计算机科学与工程学院实验报告（首页）

|  |  |
| --- | --- |
| **课程名称** | **操作系统** |

1. 实验目的

1、 理解进程的概念，明确进程和程序的区别。   
2、 理解并发执行的实质。   
3、 掌握进程的创建、睡眠、撤销等进程控制方法。

1. 实验要求

用C语言编写程序，模拟实现创建新的进程；查看运行进程；换出某个进程；杀死进程等功能。

1. 实验内容

#### 1.进程、进程控制块等的基本原理

a.为了能使程序并发执行，并且可以对并发执行的程序加以描述和控制，引入了“进程”的概念。它是资源分配和独立运行的基本单位。   
b.进程控制块（PCB）是操作系统为进程配置的一个专门的数据结构。系统利用PCB来描述进程的基本情况和活动过程，进而控制和管理进程。

#### 2.程序流程图

图示, 示意图

描述已自动生成

#### 3.程序及注释

#include "stdio.h"

#include "conio.h"

#include "stdlib.h"

struct PCB\_type

{ int pid;

int youxianji;

int daxiao;

char neirong[10];

int state; /\* 0表示不在内存，1表示在内存，2表示挂起\*/

};

struct PCB\_type neicun[20];int num=0,guaqi=0,bj,i,j,pid;/\*定义相关参数变量\*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*创建新进程\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/void create()

{

if(num>=20) /\*判断是否有存储空间\*/

printf("\n 内存已满，请删除或挂起其他程序");

else{

for(i=0;i<20;i++)

if(neicun[i].state==0) break; /\*按顺序在内存中寻找创建新进程的空间\*/

printf("\n请输入新建进程的pid值\n");

scanf("%d",&neicun[i].pid);

for(j=0;j<i;j++) /\*判断是否之前已存储相同pid进程\*/

if(neicun[j].pid==neicun[i].pid)

{ printf("\n该进程已存在于内存中");

return;

}

printf("\n请输入新进程的优先级 \n" );

scanf("%d",&neicun[i].youxianji);

printf("\n请输入新进程的大小\n" );

scanf("%d",&neicun[i].daxiao);

printf("\n请输入新进程的内容\n" );

scanf("%s",&neicun[i].neirong);

neicun[i].state=1; /\*将进程状态设为1，表示在内存中\*/

num++;/\*内存中进程数目加一\*/

}

}/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*查看当前运行进程\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/void run()

{ bj=0;

for(i=0;i<20;i++)if(neicun[i].state==1) /\*若进程状态设为1，表示在内存中，输出查看\*/

{printf("\n pid=%d",neicun[i].pid);

printf(" youxianji=%d",neicun[i].youxianji);

printf(" daxiao=%d",neicun[i].daxiao);

printf(" neirong=%s",neicun[i].neirong);

bj=1;/\*存在于内存中的进程标记为1 \*/

}

if(bj==0)

printf("\n当前没有运行该进程\n" );/\*标记为0，表示该进程未创建于内存中\*/

}/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*换出进程\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void huanchu()

{if(num==0) /\*num=0,表示内存中无进程\*/

{ printf("\n当前没有运行的进程\n" );

return;

}

printf("\n请输入要换出的进程的pid值\n" );

scanf("%d",&pid);

bj=0;

for(i=0;i<20;i++)

{

if(pid==neicun[i].pid)

{ if(neicun[i].state==1)

{neicun[i].state=2; /\*将该进程的状态设为挂起\*/

guaqi++;

printf("\n该进程已成功挂起换出\n" );

}

else if(neicun[i].state==0)

printf("\n要换出的进程不存在\n" );

else printf("\n要换出的进程已经被挂起\n" );

bj=1;break;

}

}

if(bj==0) printf("\n要换出的进程不存在\n" );

} /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*杀死进程\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/void kill()

{

if(num==0)

{ printf("\n当前没有运行的进程\n" );

return;

}

printf("\n请输入要杀死的进程的pid值\n" );

scanf("%d",&pid);

bj=0;

for(i=0;i<20;i++)

{

if(pid==neicun[i].pid)

{ if(neicun[i].state==1)

{neicun[i].state=0; /\*将该进程的状态设为不在内存中\*/

num--; /\*内存中的进程数目减少一个\*/

printf("\n该进程已成功杀死\n" );

}

else if(neicun[i].state==0)

printf("\n要杀死的进程不存在\n" );

else printf("\n要杀死的进程已经被挂起\n" );/\*剩余状态为2，挂起\*/

bj=1;break;

}

}

if(bj==0) printf("\n要杀死的进程不存在\n" );

}/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*唤醒进程\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/void huanxing()

{

if(num==0)

{ printf("当前没有运行的进程\n");

return;

}

if(guaqi==0) /\*guaqi=0,表示没有挂起的进程\*/

{ printf("\n当前没有换出的进程\n");

return;

}

printf("\n请输入要唤醒的进程的pid值:\n");

scanf("%d",&pid);

for(i=0;i<20;i++){

if(pid==neicun[i].pid)

{

if(neicun[i].state==2)

{neicun[i].state=1; /\*将该进程的状态设为挂起\*/

guaqi--;

num++;

printf("\n该进程已成功唤醒\n" );

}

else if(neicun[i].state==0)

printf("\n要唤醒的进程不存在\n" );

else printf("\n要唤醒的进程已经在内存中\n" );

}

}

} /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/void main()

{int xuhao,n=1,i;for(i=0;i<20;i++)

neicun[i].state=0; /\*使所有进程都初始设为不在内存中\*/while(n){

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\* 进程演示系统 \*");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n 1.创建新的进程 2.查看运行进程");

printf("\n 3.换出某个进程 4.杀死运行进程");

printf("\n 5.唤醒某个进程 6.退出程序 ");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n请选择(1～6):");

scanf("%d",&xuhao);

switch(xuhao)

{

case 1: create( ); break;

case 2 :run(); break;

case 3 :huanchu();break;

case 4 :kill();break;

case 5 :huanxing();break;

case 6 :exit(0);

default: n=0;

break;

}

}

}

1. 说明

本实验应在教师指导下进行，尽量不要擅自进行，否则可能会破坏原有系统。

1. 运行结果及实验结果、实验体会

本实验代码充分体现了进程的创建、进程状态的切换、杀死的过程与原理。首先源代码运用了c语言来编译整个过程。用结构体数组来储存个个“进程”的数据。其中核心的数据项是state，进程状态项。如下：

State=0：不在内存中。

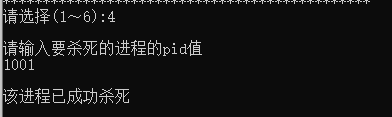
State=1；运行中。

State=2：挂起态（不在内存中）。

这三条作为核心数据项贯穿了整个代码。



此图为查看运行进程

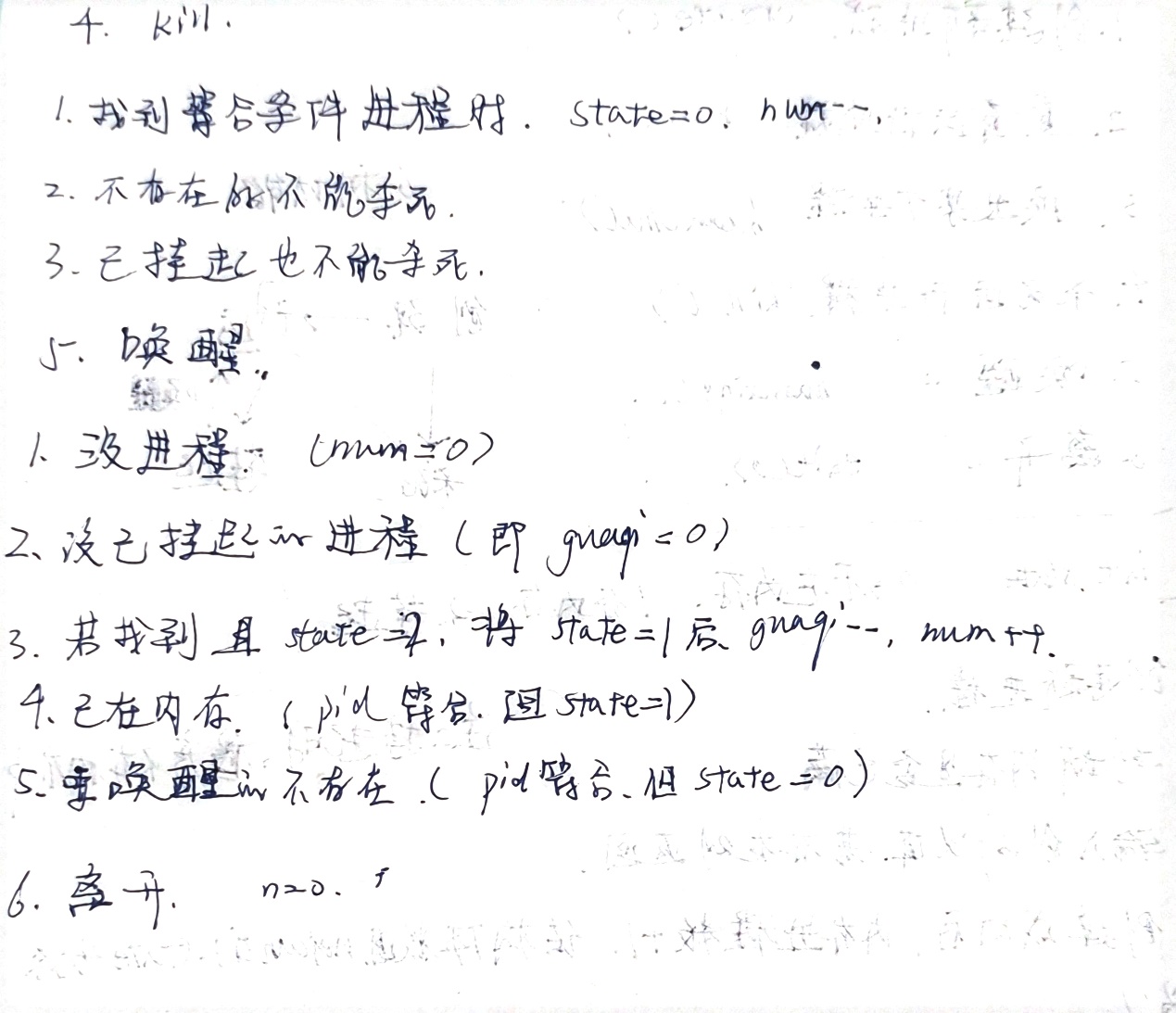
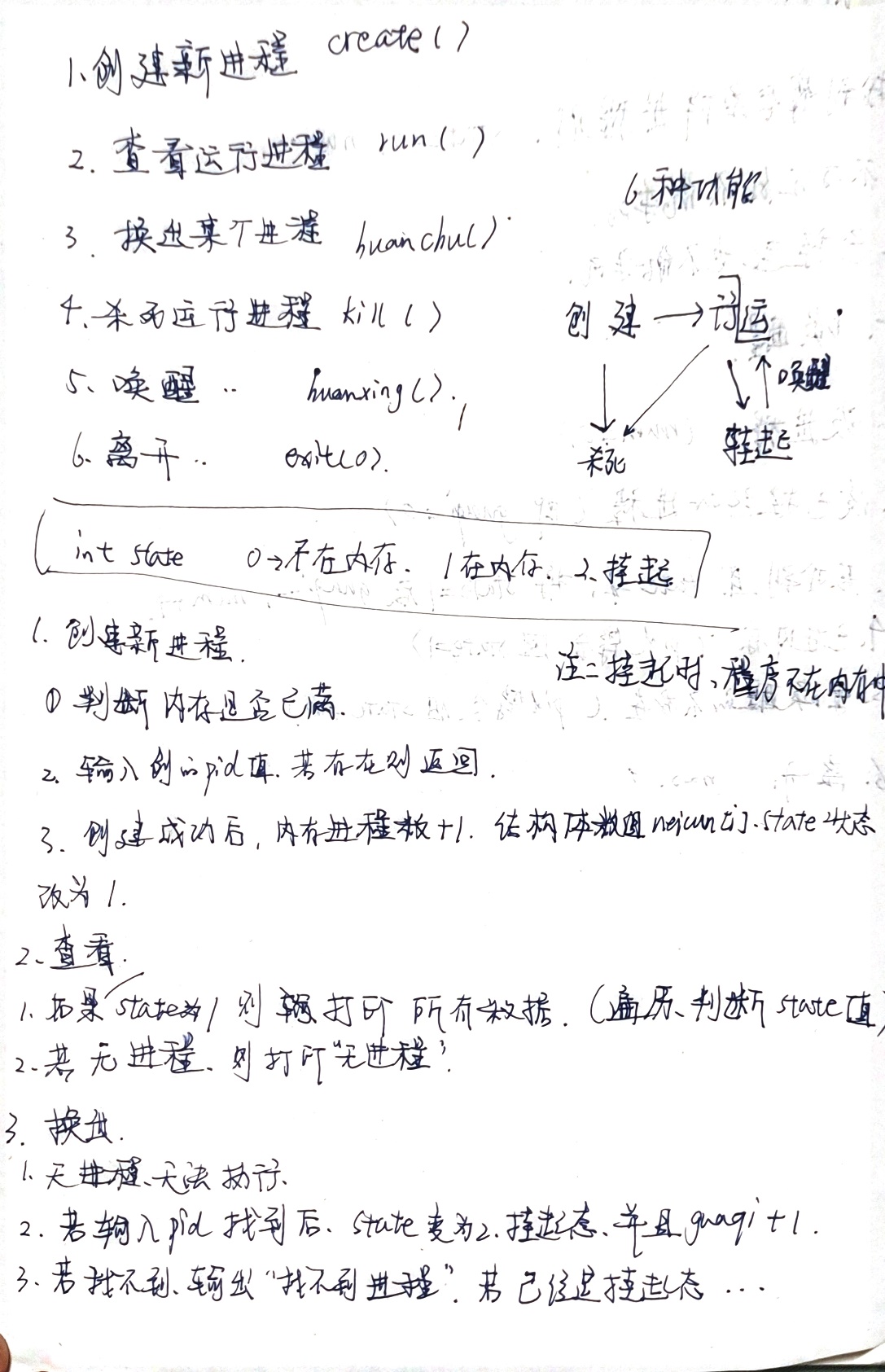


杀死进程指令

代码中还运用了大量的遍历去匹配结构数组中的pid项，结合输入找到pid进程，再在对state的状态进行判断，推断出该进程的状态是否能继续执行原有指令。

其次就是内存中运行的进程数，定义了一个num变量去实现，如果创建进程就是num++，挂起，杀死进程就是num—，还有一个挂起进程数量的guaqi变量，储存了挂起程序的熟练，以便辅助唤醒函数的实现。

以下是我读代码时做的相应的笔记：



以上是我读代码时做的相应的笔记。

代码一共实现了五种功能，利用了自定义函数的方式定义了每个功能，再用switch函数打印菜单，选择选项或者退出菜单。如果内存满了就不能再创建进程，如果内存里没有进程也不能查看已经在运行的进程。同时如果被挂起的进程也不能被杀死，如果进程不在内存中也无法被杀死。

通过此次实验，我不仅让自己读代码的能力有所提升，还对进程的创建，调度，状态理解程度更为深刻了。