2017/11/29

操作系统实验报告

实验六 磁盘调度算法

软件工程一班_秦源_1525161007

一　需求分析

问题描述：

设计程序模拟先来先服务 FCFS、最短寻道时间优先 SSTF、SCAN 和循环 SCAN 算法的工作过程。假设有 n 个磁道号所组成的磁道访问序列，给定开始磁道号 m 和磁头移动的方向（正向或者反向），分别利用不同的磁盘调度算法访问磁道序列，给出每一次访问的磁头移动距离，计算每种算法的平均寻道长度。

程序要求：

1）利用先来先服务 FCFS、最短寻道时间优先 SSTF、SCAN 和循环 SCAN 算法模拟磁道访问过程。

2）模拟四种算法的磁道访问过程，给出每个磁道访问的磁头移动距离。

3）输入：磁道个数 n 和磁道访问序列，开始磁道号 m 和磁头移动方向（对 SCAN 和循环 SCAN 算法有效），算法选择 1-FCFS，2-SSTF，3-SCAN，4-循环 SCAN。

4）输出：每种算法的平均寻道长度。

二　概要设计

先来先服务(FCFS，First Come First Served)：

这是一种最简单的磁盘调度算法。它根据进程请求访问磁盘的先后次序进行调度。

最短寻道时间优先(SSTF，Shortest Seek Time First)：

该算法选择这样的进程：其要求访问的磁道与当前磁头所在的磁道距离最近，以使每次的寻道时间最短。

扫描(SCAN)算法：

算法不仅考虑到欲访问的磁道与当前磁道间的距离，更优先考虑的是磁头当前的移动方向。例如，当磁头正在自里向外移动时，SCAN 算法所考虑的下一个访问对象，应是其欲访问的磁道既在当前磁道之外，又是距离最近的。

循环扫描(CSCAN)算法：

CSCAN 算法规定磁头单向移动，例如，只是自里向外移动，当磁头移到最外的磁道并访问后，磁头立即返回到最里的欲访问的磁道，亦即将最小磁道号紧接着最大磁道号构成循环，进行循环扫描。

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

先来先服务算法的优点是公平、简单，且每个进程的请求都能依次地得到处理，不会出现某一进程的请求长期得不到满足的情况。但此算法由于未对寻道进行优化，致使平均寻道时间可能较长。

最短寻道时间优先算法不能保证平均寻道时间最短。

SCAN 算法既能获得较好的寻道性能， 但 SCAN 也存在这样的问题：当磁头刚从里向外移动而越过了某一磁道时，恰好又有一进程请求访问此磁道，这时，该进程必须等待，待磁头继续从里向外，然后再从外向里扫描完所有要访问的磁道后，才处理该进程的请求，致使该进程的请求被大大地推迟。为了减少这种延迟，CSCAN 算法规定磁头单向移动，例如，只是自里向外移动，当磁头移到最外的磁道并访问后，磁头立即返回到最里的欲访问的磁道，亦即将最小磁道号紧接着最大磁道号构成循环，进行循环扫描。

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

先来先服务算法：

```
bogon:SourceCode qinyuan$ ./main.sh
实验六　磁盘调度算法
----------默认数据----------
磁道个数：        9
开始磁道号：      100
磁盘访问序列：    55 58 39 18 90 160 150 38 184
磁头移动方向：    向外
--------------------------------
实验数据选择 1-使用默认数据，2-输入新数据
1
算法选择：1-先来先服务FCFS 2-最短寻道时间优先SSTF 3-扫描SCAN 4-循环扫描SCAN
1
----------执行先来先服务FCFS算法----------
开始磁道号：  100
被访问的下一个磁道号：    55      移动距离：    45
被访问的下一个磁道号：    58      移动距离：    3
被访问的下一个磁道号：    39      移动距离：    19
被访问的下一个磁道号：    18      移动距离：    21
被访问的下一个磁道号：    90      移动距离：    72
被访问的下一个磁道号：    160     移动距离：    70
被访问的下一个磁道号：    150     移动距离：    10
被访问的下一个磁道号：    38      移动距离：    112
被访问的下一个磁道号：    184     移动距离：    146
平均寻道长度：    55.33
bogon:SourceCode qinyuan$ ./main.sh
```

最短寻道时间优先算法：

```
bogon:SourceCode qinyuan$ ./main.sh
实验六　磁盘调度算法
----------默认数据----------
磁道个数：        9
开始磁道号：      100
磁盘访问序列：    55 58 39 18 90 160 150 38 184
磁头移动方向：    向外
--------------------------------
实验数据选择 1-使用默认数据，2-输入新数据
1
算法选择：1-先来先服务FCFS 2-最短寻道时间优先SSTF 3-扫描SCAN 4-循环扫描SCAN
2
----------执行最短寻道时间优先SSTF算法----------
开始磁道号：  100
被访问的下一个磁道号：    90      移动距离：    10
被访问的下一个磁道号：    58      移动距离：    32
被访问的下一个磁道号：    55      移动距离：    3
被访问的下一个磁道号：    39      移动距离：    16
被访问的下一个磁道号：    38      移动距离：    1
被访问的下一个磁道号：    18      移动距离：    20
被访问的下一个磁道号：    150     移动距离：    132
被访问的下一个磁道号：    160     移动距离：    10
被访问的下一个磁道号：    184     移动距离：    24
平均寻道长度：    27.55
```

SCAN 算法：

```
bogon:SourceCode qinyuan$ ./main.sh 1
实验六   磁盘调度算法
----------默认数据-------------
磁道个数：        9
开始磁道号：      100
磁盘访问序列：    55 58 39 18 90 160 150 38 184
磁头移动方向：    向外
------------------------------
实验数据选择  1-使用默认数据，  2-输入新数据
1
算法选择：1-先来先服务FCFS  2-最短寻道时间优先SSTF  3-扫描SCAN  4-循环扫描SCAN
3
----------执行扫描SCAN算法----------
开始磁道号：  100
被访问的下一个磁道号：    150      移动距离：    50
被访问的下一个磁道号：    160      移动距离：    10
被访问的下一个磁道号：    184      移动距离：    24
被访问的下一个磁道号：    90       移动距离：    94
被访问的下一个磁道号：    58       移动距离：    32
被访问的下一个磁道号：    55       移动距离：    3
被访问的下一个磁道号：    39       移动距离：    16
被访问的下一个磁道号：    38       移动距离：    1
被访问的下一个磁道号：    18       移动距离：    20
平均寻道长度：    27.77
```

CSCAN 算法：

```
bogon:SourceCode qinyuan$ ./main.sh
实验六   磁盘调度算法
----------默认数据-------------
磁道个数：        9
开始磁道号：      100
磁盘访问序列：    55 58 39 18 90 160 150 38 184
磁头移动方向：    向外
------------------------------
实验数据选择  1-使用默认数据，  2-输入新数据
1
算法选择：1-先来先服务FCFS  2-最短寻道时间优先SSTF  3-扫描SCAN  4-循环扫描SCAN
4
----------执行循环扫描SCAN算法----------
开始磁道号：  100
被访问的下一个磁道号：    150      移动距离：    50
被访问的下一个磁道号：    160      移动距离：    10
被访问的下一个磁道号：    184      移动距离：    24
被访问的下一个磁道号：    18       移动距离：    166
被访问的下一个磁道号：    38       移动距离：    20
被访问的下一个磁道号：    39       移动距离：    1
被访问的下一个磁道号：    55       移动距离：    16
被访问的下一个磁道号：    58       移动距离：    3
被访问的下一个磁道号：    90       移动距离：    32
平均寻道长度：    35.77
```

七　附录

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

```
#! /bin/bash
. func.sh
#小数点保留位数
sc=2
#磁道个数n
n=9
#开始磁道号m
m=100
#磁盘访问序列
TrackOrder=(55 58 39 18 90 160 150 38 184)
#磁头移动方向(对SCAN和循环SCAN算法有效)定义为1向外,0向内
declare -i direction
direction=1
echo "实验六　磁盘调度算法"
echo "----------默认数据----------"
echo -e "磁道个数:\t${n}"
echo -e "开始磁道号:\t${m}"
echo -e "磁盘访问序列:\t${TrackOrder[@]}"
echo -e "磁头移动方向:\t向外"
echo "------------------------------"
echo "实验数据选择 1-使用默认数据, 2-输入新数据"
```

```
read keypressData
case "$keypressData" in
    1 )
        ;;
    2 )
        echo "请输入磁道个数n:"
        read new_n
        n="${new_n}"
        echo "请输入开始磁道号m:"
        read new_m
        m="${new_m}"
        echo "请输入磁盘访问序列:(长度${n})"
        read -a new_P
        TrackOrder=("${new_P[@]}")
        echo "请输入磁头移动方向:(向外为1,向内为0)"
        read new_direction
        direction=new_direction
        ;;
    * )
        echo "输入无效,请输入 '1' 或 '2' 选择!"
        exit
        ;;
esac
echo "算法选择: 1-先来先服务FCFS 2-最短寻道时间优先SSTF 3-扫描SCAN 4-循环扫描SCAN"
```

```
read keypressKind
case "$keypressKind" in
     1 )
          echo "----------执行先来先服务FCFS算法----------"
          FCFS
          ;;
     2 )
          echo "----------执行最短寻道时间优先SSTF算法----------
"
          SSTF
          ;;
     3 )
          echo "----------执行扫描SCAN算法----------"
          SCAN
          ;;
     4 )
          echo "----------执行循环扫描SCAN算法----------"
          CYCLESCAN
          ;;
     * )
          echo "输入无效,请输入 '1' 或 '2' 或 '3' 或 '4' 选择!"
          exit
          ;;
esac
```

```
----------------------------func.sh-------------------------------
------------------------------------------------------------------
#! /bin/bash
ININT(){
	declare -a MoveDistance
	declare AverageDistance
	declare abs_result
}
#计算两个值的绝对值
abs(){
	#第一个数大于等于第二个数
	if [[ "$1" -ge "$2" ]]; then
		abs_result=$[$1-$2]
		return 0
	#第一个数小于第二个数
	else
		abs_result=$[$2-$1]
		return 1
	fi
}
#先来先服务FCFS
FCFS(){
	local start="$m"
	local totalLength=0
	echo -e "开始磁道号:  ${start}"
```

```
    for i in `seq 0 $[$n-1]`; do
        echo -ne "被访问的下一个磁道号:\t${TrackOrder[$i]}"
        abs ${start} ${TrackOrder[$i]}
        MoveDistance[$i]=${abs_result}
        totalLength=$[${totalLength}+${abs_result}]
        start=${TrackOrder[$i]}
        echo -e "\t移动距离:   ${MoveDistance[$i]}"
    done
   AverageDistance=$(echo "scale=${sc};${totalLength}/${n}"|bc)
    echo -e "平均寻道长度:\t${AverageDistance}"
}
#最短寻道时间优先SSTF
SSTF(){
    local start="$m"
    local totalLength=0
    local tempTrackOrder=("${TrackOrder[@]}")
    #最近的磁道号
    local shortestIndex=-1
    echo -e "开始磁道号:  ${start}"
    for i in `seq 0 $[$n-1]`; do
        #定义最近距离
        shortestLength=999
        #寻找最近的磁道
        for j in `seq 0 $[$n-1]`; do
```

```
                if [[ "${tempTrackOrder[$j]}" -eq -999 ]]; then
                    continue
                fi
                abs ${start} ${tempTrackOrder[$j]}
               if [[ "${shortestLength}" -gt  "${abs_result}" ]];
then
                    shortestLength="${abs_result}"
                    shortestIndex="$j"
                fi
            done
            echo -ne "被访问的下一个磁道
号:\t${tempTrackOrder[${shortestIndex}]}"
            abs ${start} ${tempTrackOrder[${shortestIndex}]}
            MoveDistance[$i]=${abs_result}
            totalLength=$[${totalLength}+${abs_result}]
            start=${tempTrackOrder[${shortestIndex}]}
            tempTrackOrder[${shortestIndex}]=-999
            echo -e "\t移动距离:   ${MoveDistance[$i]}"
      done
    AverageDistance=$(echo  "scale=${sc};${totalLength}/${n}"|bc)
      echo -e "平均寻道长度:\t${AverageDistance}"
}
#排序从小到大
SORT(){
      tempTrackOrder=("${TrackOrder[@]}")
```

```
    for i in `seq 0 $[$n-1]`; do
        MIN=999
        MIN_Index=0
        for j in `seq 0 $[$n-1]`; do
            if [[ "${MIN}" -gt "${tempTrackOrder[$j]}" ]];
then
                MIN=${tempTrackOrder[$j]}
                MIN_Index=$j
            fi
        done
        tempTrackOrder[${MIN_Index}]=999
        TrackOrder[$i]="${MIN}"
    done
}
#扫描SCAN
SCAN(){
    SORT
    local start="$m"
    local totalLength=0
    echo -e "开始磁道号:  ${start}"
    #当开始磁道号m最小
    if [[ "${start}" -lt "${TrackOrder[0]}" ]]; then
        for i in `seq 0 $[$n-1]`; do
            echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
```

```
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
            done
            AverageDistance=$(echo
"scale=${sc};${totalLength}/${n}"|bc)
            echo -e "平均寻道长度:\t${AverageDistance}"
            return
        fi
        #当开始磁道号m最大
        if [[ "${start}" -gt "${TrackOrder[$[$n-1]]}" ]]; then
            for (( i = $[$n-1]; i >= 0; i-- )); do
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
            done
            AverageDistance=$(echo
"scale=${sc};${totalLength}/${n}"|bc)
            echo -e "平均寻道长度:\t${AverageDistance}"
```

```
            return
        fi
        #当开始磁道号在中间,寻找稍大的位置Index
        midIndex=-1
        for i in `seq 0 $[$n-1]`; do
            if [[ "$m" -lt "${TrackOrder[$i]}" ]]; then
                midIndex="$i"
                break
            fi
        done
        #根据direction判断方向
        #1时向外
        if [[ "${direction}" -eq 1 ]]; then
            for (( i = ${midIndex}; i < ${n}; i++ )); do
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
            done
            for (( i = $[${midIndex}-1]; i >= 0; i-- )); do
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
```

```
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
            done
            AverageDistance=$(echo
"scale=${sc};${totalLength}/${n}"|bc)
            echo -e "平均寻道长度:\t${AverageDistance}"
        #0时向内
        elif [[ "${direction}" -eq 0 ]]; then
            for (( i = $[${midIndex}-1]; i >= 0; i-- )); do
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
            done
            for (( i = ${midIndex}; i < ${n}; i++ )); do
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
```

```
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:    ${MoveDistance[$i]}"
        done
        AverageDistance=$(echo
"scale=${sc};${totalLength}/${n}"|bc)
        echo -e "平均寻道长度:\t${AverageDistance}"
    else
        echo "方向错误! '1'时向外 '0'向内."
    fi
}
#循环扫描SCAN
CYCLESCAN(){
    SORT
    local start="$m"
    local totalLength=0
    echo -e "开始磁道号:  ${start}"
    #开始磁道号在两侧时结果一致,默认0,不在两端
    Judge=0
    #当开始磁道号m最小
    if [[ "${start}" -lt "${TrackOrder[0]}" ]]; then
        Judge=1
    fi
    #当开始磁道号m最大
    if [[ "${start}" -gt "${TrackOrder[$[$n-1]]}" ]]; then
```

```
            Judge=1
        fi
        if [[ "${Judge}" -eq 1 ]]; then
            #1时向外
            if [[ "${direction}" -eq 1 ]]; then
                for i in `seq 0 $[$n-1]`; do
                    echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                    abs ${start} ${TrackOrder[$i]}
                    MoveDistance[$i]=${abs_result}
                    totalLength=$[${totalLength}+${abs_result}]
                    start="${TrackOrder[$i]}"
                    echo -e "\t移动距离:   ${MoveDistance[$i]}"
                done
                AverageDistance=$(echo
"scale=${sc};${totalLength}/${n}"|bc)
                echo -e "平均寻道长度:\t${AverageDistance}"
                return
            #0时向内
            elif [[ "${direction}" -eq 0 ]]; then
                for (( i = $[$n-1]; i >= 0; i-- )); do
                    echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                    abs ${start} ${TrackOrder[$i]}
                    MoveDistance[$i]=${abs_result}
```

```
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
            done
            AverageDistance=$(echo
"scale=${sc};${totalLength}/${n}"|bc)
            echo -e "平均寻道长度:\t${AverageDistance}"
            return
        else
            echo "方向错误! '1'时向外 '0'向内."
        fi
    fi
    #当开始磁道号在中间,寻找稍大的位置Index
    midIndex=-1
    for i in `seq 0 $[$n-1]`; do
        if [[ "$m" -lt "${TrackOrder[$i]}" ]]; then
            midIndex="$i"
            break
        fi
    done
    #根据direction判断方向
    #1时向外
    if [[ "${direction}" -eq 1 ]]; then
        for (( i = ${midIndex}; i < ${n}; i++ )); do
```

```
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
        done
        for (( i = 0; i < ${midIndex}; i++ )); do
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
        done
        AverageDistance=$(echo
"scale=${sc};${totalLength}/${n}"|bc)
        echo -e "平均寻道长度:\t${AverageDistance}"
    #0时向内
    elif [[ "${direction}" -eq 0 ]]; then
        for (( i = $[${midIndex}-1]; i >= 0; i-- )); do
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                abs ${start} ${TrackOrder[$i]}
```

```
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
            done
            for (( i = $[${n}-1]; i >= ${midIndex}; i-- )); do
                echo -ne "被访问的下一个磁道
号:\t${TrackOrder[$i]}"
                abs ${start} ${TrackOrder[$i]}
                MoveDistance[$i]=${abs_result}
                totalLength=$[${totalLength}+${abs_result}]
                start="${TrackOrder[$i]}"
                echo -e "\t移动距离:   ${MoveDistance[$i]}"
            done
            AverageDistance=$(echo
"scale=${sc};${totalLength}/${n}"|bc)
            echo -e "平均寻道长度:\t${AverageDistance}"
        else
            echo "方向错误! '1'时向外 '0'向内."
        fi
}
```