实验6

struct PCB /\*进程控制块结构体,包含标识符、优先数、运行时间、状态、前后指针\*/

{

   charname[6];  /\*进程标识符\*/

   intrun\_time;  /\*进程运行时间\*/

   intprior\_num;  /\*进程优先数\*/

   charstatus;  /\*进程状态：R-就绪,E-结束\*/

   structPCB \*pre;  /\*指向后一进程的指针\*/

   structPCB \*next;  /\*指向后一进程的指针\*/

};

/\*数据结构为双链表\*/

struct PCB \*head;  /\*进程链表的头指针\*/

struct PCB \*tail;  /\*进程链表的尾指针\*/

（四）源程序

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int num = 5;  /\*假定进程数为5\*/

struct PCB\*head;  /\*进程链表的头指针\*/

struct PCB\*tail;  /\*进程链表的尾指针\*/

intglobal\_time;  /\*定义全局时间\*/

struct PCB\*PCBinit(struct PCB \*q);   /\*初始化进程链表\*/

struct PCB\*init(struct PCB \*p, int i);  /\*初始化进程\*/

void sort(structPCB \*phead);  /\*冒泡排序链表\*/

voidexchange(struct PCB \*p, struct PCB \*max); /\*交换相邻两个进程的指针\*/

void run(structPCB \*p);  /\*模拟运行进程\*/

voidshowinfor(struct PCB \*head);  /\*输出进程信息\*/

voidcheck\_runtime(struct PCB \*p);/\*判断运行时间是否为0\*/

struct PCB  /\*进程控制块结构体,包含标识符、优先数、运行时间、状态、前后指针\*/

{

       char name[6];  /\*进程标识符\*/

       int run\_time;  /\*进程运行时间\*/

       int prior\_num;  /\*进程优先数\*/

       char status;  /\*进程状态：R-就绪,E-结束\*/

       struct PCB \*pre;  /\*指向后一进程的指针\*/

       struct PCB \*next;  /\*指向后一进程的指针\*/

};

//初始化进程

struct PCB\*init(struct PCB \*p, int i){

       //初始化进程名

       p->name[0] = 'P';

       p->name[1] = 'C';

       p->name[2] = 'B';

       p->name[3] = i+1+'0';

       //为进程指定运行时间

       printf("进程 %s\n",p->name);

       printf("请确定该进程的运行时间：");

       scanf("%d", &p->run\_time);

       //为进程指定优先数

       printf("请确定该进程的优先数：");

       scanf("%d",&p->prior\_num);

       printf("\n");

       //初始化进程状态为就绪

       p->status = 'R';

       //初始化指向后一进程的指针为空

       p->next = NULL;

       //返回进程

       return p;

}

//初始化进程链表

struct PCB\*PCBinit(struct PCB \*q){

       int i;

       struct PCB \*p = NULL;  /\*p为待运行队列PCB指针\*/

       head = tail = NULL; /\*初始化头尾指针\*/

       for(i = 0; i < num; i++){

              p = (struct PCB\*)malloc(sizeof(struct PCB));  //分配空间，让p指向这个PCB

              init(p,i); //初始化进程

              p->next = NULL;

              if(head == NULL){       //连接进程

                     tail = head = p;

                     p->pre = NULL;

              }else{

                     p->pre = tail;

                     tail->next = p;

                     tail = p;

              }

       }

       return p;

}

//冒泡排序链表

void sort(structPCB \*phead){

       struct PCB \*a, \*b;  //定义进程a,b

       int i;

       for(i=0;i<num;i++){  //外循环循环次数取决于全局变量num，相当于链表长度

              a = head;  //初始化进程a为head

              b = head->next;   //初始化进程b为head->next

              while(b != NULL){  //b非空

                     if(a->prior\_num <b->prior\_num){  //比较进程a,b的优先数

                            exchange(a,b);  //调用exchange函数交换a,b进程指针

                            a = a->pre;  //实现一次指针交换要重置a,b进程位置

                            b = b->next;  //因为指针交换数据不变

                     }

              a = a->next; //完成一次内循环后

              b = b->next; //指针顺延

              }

       }

}

//交换相邻两个进程的指针

void exchange(structPCB \*p, struct PCB \*max){

       if(p == max | max != p->next){  //判断若两个进程不相邻，则返回

              return;

       }

       if(max == p->next){  //进程相邻

          if(p->pre != NULL){  //进程p指向前一进程的指针不为空

              p->pre->next = max;  //将p的前一进程的后指针指向max

          }else{  //进程p的前一进程指针为空。则说明p原是head指向的进程

              head = max; //将头指针指向max

          }

          if(max->next != NULL){  //max的下一进程指针不为空

              max->next->pre=p;  //将max的下一进程的前指针指向p

          }

       max->pre = p->pre;  //将max的前一指针指向p的前一指针

       p->next = max->next;  //将p的后一指针指向max的后一指针

       max->next = p;  //max的后一指针指向p

       p->pre = max;  //p的前一指针指向max

       }//本方法用于实现冒泡排序中进程结构体只转换指针，不转换数据

}

//输出进程信息

voidshowinfor(struct PCB \*phead){

       struct PCB \*p;

       for(p = phead; p != NULL; p =p->next){

              printf("进程 %s\t 优先数 %d\t 运行时间 %d\t 状态 %c\n",p->name,p->prior\_num,p->run\_time,p->status);

       }

}

//运行函数

void run(structPCB \*p){

       //输出这个任务的状态，全局时间加1，任务还需要的时间减1，优先级减1

    global\_time++;  //全局时间加1

    p->run\_time--;  //运行时间减1

    if(p->prior\_num > 0)   // 优先数需大于0

        p->prior\_num--;   //优先数减1

       printf("第 %d 次运行:\n",global\_time);

       printf("当前进程: %s\t 优先数 %d\t 运行时间:%d\n", p->name, p->prior\_num, p->run\_time);

       printf("\n");

}

//判断运行时间是否为0

voidcheck\_runtime(struct PCB \*p){

       if(p->run\_time <= 0){  //当运行时间为0，结束进程

                     p->status = 'E';  //修改进程状态

                     printf("进程 %s 已结束",p->name);

                     printf("\n");

                     printf("进程当前状态为：\n");

                     showinfor(head);  //显示进程信息

                     printf("\n");

                     printf("-----------------------------------------------------\n");

                     printf("请按回车键进行下一进程");

                     printf("\n");

                     getchar();  //接收回车键

                     free(p); //释放p内存

                     head = p->next;  //将头指针顺延

              }

}

void main(){

       struct PCB \*p = NULL;  /\*p为待运行队列PCB指针\*/

       //初始化进程链表

       PCBinit(p);

       //按优先数递减进行进程链表排序

       sort(head);

       printf("进程当前状态为：\n");

       showinfor(head);  //显示进程信息

       printf("\n");

       //若链表中还有PCB是执行循环语句

       while(head != NULL){

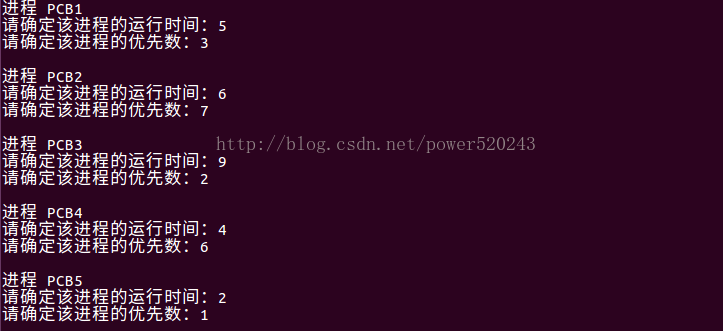
              p = head;  //p指向第一个进程

              run(p);   //进行调度

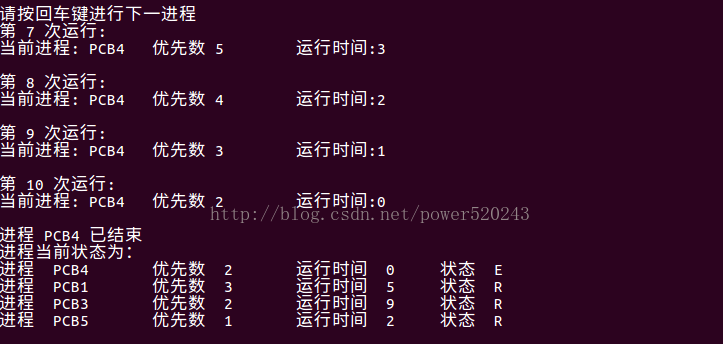
              check\_runtime(p); //判断运行时间是否为0

       }

}







![W`2A](~%11{G02LQY[VNQJY](data:image/png;base64,)