题目一解答

对于长期调度程序而言，需要适当的组合所调度的进程组合，即需要CPU约束型进程和I/O约束型进程相组合；否则当只有CPU约束型时，I/O设备几乎完全被闲置；而只有I/O约束型时，短期调度程序又几乎无事可做；

区分CPU约束型和I/O约束型进程，可以使得调度程序根据二者的特点而组合进程组，使得对CPU和I/O设备的利用率都得以提升，从而让计算机在整体上获得更好的性能。

题目二解答

1.先到先服务

不会导致饥饿；进程按照到达CPU时间而先后被执行，所有到达的进程终将会被执行；

2.最短作业优先

可能导致饥饿；这种调度算法对于CPU周期较长的进程不友好，在极端情况下有可能长进程始终没有机会占据CPU；

3.轮转法

不会导致饥饿；每个进程以相等几率占据CPU的固定时间片；

4.优先级

可能导致饥饿；在没有考虑老化的情况下，对于优先级低的进程不友好，优先级最低的进程可能永远不会被执行；

题目三解答

CPU利用率=$\frac{CPU_{执行用户进程时间}}{CPU_{工作总时间}}$

1.时间片为1ms

以一次循环为例，对于这1ms的时间片，无论是I/O型还是CPU型，都会充分利用这1ms来执行CPU计算，之后每个进程在时间片结束之后都需要0.1ms来执行上下文切换，这一部分上下文切换的时间开销是内核占用的，故不算在进程的CPU使用时间中。故CPU利用率=$\frac{0.1*11}{(1+0.1)*11}=\frac{1}{1.1}\approx 90.91\%$；

2.时间片为10ms

同样以一次循环为例，对于这10ms的时间片，CPU型可以充分利用，但是I/O型只用到1ms，然后便执行上下文切换；所以这时CPU工作的总时间=$10(CPU_{型})+1*10(I/O_{型})+0.1*11(_{上下文切换})=21.1$；故CPU利用率=$\frac{10+1*10}{21.1}\approx 94.79\%$；

题目四解答

1.画出甘特图

FCFS

SJF（非抢占）

SJF（抢占式）

抢占式优先级调度

RR

2.计算平均周转时间

FCFS	SJF(非抢占)	SJF(抢占)	抢占式优先级调度	RR
14.2ms	13.2ms	9ms	13ms	13.8ms

3.计算平均等待时间

FCFS	SJF(非抢占)	SJF(抢占)	抢占式优先级调度	RR
9.6ms	8.6ms	4.4ms	8.4ms	9.2ms

题目五解答

查找文献了解Linux的调度设计，并就至少两个细节说明其好处；

Linux把进程区分为实时进程和非实时进程, 其中非实时进程进一步划分为交互式进程和批处理进程：

类型	描述	示例

类型	描述	示例
交互式进程(interactive process)	此类进程经常与用户进行交互, 因此需要花费很多时间等待键盘和鼠标操作. 当接受了用户的输入后, 进程必须很快被唤醒, 否则用户会感觉系统反应迟钝;	shell, 文本编辑程序和图形应用程序;
批处理进程(batch process)	此类进程不必与用户交互, 因此经常在后台运行. 因为这样的进程不必很快相应, 因此常受到调度程序的怠慢;	程序语言的编译程序, 数据库搜索引擎以及科学计算;
实时进程(real-time process)	这些进程由很强的调度需要, 这样的进程绝不会被低优先级的进程阻塞并且他们的响应时间要尽可能的短;	视频音频应用程序, 机器人控制程序以及从物理传感器上收集数据的程序;

在Linux中, 调度算法可以明确的确认所有实时进程的身份, 但是没办法区分交互式程序和批处理程序, linux2.6的调度程序实现了基于进程过去行为的启发式算法, 以确定进程应该被当做交互式进程还是批处理进程. 当然与批处理进程相比, 调度程序有偏爱交互式进程的倾向;

根据进程的不同分类Linux采用不同的调度策略:

①对于实时进程，采用FIFO或者Round Robin的调度策略;

②对于普通进程，则需要区分交互式和批处理式的不同。传统Linux调度器提高交互式应用的优先级，使得它们能更快地被调度。而CFS和RSDL等新的调度器的核心思想是“完全公平。这个设计理念不仅大大简化了调度器的代码复杂度，还对各种调度需求的提供了更完美的支持；注意Linux通过将进程和线程调度视为一个，同时包含二者。进程可以看做是单个线程，但是进程可以包含共享一定资源（代码和/或数据）的多个线程。因此进程调度也包含了线程调度的功能;

③对于非实时进程，调度策略比较简单， 因为实时进程值只要求尽可能快的被响应，基于优先级, 每个进程根据它重要程度的不同被赋予不同的优先级，调度器在每次调度时，总选择优先级最高的进程开始执行，低优先级不可能抢占高优先级，因此FIFO或者Round Robin的调度策略即可满足实时进程调度的需求;

下面是CFS调度的两个实现模块:

时间记账
所有的调度器都必须对进程的运行时间做记账。CFS不再有时间片的概念，维护了每个进程运行的时间记账，因为每个进程只在公平分配给它的处理器时间内运行。关键数据结构如下：vruntime: 虚拟运行时间是在所有可运行基础的总数上计算出一个进程应该运行多久，计算时相应的nice值在CFS被作为进程获得处理器运行比的权重：越高的nice（越低优先级）值，获得更低的处理器权重，更低的nice值获得更高的处理器使用权重。
个人理解：CFS的vruntime += 处理器运行时间 * nice对应的权重

进程选择使用红黑树
进程选择是CFS调度算法的最重要的模块，当CFS调度器选择下一个要进行调度的进程时，就会选择具有最小vruntime的任务。涉及到获取最小值，以及有序数据结构，在各种场景下都很适用的红黑树就发挥了其作用。即用红黑树维护以vruntime为排序条件，存储着任务的运行情况。