## 程式设计专案

### 调度算法

该项目涉及实现几种不同的流程调度算法。将为调度程序分配一组预定义的任务，并将根据所选的调度算法来调度任务。每个任务都分配了一个优先级和CPU突发。将实施以下调度算法：

- 先来先服务（FCFS），按任务请求CPU的顺序调度任务。
- 最短作业优先（SJ），按任务的下一个CPU突发的长度顺序调度任务。
- 优先级调度，它根据优先级调度任务。
- 轮循（RR）调度，其中每个任务运行一个时间段（或其CPU突发剩余时间）。
- 具有循环优先级，该优先级按优先级调度任务，并对具有相同优先级的任务使用循环调度。

优先级范围为1到10，其中较高的数值表示较高的相对优先级。对于循环调度，时间量的长度为10毫秒。

#### I. 实作

该项目的实现可以用C或Java来完成，并且在文本的源代码下载中提供了支持这两种语言的程序文件。这些支持文件会读取任务计划，将任务插入列表，然后调用计划程序。

任务计划的格式为[任务名称] [优先级] [CPU突发次数]，格式如下：

```
T1, 4, 20
T2, 2, 25
T3, 3, 25
T4, 3, 15
T5, 10, 10
```

因此，任务T1的优先级为4，CPU突发时间为20毫秒，依此类推。假定所有任务都同时到达，因此您的调度程序算法不必支持较高优先级的进程，从而抢占了较低优先级的进程。此外，不必以任何特定顺序将任务放入队列或列表中。

第5.1.2节首先介绍了几种组织任务列表的策略。一种方法是将所有任务放在一个无序列表中，其中任务选择的策略取决于调度

算法。例如，SJF调度将搜索列表以查找下一个CPU突发最短的任务。或者，可以根据调度标准（即，按优先级）对列表进行排序。另一种策略是为每个唯一的优先级设置一个单独的队列，如图5.7所示。这些方法在5.3.6节中简要讨论。还值得强调的是，我们在使用术语列表和队列时有些可互换。但是，队列具有非常特定的FIFO功能，而列表没有严格的插入和删除要求。您可能会发现一般列表的功能更适合完成此项目。

## II. C实施细节

文件driver.c读取任务计划，将每个任务插入链表，然后通过调用schedule（）函数来调用流程计划程序。schedule（）函数根据指定的调度算法执行每个任务。选择要在CPU上执行的任务由pick-NextTask（）函数确定，并通过调用CPU.c文件中定义的run（）函数来执行。Makefile用于确定驱动程序将调用的特定调度算法。例如，要构建FCFS调度程序，我们将输入

```
使FCFS
```

并将执行调度程序（使用任务schedule.txt的调度），如下所示：

```
./fcfs schedule.txt
```

有关更多详细信息，请参阅源代码下载中的README文件。在继续之前，请确保熟悉提供的源代码以及Makefile。

## III. Java实现细节

Driver.java文件读取任务计划，将每个任务插入Java ArrayList，然后通过调用schedule（）方法来调用流程计划程序。以下接口标识了一种通用调度算法，这五个不同的调度算法将实现该算法：

```
公共接口算法
{
 //调度算法的实现public void
  schedule（）；

  //选择要调度的下一个任务public
  Task pickNetTask（）；
}
```

schedule（）方法通过调用pickNextTask（）方法来获取下一个要在CPU上运行的任务，然后通过调用CPU.java类中的static run（）方法来执行此Task。

该程序运行如下：

```
Java驱动程序fcfs schedule.txt
```

有关更多详细信息，请参阅源代码下载中的README文件。在继续之前，请确保熟悉源代码下载中提供的所有Java源文件。

## IV.进一步的挑战

此项目提出了两个其他挑战：

1. 提供给调度程序的每个任务都分配有一个唯一的任务（tid）。如果调度程序在每个CPU分别运行其自己的调度程序的SMP环境中运行，则用于分配任务标识符的变量可能存在竞争状态。使用原子整数修复此竞争条件。

   在Linux和macOS系统上，sync_fetch和add()函数可用于原子地递增整数值。例如，以下代码示例自动将value增加1：

   ```
   int值= 0;
   _同步获取并添加（&value，1）；
   ```

   有关如何对Java程序使用AtomicInteger类的详细信息，请参考Java API。

2. 计算每种调度算法的平均周转时间，等待时间和响应时间。