**黑龙江大学**

**实 验 报 告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | **计算机操作系统** | | | | | |
| **实验项目名称** | **进程控制** | | | | | |
| **实验时间**  **（日期及节次）** | **2016年09月13日 1-2节** | | | | | |
| **专业** | **计算机** | | **学生所在学院** | | **计算机科学技术学院** | |
| **年级** | **2014级** | | **学号** | | **20123349** | |
| **姓名** | **刘伟** | | **指导教师** | | **朱敬华** | |
| **实验室名称** |  | | | | | |
| **实验成绩** | **预习情况** | **操作技术** | **实验报告** | **附加：综合创新能力** | | **实验**  **综合成绩** |
|  |  |  |  | |  |
| **教师签字** |  | | | | | |

**黑龙江大学教务处**

**实验一 进程控制**

**一．实验名称：**进程控制

**二．实验目的：**利用简单的结构和控制方法模拟进程结构、进程状态和进程控制。

**三．实验内容：**

用PCB表示整个进程实体，利用随机数方法或键盘控制方法模拟进程执行中产生的事件。或者利用鼠标或者键盘中断的基于图形接口方式的进程控制管理。

**四．实验代码**

/\*\*

\* 定义一个队列类

\* @dataStore 空数组

\* @enqueue 入队

\* @dequeue 出队

\* @front 返回队首元素

\* @back 返回队尾元素

\* @show 打印队列

\* @isEmpty 判断队列是否为空

\*/

function Queue()

{

this.dataStore = [];

}

Queue.prototype.enqueue = function(element)

{

this.dataStore.push(element);

}

Queue.prototype.dequeue = function()

{

return this.dataStore.shift();

}

Queue.prototype.front = function()

{

return this.dataStore[0];

}

Queue.prototype.back = function()

{

return this.dataStore[this.dataStore.length-1];

}

Queue.prototype.show = function()

{

var retStr = "";

for(var i = 0; i < this.dataStore.length; i++)

{

retStr += this.dataStore[i] + "\n";

}

return retStr;

}

Queue.prototype.isEmpty = function()

{

if(this.dataStore.length == 0)

{

return true;

}

else

{

return false;

}

}

/\*\*

\* 定义一个 PCB 类

\* @pid 进程pid

\* @title 进程title

\* @get\_pid 返回pid

\* @get\_title 返回title

\*/

function PCB(pid,title)

{

this.pid = pid;

this.title = title;

}

PCB.prototype.get\_pid = function()

{

return this.pid;

};

PCB.prototype.get\_title = function()

{

return this.title;

};

/\*\*

\* 生成一个随机pid

\*/

var create\_pid = function()

{

var myDate = new Date();

var myYear = myDate.getYear();

var myMonth = myDate.getMonth();

var myHour= myDate.getHours();

var myMinute = myDate.getMinutes();

var mySecond = myDate.getSeconds();

var myRandom = ~~(Math.random()\*1000);

var pid = myYear+myMonth+myHour+myMinute+mySecond+myRandom;

return pid;

}

/\*\*

\* 创建进程 create\_process

\*/

var create\_process = function(title)

{

var pid = create\_pid();

var title = title;

var pcb = new PCB(pid,title);

readyQue.enqueue(pcb);

}

/\*\*

\* 阻塞进程 block\_process

\*/

var block\_process = function()

{

if(readyQue.isEmpty())

{

console.log("No ready process running");

}

else

{

blockQue.enqueue(readyQue.front());

readyQue.dequeue();

}

}

/\*\*

\* 唤醒进程 wake\_process

\*/

var wake\_process = function()

{

if(blockQue.isEmpty())

{

console.log("No blocked process running");

}

else

{

readyQue.enqueue(blockQue.front());

blockQue.dequeue();

}

}

/\*\*

\* 时间片到 time\_end

\*/

var time\_end = function()

{

if(readyQue.isEmpty())

{

console.log("No ready process running");

}

else

{

running = readyQue.front();

readyQue.dequeue();

readyQue.enqueue(running);

}

}

/\*\*

\* 结束进程 kill\_process

\*/

var kill\_process = function()

{

if(readyQue.isEmpty())

{

console.log("No ready process running");

}

else

{

readyQue.dequeue();

running = readyQue.front();

}

}

/\*\*

\* 显示进程信息 show\_process

\*/

var show\_process = function()

{

var readyStr = "";

var blockStr = "";

if(readyQue.front())

{

console.log("running: " + readyQue.front().title);

}

for(var j=1; j<readyQue.dataStore.length; j++)

{

readyStr += readyQue.dataStore[j].title+" ";

}

console.log("ready: " + readyStr);

for(var k=0; k<blockQue.dataStore.length; k++)

{

blockStr += blockQue.dataStore[k].title+" ";

}

console.log("blocked: " + blockStr);

}

var running = "";

var readyQue = new Queue();

var blockQue = new Queue();

var readline = require("readline");

var rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.setPrompt("OS Demo > ");

rl.prompt();

rl.on('line',function(input){

var shell = input.trim();

switch(shell){

case '-h':

console.log("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"+"\n-h ---- helpInfo\n-a ---- authorInfo\n-i ---- demoInfo\n-c ---- create a process\n-b ---- blocked a process\n-w ---- wake a process\n-t ---- time end\n-k ---- kill currenty process\n exit ---- exit\n"+"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

break;

case '-i':

console.log("\n This is a OS Process Management Demo! \n");

break;

case '-a':

console.log("\n liuwei NO.20123349 \n");

break;

case '-c':

rl.question('please enter process title:',function(answer){

var title = answer;

create\_process(title);

show\_process();

});

break;

case '-b':

block\_process();

show\_process();

break;

case '-w':

wake\_process();

show\_process();

break;

case '-t':

time\_end();

show\_process();

break;

case '-k':

kill\_process();

show\_process();

break;

case 'exit':

console.log("\n You had closed OS Process Management Demo! \n");

rl.close();

process.exit(0);

break;

default:

console.log("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"+

"\*This is a OS Process Management Demo,you can input '-h' see more information!\*"+

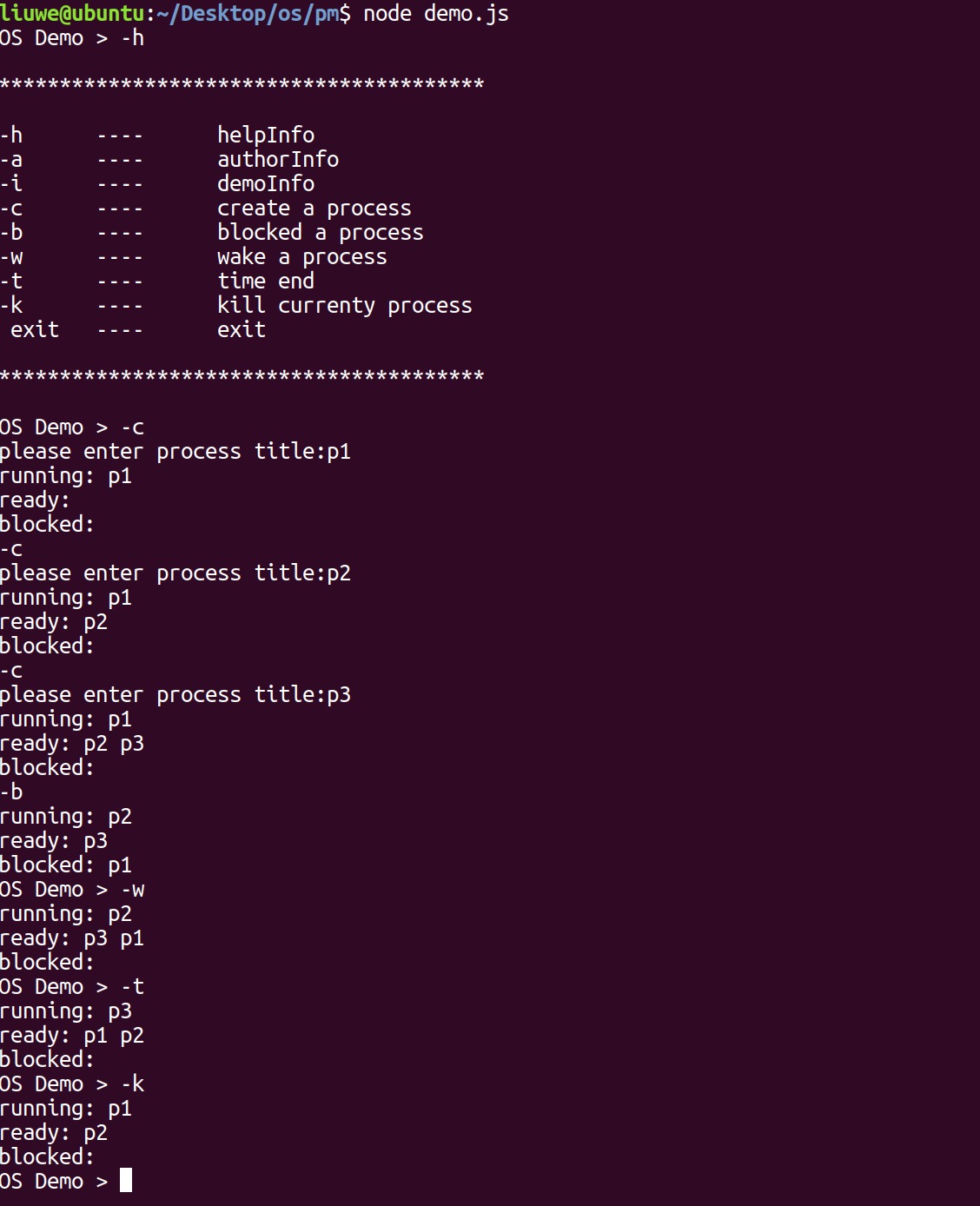
"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

); }

rl.prompt();

});

**五．实验截图：**



**六．实验心得：**

通过此次试验了解了进程控制块PCB的定义与创建方式，了解了进程控制的基本概念，了解了进程各种状态之间的转换以及转换发生的基本条件。