数量的数据很有用，因为无需避免FIFO 中同步管道的打开和关闭时可能产生的冲突

缺点

消息队列的读取和写入的过程，都会有发生用户态与内核态之间的消息拷贝过程

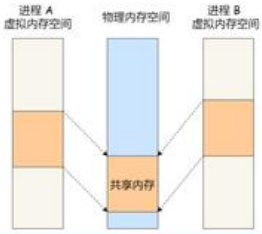
用户进程写入数据到内存中的消息队列时，会发生从用户态拷贝数据到内核态的过程

用户进程读取内存中的消息数据时，会发生从内核态拷贝数据到用户态的过程

如果数据量较大，使用消息队列就会造成频繁的系统调用，即需要消耗更多的时间以便内核介入

共享内存

共享内存的机制：就是拿出一块虚拟地址空间来，映射到相同的物理内存中，所有进程都可以访问共享内存中的地址



优缺点

优点

不需要像消息队列那样频繁的拷贝消息、进行系统调用，共享速度快

缺点

共享内存机制可能会发生冲突：

如果多个进程同时修改同一个共享内存，先来的那个进程写的内容就会被后来的覆盖

信号量

信号量定义

操作系统提供的一种协调共享资源访问的方法，主要用于实现进程间的互斥与同步

实际上就是一个整型计数器(sem)，用于表示资源的数量

P 操作：将 sem 减 1，相减后，如果 sem < 0，则进程进入阻塞等待；如果sem>=0，表明还有资源可用，进程可正常继续执行

V 操作：将 sem 加 1，相加后，如果 sem <= 0，唤醒一个等待中的进程；如果sem>0，表明当前没有阻塞中的进程

P操作用于进入共享资源之前，V操作用于离开共享资源之后；两个操作必须成对出现

信号量和 PV 操作具体的定义(伪代码)

```
// 信号量数据结构
type struct sem_t{
    int sem;    // 资源个数
    queue_t *q;// 等待队列
} sem_t;

// 初始化信号量
void init(sem_t *s, int sem) {
    s->sem = sem;
    queue_init(s->q);
```

控制信号量的原子操作

- 信号
  - 信号定义：信号是进程通信机制中唯一的异步通信机制，它可以在任何时候发送信号给某个进程，以迫使进程执行信号处理程序
  - 信号事件的来源
    - 硬件来源
      - Ctrl+C 产生 SIGINT 信号，表示终止该进程
      - Ctrl+Z 产生 SIGTSTP 信号，表示停止该进程，但还未结束
    - 软件来源
      - 输入kill命令如`kill -9 1111`，表示给 PID 为 1111 的进程发送 SIGKILL 信号，让其立即结束
- Socket
  - Socket概念
    - Socket可以完成跨网络与不同主机上的进程进行通信
    - Socket 的本质其实是一个API，它把复杂的 TCP/IP 协议族隐藏在 Socket 接口后面
  - 相关接口
    - int socket(int domain, int type, int protocal)
      - domain 参数：用来指定协议族
        - AF _ INET用于IPV4
        - AF _ INET6用于IPV6
        - AF _ LOCAL / AF _ UNIX用于本机
      - type 参数用来指定通信特性
        - SOCK _ STREAM表示的是字节流，对应TCP
        - SOCK _ DGRAM 表示的是数据报，对应UDP
        - SOCK _ RAW表示的是原始套接字
      - protocal 参数原本是用来指定通信协议的，但现在基本废弃。因为协议已经通过前面两个参数指定完成，protocol目前一般写成0即可
  - Socket通信方式
    - 实现TCP字节流通信
      - socket类型是AF _ INET和SOCK _ STREAM
    - 实现UDP数据报通信
      - socket类型是AF _ INET和SOCK _ DGRAM
    - 实现本地进程间通信
      - 本地字节流Socket
        - 类型是AF _ LOCAL和SOCK _ STREAM
      - 本地数据报Socket
        - 类型是AF _ LOCAL和SOCK _ DGRAM