实 验 报 告

课程名称：操作系统试验

实 验 二：进程调度

班　　级：

学生姓名：

学　　号：

专　　业：智能科学与技术

指导教师：

学　　期：2022－2023 学年秋季学期

成　　绩：

云南大学信息学院

一、实验目的

1、熟悉进程的定义和描述，熟悉进程控制块；

2、掌握进程的状态定义及其转换过程；

3、掌握进程的基本调度算法，包括先来先服务，轮转法，优先级法，多级反馈轮转法，最短进程优先法，最高响应比优先法等；

二、知识要点

1、进程控制块 PCB；

2、进程的初始化、就绪、执行、等待和终止状态；

3、先来先服务，轮转法，优先级法，多级反馈轮转法，最短进程优先法，最高响应比优先法等调度算法；

三、实验预习（要求做实验前完成）

1、了解 linux 系统中常用命令的使用方法；

2、掌握进程 PCB 控制块的内容和描述；

3、掌握系统状态的转换过程；

4、掌握常用进程调度算法的原理

四、实验内容和试验结果

结合课程所讲授内容以及课件中的试验讲解，完成以下试验。请分别对试验过程和观察到的情况做描述和总结，并将试验结果截图附后。

1、通过编程模拟实现轮转法进程调度算法。

系统中的每个进程用一个进程控制块 PCB 表示；将多个进程按输入顺序排成就绪队列链表（进程信息从键盘录入）；按进程在链表中的顺序依次调度，每个被调度的进程执行一个时间片，然后回到就绪队列，“已运行时间”加 1；若进程“要求运行时间”==“已运行时间”，则将其状态置为“结束”，并退出队列；运行程序，显示每次调度时被调度运行的进程 id，以及各进程控制块的动态变化过程。

```
// 先来先服务
int Init_PCB_FCFS()
{
    int i; // 在循环输入 PCB 信息时的计数器
    Proc temp, ptr;
    // 输入第一个进程的信息
    printf("Please input the number of processes|请输入进程个数:");
```

```
    scanf("%d", &Proc_num);
    printf("There are %d processes, please input PCB info|有%d 个进程，请依次输入 PCB 信息:\n", Proc_num, Proc_num);
    temp = (Proc)malloc(sizeof(struct PCB)); // 进程 temp
    printf("----------------Process %2d----------------\n", 1);
    printf("\tProcess id|PID/进程 ID:");
    scanf("%d", &temp->PID);
    printf("\tCPU time required|所需 CPU 时间:");
    scanf("%d", &temp->Total_time);
    temp->State = true;
    temp->CPU_time = 0; // 就绪，且已运行时间为 0
    head = temp;
    tail = temp;
    // 依次输入之后 Proc_num-1 个进程的信息
    for (i = Proc_num; i > 1; i--)
    {
        ptr = temp; // 先指向上一次的结点
        temp = (Proc)malloc(sizeof(struct PCB));
        printf("----------------Process %2d----------------\n", Proc_num - i + 2);
        printf("\tProcess id|PID/进程 ID:");
        scanf("%d", &temp->PID);
        printf("\tCPU time required|所需 CPU 时间:");
        scanf("%d", &temp->Total_time);
        temp->State = true;
        temp->CPU_time = 0;
        ptr->Next_PCB = temp;
    }
    tail = temp;
    tail->Next_PCB = head;
    printf("----------------初始化完成----------------\n");
    return 0;
}
```

```
// 先来先服务
void Display_FCFS()
{
    int i;
    Proc ptr = head;
    printf(" | PID|进程 ID\tCPU_Time|已运行时间\tReq_Time|进程所需总时间|\n");
    for (i = 0; i < Proc_num; i++)
    {
        printf(" | %-10d\t%-19d\t%-23d |\n", ptr->PID, ptr->CPU_time, ptr->Total_time);
```

```
        ptr = ptr->Next_PCB;
    }
}
```

```
// 先来先服务
void Sched_FCFS()
{
    int round = 1, i;
    Proc temp = tail;
    Proc p = head;
    while (p->Total_time > p->CPU_time)
    {
        printf("\nRound %d, Process %d is running|第%d 次轮转，进程%d 运行中
\n", round, p->PID, round, p->PID);
        p->CPU_time++;
        Display_FCFS();
        if (p->Total_time == p->CPU_time)
        {                    // 进程运行时间已达到要求的时间，表示该进程已执行
完毕
            p->State = false; // 置结束标志
            Proc_num--;
            temp->Next_PCB = p->Next_PCB;
            if (p == head)
                head = p->Next_PCB; // 队列后移
            printf("\tProcess %d is finished|进程%d 执行完成\n", p->PID,
p->PID);
        }
        else
            temp = p;
        p = p->Next_PCB;
        round++;
    }
}
```

```
Please input the number of processes|请输入进程个数:3
There are 3 processes, please input PCB info|有3个进程，请依次输入PCB信息:
-----------------Process  1-----------------
        Process id|PID/进程ID:1
        CPU time required|所需CPU时间:3
-----------------Process  2-----------------
        Process id|PID/进程ID:2
        CPU time required|所需CPU时间:2
-----------------Process  3-----------------
        Process id|PID/进程ID:3
        CPU time required|所需CPU时间:6
-----------------初始化完成-----------------
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 1           0                      3                      |
 | 2           0                      2                      |
 | 3           0                      6                      |
```

```
Round 1, Process 1 is running|第1次轮转，进程1运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 1            1                       3                      |
 | 2            0                       2                      |
 | 3            0                       6                      |

Round 2, Process 2 is running|第2次轮转，进程2运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 1            1                       3                      |
 | 2            1                       2                      |
 | 3            0                       6                      |

Round 3, Process 3 is running|第3次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 1            1                       3                      |
 | 2            1                       2                      |
 | 3            1                       6                      |

Round 4, Process 1 is running|第4次轮转，进程1运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 1            2                       3                      |
 | 2            1                       2                      |
 | 3            1                       6                      |

Round 5, Process 2 is running|第5次轮转，进程2运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 1            2                       3                      |
 | 2            2                       2                      |
 | 3            1                       6                      |
       Process 2 is finished|进程2执行完成

Round 6, Process 3 is running|第6次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 1            2                       3                      |
 | 3            2                       6                      |

Round 7, Process 1 is running|第7次轮转，进程1运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 1            3                       3                      |
 | 3            2                       6                      |
       Process 1 is finished|进程1执行完成

Round 8, Process 3 is running|第8次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 3            3                       6                      |

Round 9, Process 3 is running|第9次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 3            4                       6                      |

Round 10, Process 3 is running|第10次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 3            5                       6                      |

Round 11, Process 3 is running|第11次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 |
 | 3            6                       6                      |
       Process 3 is finished|进程3执行完成
```

2、参考第一题的描述，通过编程模拟实现动态优先级轮转调度算法。

```
// 动态最高优先级
int Init_PCB_DHPF()
{
    int i; // 在循环输入 PCB 信息时的计数器
    Proc temp;
    // 输入第一个进程的信息
    printf("Please input the number of processes|请输入进程个数:");
    scanf("%d", &Proc_num);
    printf("There are %d processes, please input PCB info|有%d 个进程，请依
次输入 PCB 信息:\n", Proc_num, Proc_num);
    temp = (Proc)malloc(sizeof(struct PCB)); // 进程 temp
    printf("----------------Process %2d----------------\n", 1);
    printf("\tProcess id|PID/进程 ID:");
    scanf("%d", &temp->PID);
    printf("\tCPU time required|所需 CPU 时间:");
    scanf("%d", &temp->Total_time);
    printf("\tProcess priority|进程优先级:");
    scanf("%d", &temp->Priority);
    temp->State = true;
    temp->CPU_time = 0; // 就绪，且已运行时间为 0
    head = temp;
    tail = temp;
    // 依次输入之后 Proc_num-1 个进程的信息
    for (i = Proc_num; i > 1; i--)
    {
        temp = (Proc)malloc(sizeof(struct PCB));
        printf("----------------Process %2d----------------\n", Proc_num
- i + 2);
        printf("\tProcess id|PID/进程 ID:");
        scanf("%d", &temp->PID);
        printf("\tCPU time required|所需 CPU 时间:");
        scanf("%d", &temp->Total_time);
        printf("\tProcess priority|进程优先级:");
        scanf("%d", &temp->Priority);
        temp->State = true;
        temp->CPU_time = 0;
        if (temp->Priority >= head->Priority)
        { // 进程 temp 优先级最高，将其放在队头
            temp->Next_PCB = head;
            head = temp;
        }
        else
        {
```

```
            if (temp->Priority <= tail->Priority)
            { // 进程 temp 优先级最低，将其放在队尾
                tail->Next_PCB = temp;
                tail = temp;
            }
            else
            { // temp 优先级在队列中，则循环找到应该插入队列的地方
                Proc pf = head, pa = head->Next_PCB;
                while (1)
                {
                    if (temp->Priority >= pa->Priority)
                    {
                        pf->Next_PCB = temp;
                        temp->Next_PCB = pa;
                        break;
                    }
                    else
                    {
                        pf = pf->Next_PCB;
                        pa = pa->Next_PCB;
                    }
                }
            }
        }
    }
    tail->Next_PCB = head;
    printf("----------------初始化完成----------------\n");
    return 0;
}
```

```
// 动态最高优先级
void Display_DHPF()
{
    int i;
    Proc ptr = head;
    printf(" | PID|进程 ID\tCPU_Time|已运行时间\tReq_Time|进程所需总时间
\tPriority|优先级 |\n");
    for (i = 0; i < Proc_num; i++)
    {
        printf(" | %-10d\t%-19d\t%-23d\t%-15d |\n", ptr->PID,
ptr->CPU_time, ptr->Total_time, ptr->Priority);
        ptr = ptr->Next_PCB;
    }
}
```

```
// 动态最高优先级(和 FCFS 的唯一区别就是多了一个优先级--)
```

```
void Sched_DHPF()
{ // 由于在初始化的时候就按照优先级降序进行过排列了，所以在顺序执行的时候优先级降序仍然维持。
    int round = 1, i;
    Proc temp = tail;
    Proc p = head;
    while (p->Total_time > p->CPU_time)
    {
        printf("\nRound %d, Process %d is running|第%d 次轮转，进程%d 运行中\n", round, p->PID, round, p->PID);
        p->CPU_time++;
        p->Priority--;
        Display_DHPF();
        if (p->Total_time == p->CPU_time)
        {                    // 进程运行时间已达到要求的时间，表示该进程已执行完毕
            p->State = false; // 置结束标志
            Proc_num--;
            temp->Next_PCB = p->Next_PCB;
            if (p == head)
                head = p->Next_PCB; // 队列后移
            printf("\tProcess %d is finished|进程%d 执行完成\n", p->PID, p->PID);
        }
        else
            temp = p;
        p = p->Next_PCB;
        round++;
    }
}
```

```
Please input the number of processes|请输入进程个数:3
There are 3 processes, please input PCB info|有3个进程，请依次输入PCB信息:
-----------------Process  1-----------------
        Process id|PID/进程ID:1
        CPU time required|所需CPU时间:3
        Process priority|进程优先级:1
-----------------Process  2-----------------
        Process id|PID/进程ID:2
        CPU time required|所需CPU时间:2
        Process priority|进程优先级:2
-----------------Process  3-----------------
        Process id|PID/进程ID:3
        CPU time required|所需CPU时间:6
        Process priority|进程优先级:3
-----------------初始化完成-----------------
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3           0                      6                      3              |
 | 2           0                      2                      2              |
 | 1           0                      3                      1              |
```

```
Round 1, Process 3 is running|第1次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            1                       6                       2              |
 | 2            0                       2                       2              |
 | 1            0                       3                       1              |

Round 2, Process 2 is running|第2次轮转，进程2运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            1                       6                       2              |
 | 2            1                       2                       1              |
 | 1            0                       3                       1              |

Round 3, Process 1 is running|第3次轮转，进程1运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            1                       6                       2              |
 | 2            1                       2                       1              |
 | 1            1                       3                       0              |

Round 4, Process 3 is running|第4次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            2                       6                       1              |
 | 2            1                       2                       1              |
 | 1            1                       3                       0              |

Round 5, Process 2 is running|第5次轮转，进程2运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            2                       6                       1              |
 | 2            2                       2                       0              |
 | 1            1                       3                       0              |
        Process 2 is finished|进程2执行完成

Round 6, Process 1 is running|第6次轮转，进程1运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            2                       6                       1              |
 | 1            2                       3                       -1             |

Round 7, Process 3 is running|第7次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            3                       6                       0              |
 | 1            2                       3                       -1             |

Round 8, Process 1 is running|第8次轮转，进程1运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            3                       6                       0              |
 | 1            3                       3                       -2             |
        Process 1 is finished|进程1执行完成

Round 9, Process 3 is running|第9次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            4                       6                       -1             |

Round 10, Process 3 is running|第10次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            5                       6                       -2             |

Round 11, Process 3 is running|第11次轮转，进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Priority|优先级 |
 | 3            6                       6                       -3             |
        Process 3 is finished|进程3执行完成
```

3、参考第一题的描述，通过编程模拟实现最高响应比优先进程调度算法。

```
// 最高响应比（和 FCFS 的唯一区别就是没有将队尾的 NEXT 指针指向队首）
int Init_PCB_HRRF()
```

```
{
    int i; // 在循环输入 PCB 信息时的计数器
    Proc temp, ptr;
    // 输入第一个进程的信息
    printf("Please input the number of processes|请输入进程个数:");
    scanf("%d", &Proc_num);
    printf("There are %d processes, please input PCB info|有%d 个进程，请依次输入 PCB 信息:\n", Proc_num, Proc_num);
    temp = (Proc)malloc(sizeof(struct PCB)); // 进程 temp
    printf("----------------Process %2d----------------\n", 1);
    printf("\tProcess id|PID/进程 ID:");
    scanf("%d", &temp->PID);
    printf("\tCPU time required|所需 CPU 时间:");
    scanf("%d", &temp->Total_time);
    temp->State = true;
    temp->CPU_time = 0;
    temp->Wait_time = 0; // 就绪，且已运行时间为 0
    head = temp;
    tail = temp;
    // 依次输入之后 Proc_num-1 个进程的信息
    for (i = Proc_num; i > 1; i--)
    {
        ptr = temp; // 先指向上一次的结点
        temp = (Proc)malloc(sizeof(struct PCB));
        printf("----------------Process %2d----------------\n", Proc_num - i + 2);
        printf("\tProcess id|PID/进程 ID:");
        scanf("%d", &temp->PID);
        printf("\tCPU time required|所需 CPU 时间:");
        scanf("%d", &temp->Total_time);
        temp->State = true;
        temp->CPU_time = 0;
        temp->Wait_time = 0;
        ptr->Next_PCB = temp;
    }
    tail = temp;
    tail->Next_PCB = NULL;
    printf("----------------初始化完成----------------\n");
    return 0;
}
```

```
// 最高响应比
void Display_HRRF()
{
    int i;
```

```
    Proc ptr = head;
    printf(" | PID|进程 ID\tCPU_Time|已运行时间\tReq_Time|进程所需总时间
\tResponse_ratio|响应比 |\n");
    for (i = 0; i < Proc_num; i++)
    {
        printf(" | %-10d\t%-19d\t%-23d\t%-21.4f |\n", ptr->PID,
ptr->CPU_time, ptr->Total_time, 1 + ptr->Wait_time /
(float)(ptr->Total_time));
        ptr = ptr->Next_PCB;
    }
}
```

```
// 最高响应比
void Sched_HRRF()
{
    Proc p = head;
    float maxR = 1 + p->Wait_time / (float)(p->Total_time);
    Proc maxP = p;
    while (p->Total_time > p->CPU_time && Proc_num > 0)
    {
        while (p != NULL) // 找出最高响应比的进程
        {
            if (1 + p->Wait_time / (float)(p->Total_time) > maxR)
            {
                maxP = p;
                maxR = 1 + p->Wait_time / (float)(p->Total_time);
            }
            p = p->Next_PCB;
        }
        printf("\nProcess %d is running|进程%d 运行中\n", maxP->PID,
maxP->PID);
        maxP->CPU_time += 2;
        if (maxP->Total_time <= maxP->CPU_time)
            printf("\tProcess %d is finished|进程%d 执行完成\n", maxP->PID,
maxP->PID);
        // 其余各进程增加更新时间
        p = head;
        while (p != NULL)
        {
            if (p != maxP)
                p->Wait_time += 2;
            p = p->Next_PCB;
        }
        // 进程完成，将其从队列中清除
        if (maxP->Total_time <= maxP->CPU_time)
```

```
        {
            // maxP 运行完成
            maxP->State = false;
            if (maxP == head)
            {
                head = head->Next_PCB;
                if (head) // head 非 NULL，说明还有进程没做完
                {
                    maxP = head;
                    maxR = 1 + maxP->Wait_time /
(float)(maxP->Total_time);
                }
                else // 全部进程已完成
                    break;
            }
            else
            {
                Proc temp = head;
                while (temp != NULL)
                {
                    if (temp->Next_PCB == maxP)
                    {
                        temp->Next_PCB = maxP->Next_PCB;
                        break;
                    }
                    temp = temp->Next_PCB;
                }
            }
            Proc_num--;
        }
        Display_HRRF();
        // 复位
        p = head;
        maxP = p;
        maxR = 1 + p->Wait_time / (float)(p->Total_time);
    }
}
```

```
Please input the number of processes|请输入进程个数:3
There are 3 processes, please input PCB info|有3个进程，请依次输入PCB信息:
————————————Process  1————————————
        Process id|PID/进程ID:1
        CPU time required|所需CPU时间:3
————————————Process  2————————————
        Process id|PID/进程ID:2
        CPU time required|所需CPU时间:2
————————————Process  3————————————
        Process id|PID/进程ID:3
        CPU time required|所需CPU时间:6
————————————初始化完成————————————
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Response_ratio|响应比 |
 | 1           0                       3                       1.0000                |
 | 2           0                       2                       1.0000                |
 | 3           0                       6                       1.0000                |

Process 1 is running|进程1运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Response_ratio|响应比 |
 | 1           2                       3                       1.0000                |
 | 2           0                       2                       2.0000                |
 | 3           0                       6                       1.3333                |

Process 2 is running|进程2运行中
        Process 2 is finished|进程2执行完成
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Response_ratio|响应比 |
 | 1           2                       3                       1.6667                |
 | 3           0                       6                       1.6667                |

Process 1 is running|进程1运行中
        Process 1 is finished|进程1执行完成
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Response_ratio|响应比 |
 | 3           0                       6                       2.0000                |

Process 3 is running|进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Response_ratio|响应比 |
 | 3           2                       6                       2.0000                |

Process 3 is running|进程3运行中
 | PID|进程ID   CPU_Time|已运行时间     Req_Time|进程所需总时间 Response_ratio|响应比 |
 | 3           4                       6                       2.0000                |

Process 3 is running|进程3运行中
        Process 3 is finished|进程3执行完成
```