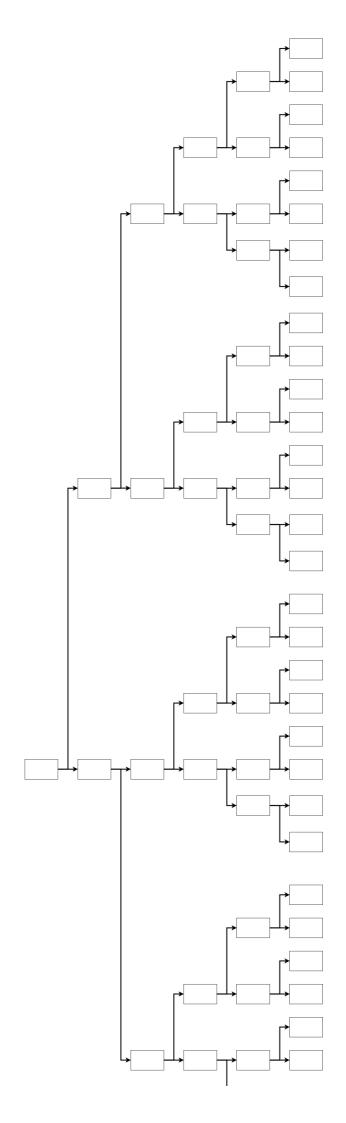
每个概念被创造都有意义,请简述"进程"这个概念在Linux系统中有什么用途

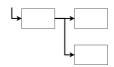
```
//代码段A
int i;
for(i=0;i<5;i++)
{
    pid=fork();
}
```

```
//代码段B
int i;
for(i=0;i<5;i++)
{
    if((pid=fork())==0)
        break;
}
```

阅读以上代码段,回答代码段A和代码段B的执行结果有什么区别?并解释为什么会有这种区别。

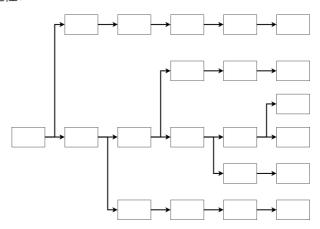
如图: 代码段A执行过程:





对于代码段A,由 $a_n=2a_{n-1},\ a_0=1$ 累乘解得,最后的状态数为 $a_5=2^5=32$ 种状态。

如下图: 代码段B执行过程:



对于代码段B,容易看出 $a_n=a_{n-1}+1, a_0=1$,从而求得最终的状态数为 $a_5=5$ 种状态。

执行结果的区别即最终结果中子进程的多少;造成这种区别的原因是代码段A与代码段B相比,在循环中 fork() 的时候,子进程将通过 break 退出循环。

Q2

用自己的话阐述什么是僵尸进程,并描述进程通过调用wait()捕获僵尸态的子进程的过程。

僵尸进程是比父进程先结束却继续占用资源而没有被父进程回收的子进程。

父进程调用了wait()后,立即阻塞自己。由wait()自动分析是否当前进程的某个子进程已经退出,如果让它找到了这样一个已经变成僵尸进程的子进程,wait就会收集这个子进程的信息,并把它彻底销毁后返回;如果没有找到这样一个子进程,父进程将挂起,直到有一个子进程退出为止。(wait()等待处理第一个结束的子进程)

Q3

请阐述信号在Linux系统中的作用。

信号是进程间通讯的机制,是发送给进程特殊的异步消息。

Q4

请简述信号什么时候处于未决状态,并简述信号存在未决状态的作用。

信号从产生到递达之间的状态称为信号未决状态,即已经产生的信号还未传递给任何进程的状态。

当信号被堵塞的时候,可以保持在未决状态,直到进程解除对此信号的堵塞,才执行递达的动作。

请设计一种通过信号量来实现共享内存读写操作同步的方式,文字阐述即可,不需要代码实现。 (提示:在写进程操作未完成时,需要防止其他进程从共享内存中读取数据)

设计一种信号量 SIGRDWR ,其结构体由一个保护信号量的自旋锁 lock ,现有资源的数量 count 与等待获取此锁的进程序列组成 wait_list 。

当自旋锁为打开状态时,如果现有资源数量满足等待序列头部进程所需内存,则从等待进程索引链表头部得到的进程进行共享内存读写操作,并关闭自旋锁;当自旋锁所为关闭状态时,等待当前进程的读写操作,当共享内存重新处于空闲状态时,打开自旋锁。

当一个新的进程想要进行共享内存的的写操作时,将排到等待队列的尾端。可通过 sigaction 改变默认的处理信号的函数指针