LEC5

一. 线程调度

1. 竞争范围

1. 竞争范围（PCS）：用户级线程访问内核资源之间的竞争。
    - 进程竞争范围（LCS）：指的是在一个进程内部的竞争范围。
    - 系统竞争范围（SCS）：指的是系统中所有进程使用的竞争范围。

线程调度的基本级别有两种方案：

1. 针对进程内的竞争范围内的线程进行调度。
2. 针对系统中使用的多个进程进行调度。

线程的优先级在创建时由应用程序开发者指定，这决定了在PCS范围内的调度顺序。而系统竞争范围的调度则涉及到所有使用的进程。

内核级线程是CPU调度的最小单位，CPU的竞争发生在系统中所有使用的进程之间。线程库将用户级线程调度到可用的工作线程池（LWP），并在CPU上运行。操作系统负责决定哪个内核级线程调度到CPU上，这涉及到SCS的作用。

PCS和SCS的调度根据优先级执行，而将线程调度到可用的LWP则是SCS调度的一部分。

总结起来，内核级线程是由操作系统调度的最小单位，在PCS范围内，线程的调度是基于优先级的；而在SCS范围内，调度涉及到系统中所有进程的竞争。SCS调度将内核级线程调度到可用的LWP上运行。

二. 多处理器调度方法

- 多处理器调度中的问题：如何将进程分配到处理器上。
- 静态分配方式可能会导致负载不均衡的问题，因为当进程正在执行时，无法确定每个处理器上要处理的进程数量，可能会导致一个处理器忙碌而另一个处理器空闲。

1. 解决负载不均衡问题的方法：
    - 使用一个公共的就绪进程队列，该队列维护着所有准备执行的进程。每个处理器都可以从该队列中获取任务来执行，以实现负载均衡。
2. 多处理器的组织方式：
    - 主/从式（非对称处理）：其中一个处理器充当主节点，负责调度任务，其他处理器作为从节点，接受主节点的调度。

- 对称式（对称处理）：所有处理器都可以执行调度任务，由主处理器决定将进程分配给哪个处理器执行。

3. 主/从式和对称式的区别：
   - 主/从式效率较高，但会造成处理器瓶颈，因为只有主处理器负责调度任务。
   - 对称式允许处理器自我调度，提高了并行处理能力。
4. 对称多处理的优点和缺点：
   - 优点：系统中的任何处理器都可以执行调度任务，即使其中一个处理器故障，系统仍然可以正常运行。
   - 缺点：操作系统需要确保两个处理器不会同时处理同一个进程，以防止进程状态丢失，这增加了系统的复杂性。

三. 多处理器线程调度方法

1. 共享加载：进程不是分配到特定处理器，而是将公共的就绪队列分配给处理器，基于负载平衡的原则。
2. 组调度：将相关的线程组成组，并将组调度到一组处理器上执行。
3. 专用处理器分配：每个进程被分配给一组处理器，处理器数量与线程数相适应。
4. 动态调度：在执行过程中动态改变进程中线程的数量。

共享加载优点是充分利用处理器间的负载平衡，不采用集中式调度程序方案，可以采用先来先服务、最少进程数优先或最少线程数优先等调度策略。然而，可能存在访问共享资源的瓶颈，导致效率下降。

组调度的优点在于如果相关的进程可以并行执行，同步的开销会减少，只需要很少的进程切换，从而提高性能。同时，由于决策集合可以根据不同的时间片和处理器调整，因此减少了管理开销。

专用处理器分配是当一个应用程序被调度时，其每个线程都被分配给一个处理器，直到应用程序结束。

动态调度允许应用程序根据需要动态地改变进程中的线程数量，以适应负载情况的变化。应用程序可以通过提供语言和系统工具来实现动态调度，在执行过程中调整线程数量，以优化系统性能。

多处理器系统中，缓存的亲和性指的是利用共享缓存的优势。处理器的亲和性指的是任务是否具有指定处理器的能力。保持进程运行在负载均衡状态下是重要的，因为在多任务环境中容易出现负载不均衡，可能导致处理器间的工作窃取和负载切换的问题。

四. 实时操作系统

实时操作系统（RTOS）是为了满足特定时间期限而设计的，其中任务具有一些时间限制。这些任务可以根据其截止时间造成的后果进行分类：

1. 硬实时任务：必须在截止日期之前完成，例如导弹拦截系统。
2. 软实时任务：可以偶尔错过截止日期，但尽可能在截止日期前完成，例如视频播放。

这些任务可以进一步分类为周期任务和非周期任务。

- 周期任务：根据任务的周期性进行调度，例如周期性的数据采集任务。
- 非周期任务：没有固定的执行周期，但是有最迟开始时间和最迟结束时间的限制。

实时操作系统的特点包括：

1. 可确定性：对外部事件进行快速响应。
2. 可响应性：尽快处理中断，以便快速响应事件。

中断延迟：
中断是指计算机系统中某些事件（如硬件设备发出的信号或软件生成的请求）打断正常程序执行流程的机制。中断可以是内部的（如程序错误或异常）或外部的（如硬件设备发生状态变化）。中断延迟是指在发生中断事件和系统开始处理该中断之间的时间间隔。这包括中断信号传输到 CPU、CPU 切换到中断处理程序的执行上下文、保存当前执行状态等操作所需的时间。中断延迟的大小直接影响着系统对于外部事件的响应速度。

分派延迟：
分派是指操作系统将 CPU 分配给某个特定的任务或进程的过程。分派延迟是指从操作系统决定将 CPU 分配给某个任务到实际开始执行该任务之间的时间间隔。在这段时间内，可能需要保存和恢复上一个任务的状态、进行上下文切换、执行调度算法等操作。分派延迟的大小直接影响着系统对于任务切换的效率和响应速度。

实时调度算法可以根据静态和动态的方式进行分类：

静态优先级调度策略（RM）：

按照任务的周期性和最早启动时间确定任务的优先级。优先级越高的任务越先执行，任务的周期越短，优先级越高。RMS（Rate-Monotonic Scheduling）是这种策略的一个例子。该策略要求事先知道每个任务的周期，并根据周期动态地分配优先级。任务的周期越短，说明任务越紧急，优先级越高。RMS采用抢占式调度，高优先级的任务可以抢占低优先级的任务，并在任务开始执行前根据任务属性确定优先级，包括周期和截止时间。

动态优先级调度策略（EDF）：

根据截止时间确定任务的优先级。截止时间越近的任务优先级越高，这样可以保证处理器最早处理截止时间最近的任务。该策略会根据任务的截止时间动态地为任务分配优先级。调度器只需要在非周期任务中了解任务的截止时间这一信息，其他信息如周期和处理时间不是必需的。任务的最晚启动时间在调度器决定何时分派任务开始执行时是非常重要的，因为它决定了任务是否会错过截止时间。如果

调度器可以成功调度任务使其在最晚启动时间之前开始执行，那么这个调度是成功的；否则，就会发生调度失败。

在比例共享调度策略中：

操作系统的主要目标是确保每个任务都获得了它所需的CPU时间片，而不是简单地优化周转时间和响应时间。操作系统会将 CPU 时间按照各个应用程序持有的份额分配给系统内的所有进程。假设一个应用程序有自己的 CPU 时间份额，而系统总共有多个这样的应用程序，那么每个应用程序将按照其持有的份额比例分配 CPU 时间。举例来说，如果一个应用程序拥有总处理时间的一半，那么它将获得总 CPU 处理时间的一半。

五. 调度算法评估

调度算法的评估可以通过以下几种方式进行：

1. 确定性评估： 根据系统需求确定特定的预设工作负载，并为每种算法定义评估性能的指标。例如，为了评估算法的平均等待时间，可以选择最小平均等待时间作为评估指标。不同的系统可能对性能指标有不同的要求，因此需要根据具体情况选择合适的评估标准。
2. 排队模型： 在排队模型中，可以利用特尔定律(Little's Law)来建模。通过定义就绪队列和CPU等待队列，并使用队列理论来评估多种算法的性能。通过设定队列长度、进程到达速率等参数，可以对系统的稳定状态进行模拟，并评估算法的性能。
3. 模拟： 利用仿真模拟的方法来评估算法的性能。通过编程生成随机数来模拟系统的状态，并在模拟执行时输出指示算法性能的统计信息。模拟可能不够准确，但可以通过监视实际系统并记录事件序列来创建跟踪磁带，以驱动模拟的方式来提高准确性

Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

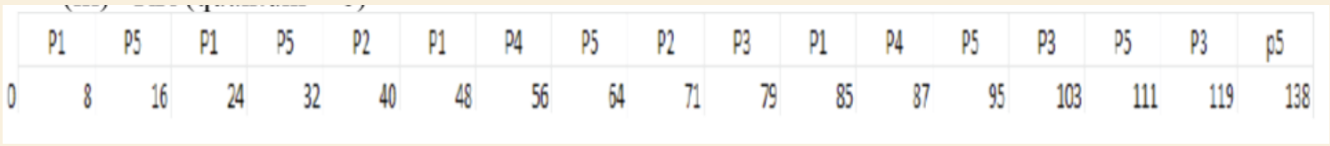
Process	Arrival Time(ms)	Burst Time(ms)
P1	0	30
P2	19	15
P3	42	24
P4	27	10
P5	3	59

The execution of these processes is on the **Round-Robin (quantum = 8)** scheduling algorithm.

The **average waiting time** of the system is 31.8 ✖ *[54.2]* milliseconds.

The **average turnaround time** of the system is 83.8 ✖ *[81.8]* milliseconds.

Your answer is incorrect.



Waiting time/process

RR	55	37	53	50	76

AWT = (55+37+53+50+76)/5 = 54.2

Turnaround time /process

RR	85	52	77	60	135

ATAT = (85+52+77+60+135)/5 = 81.8