**操作系统课程设计 实验报告**

姓名：刘天祺

班级：07121502

学号：1320151097

学院：计算机学院

专业：物联网工程

日期：2018年3月21日

**实验二　进程控制**

1. 实验要求

设计并实现类似Unix的“time”命令：“mytime”。“mytime”命令通过命令行参数接收要运行的程序，创建一个独立的进程来运行该程序，并记录该程序运行的时间。

实现Windows版本和Linux版本。

1. 实验环境

2.1 Linux环境

操作系统：Ubuntu 14.04.5 LTS 64bit

Shell：zsh 5.0.5 (x86\_64-pc-linux-gnu)

编译器：gcc 4.8.5 (Ubuntu 4.8.5-2ubuntu1~14.04.1)

2.2 Windows环境

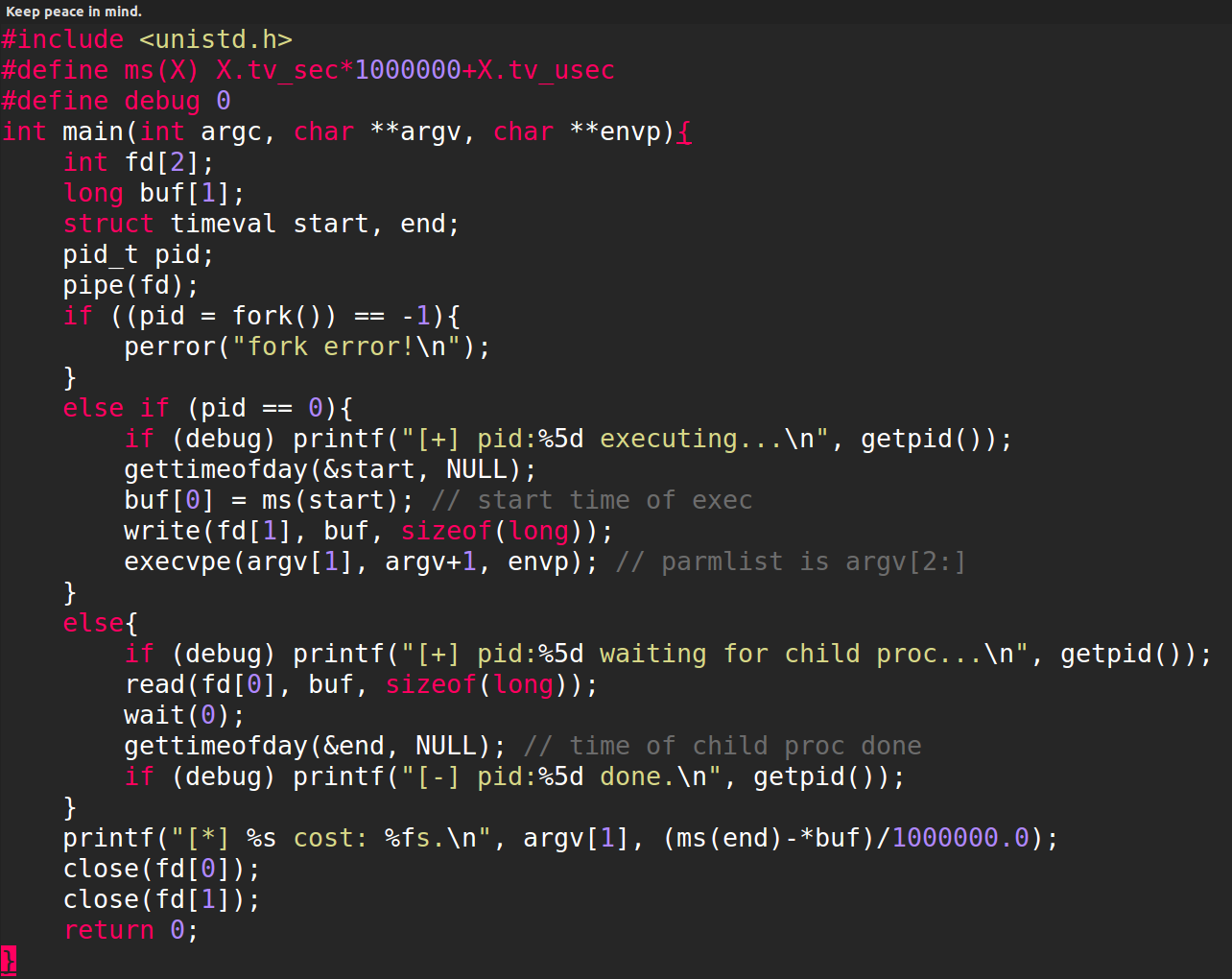
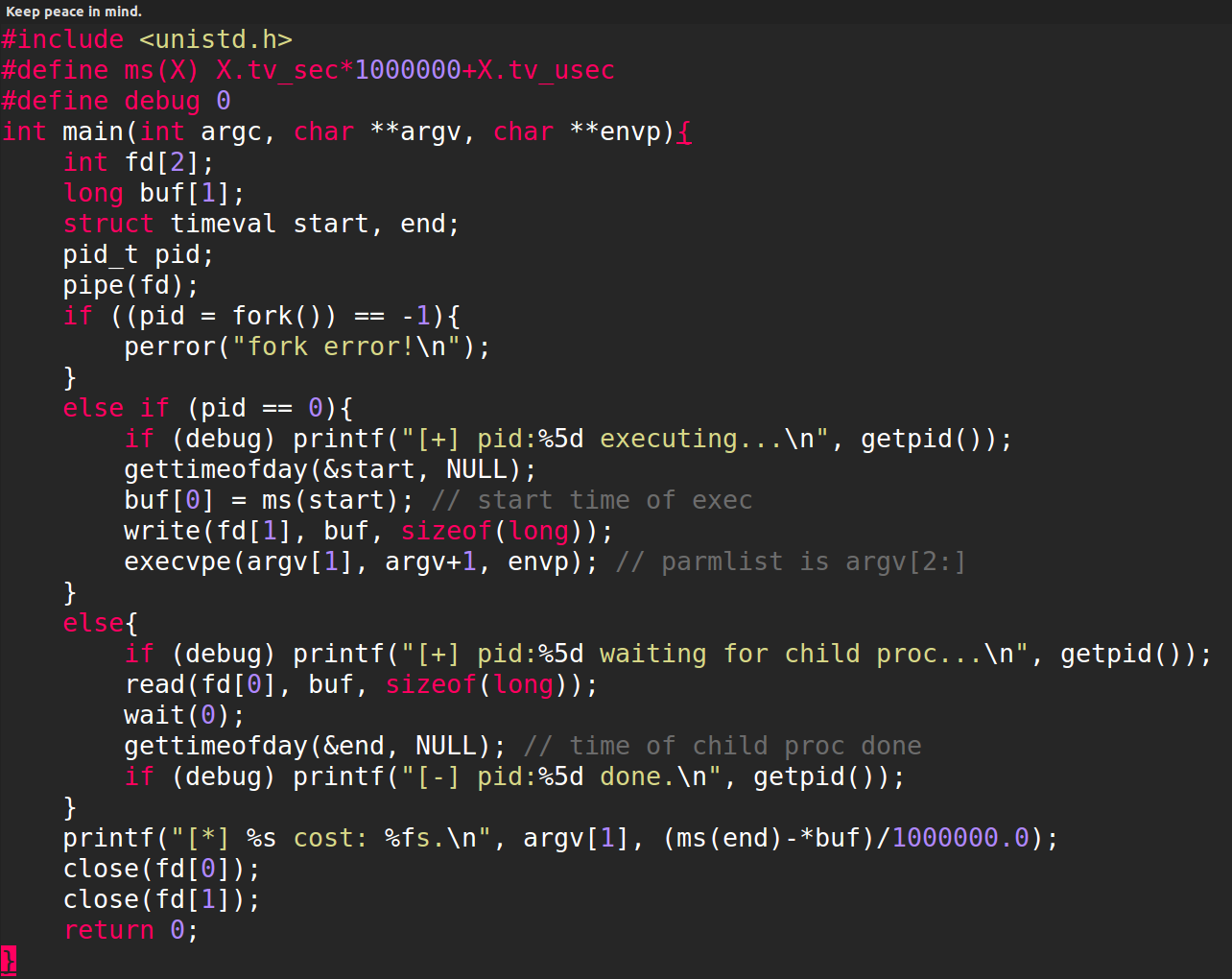
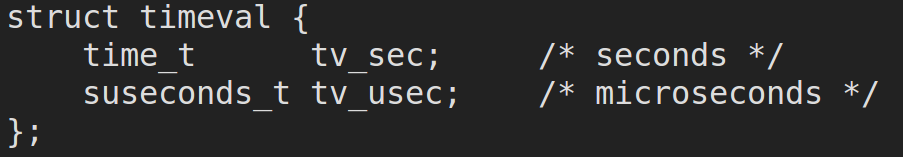
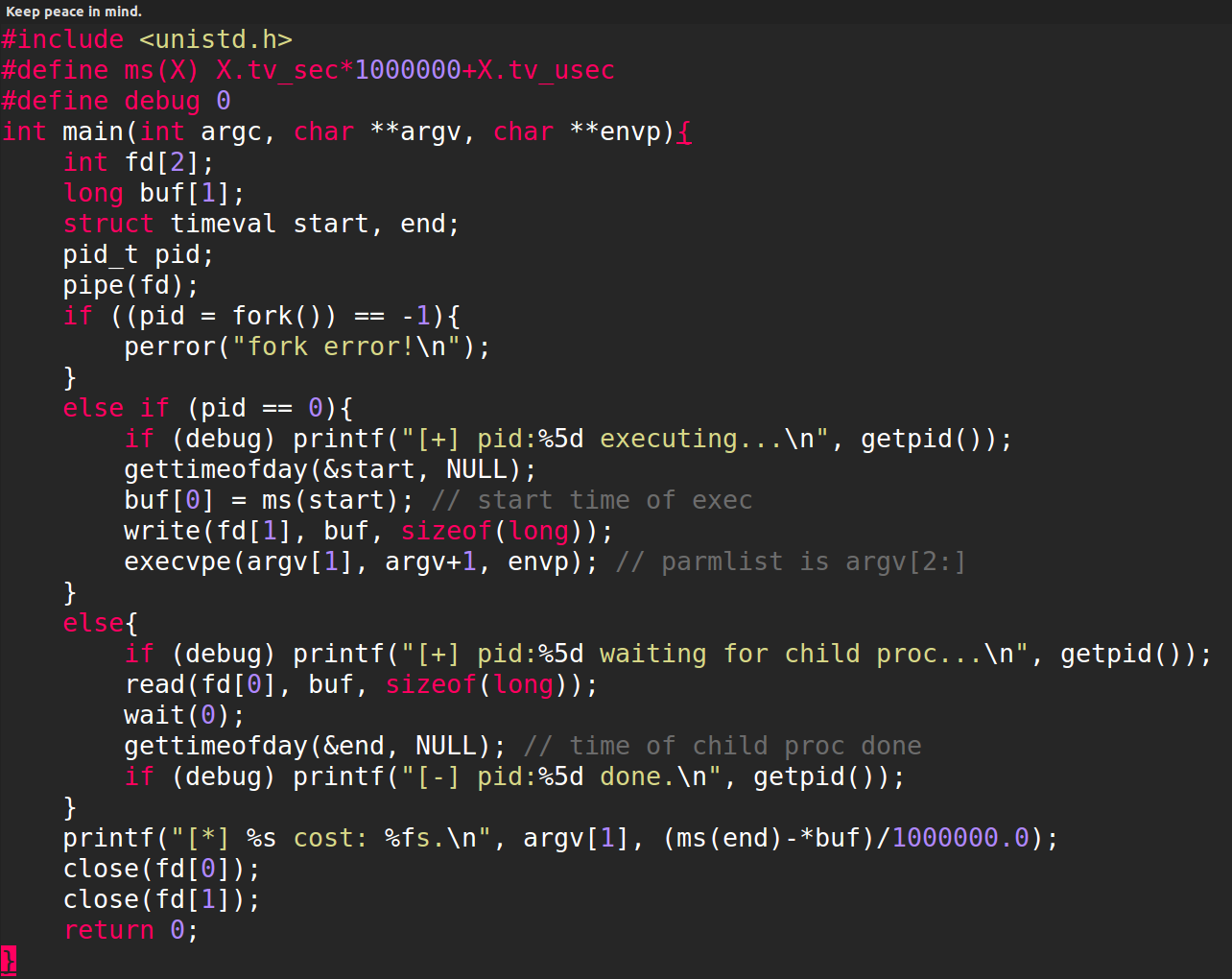
操作系统：Windows 10 64bit

Shell：cmder 160710

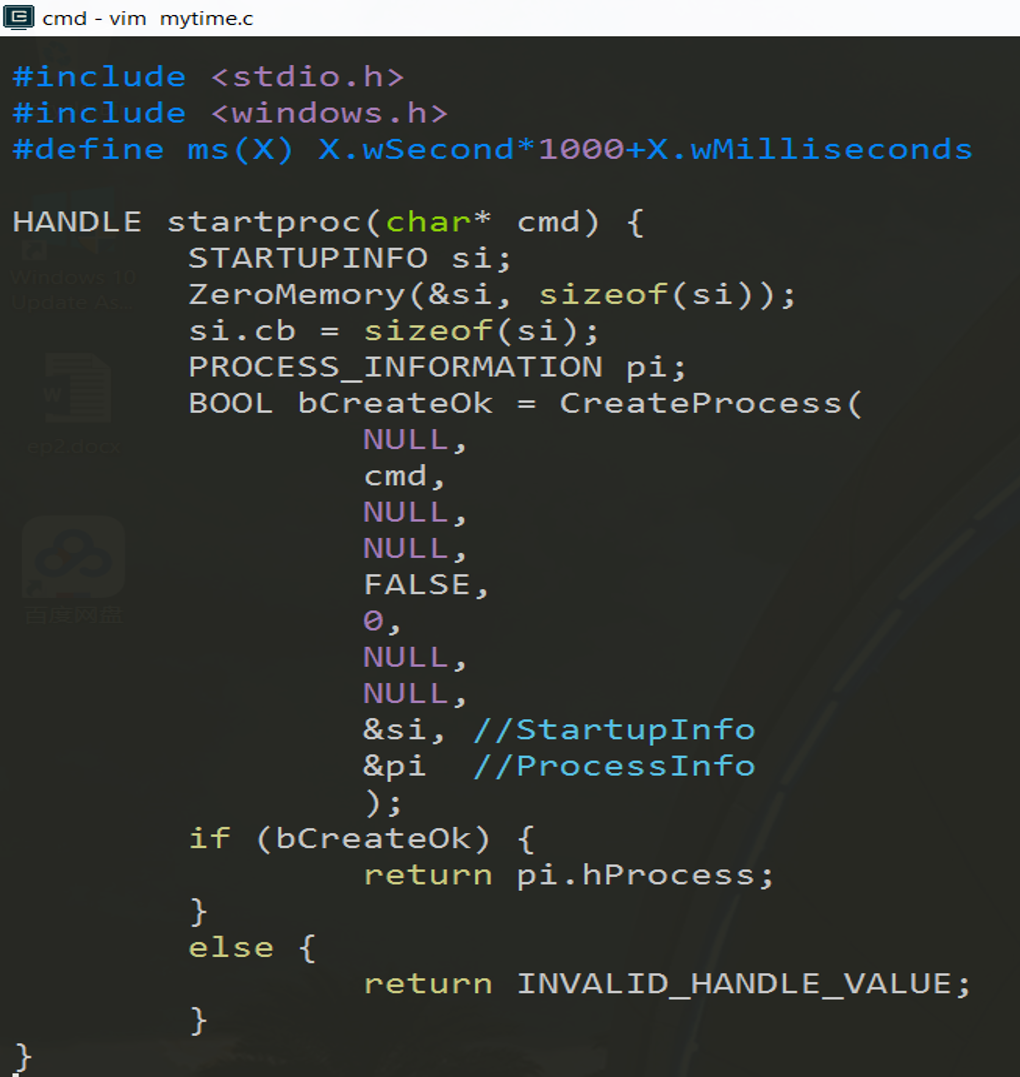
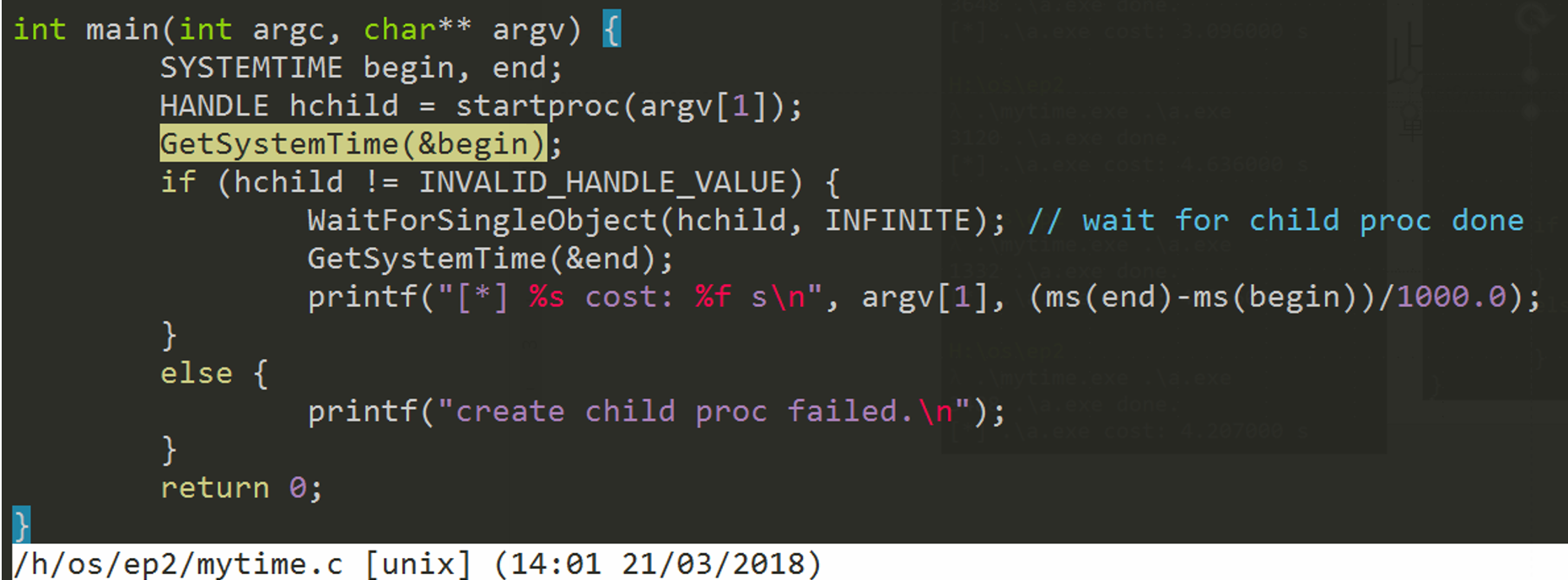
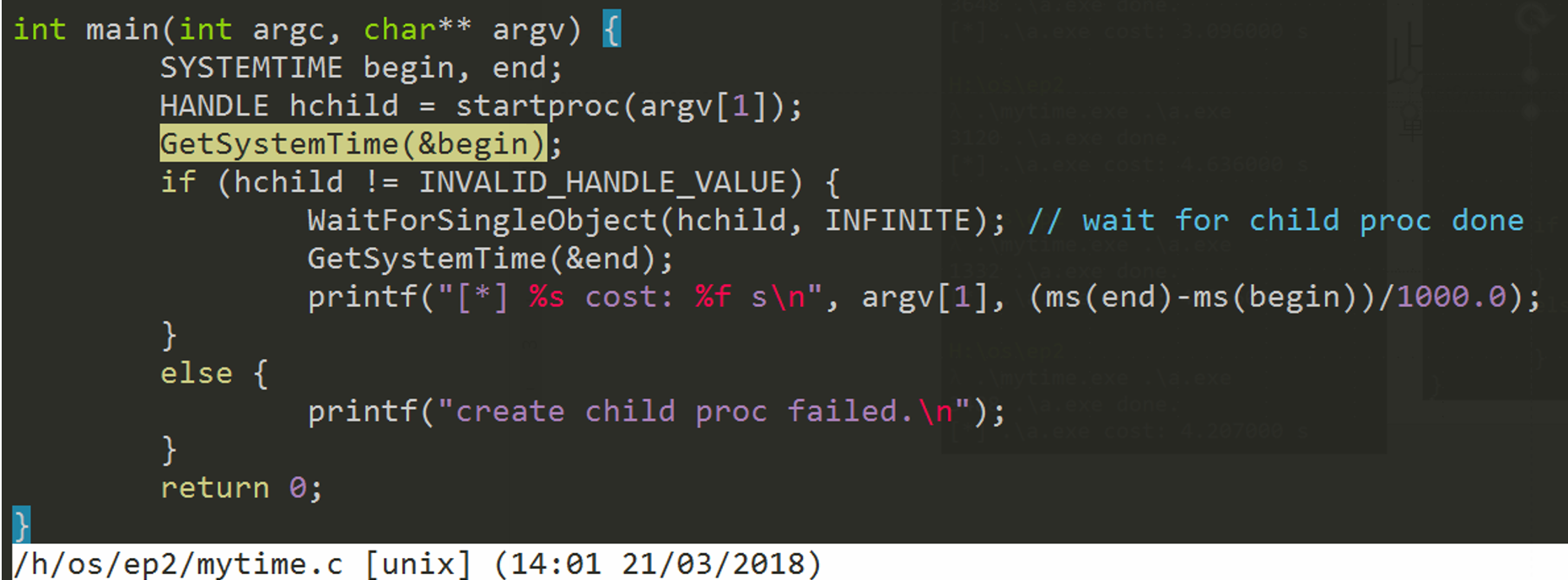
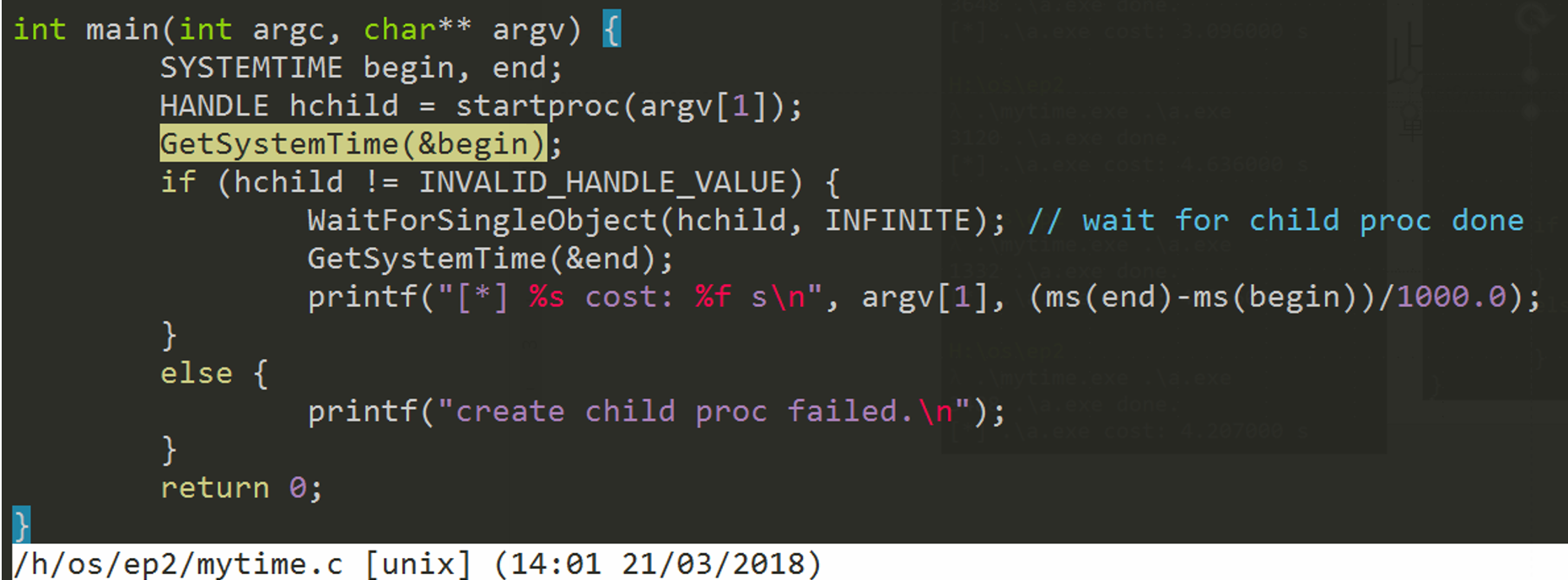
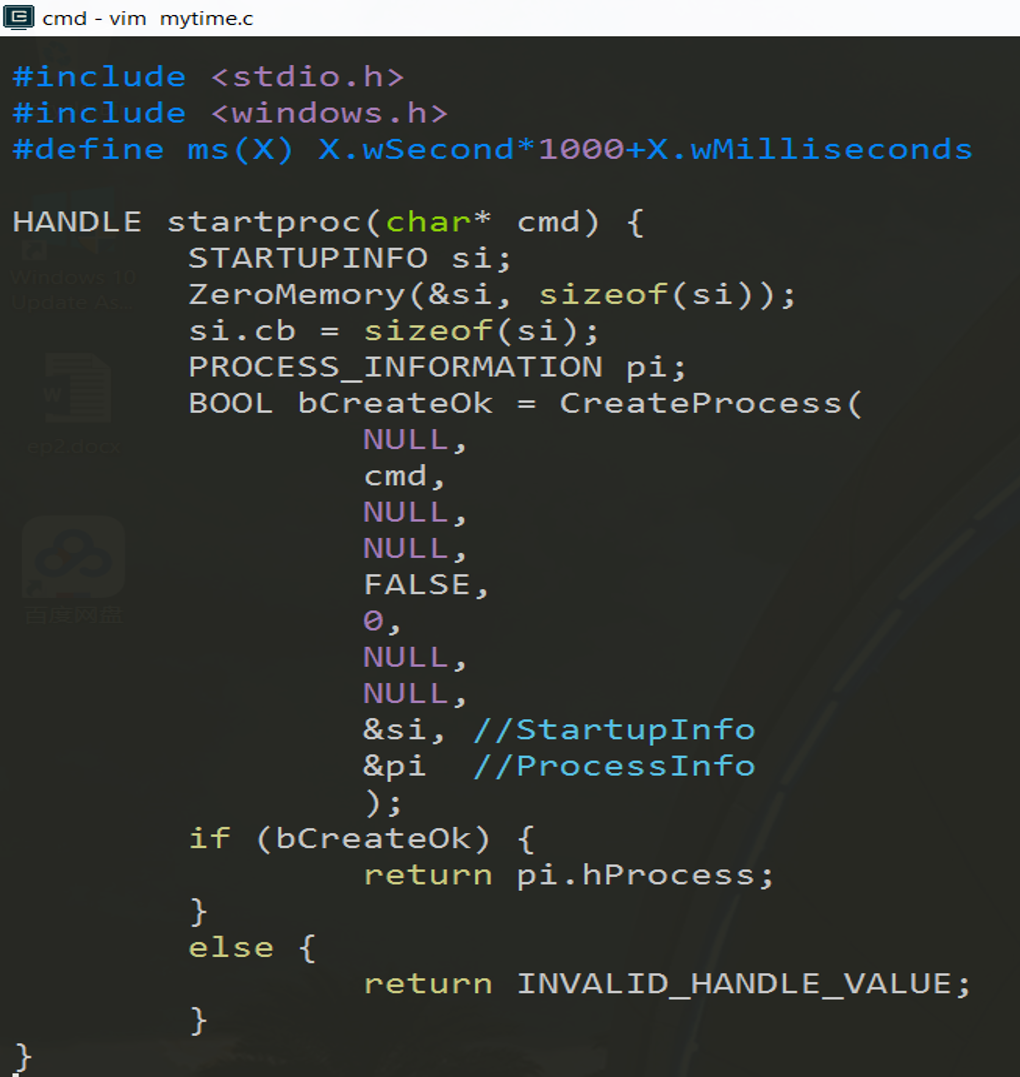
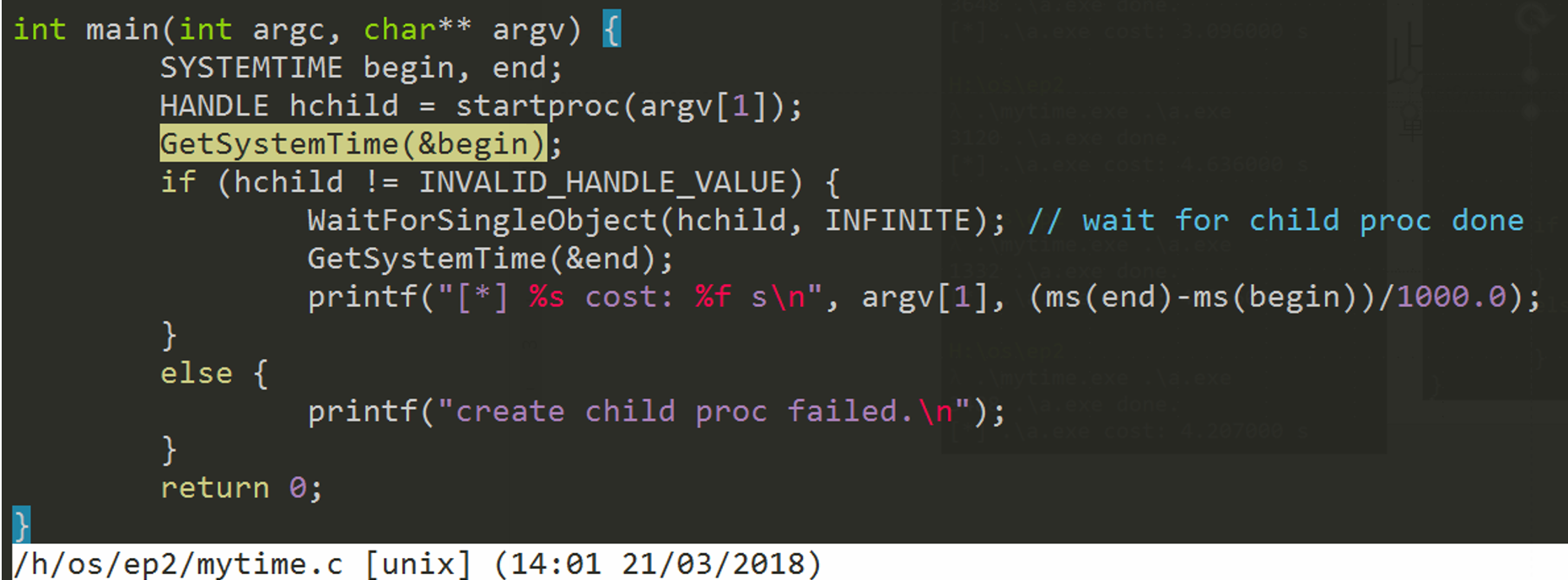
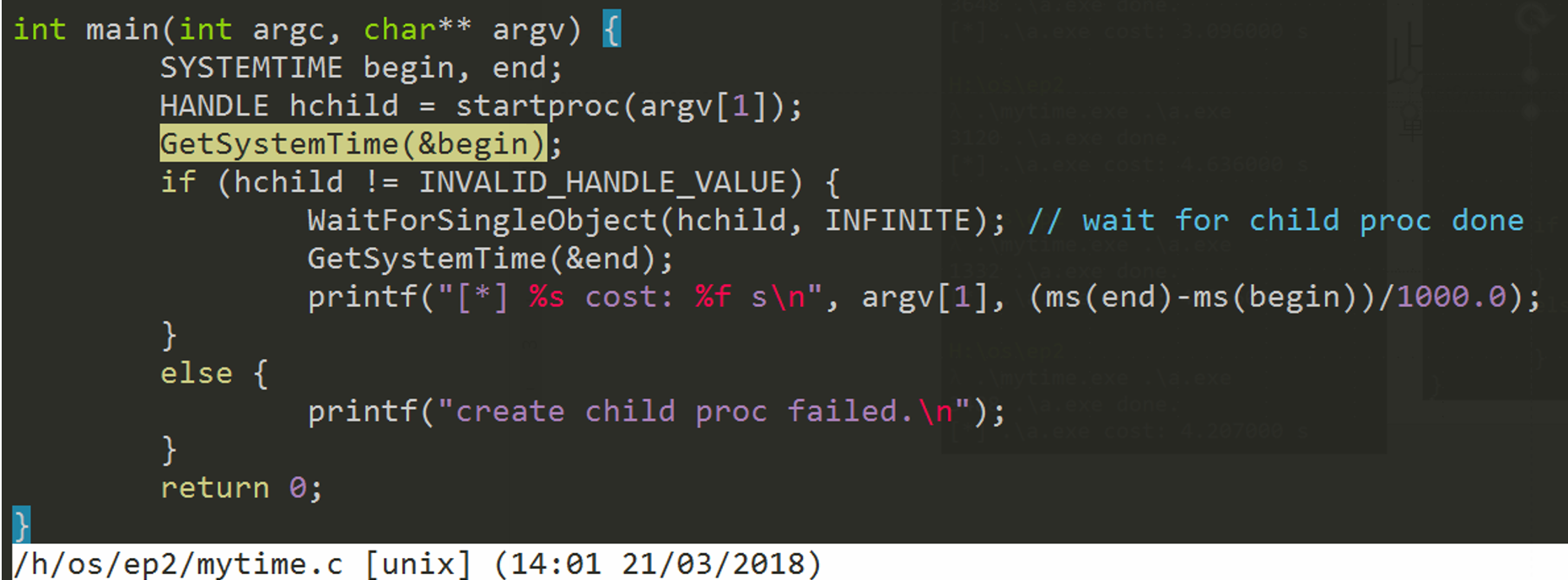
编译器：gcc 3.4.5 (mingw-vista special r3)

1. 实验步骤

3.1 在Linux环境实现

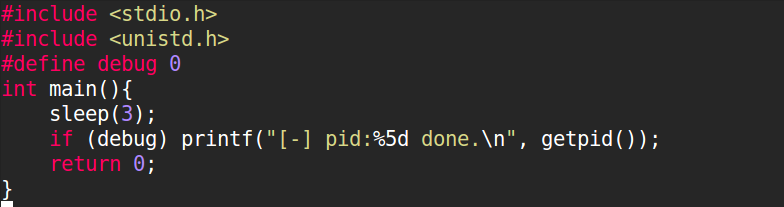
* 使用fork()/execvpe()来创建进程运行程序  
  **使用fork()创建子进程后，在子进程中使用execvpe()运行程序P。  
  P为mytime的第一个参数，mytime的第二个及其后面的参数为P的参数，P的运行环境与mytime相同。
* 使用wait()等待新创建的进程结束  
  **
* 调用gettimeofday()来获取时间  
  gettimeofday函数定义在头文件<sys/time.h>中，用于获取系统时间。其用于存储时间信息的结构体timeval，定义如下：  
    
  方便起见，本实验中定义ms宏，将timeval类型的变量转化为以微秒为单位的时间戳，ms宏定义如下：  
  
* 计算程序运行的时间**由于使用fork()创建子进程，子进程与父进程的变量不共享，故在程序中使用pipe()进行进程间的通讯：子进程开始执行时，获取系统时间，即子进程开始执行的时间，随后将其开始时间发送至写管道；父进程通过读管道获取子进程的开始时间，在等待子进程结束执行后，再次获取系统时间，即子程序执行完毕的时间。  
  　　二者的时间差即为mytime通过命令行参数接收到的要运行的程序的执行时间。

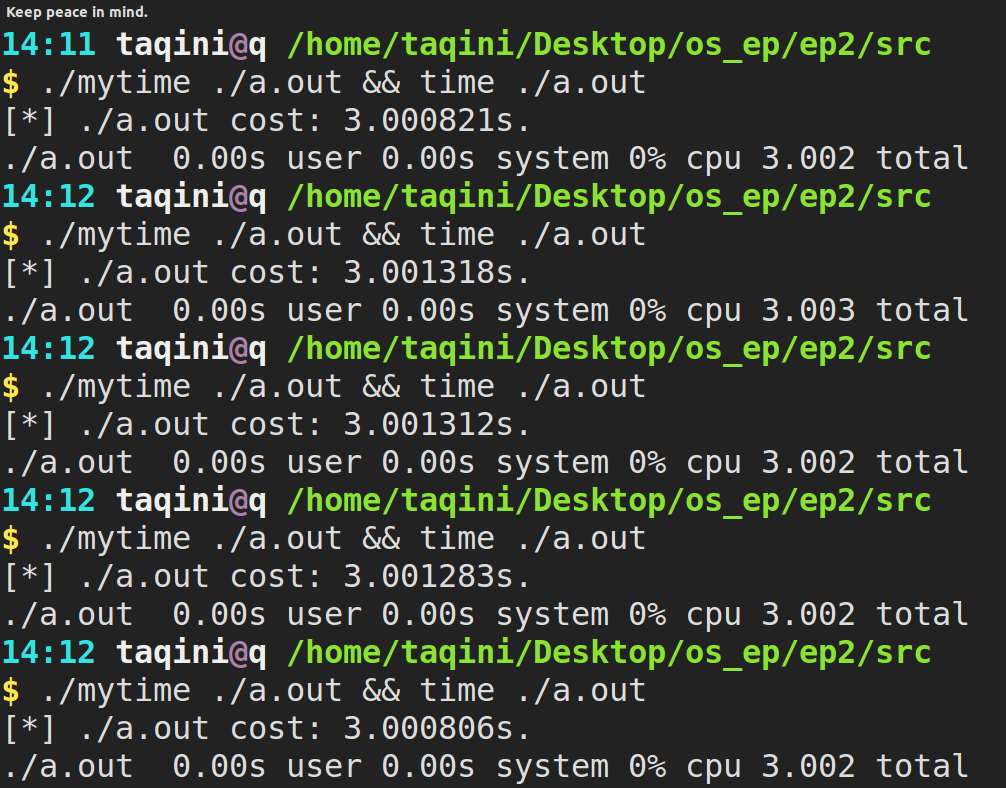
3.2 在Windows环境实现

* 使用CreateProcess()来创建进程  
  通过指定CreateProcess的第二个参数(lpCommandLine)为要运行的程序，即mytime的首个参数。实现了创建进程并运行程序。  
  
* 使用WaitForSingleObject()在“mytime”命令和新创建的进程之间同步  
    
  
* 调用GetSystemTime()来获取时间  
  该API使用SYSTEMTIME类型记录时间信息。  
  本次实验中使用其结构体中的wSecond和wMilliseconds。  
  方便起见，定义ms宏，将SYSTEMTIME类型变量转换为以毫秒为单位的时间戳，ms宏定义如下：  
  
* 计算程序运行的时间  
  在进程创建并执行后获取系统时间，如下：  
  在等待进程结束后再次获取系统时间，如下：  
    
  二者的时间差即为mytime通过命令行参数接收到的要运行的程序的执行时间。

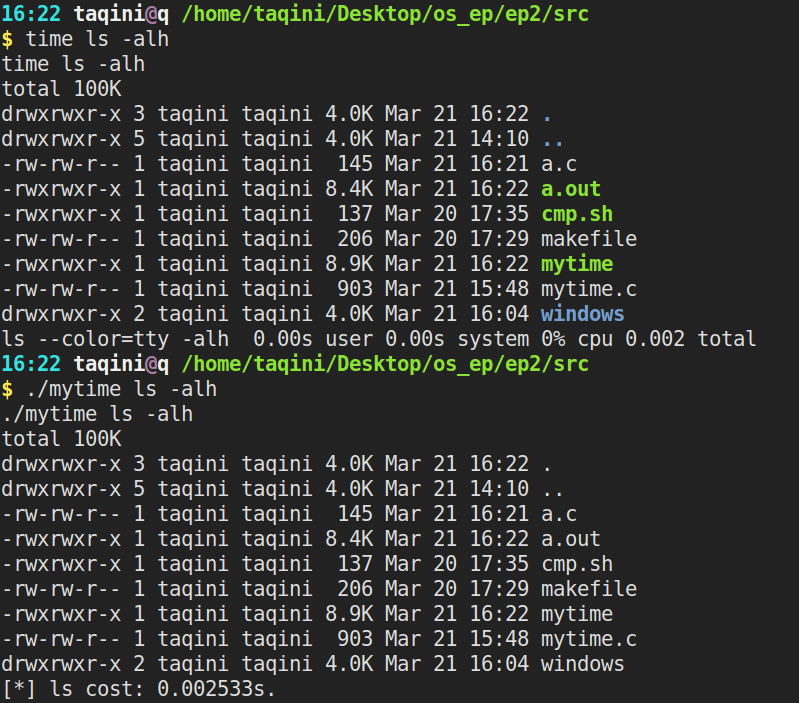
1. 实验结果

4.1 在Linux环境下的实验结果

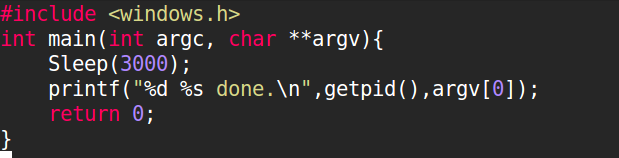
要运行的测试程序源代码a.c如下：  
使用gcc将测试程序源代码a.c和mytime命令源代码mytime.c分别编译为可执行文件mytime和a.out。

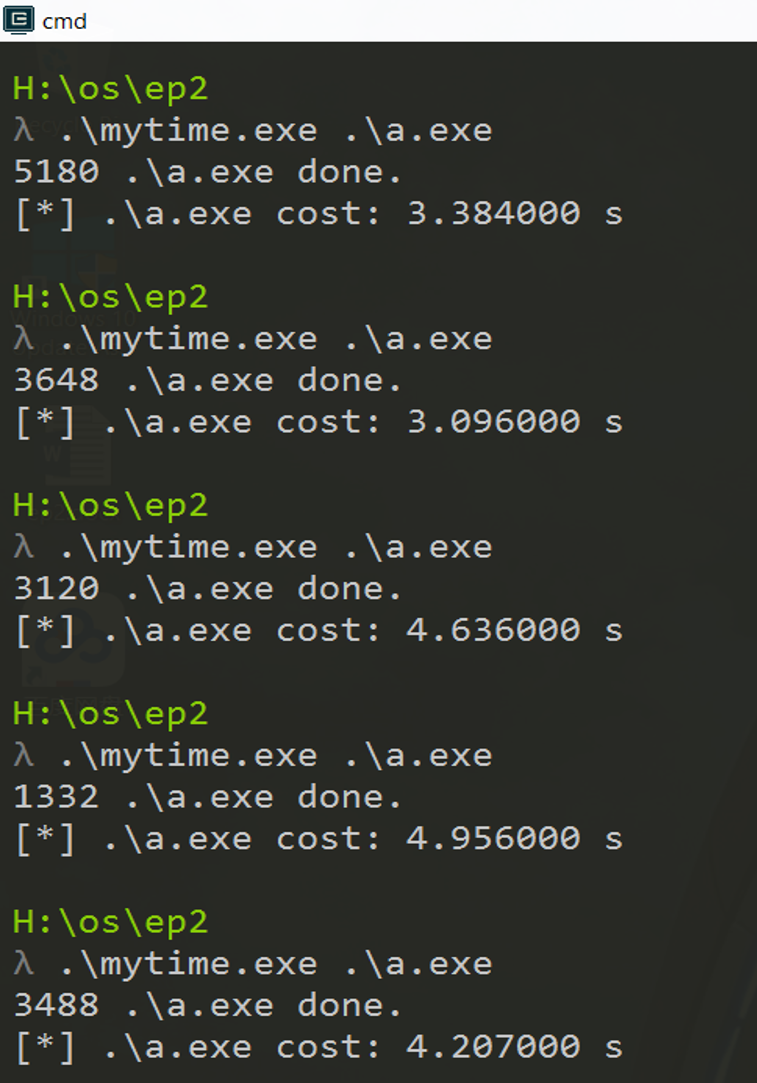
指定./a.out文件作为参数，先后执行mytime命令和系统time命令，执行5次后的结果如下图：  
  


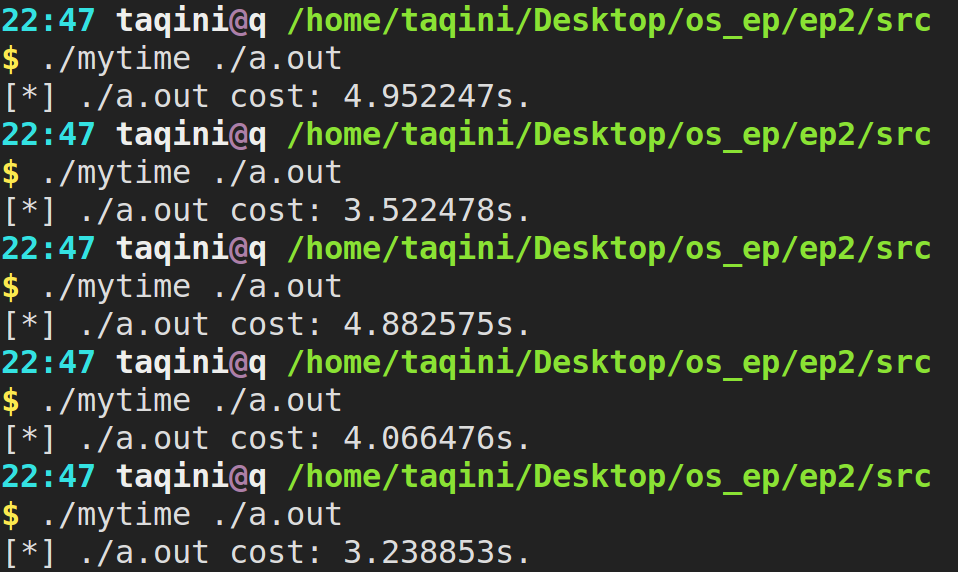
由mytime命令统计的a.out的执行时间平均为3.001108秒；  
由time命令统计的a.out的执行时间平均为3.0022秒。  
mytime命令的误差约为0.04%。

此外，本实验中编写的mytime程序实现了与time命令类似的功能，可以统计**带参数**的程序执行的时间，如下：

4.2 在Windows环境下的实验结果

要运行的测试程序源代码a.c如下：使用gcc将测试程序源代码a.c和mytime命令源代码mytime.c分别编译为可执行文件mytime.exe和a.exe。

指定.\a.exe文件作为参数，执行mytime.exe命令5次后的结果如下图：  
由mytime命令统计的a.exe的执行时间平均为4.0558秒。

1. 实验总结  
    在linux环境下，mytime命令在统计新运行的程序的时间时，对于同样一个主函数中只有sleep(3)的程序，如果不使用管道通讯，在主进程开始时记录开始时间，在主进程等待子进程结束后记录结束时间，则他们的差值在4s左右，和使用管道通讯得到的差值3s相差不少。我认为原因在于子进程和主进程开始的时间不同，fork()调用之后，创建了子进程，此时可能因为操作系统任务调度、内存换页等因素导致主进程并未立即执行下去。因此在调用execv()之前记录开始时间较为合理，对比于系统的time命令，mytime和time统计的时间相近，因此可以猜测，time命令的源码中，可能就是通过pipe()实现的子进程和主进程之间通讯，从而获取的子进程执行时间。  
    未使用管道的mytime执行结果，如下图：  
   

相较于Win32 API，linux的系统调用更加简练明了，给人一种短小精悍的感觉，Win32 API虽然功能强大，但是使用起来过于复杂，不光函数名字长，函数的参数也多。API固然方便开发者使用，但是同时也限制住了开发者的思维，如果把编程比作搭建一间房子，linux的系统调用像是一块砖一片瓦，程序员做的事情是用水泥把他砖头一块一块的垒起来，把瓦片一片一片的铺上去，而Win32 API则像是请建筑工队帮忙搭建房间，程序员做的事情则是指挥建筑工队如何搭房子。对于我来说，使用linux的系统调用编程更加轻松愉快。