操作系统实验报告

实验二 进程控制

2019-3-15

北京理工大学计算机学院

谭超 1120161874

操作系统课程设计实验报告

实验名称：　　进程控制

姓名/学号：　　谭超　1120161874

一、　实验目的

分别在 Windows 系统下和 Linux 系统设计并实现 Unix 的”time”命令。”mytime”命令通过命令行参数接受要运行的程序，创建一个独立的进程来运行该程序，并记录程序运行的时间。

二、　实验内容

在 Windows 系统下实现：

使用 CreateProcess()来创建进程

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

在 Linux 系统下实现：

使用 fork()/vfork()/exec()来创建进程运行程序

调用 wait()等待新创建的进程结束

调用 gettimeofday()来获取时间

mytime 的用法：

$ mytime.exe program　　//program 是一个可执行程序

$ mytime.exe program time　//time 是一个时间参数，单位为秒

三、　实验环境

Windows:

Windows10 1803

处理器 Intel Core i5-6300HQ 2.3GHz ,RAM 8G

Linux:

VMware Workstation 14.1.3

Ubuntu 4.13.0.36

四、　程序设计与实现

Windows:

首先要准备创建新进程所需的参数，由于我们需要从命令行读取可执行文件名称以及运行时间，所以我将可执行文件名通过 lpCommendLine 传给新进程。lpProcessAttributes、lpThreadAttributes、lpEnvironment 和 lpCurrentDirectory 都设为默认，然后不需要继承原有进程的句柄，所以 bInheritHandles 参数设为 false。然后还有 lpStartupInfo 和 lpProcessInformation 两个参数，lpStartupInfo 是一个 STARTUPINFO 类型的参数，故而定义一个 SATRTUPINFO 类型的变量，并对其初始化，lpProcessInformation 则是一个 PROCESS_INFORMATION 类型的变量，

```
STARTUPINFO si; //进程启动相关信息的结构体
memset(&si, 0, sizeof(STARTUPINFO));
si.cb = sizeof(STARTUPINFO);     //应用程序必须将cb初始化为sizeof(STARTUPINFO)
si.dwFlags = STARTF_USESHOWWINDOW;  //窗口标志
si.wShowWindow = SW_SHOW;
PROCESS_INFORMATION pi; //必备参数设置结束
```

确定了参数，然后就可以创建一个新的进程：

```
CreateProcess(NULL, argv[1], NULL, NULL, FALSE, CREATE_NEW_CONSOLE, NULL, NULL, &si, &pi);
```

然后调用 GetSystemTime 函数来获取当前时间：

```
GetSystemTime(&time_start);
```

创建新的进程后，通过有无时间参数来决定程序运行的时间：

```
if (argv[2] == NULL)
    WaitForSingleObject(pi.hProcess, INFINITE);
else
    WaitForSingleObject(pi.hProcess, atoi(argv[2])*1000);//程序运行时间以秒为单位
```

新进程运行结束后再次调用 GetSystemTime 获取当前时间：

```
GetSystemTime(&time_end);
```

然后通过两次时间之差得到新线程的运行时间。

结果：

```
D:\SourceCode\C++SourceCode\VS2017\mytime\Debug>mytime out.exe
创建进程成功!
程序运行时间: 17秒163毫秒
```

```
D:\SourceCode\C++SourceCode\VS2017\mytime\Debug>mytime out.exe 1
创建进程成功!
程序运行时间: 1秒16毫秒
```

Linux:

首先定义一个 pid 用来指向子进程的标识，然后通过 fork()函数创建一个子进程：

```
pid_t pid;
pid = fork();
```

此时调用 gettimeofday()函数获取当前时间

```
gettimeofday(&time_start,NULL);
```

然后判断是子进程还是父进程，在子进程中，通过判断有无时间参数即第二个参数来判断 execl()函数参数，由于在命令行调用一个程序时，argv[0]为程序名称，此处为了和外部调用保持一致，将 execl()的第一个参数也设为程序名，如果时间参数则将第二个参数为运行时间：

```
if(argv[2]==NULL)
    execl(argv[1],argv[1],NULL);
else
    execl(argv[1],argv[1],argv[2],NULL);
```

在父进程中，首先等待子进程运行结束，然后通过 gettimeofday()获取当前时间，通过两次时间之差获得子线程运行时间，此处注意时间差的微秒可能为负，所以要进行判断，最后输出时间：

```
wait(NULL); //等待子进程结束
gettimeofday(&time_end,NULL);
long time_sec=time_end.tv_sec - time_start.tv_sec;
long time_usec=time_end.tv_usec - time_start.tv_usec;
if(time_usec<0)
{
    time_sec-=1;
    time_usec+=1000000;
}
printf("Program running time: %lds%ldus\n",time_sec,time_usec);
```

自此，mytime 程序编写完成，开始编译：

```
new@ubuntu:~/test1/OS_experiments$ g++ -o mytime mytime.cpp
```

对于被调用的程序，我将其命名为 out,当没有时间参数时，输出 0 到 10000，当有时间参数时，从 0 开始自加输出，直到运行时间接近给定的时间参数。

```
new@ubuntu:~/test1/OS_experiments$ g++ -o out out.cpp
```

然后分别运行没有时间参数的和有时间参数的。

```
new@ubuntu:~/test1/OS_experiments$ ./mytime /home/new/test1/OS experiments/out
Program running time: 0s34374us
new@ubuntu:~/test1/OS_experiments$ ./mytime /home/new/test1/OS experiments/out 5.2
Program running time: 5s267689us
```

五、 实验收获与体会

通过本次实验，我对进程有了更加深入的认识，特别是对于创建进程的过程以及子进程和父进程之间的关系。在 Windows 系统中，对于 CreateProcess()参数的传递、WaitForSingleObject()函数的作用以及 GetSystemTime()函数的返回类型的结构等都有了一个比较深刻的认识。在 Linux 系统中，也初步掌握了创建进程的函数 fork()、获取当前时间的函数 gettimeofday()以及进程中调用程序的函数 execl()，另外对于进程的并行性有了一个更加深入的认识。