|  |  |
| --- | --- |
| **成绩** |  |

西安交通大学城市学院

操作系统 实验报告

班级 软件191 姓名 胡瑞平 学号 19031198 专业 软件工程

实验日期： 2021 年 12 月 19 日

审阅教师： 年 月 日

实验名称： 进程调度实验 实验地点： 实验楼105

|  |
| --- |
| 实验任务概要：  高优先级优先算法 |
| 学生实验心得及建议：  再接再厉！ |
| 教师批阅记录： |

**计算机系制**

**二 0二 0 年一月**

高优先级优先算法的C语言实现

#include <stdio.h>

#include <stdlib.h>

#define getpch(type) (type \*)malloc(sizeof(type))

struct pcb

{

char name[10];

char state;

int super;

int ntime;

int rtime;

struct pcb \*link;

} \*ready = NULL, \*p;

typedef struct pcb PCB;

void sort()

{

PCB \*first, \*second;

int insert = 0;

if ((ready == NULL) || ((p->super) > (ready->super)))

{

p->link = ready;

ready = p;

}

else

{

first = ready;

second = first->link;

while (second != NULL)

{

if ((p->super) > (second->super))

{

p->link = second;

first->link = p;

second = NULL;

insert = 1;

}

else

{

first = first->link;

second = second->link;

}

}

if (insert == 0)

{

first->link = p;

}

}

}

void input()

{

puts("输入六个进程信息");

for (int i = 0; i < 6; i++)

{

printf("进程号%d", i + 1);

p = getpch(PCB);

puts("进程名");

scanf("%s", p->name);

puts("进程优先数");

scanf("%d", &p->super);

puts("进程运行时间");

scanf("%d", &p->ntime);

p->rtime = 0;

p->state = 'w';

p->link = NULL;

sort();

}

}

int space()

{

int l = 0;

PCB \*pr = ready;

while (pr != NULL)

{

l++;

pr = pr->link;

}

return l;

}

void disp(PCB \*pr)

{

printf("%s\t%c\t%d\t%d\t%d\n", pr->name, pr->state, pr->super, pr->ntime, pr->rtime);

}

void check()

{

PCB \*pr;

printf("当前正在运行的进程是%s", p->name);

disp(p);

pr = ready;

puts("当前就绪队列状态为");

while (pr != NULL)

{

disp(pr);

pr = pr->link;

}

}

void destroy()

{

printf("进程%s已完\n", p->name);

free(p);

}

void running()

{

(p->rtime)++;

if (p->rtime == p->ntime)

{

destroy();

}

else

{

(p->super)--;

p->state = 'w';

sort();

}

}

void high\_priority()

{

int len, h = 0;

char ch;

input();

len = space();

while ((len != 0) && (ready != NULL))

{

ch = getchar();

h++;

printf("执行号%d\n", h);

p = ready;

ready = p->link;

p->link = NULL;

p->state = 'r';

check();

running();

}

puts("进程已完成");

}

int main(void)

{

high\_priority();

return 0;

}