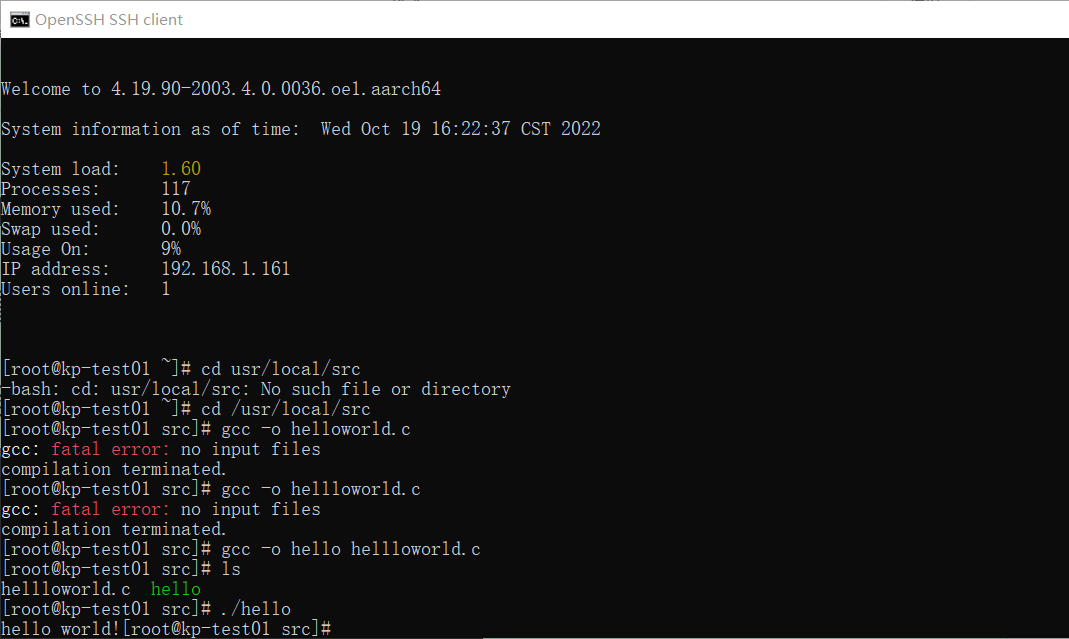
**任务1**

**1. 进程相关编程实验**

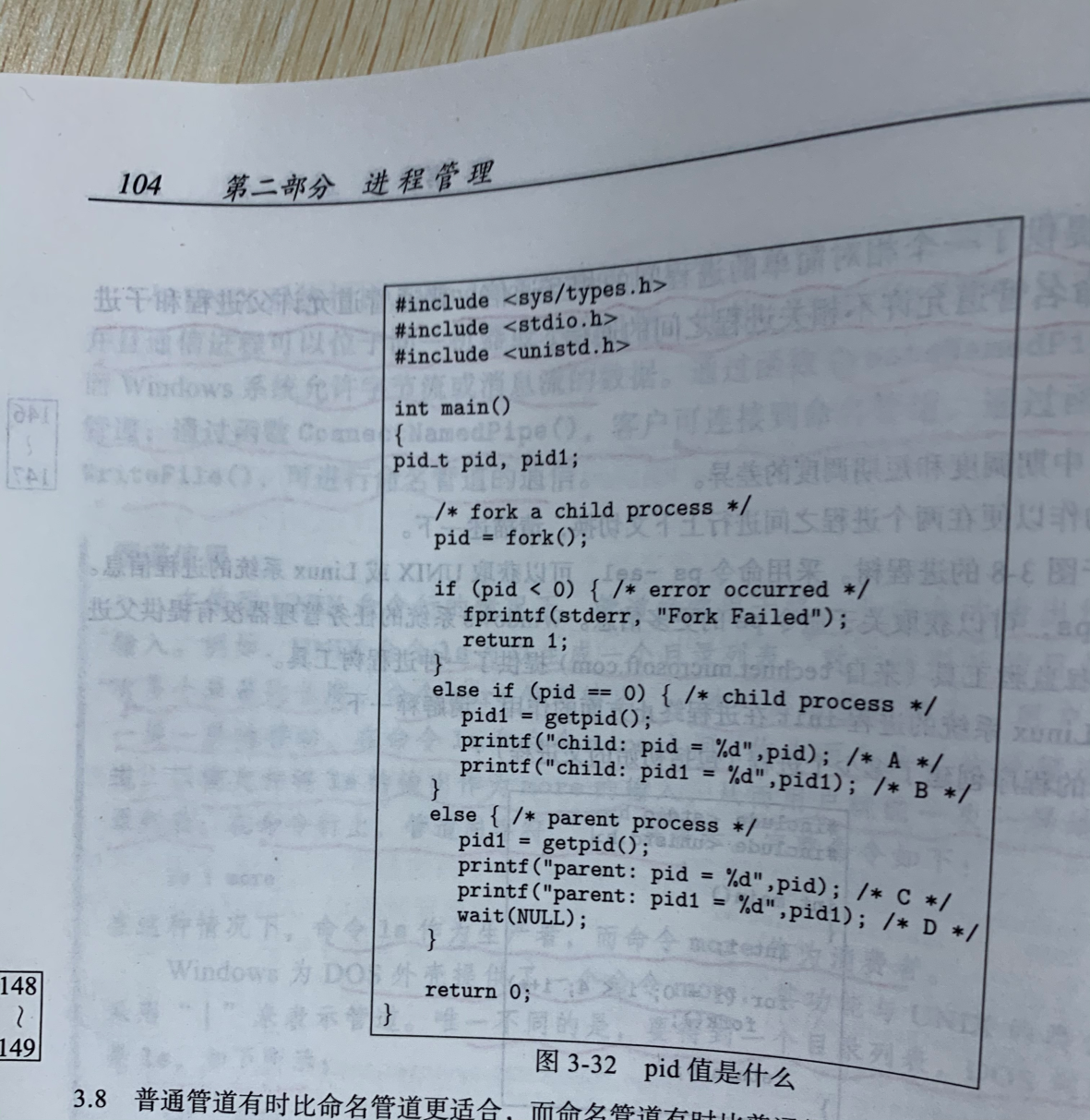
**1.1熟悉操作系统命令**

**在本地写一个简单的helloworld.c文件，通过winscp连接到 华为服务器，将该文件放置到指定的文件目录下，利用openSSH连接到主机，使用命令行进入到指定目录下再编译运行即可**

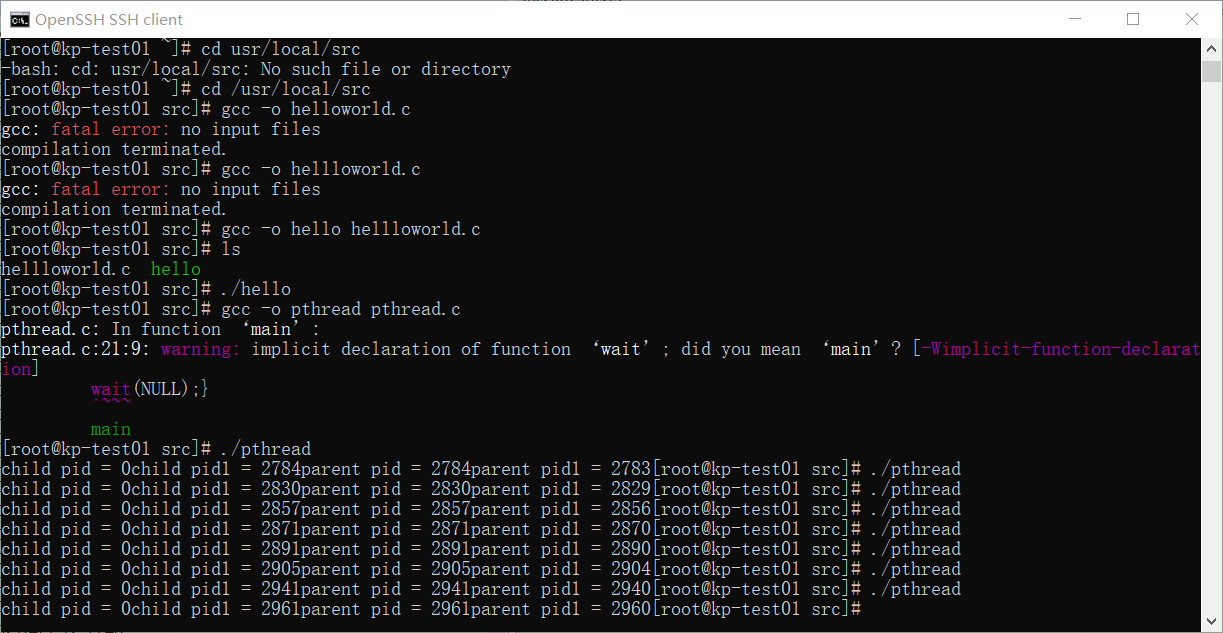
****

**1.2．1运行进程的程序**

**要运行的程序如图：**

****

**同上述操作，多次运行结果如下：**

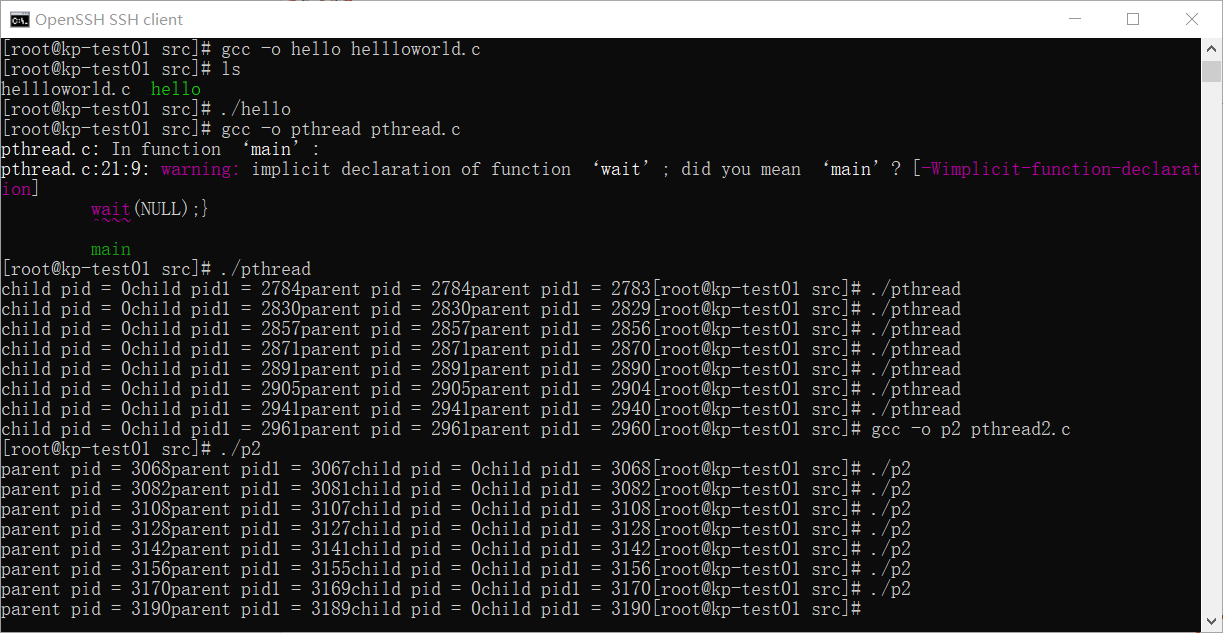
****

**分析：由图可知在子进程中，pid的值为0；在父进程中，其自身pid值pid1的值为2783，这是因为fork（）函数在子进程中的返回值为0，在父进程中返回子进程的pid值。**

**对于getpid函数，它的功能就是返回当前进程的pid值。**

**去除wait（）函数的情况：**

**运行结果如下：**

****

**分析与解释：父进程在执行完后由于有了wait会等待子函数执行完并将其销毁后结束，从而在输出缓冲区中先出来的是子进程的输出再是父进程的输出，如同上面一大段输出child pid在前，parent pid在后，如果没有wait，父进程直接输出结束，子进程输出后被内核进程回收，正如输出所示，parent pid在前，child pid 在后。**

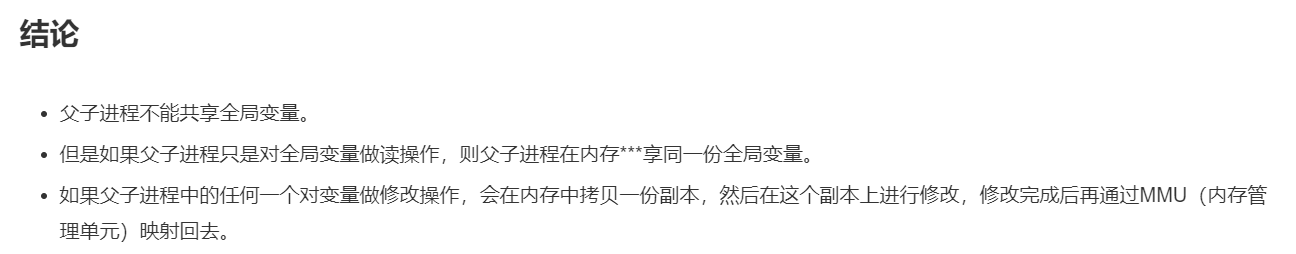
**1.2.2修改程序为情况A**

**A：添加一个全局变量并在父进程和子进程中对这个变量做不同操作，输出操作结果并解释，同时输出两种变量的地址观察并分析。**

**运行结果如下：**

**C:\Users\lenovo\Documents\Tencent Files\1162984920\Image\Group2\W0\R2\W0R2}%GLYPWI%]@OG$5([T0.png**

**分析：（注意global为程序设定的全局变量初始值为0，在子进程中global变量自增一次，并且将打印其值和地址；在父进程中global将自减，同样打印其值与地址）可以看见这两个进程都是将全局变量分别保存，所以都是对值为0 的 glb进行操作。两个进程打印的glb的地址都一样是因为：此处的地址并非是实际的物理地址，而是虚拟地址，两进程所保存的虚拟地址是一样的但是实际物理地址不一样。虚拟地址一样的原因是：子进程是会完全复制父进程的数据段的。对于涉及到修改操作的全局变量的情况，子进程会复制该全局变量，并对复制的变量进行操作。可见，就算是全局变量，两个进程也无法分享。若是父进程做只读不修改的操作，查阅资料知可以共享。**



**（**[**https://blog.nowcoder.net/n/a6cdab4f10644fe18d85c9ad704b4638**](https://blog.nowcoder.net/n/a6cdab4f10644fe18d85c9ad704b4638)**）**

**1.2.3修改程序为情况B**

**B：在return前增加对全局变量的操作并输出结果，观察并解释；**

**运行结果如下：**

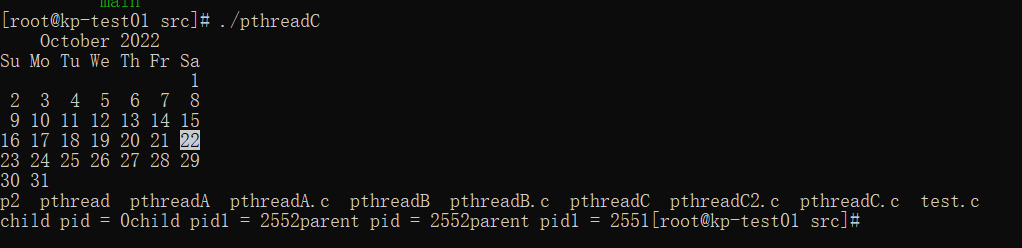
**C:\Users\lenovo\Documents\Tencent Files\1162984920\Image\Group2\3T\J`\3TJ`RJO2[{1B(AG~3~8BMPY.png**

**分析：由于是在return前的操作，glb在子进程中，自增一次后又自增一次，故值为12；在父进程中先自减后自增，值为10不变；**

**1.2.3修改程序为情况C**

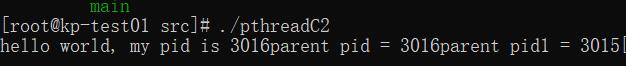
**C：修改程序体会在子进程中调用system函数和在子进程中调用exec族函数执行自己写的一段程序，在此程序中输出进程PID进行比较并说明原因**

**调用system（）运行结果如下：**



**分析：在子进程中调用了system（）函数中的 cal（显示日期） 和ls （显示当前目录下的所有文件名）在调用完后重新进入子进程执行接下来的代码。**

**调用execve（）运行结果如下：**



**分析：在子进程中调用了execve（）函数，调用了另一段打印hello world以及自己的pid值的代码，可以看到，子进程在调用完该程序后直接结束了，且调用的该进程的pid和原来子进程的pid的值是一样的，查阅知：exec（）函数族是把调用自己的进程的存储空间清空后写入自己的数据，原进程的数据没有保存而全部丢失。**

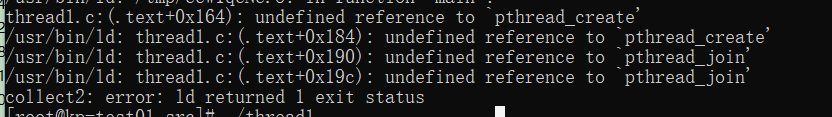
**2.线程相关编程实验**

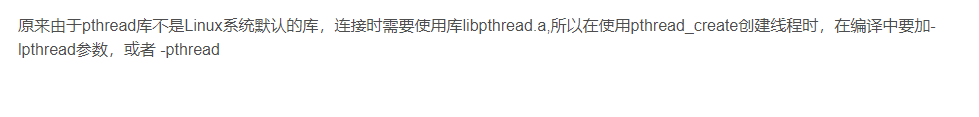
**2.1.1**

**程序要求：在进程中给一变量赋初值并创建两个线程，在两个线程中分别对此变量循环五千次以上做不同的操作并输出结果；**

**进一步要求：多运行几遍程序观察运行结果，如果发现每次运行结果不同，请解释原因并修改程序解决，考虑如何控制互斥和同步**

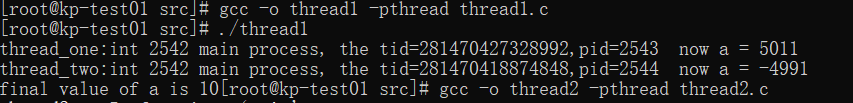
**错误：在编译程序的时候报错，查询知为linux系统中的pthread不是默认的库所以编译时需要添加-pthread命令**

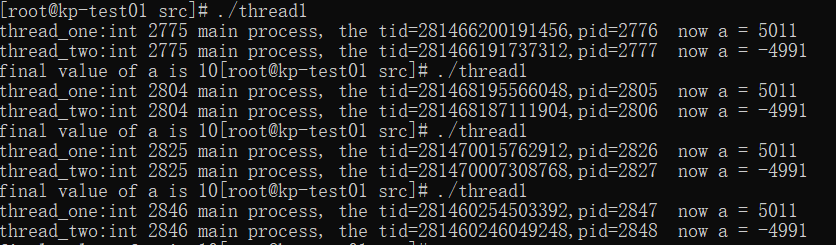
****



**（**<https://blog.csdn.net/freiheit_zz/article/details/118030587>**）**

**运行的结果如下：**

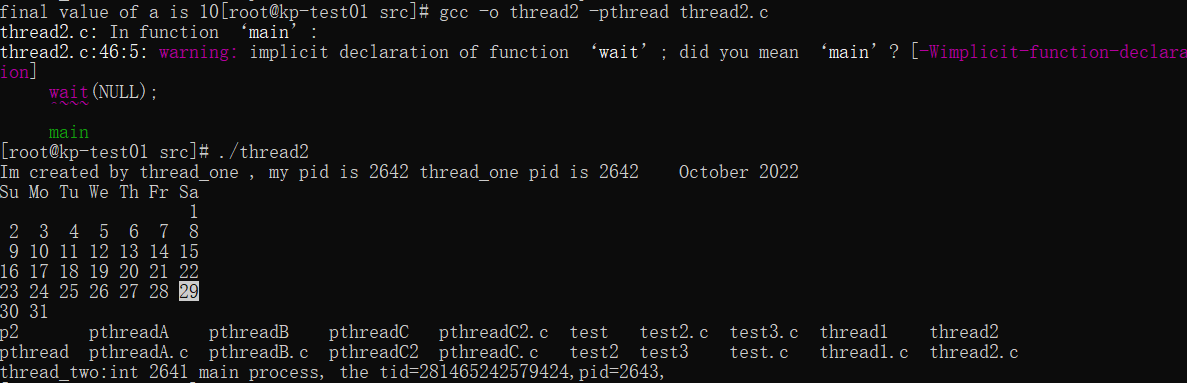
****



**分析：运行的结果都是相同的，程序中两个线程分别拷贝了变量a的值并对拷贝的值进行修改，而对a没有影响，故两个线程分别访问a变量的顺序就算不同也不会有影响。考虑互斥，可以考虑增加一个互斥锁，在一个线程访问a的时候打开互斥锁，访问结束之后解锁即可，而同步可以引入条件变量或者信号量，当A线程在访问a变量时，B线程在等待条件变量，从而达到同步的目的。**

**进一步修改程序：将任务一中第一个实验调用system函数和调用exec族函数改成在线程中实现，观察运行结果输出进程PID与线程TID进行比较并说明原因**

**运行的结果如下：**

****

**分析：execve函数族并不创建新的进程，故其pid的值是不变的，故均为2642. 父进程创建了新的线程，可以发现，新线程的tid（此处最后一个pid是线程的）与原进程pid的值不同，可推测是由于线程的创建过程与进程类似，导致了pid的不同。**