**一、线程基础**

**1.1 基本概念**

**1.1.1 背景**

进程是资源分配的基本单位，每个进程拥有独立的虚拟地址空间、文件描述符表、全局变量等资源。随着计算机应用复杂度的提升，其局限性逐渐显现（资源开销大​，共享数据困难​，无法充分利用多核），因此设计了轻量级的进程---线程

**1.1.2 线程**

线程（Thread）是操作系统能够进行运算调度的最小单位，被包含在进程之中，是进程中的实际运作单位。一个进程可以拥有多个线程，这些线程共享进程的内存空间、文件描述符等资源，但每个线程有自己独立的栈空间、寄存器上下文和程序计数器（PC）。相比进程，线程的创建和切换开销更小，适合实现并发任务。

**二、线程相关函数**

**2.1 基本函数**

**1. 创建线程 pthread\_create**

​在一个已存在的进程中创建一个新的线程，新线程从指定的启动函数开始执行。

多个线程可以同属于一个进程，可以通过命令pstree -p pid，可以查看进程中的线程。

**2. 结束线程**

|  |  |
| --- | --- |
| **pthread\_exit** | 终止当前线程的执行，并设置退出状态 |
| **pthread\_cancel** | 向目标线程发送取消请求，使其提前终止。 |

**3.线程资源回收**

|  |  |  |
| --- | --- | --- |
| **pthread\_join** | 阻塞版本 | 阻塞当前线程，等待指定线程终止，并获取其退出状态 |
| **pthread\_detach** | 非阻塞版本 | 将线程标记为“分离”状态，使其终止后自动回收资源(不需要轮询操作) |

**2.2 错误处理**

常用的错误解析函数有perror、strerror，其中perror函数依赖全局变量(errno)，当系统调用或部分库函数执行失败时，会将错误码写入errno,perror函数会根据当前errno输出对应的错误信息。

由于pthread库中有些函数遵循 ​​POSIX 标准的错误处理规范​​：成功返回0，失败返回错误码（不会修改全局变量 errno），因此不能使用perror，但可以使用strerror。

示例代码

​    ret = pthread\_create(&tid,NULL,pthread\_handler,NULL);

    if(ret){

        fprintf(stderr,"failed:%s\n",strerror(ret));

        exit(-1);

    }​

**2.3 同步与互斥**

在多线程编程中，​​线程同步与互斥​​是解决共享资源竞争、确保程序正确性的核心技术。由于多线程共享进程的内存空间，当多个线程同时访问共享资源（如变量、文件、网络连接等）时，可能导致数据不一致或逻辑错误。

**2.3.1 基本概念**

同步​​是指多个线程按​​特定顺序执行​​，确保操作的先后顺序符合业务逻辑。

互斥​​是指多个线程对共享资源的​​独占访问​​：同一时间仅允许一个线程操作共享资源，避免数据被破坏。

**2.3.2 信号量**

原理​​：信号量是一个计数器（sem\_t），表示可用资源的数量。线程通过 sem\_wait() 减少计数器（若计数器为0则阻塞），通过 sem\_post() 增加计数器（唤醒等待线程）

pthread库 信号量 常用函数有

典型用法： 1.实现同步效果 2.实现互斥效果

|  |  |  |
| --- | --- | --- |
| int sem\_init(sem\_t \*sem, int pshared, unsigned int value); | 初始化 |  |
| int sem\_wait(sem\_t \*sem); | 获取信号量（-1） 等待 | P操作​​（Proberen） |
| int sem\_post(sem\_t \*sem); | 释放信号量（+1） 发布 | V操作​​（Verhogen） |
| int sem\_destroy(sem\_t \*sem); | 销毁 |  |

**2.3.3 互斥锁**

|  |  |
| --- | --- |
| int pthread\_mutex\_init(pthread\_mutex\_t \*mutex,const pthread\_mutexattr\_t \*restrict attr); | 初始化 |
| int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex); | 加锁 |
| int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex); | 解锁 |
| int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex); | 销毁 |

**三、函数备注**

int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr,

void \*(\*start\_routine) (void \*), void \*arg);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：创建线程

Input:thread：线程id

attr：线程属性 默认NULL

start\_routine：线程函数

arg：参数

Return：成功创建返回值为0，错误返回错误号。

other: 不能用perror()进行打印，采用strerror();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pthread\_t pthread\_self(void); //返回线程id

void pthread\_exit(void \*retval);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：线程退出

Input:retval：线程退出状态

Return：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int pthread\_cancel(pthread\_t thread);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：取消线程

Input:thread：线程id

Return：成功创建返回值为0，错误返回错误号。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int pthread\_join(pthread\_t thread, void \*\*retval);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：阻塞等待线程退出，回收线程的资源

Input:thread：线程id

retval：线程的状态

Return：成功创建返回值为0，错误返回错误号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int pthread\_detach(pthread\_t thread);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：线程在运行结束后会自动释放系统分配的资源

Input:thread：线程id

Return：成功创建返回值为0，错误返回错误号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int sem\_init(sem\_t \*sem, int pshared, unsigned int value);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：信号量初始化

Input:sem：信号量

pshared:指明信号量的类型,共享区域(0:线程间,1:进程间)

value：信号量初值

Return：成功返回0， 出错返回-1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int sem\_wait(sem\_t \*sem);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description： 获取信号量（信号量减1） (P操作)

Input:sem：信号量

Return：成功返回0， 出错返回-1

注意： 减到0值时阻塞

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int sem\_post(sem\_t \*sem);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description： 释放信号量（信号量加1） (V操作)

Input:sem：信号量

Return：成功返回0， 出错返回-1

注意： 不会阻塞

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int sem\_destroy(sem\_t \*sem);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：销毁

Input:sem：信号量

Return：成功返回0， 出错返回-1

注意：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int pthread\_mutex\_init(pthread\_mutex\_t \*mutex,const pthread\_mutexattr\_t \*restrict attr);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：初始化锁

Input:mutex：锁 restrict attr：属性

Return：成功返回0， 出错返回错误码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：加锁

Input:mutex：锁

Return：成功返回0， 出错返回错误码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：解锁

Input:mutex：锁

Return：成功返回0， 出错返回错误码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description：销毁

Input:mutex：锁

Return：成功返回0， 出错返回错误码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**作业**

1.练习线程的同步

线程1 输入字符串，例如 args/62/13/27

线程2 对字符串进提取，提取输出数字并打印 例如 62,13,27