### 先来先服务算法

按作业或进程到达的先后顺序进行调度，即每次在后备作业（就绪进程）队列中选择先到达的作业（或进程）投入运行，它最简单可用于作业调度和进程调度。

## 2. ****短作业优先算法****

按作业（或进程）估计运行时间长短来组织后备作业队列，每次选择运行时间最短的作业（或进程）投入运行，目的是为了提高系统的吞吐率，减少进程的平均周转时间。后来的作业不抢占正在执行的作业适合作业调度和进程调度，其中短作业优先(SJF)的调度算法：从后备队列中选择一个或若干个估计运行时间最短的作业，将它们调入内存运行短进程优先(SPF)调度算法：从就绪队列中选出一估计运行时间最短的进程，将处理机分配给它，使它立即执行并一直执行到完成，或发生某事件而被阻塞放弃处理机时，再重新调度

## 3. ****优先级调度算法****

每次在作业后备队列（或就绪进程队列）中挑选优先级最高的作业（或进程）投入运行

优先权调度算法的类型

（1）非抢占式优先权算法：在这种方式下，系统一旦把处理机分配给就绪队列中优先权最高的进程后，该进程便一直执行下去，直至完成； 或因发生某事件使该进程放弃处理机时，系统方可再将处理机重新分配给另一优先权最高的进程。这种调度算法主要用于批处理系统中；也可用于某些对实时性要求不严的实时系统中。

（2）抢占式优先权算法：在这种方式下，系统同样是把处理机分配给优先权最高的进程，使之执行。但在其执行期间，只要又出现了另一个其优先权更高的进程，进程调度程序就立即停止当前进程(原优先权最高的进程)的执行，重新将处理机分配给新到的优先权最高的进程。因此，在采用这种调度算法时，每当系统中出现一个新的就绪进程i时，就将其优先权Pi与正在执行的进程j的优先权Pj进行比较。如果Pi≤Pj，原进程Pj便继续执行；但如果是Pi＞Pj, 则立即停止Pj的执行，做进程切换，使i进程投入执行。显然，这种抢占式的优先权调度算法，能更好地满足紧迫作业的要求，故而常用于要求比较严格的实时系统中，以及对性能要求较高的批处理和分时系统中

优先权的类型

静态优先权：在进程创建时确定，在进程运行过程中其优先数不会发生改变。一般地，优先权是利用某一范围内的一个整数来表示的，例如，0~7或0~255中的某一整数， 又把该整数称为优先数。只是具体用法各异：有的系统用“0”表示最高优先权，当数值愈大时，其优先权愈低；而有的系统恰恰相反

动态优先权

在进程创建时确定，但进程的优先数会在进程推进过程中或随其等待时间的增加而改变例如，规定在就绪队列中的进程随其等待时间的增长，其优先权以速率a提高。若所有的进程都具有相同的优先权初值，则显然是最先进入就绪队列的进程，将因其动态优先权变得最高而优先获得处理机，此即FCFS算法。若所有的就绪进程具有各不相同的优先权初值，那么，对于优先权初值低的进程，在等待了足够的时间后，其优先权便可能升为最高，从而可以获得处理机。当采用抢占式优先权调度算法时，如果再规定当前进程的优先权以速率b下降，则可防止一个长作业长期地垄断处理机确定进程优先权的依据

原则

进程类型

进程对资源的需求

用户要求

作业优先级的确定（静态）

根据用户要求或用户身份确定作业的优先级。

根据作业的类型确定作业的优先级：一般情况下，I/O型作业的优先级高于CPU型作业的优先级。

根据作业需要资源的多少来确定其优先级，原则上需要资源多的作业的优先级低于需要资源少的作业的优先级。

进程优先级的确定（静态）

按进程的属性把进程分为系统进程和用户进程。其中，系统进程的优先级高于用户进程的优先级。

按进程的类型把进程分为I/O型进程、CPU型进程以及I/O与CPU均衡的进程，一般情况下，I/O型进程的优先级最高，I/O与CPU均衡的进程优先级次之，CPU型的优先级最低。

其他方法。

进程优先级的确定（动态）

根据进程占用CPU的时间长短来决定，进程占用CPU时间越长，其优先级就越低。

根据进程等待CPU的时间长短来决定，进程等待CPU的时间越长，其优先级就越高。

4.最高响应比优先调度算法（HRRN）

HRRN算法同时兼顾每个作业的等待时间和运行时间两个方面的因素，每次在作业后备队列中挑选响应比最高的作业投入运行。其中响应比计算公式如下：

响应比=（等待时间+要求服务时间）/要求服务时间 = 响应时间/要求服务时间如果作业的等待时间相同，则要求服务的时间越短，其优先权越高，有利于短作业。当要求服务的时间相同时，作业的优先权决定于其等待时间，等待时间越长的进程，其优先权越高，因而它实现了先来先服务。对于长作业，作业的优先级可以随等待时间的增加而提高，当其等待时间足够长时，进程的优先级便可升到很高，从而也可获得处理机。对于长作业，作业的优先级可以随等待时间的增加而提高，当其等待时间足够长时，进程的优先级便可升到很高，从而也可获得处理机。

5.时间片轮转调度算法（RR）

RR算法把CPU的处理时间分成固定大小的时间片，时间片长度为几ms到几百ms每次调度时，把CPU分配给队首进程，并令其执行一个时间片，当执行的时间片用完时，由一个计时器发出时钟中断请求，调度程序便据此信号来停止当前进程的执行，并将它送往就绪队列的末尾，然后再把处理机分配给就绪队列中新的队首进程，同时也让它执行一个时间片，如此反复。

6.多级反馈队列调度算法（RRMF）

为提高系统吞吐量而照顾短作业。

为得到较好的I/O设备利用率和对交互用户的及时响应而照顾I/O型作业。根据运行情况动态考虑作业性质，根据其当前性质进行相应调度。

RRMF性能分析多级反馈队列调度算法具有较好的性能，能较好地满足各种类型用户的需要。