## 《操作系统》实验报告

学号： 姓名： 上课时间：周三上午

|  |  |
| --- | --- |
| **实验名称：** | 进程与线程-Linux进程与线程通讯 |
| **实验目的：** | 1）深刻理解线程和进程的概念，掌握线程与进程在组成成分上的差别，以及与其相适应的通讯方式和应用目标。  2）了解Linux系统中fork()语义和clone()系统调用。 |
| 实验内容： | 1）以Linux系统进程和线程为背景，掌握fork()和clone()系统调用的形式和功能。   1. 以生产者消费者问题为例，通过实验理解fork()和clone()两个系统调用的区别。 |
| **程序框图：**   |  |  | | --- | --- | |  |  | |  |  | | |
| **程序关键代码和解读：**   |  |  | | --- | --- | |  | 通过返回值判断为父进程还是子进程。对子进程来说返回值为0，调用producer()/consumer()，而对父进程来说返回值为非负整数，为子进程的pid。 | |  | 如果父进程不关闭管道的读端和写端，那么管道将一直保持打开状态。 | |  | 在每次循环中，wait(&status)会暂停父进程的执行，直到任一子进程退出。一旦有子进程退出，wait(&status)会返回被终止子进程的进程ID，并将子进程的退出状态保存在status中。当四个子进程都退出后，父进程结束。 | |  | 生产者进行写操作，若返回-1则写入失败。 | |  | 消费者进行读操作，若读取的字节数为0，则退出。 | |  | 使用“或”操作设定clone\_flag的值。 | |  | 动态申请栈，并将栈分配给即将产生的生产者和消费者子线程。生产者消费者的编号由参数arg传入。 | |  | P信号量。获取互斥锁，保证每次只有一个线程进行读/写。解锁。V信号量。消费者V一个warehouse信号量表示可以继续生产，生产者V一个product信号量表示可以消费。 | |  | 实验(2)中，原来的程序在主进程创建完子线程后，会很快退出，导致子线程也强制退出，无法得到正确结果。尝试wait()、pthread\_join()、pthread\_create()等多种方式等待子线程都无效后，只能采用sleep()函数进行休眠，等待子线程结束。 | |  |  | |  |  | |  |  | | |
| **实验结果分析与体会：**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | |  | 第一个实验中，尝试父进程不关闭管道的读写段，使管道保持打开状态，发现程序无法正常结束。若父进程不使用wait()等待子进程退出而直接退出，也会导致子进程直接退出，无法得到正常实验结果。  在消费者代码中，消费者在读取数据前将写缓冲区的内容设置为ccc/ddd，实际上是无效的，因为管道的写端已经关闭。且在循环中将读缓冲设置为eee，也是无效的，会被后面读取的内容覆盖。 | |  | 实验二中，编译程序发现clone\_flag中的宏未定义，添加了宏“#define \_GNU\_SOURCE”来解决。  原本的程序中，父进程没有等待子线程结束就直接退出，导致子线程在父进程退出后也终止，无法得到正常实验结果。于是尝试使用类似实验一中的wait()函数进行等待，不可行。原因是这是对于进程来使用的。又尝试使用“pthread\_join(tid,NULL)”进行等待，在创建子线程后，将返回值转换为“pthread\_t”类型并保存，最后在父进程中用循环等待四个子线程结束。发现仍然不可行。后来使用“pthread\_create(&tid,null,producer,NULL)”函数来进行子线程的创建，代替clone()，发现无法实现。  最后只能使用sleep()函数，父进程保持休眠，等待子线程操作完毕，可行。 | |  | |  |  | | |
| **思考题：**   |  | | --- | | （1）采用msg实现的生产者/消费者问题。 | | **（2）在Linux系统中，pipe、clone、shm和msg是四种常用的高级通信方法。它们各自有不同的优点和适应环境，下面是对它们的比较：**  **1. 管道（pipe）：**  **- 优点：**  **- 简单易用，是最基本的进程间通信方法之一。**  **- 只需要一个文件描述符，适用于两个相关进程之间的通信。**  **- 缺点：**  **- 只能在有亲缘关系（父子进程或者兄弟进程）的进程之间使用。**  **- 只支持单向通信。**  **- 数据流必须按顺序传递，不支持随机访问。**  **- 传输的数据量有限，缓冲区大小有限。**  **- 适应环境：**  **- 适用于父子进程或者兄弟进程之间的通信，且通信量较小的场景。**  **2. 克隆（clone）：**  **- 优点：**  **- 可以在不同的命名空间中创建新的进程，包括网络命名空间、文件系统命名空间等。**  **- 可以选择性地共享进程资源，如文件描述符表、内存空间等。**  **- 缺点：**  **- 使用复杂，需要了解和处理命名空间、共享资源等概念。**  **- 不适合简单的进程间通信，更适用于创建具有特定环境要求的进程。**  **- 适应环境：**  **- 适用于需要创建特定命名空间或共享资源的场景，如容器化技术中的进程隔离。**  **3. 共享内存（shm）：**  **- 优点：**  **- 可以实现高效的内存共享，避免了数据的复制和传输开销。**  **- 可以实现进程间的实时通信。**  **- 缺点：**  **- 需要额外的同步机制来保证数据的一致性和互斥访问。**  **- 不适合传输大型数据结构，需要事先协调好数据结构的布局。**  **- 适应环境：**  **- 适用于需要高效共享数据、实时通信的场景，如多进程并行计算、数据流处理等。**  **4. 消息队列（msg）：**  **- 优点：**  **- 提供了灵活的消息传递机制，支持进程间的异步通信。**  **- 可以实现多对多的通信模式。**  **- 具有较高的消息容量和可靠性。**  **- 缺点：**  **- 使用稍微复杂，需要使用特定的消息队列函数进行操作。**  **- 对消息的处理顺序和错误处理需要额外的考虑。**  **- 适应环境**  **- 适用于需要异步通信、多对多通信模式或者大容量消息传递的场景，如进程间任务调度、分布式系统等。** | | |