第二章作业

1.什么是前趋图?为什么要引入前趋图？

前趋图是一个有向无循环图，记为DAG，用于描述进程之间执行的前后关系。

程序分为顺序执行以及并行执行，前趋图可以描述进程之间的前后关系。

2.画出前趋图：

5.在操作系统中为什么要引入进程的概念？它会产生什么影响？

因为在多道程序环境下，允许多个程序并发执行，正是因为程序并发执行的特征才导致操作系统引入进程的概念，多个程序并发执行，必须保证其独立性，而且可以控制，所以引入了PCB，程序段，数据段组成了进程实体。

它产生的影响，解决了多个程序并发执行的问题，是系统资源分配和调度的一个独立单位，更好地运用了系统资源，极大的拓展了计算机的能力。

6.试从动态性，并发性和独立性上比较进程和程序。

1. 动态性

进程具有创建🡪调度执行🡪撤消消亡的生命期，是动态的。而程序是一组有序指令的集合，是静态的。

1. 并发性

程序并发性是进程的重要特征，同时也是os的重要特征。引入进程的目的正是为了使其程序能和其它建立了进程的程序并发执行，而程序本身是不能并发执行的。

1. 独立性

独立性是指进程实体是一个能独立运行的基本单位，同时也是系统中独立获得资源和独立调度的基本单位。而对于未建立任何进程的程序，都不能作为一个独立的单位来运行。

7试说明PCB的作用具体表现在哪些方面？为什么说PCB是进程存在的唯一标志？

PCB具体作用：

1. 存放进程标识符，唯一标识一个进程
2. 存放处理机状态信息
3. 存放进程调度和进程对换有关的信息
4. 存放进程控制信息

进程控制块的作用是使一个在多道程序环境下不能独立运行的程序（含数据），成为一个能独立运行的基本单位—进程。在进程的整个生命期中，操作系统总是通过PCB对进程进行控制的。所以说，PCB是进程存在的唯一标志。

11试说明进程在三个基本状态之间转换的典型原因。

时间片用完/被抢占

I/O请求

进程调度

I/O完成

12 为什么要引入挂起状态？该状态有哪些性质？

引入挂起状态的原因：

1. 负荷调节的需要。当实时系统中的工作负荷较重，把一些不重要的进程挂起，以保证系统能正常运行。
2. 操作系统的需要。操作系统有时希望挂起某些进程，以便检查运行中的资源使用情况或进行记账。

挂起状态的性质：

1. 挂起的进程不能直接转化为运行态，必须先转化为活动态
2. 挂起状态的进程储存在外存中。
3. 激活挂起的进程是，先把它从外存调入到内存中，再进行检查。

13 在进程切换时，所要保存的处理机状态信息有哪些？

1. 通用寄存器。
2. 指令计数器PC。
3. 程序状态字PSW。
4. 用户栈指针SP。

14 试说明引起进程创建的主要事件。

1. 用户登陆。
2. 作业调度。
3. 提供服务。如I/O请求。
4. 应用请求。

15 试说明引起进程被撤销的主要事件。

1. 正常结束。
2. 异常结束
   1. 越界错误。
   2. 保护错。
   3. 非法指令。
   4. 特权指令错。
   5. 运行超时。
   6. 算术运算错，除零错误。
   7. I/O故障。
3. 外界干预。
   1. 操作员或者操作系统干预。
   2. 父进程请求终止该进程。
   3. 父进程终止，其所有的子进程也即将终止。

16 在创建一个进程时所主要完成的主要工作是什么？

（1）申请空白PCB。

（2）为新进程分配资源。

（3）初始化进程控制块。

1. 初始化标识信息。
2. 初始化处理机状态信息。使程序计数器指向程序的入口地址，使栈指针指向栈顶；
3. 初始化处理机控制信息：进程的状态、优先级。

（4）将新进程插入就绪队列，启动调度。

17 在撤销一个进程所要完成的主要工作是什么？

（1）根据被终止进程的PID找到它的PCB，从中读出该进程的状态。

（2）若被终止进程正处于执行状态，应立即终止该进程的执行，重新进行调度。

（3）若该进程还有子孙进程，立即将其所有子孙进程终止。

（4）将被终止进程所拥有的全部资源，归还给其父进程，或者归还给系统。

（5）将被终止进程的PCB从所在队列中移出。

18 时说明引起进程阻塞或者被唤醒的主要事件是什么？

1. 引起进程阻塞的主要事件。
   1. 请求系统服务。
   2. 启动某种操作：如I/O操作。
   3. 新数据尚未到达。
   4. 无新工作可做，使用sleep()进入休眠。
2. 进程唤醒事件
   1. 当被阻塞进程所期待的事情出现时，则进程被唤醒。

21 试从调度性，并发性，拥有资源及系统开销方面对进程和线程进行比较。

1. 调度上

在传统操作系统中，调度和分派的单位是进程。

在引入线程的操作系统中，调度和分派的单位是线程。

1. 并发性

在引入线程之间操作系统中，不仅进程之间可以并发执行，而且一个进程的多个线程之间也可以并发执行，从而可以更为有效地使用资源，并提高系统的吞吐量

1. 拥有资源

进程是拥有资源的基本单位。

线程自己不拥有系统资源(也有一点必不可少的资源),但它可以访问其隶属进程的资源，同一进程中的多个线程共享其资源。

1. 系统开销

由于在创建,撤销或切换进程时,系统都要为之分配或回收资源,保存CPU现场.因此,操作系统所付出的开销将显著地大于在创建,撤销或切换线程时的开销。

23 何谓用户级线程和内核支持线程。

用户进程：该进程仅存在于用户空间中。对于这种线程的创建、撤消、线程之间的同步与通信等功能，都无须内核来实现。

内核支持线程：只运行在内核模式，即无论是用户进程中的线程，还是系统进程中的线程，他们的创建、撤消和切换等，是依靠内核实现的。

24 试说明用户级线程的实现方法。

1. 用户级线程是在用户空间实现的。所有用户级线程都具有相同的数据结构，它们都运行在一个中间系统上。
2. 当前有两种方式实现的中间系统。
   1. 运行时系统

是用于管理和控制线程的函数的集合，又称为线程库。包括创建、撤消线程函数、线程同步和通信函数、线程调度函数等。

用户级线程不能直接利用系统调用，必须通过运行时系统间接利用系统调用。

* 1. 内核控制线程

这种线程又称为轻型进程LWP（Light Weight Process）。每个进程都可拥有多个LWP，每个LWP都有自己的TCB，其中包括线程标识符、优先级、状态、栈和局部存储区等。

LWP可通过系统调用来获得内核提供的服务，当一个用户级线程运行时，只要将它连接到一个LWP上，它便具有了内核支持线程的所有属性。

25 试说明内核级线程的实现方法。

在仅设置了内核支持线程的OS中，一种可能的线程控制方法是，系统在创建一个新进程时，便为它分配一个任务数据区PTDA（Per Task Data area),其中包括若干个线程控制块TCB空间。

|  |
| --- |
| PTDA 进程资源 |
| TCB#1 |
| TCB#2 |
| TCB#3 |

当进程要创建一个线程时，便为新线程分配一个TCB，将有关信息填入TCB，并为之分配必要的资源。新线程便可立即执行。

当进程要撤消一个线程时，也应收回线程的TCB和所有资源。

26 试修改下面生产者-消费者问题解法中的错误：

consumer:

begin

repeat

wait(mutex);

wait(empty);

nextc:=buffer(out);

out:=out+1;

signal(mutex);

consumer item in nextc;

until false;

end

producer:

begin

repeat

…

produce an item in nextp;

wait(mutex);

wait(full);

buffer(in):=nextp;

signal(mutex);

until false;

end

错误点：判断缓冲为空或者满时，不能放在临界区内，否则如果满，消费者不能消费，一直满，造成阻塞。

改为：

consumer:

begin

repeat

wait(empty);

wait(mutex);

nextc:=buffer(out);

out:=out+1;

signal(mutex);

singal(empty);

consumer item in nextc;

until false;

end

producer:

begin

repeat

…

produce an item in nextp;

wait(full);

wait(mutex);

buffer(in):=nextp;

signal(mutex);

signal(full)

until false;

end

27 某机场的入口有1条通道，该通道有3名证件查验员，再往里是一个安检室，内有5名安检员。安检室可容纳五人等待（另有五人正在被检查）。入场者可让任意一名证件查验员检查，在证件查验员核对正确并放行后进入安检室等待并被检验，安检员放行后进入机场。证件查验员无人时休息，来人时被来人唤醒，查验来人证件后，确认安检室有空位后让来人入安检室。安检员无人时休息，直到被来人唤醒，检查完来人后放行来人入机场。请用信号量机制实现上述过程中的同步和互斥。

process:

begin:

repeat

show a consumer;

wait(check\_idCard); //等待证件检查员

checkIDcard(); //证件检查

wait(wait\_securityCheck);//等待空位

singal(check\_idCard);//释放证件检查员

wait(securityCheck);//等待安全检查

singal(wait\_securityCheck);//释放等待空位

securityCheck();//进行安全检查

singal(securityCheck);//释放安全检查员