|  |  |  |  |
| --- | --- | --- | --- |
|  | 2017/11/29 |  | |
| |  |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | | |  | |
| 操作系统实验报告  *实验一 先来先服务FCFS和短作业优先SJF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先来先服务FCFS和短作业优先SJF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。分别采用先来先服务FCFS和短作业优先SJF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；选择算法1-FCFS，2-SJF。

2）要求采用先来先服务FCFS和短作业优先SJF分别调度进程运行，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

FCFS算法，每次调度是从就绪队列中选择一个最先进入该队列的进程，为之分配处理机，使之投入运行。该进程一直运行到完成或发生某事件而阻塞后才放弃处理机。

SJF算法，则是从就绪队列中选出一个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。

# 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

FCFS算法比较有利于长作业(进程)，而不利于短作业(进程)。

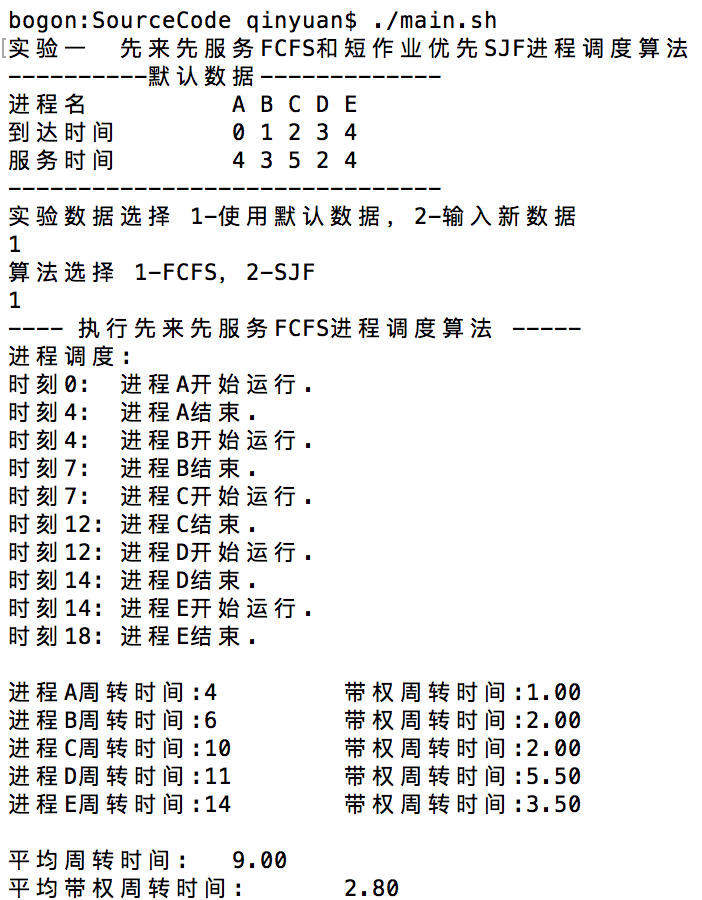
SJF调度算法能有效地降低作业的平均等待时间，提高系统吞吐量，但该算法对长作业不利。

# 五 用户使用说明

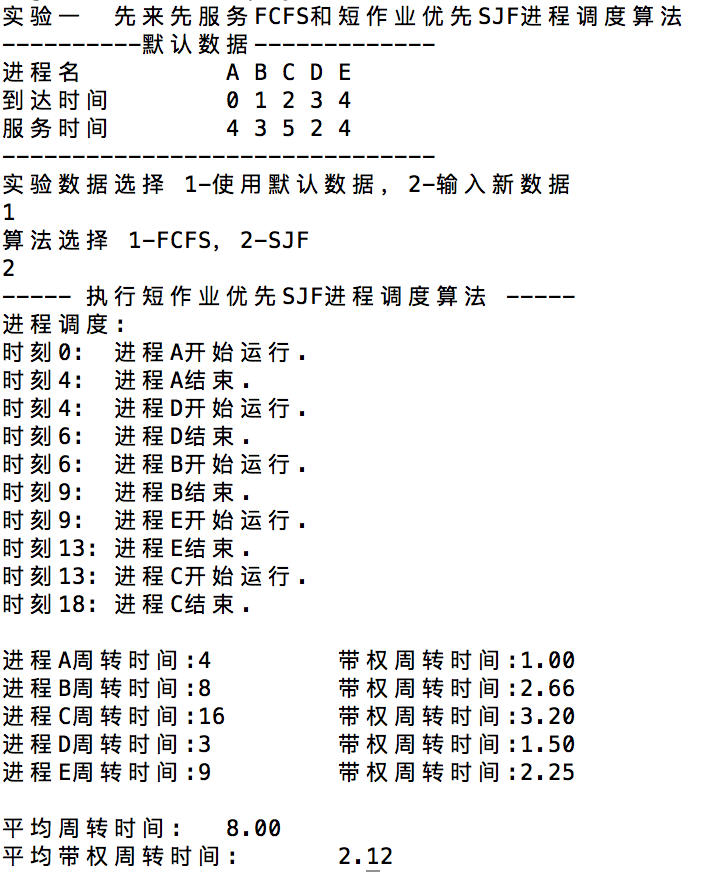
使用终端，将运行目录切换到源代码所在路径，运行main.sh。根据提示即可使用。

# 六 测试结果

FCFS:



SJF:



# 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)

echo "实验一 先来先服务FCFS和短作业优先SJF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 "-------------------------------"

echo "实验数据选择 1-使用默认数据，2-输入新数据"

read keypressData

case "$keypressData" in

1)

;;

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

ArrivalTime=("${arrive\_array[@]}")

ServiceTime=("${service\_array[@]}")

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 "-------------------------------"

echo

;;

\*)

echo "输入无效,请输入 '1' 或 '2' 选择!"

;;

esac

echo "算法选择 1-FCFS，2-SJF"

read keypress

case "$keypress" in

1)

echo "---- 执行先来先服务FCFS进程调度算法 -----"

FCFS

;;

2)

echo "----- 执行短作业优先SJF进程调度算法 -----"

SJF

;;

\*)

echo "输入无效,请输入 '1' 或 '2' 选择!"

;;

esac

---------------------------func.sh----------------------------------------------------------------------------------------------

#! /bin/bash

#初始化

INIT(){

#小数点保留位数

sc=2

}

#先来先服务

FCFS(){

INIT

#完成时间 FinishTime

declare -a FinishTime

#进程数量 p\_number

declare -i p\_number

p\_number="${#ArrivalTime[@]}"

#第一个进程

echo "进程调度:"

echo -e "时刻${ArrivalTime[0]}:\t进程${PID[0]}开始运行."

FinishTime[0]=$[${ArrivalTime[0]}+${ServiceTime[0]}]

echo -e "时刻${FinishTime[0]}:\t进程${PID[0]}结束."

#计数从1开始,从第二个进程开始

temp\_number=$[$p\_number - 1]

for i in `seq 1 $temp\_number`

do

#当前进程到达时，上一进程未结束

if [ ${ArrivalTime[i]} -lt ${FinishTime[i-1]} ];then

echo -e "时刻${FinishTime[i-1]}:\t进程${PID[i]}开始运行."

FinishTime[i]=$[${ServiceTime[i]} + ${FinishTime[i-1]}]

echo -e "时刻${FinishTime[i]}:\t进程${PID[i]}结束."

else

#当前进程到达时，上一进程已结束

echo -e "时刻${ArrivalTime[i]}:\t进程${PID[i]}开始运行."

FinishTime[i]=$[${ServiceTime[i]} + ${ArrivalTime[i]}]

echo -e "时刻${FinishTime[i]}:\t进程${PID[i]}结束."

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\_FCFS=0

#平均带权周转时间

AverageWWT\_FCFS=0

#计算周转时间及计算带权周转时间

echo

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\_FCFS=$(echo "scale=$sc;${AverageWT\_FCFS}+${WholeTime[i]}"|bc)

AverageWWT\_FCFS=$(echo "scale=$sc;${AverageWWT\_FCFS}+${WeightWholeTime[i]}"|bc)

done

AverageWT\_FCFS=$(echo "scale=$sc;${AverageWT\_FCFS}/${p\_number}"|bc)

AverageWWT\_FCFS=$(echo "scale=$sc;${AverageWWT\_FCFS}/${p\_number}"|bc)

echo

echo "平均周转时间: $AverageWT\_FCFS"

echo "平均带权周转时间: $AverageWWT\_FCFS"

}

#短作业优先

#当前FinishTime

nowFinishTime=0

#返回当前服务时间最短且已到达的进程id

NEXT(){

local minTime=999

local resultId=0

local ts=$[$#-1]

shift

for i in `seq 1 $ts`

do

#当前进程已结束

if [ $1 -eq 999 ];then

shift

continue

fi

#当前进程还未到

if [[ ${ArrivalTime[i]} -gt ${nowFinishTime} ]];then

shift

continue

fi

#当前进程服务时间最小

if [[ ${ServiceTime[i]} -lt $minTime ]];then

minTime=${ServiceTime[i]}

resultId=$i

fi

shift

done

#如果所有进程都未到达,返回到达值最小的

if [[ "${resultId}" -eq 0 ]]; then

for i in `seq 1 $[${#ServiceTime[@]}-1]`; do

if [[ "${tempServiceTime[$i]}" -ne 999 ]]; then

nowFinishTime="${ArrivalTime[$i]}"

resultId=$i

break

fi

done

fi

return "$resultId"

}

SJF(){

INIT

#完成时间 FinishTime

declare -a FinishTime

#进程数量 p\_number

declare -i p\_number

p\_number="${#ArrivalTime[@]}"

#第一个进程

echo "进程调度:"

echo -e "时刻${ArrivalTime[0]}:\t进程${PID[0]}开始运行."

FinishTime[0]=$[${ArrivalTime[0]}+${ServiceTime[0]}]

echo -e "时刻${FinishTime[0]}:\t进程${PID[0]}结束."

nowFinishTime=${FinishTime[0]}

#临时数组储存ServiceTime以供删除

tempServiceTime=("${ServiceTime[@]}")

#完成的进程ServiceTime标记为999

tempServiceTime[0]=999

#从第二个进程开始

left\_number=$[$p\_number - 2]

for i in `seq 0 $left\_number`

do

NEXT "${tempServiceTime[@]}"

NEXT\_ID="$?"

echo -e "时刻${nowFinishTime}:\t进程${PID[$NEXT\_ID]}开始运行."

FinishTime[$NEXT\_ID]=$[${nowFinishTime}+${ServiceTime[NEXT\_ID]}]

nowFinishTime=${FinishTime[$NEXT\_ID]}

echo -e "时刻$nowFinishTime:\t进程${PID[$NEXT\_ID]}结束."

tempServiceTime[$NEXT\_ID]=999

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\_SJF=0

#平均带权周转时间

AverageWWT\_SJF=0

#计算周转时间及计算带权周转时间

echo

temp\_number=$[$p\_number - 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\_SJF=$(echo "scale=$sc;${AverageWT\_SJF}+${WholeTime[i]}"|bc)

AverageWWT\_SJF=$(echo "scale=$sc;${AverageWWT\_SJF}+${WeightWholeTime[i]}"|bc)

done

AverageWT\_SJF=$(echo "scale=$sc;${AverageWT\_SJF}/${p\_number}"|bc)

AverageWWT\_SJF=$(echo "scale=$sc;${AverageWWT\_SJF}/${p\_number}"|bc)

echo

echo "平均周转时间: $AverageWT\_SJF"

echo "平均带权周转时间: $AverageWWT\_SJF"

}