**实验7 信号量应用**

实验指导

一、Linux线程创建

1．线程创建函数pthread\_create()

头文件：#include <pthread.h>

函数定义：

int pthread\_create(pthread\_t \*thread, pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg);

参数说明：

thread：指向pthread\_create类型的指针，用于引用新创建的线程。

attr：用于设置线程的属性，一般不需要特殊的属性，所以可以简单地设置为NULL。 \*(\*start\_routine)(void \*)：传递新线程所要执行的函数地址。

arg：新线程所要执行的函数的参数。 调用如果成功，则返回值是0，如果失败则返回错误代码。

2. 线程终止函数pthread\_exit()

头文件：#include <pthread.h>

函数定义：void pthread\_exit(void \*retval);

功能：线程退出

参数说明：

retval：是线程的返回值。有pthread\_join()检测获得。

3. 等待线程终止 pthread\_join()

头文件：#include <pthread.h>

函数定义：int pthread\_join(pthread\_t th, void \*\*thread\_return);

功能：等待指定的线程结束

参数说明：

th：等待线程的ID，线程通过pthread\_create返回的标识符来指定。

thread\_return：用户自定义的指针，用来保存被等待线程的返回值。

这个函数是线程阻塞函数，调用它的函数将一直阻塞到被等待的线程结束为止，当函数返回时，被等待线程的资源被回收。

二、Linux线程信号量

1. 信号量创建sem\_init

头文件：#include <semaphore.h>

函数定义：int sem\_init(sem\_t \*sem, int pshared, unsigned int value);

参数说明：

sem：信号量对象。

pshared：控制信号量的类型，0表示这个信号量是当前进程的局部信号量，否则，这个信号量就可以在多个进程之间共享。

value：信号量的初始值。

2. 信号量操作

头文件：#include <semaphore.h>

函数定义：int sem\_wait(sem\_t \*sem); //相当于P操作

int sem\_post(sem\_t \*sem); //相当于V操作

sem\_post的作用是以原子操作的方式给信号量的值加1。

sem\_wait的作用是以原子操作的方式给信号量的值减1。

3. 信号量销毁

头文件：#include <semaphore.h>

函数定义：int sem\_destory(sem\_t \*sem);

函数功能：用完信号量后对它进行清理，清理该信号量所拥有的资源。

示例程序

#include<stdio.h>

#include<pthread.h>

#include <semaphore.h>

#include <unistd.h>

void pthread1();

void pthread2();

sem\_t sem1,sem2; //定义信号量sem1，sem2；

int main()

{ pthread\_t th1,th2;

int ret;

sem\_init(&sem1,0,1); //初始化线程信号量sem1，初值为1

sem\_init(&sem2,0,0); //初始化线程信号量sem2，初值为0

ret=pthread\_create(&th1,NULL,(void \*)pthread1, NULL);

if(ret!=0) perror("pthread cread1");

ret=pthread\_create(&th2,NULL,(void \*)pthread2, NULL);

if(ret!=0) perror("pthread cread2");

pthread\_join(th1,NULL);

pthread\_join(th2,NULL);

return 0;

}

void pthread1()

{ int k,ai=0;

sem\_wait(&sem1); //执行信号量sem1的p操作，相当于 P（sem1）

printf("I am father\n");

sleep(1);

printf("I am putting an apple %d! in the plate!\n",ai);

sleep(1);

sem\_post(&sem2); //执行信号量sem2的v操作，相当于 v（sem2）

ai++;

}

void pthread2()

{ int k,ai=0;

sem\_wait(&sem2); //执行信号量sem2的p操作，相当于 P（sem2）

printf("I am son\n");

sleep(1);

printf("I am picking an apple %d! from the plate!\n",ai);

sleep(1);

ai++;

sem\_post(&sem1); //执行信号量sem1的v操作，相当于 v（sem1）

}