2017/11/29

操作系统实验报告

实验二 时间片轮转 RR 进程调度算法

软件工程一班_秦源_1525161007

一　需求分析

问题描述：

设计程序模拟进程的时间片轮转 RR 调度过程。假设有 n 个进程分别在 T1，…，Tn 时刻到达系统，它们需要的服务时间分别为 S1，…，Sn。分别利用不同的时间片大小 q，采用时间片轮转 RR 进程调度算法进行调度，计算每个进程的完成时间、周转时间和带权周转时间，并且统计 n 个进程的平均周转时间和平均带权周转时间。

程序要求：

1）进程个数 n；每个进程的到达时间 T1，…，Tn 和服务时间 S1，…，Sn；输入时间片大小 q。

2）要求时间片轮转法 RR 调度进程运行，计算每个进程的周转时间和带权周转时间，并且计算所有进程的平均周转时间和带权平均周转时间；

3）输出：要求模拟整个调度过程，输出每个时刻的进程运行状态，如“时刻 3：进程 B 开始运行”等等；

4）输出：要求输出计算出来的每个进程的周转时间、带权周转时间、所有进程的平均周转时间以及带权平均周转时间。

二　概要设计

时间片轮转法中，系统将所有的就绪进程按先来先服务的原则排成一个队列，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该进程的执行，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。

这样就可以保证就绪队列中的所有进程在一给定的时间内均能获得一时间片的处理机执行时间。换言之，系统能在给定的时间内响应所有用户的请求。

三　详细设计

main.sh文件声明了静态变量以及默认数据。

Func.sh文件定义函数以及主要的算法。(细节见原代码注释)

四　调试分析

在时间片轮转算法中，时间片的大小对系统性能有很大的影响，如选择很小的时间片将有利于短作业，因为它能较快地完成，但会频繁地发生中断、进程上下文的切换，从而增加系统的开销；反之，如选择太长的时间片，使得每个进程都能在一个时间片内完成，时间片轮转算法便退化为FCFS算法，无法满足交互式用户的需求。

一个较为可取的大小是，时间片略大于一次典型的交互所需要的时间。这样可使大多数进程在一个时间片内完成。

五　用户使用说明

使用终端，将运行目录切换到源代码所在路径，运行main.sh。根据提示即可使用。

六　测试结果

```
bogon:SourceCode qinyuan$ ./main.sh
实验二　时间片轮转RR进程调度算法
----------默认数据-------------
进程名            A B C D E
到达时间          0 1 2 3 4
服务时间          4 3 5 2 4
时间片长度        1
------------------------------
实验数据选择 1-使用默认数据， 2-输入新数据
1
---- 执行时间片长度为1的轮转RR进程调度算法 -----
时间0:  进程A开始运行.
时间1:  进程A停止.
时间1:  进程B开始运行.
时间2:  进程B停止.
时间2:  进程A开始运行.
时间3:  进程A停止.
时间3:  进程C开始运行.
时间4:  进程C停止.
时间4:  进程B开始运行.
时间5:  进程B停止.
时间5:  进程D开始运行.
时间6:  进程D停止.
时间6:  进程A开始运行.
时间7:  进程A停止.
时间7:  进程E开始运行.
时间8:  进程E停止.
时间8:  进程C开始运行.
时间9:  进程C停止.
时间9:  进程B开始运行.
时间10: 进程B结束.
时间10: 进程D开始运行.
时间11: 进程D结束.
时间11: 进程A开始运行.
时间12: 进程A结束.
时间12: 进程E开始运行.
时间13: 进程E停止.
时间13: 进程C开始运行.
时间14: 进程C停止.
时间14: 进程E开始运行.
时间15: 进程E停止.
时间15: 进程C开始运行.
时间16: 进程C停止.
时间16: 进程E开始运行.
时间17: 进程E结束.
时间17: 进程C开始运行.
时间18: 进程C结束.
```

```
进程A周转时间:12        带权周转时间:3.00
进程B周转时间:9         带权周转时间:3.00
进程C周转时间:16        带权周转时间:3.20
进程D周转时间:8         带权周转时间:4.00
进程E周转时间:13        带权周转时间:3.25

平均周转时间:   11.60
平均带权周转时间:      3.29
```

七　附录

----------main.sh----------

```
#! /bin/bash
. func.sh
#进程名对照数组
PID=('A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L')
#到达时间
ArrivalTime=(0 1 2 3 4)
#服务时间
ServiceTime=(4 3 5 2 4)
#时间片长度
q=1
echo "实验二　时间片轮转RR进程调度算法"
echo "----------默认数据----------"
echo -e "进程名　\t${PID[@]:0:${#ArrivalTime[@]}}"
echo -e "到达时间\t${ArrivalTime[@]}"
echo -e "服务时间\t${ServiceTime[@]}"
echo -e "时间片长度\t$q"
```

```
echo "------------------------------"

echo "实验数据选择 1-使用默认数据, 2-输入新数据"
read keypress
case "$keypress" in
     1)
          echo "---- 执行时间片长度为${q}的轮转RR进程调度算法 ----"
          RR
          ;;
     2)
          echo "请输入到达时间:"
          read -a arrive_array
          if [[ "${#arrive_array[@]}" -eq 0 ]]; then
               echo "到达时间的长度不能为0"
               exit
          fi
          echo "请输入服务时间:"
          read -a service_array
          if [[ "${#arrive_array[@]}" -nt "${#service_array[@]}" ]]; then
               echo "到达时间与服务时间长度不匹配"
               exit
          fi
          echo "请输入时间片长度:"
```

```
            read new_q
            if [[ "${new_q}" -eq 0 ]]; then
                echo "时间片的长度不能为0"
                exit
            fi
            ArrivalTime=("${arrive_array[@]}")
            ServiceTime=("${service_array[@]}")
            q="${new_q}"
            echo "------------新数据-------------"
            echo -e "进程名  \t${PID[@]:0:${#ArrivalTime[@]}}"
            echo -e "到达时间\t${ArrivalTime[@]}"
            echo -e "服务时间\t${ServiceTime[@]}"
            echo -e "时间片长度\t$q"
            echo "------------------------------"
            echo
            echo "----- 执行时间片为${q}的轮转RR进程调度算法 -----
"
            RR
            ;;
        *)
            echo "输入无效,请输入 '1' 或 '2' 选择!"
            ;;
esac
-------------------------func.sh-----------------------------
------------------------------------------------------------
```

```
#! /bin/bash


#初始化条件
INIT(){
#小数点保留位数
sc=2
#结束时间
declare -a FinishTime
#临时存储到达时间,当数组为空时所有进程进入队列
TempArrivalTime=("${ArrivalTime[@]}")
#临时存储服务时间,当服务时间为0时进程结束
TempServiceTime=("${ServiceTime[@]}")
}
#判断当前时间是否有新进程到达, 到达加入到NowQueue中
ADD(){
#当存在进程未开始时
until [[ "${#TempArrivalTime[@]}" -eq 0 ]]; do
      #当前进程到达时间小于等于当前时间时加入到队列中
      if [[ "${TempArrivalTime[0]}" -le "${NowTime}" ]]; then
            NowQueue[${#NowQueue[@]}]=$[${#ArrivalTime[@]}-
${#TempArrivalTime[@]}]
            unset TempArrivalTime[0]
            TempArrivalTime=("${TempArrivalTime[@]}")
      else
            break
```

```
        fi
done
}
RR(){
INIT
#当前时刻,进程1到达时间为初次当前时刻
NowTime="${ArrivalTime[0]}"
#当前进程队列,进程1在队列中
declare -a NowQueue
#结束进程个数
Finished=0
#当所有进程结束时退出循环
until [[ "$Finished" -eq "${#ServiceTime[@]}" ]]; do
        #当存在未加入执行队列的进程时检测
        if [[ "${#TempArrivalTime[@]}" -gt 0 ]]; then
                ADD
        fi
        #开始执行进程
        echo -e "时间$NowTime:\t进程${PID[${NowQueue[0]}]}开始运
行."
        #时间片长度大于等于当前进程剩余时间长度
        if [[ "$q" -ge  "${TempServiceTime[${NowQueue[0]}]}" ]];
then

        NowTime=$[${NowTime}+${TempServiceTime[${NowQueue[0]}]}]
```

```
            FinishTime[${NowQueue[0]}]="$NowTime"
            echo -e "时间$NowTime:\t进程${PID[${NowQueue[0]}]}结束."
            unset NowQueue[0]
            NowQueue=("${NowQueue[@]}")
            Finished=$[${Finished}+1]
      #时间片长度小于当前进程剩余时间长度
      else
            LeftServiceId=${NowQueue[0]}
            unset NowQueue[0]
            NowQueue=("${NowQueue[@]}")
            NowTime=$[${NowTime}+$q]
            if [[ "${#TempArrivalTime[@]}" -gt 0 ]]; then
                  ADD
            fi
            NowQueue[${#NowQueue[@]}]="$LeftServiceId"

      TempServiceTime[${LeftServiceId}]=$[${TempServiceTime[${LeftServiceId}]}-$q]
            echo -e "时间$NowTime:\t进程${PID[$LeftServiceId]}停止."
      fi
done
#周转时间  WholeTime
declare -a WholeTime
#带权周转时间   WeightWholeTime
```

```
declare -a WeightWholeTime
#平均周转时间
AverageWT=0
#平均带权周转时间
AverageWWT=0
#计算周转时间及计算带权周转时间
echo
temp_number=$[${#ArrivalTime[@]} - 1]
for i in `seq 0 $temp_number`
do
     WholeTime[i]=$[${FinishTime[i]} - ${ArrivalTime[i]}]
     echo -n -e "进程${PID[i]}周转时间:${WholeTime[i]} \t"
     WeightWholeTime[i]=$(echo
"scale=$sc;${WholeTime[i]}/${ServiceTime[i]}"|bc)
     echo "带权周转时间:${WeightWholeTime[i]} "
     AverageWT=$(echo
"scale=$sc;${AverageWT}+${WholeTime[i]}"|bc)
     AverageWWT=$(echo
"scale=$sc;${AverageWWT}+${WeightWholeTime[i]}"|bc)
done
AverageWT=$(echo "scale=$sc;${AverageWT}/${#ArrivalTime[@]}"|bc)
AverageWWT=$(echo
"scale=$sc;${AverageWWT}/${#ArrivalTime[@]}"|bc)
echo
echo "平均周转时间:  $AverageWT"
echo "平均带权周转时间:   $AverageWWT"
```

```
}
```