操作系统课程设计实验报告

实验名称	进程控制		
学号	1120191600	姓名	张驰

一、实验目的

设计并实现 Unix 的“time”命令。用 mytime 命令记录某可执行程序的运行时间。要求用命令行参数接受某可执行程序,并为该可执行程序创建一个独立的进程。“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

设计并实现 Unix 的“time”命令，取名为“mytime”。

用 mytime 命令记录某可执行程序的运行时间。要求用命令行参数接受某可执行程序，并为该可执行程序创建一个独立的进程。

在 Windows 下实现:

1. 使用 CreateProcess()来创建进程

2. 使用 WaitForSingleObject()在“mytime”命令中和新创建的进程之间同步

3. 调用 GetSystemTime()来获取时间

在 Linux 下实现:

1. 使用 fork()/execv()来创建进程并运行新的可执行程序

2. 使用 wait()等待新创建的进程结束

3. 调用 gettimeofday()来获取时间

通过使用 mytime.exe 程序，在命令行中调用 windows 中的其他程序，例如 notepad，同时使用 mytime.exe 程序调用另一个程序 program2.exe，控制其运行的时间。

三、实验环境及配置方法

Windows10

VMware workstation pro

Ubuntu18

四、实验方法和实验步骤（程序设计与实现）

mytime.exe 的设计思路如下:

1. 首先通过设定进程信息、进程启动信息等参数，初始化进程控制所需要的信息

2. 创建子进程，并将 argv[]中的参数传递给子进程，同时让主进程开始计时

3. 主进程等待子进程结束，子进程结束后主进程立刻停止计时

4. 结束时间减去开始时间，对结果进行借位处理，输出运行的时间。

下面介绍在 Windows 系统和 Linux 系统中的具体实现：

1. 在 Windows 系统中的实现：

(1)制定系统的时间变量，初始化进程的启动信息和进程的信息：

```
SYSTEMTIME starttime, endtime;//制定系统时间变量，为启动时间，结束时间
PROCESS_INFORMATION pro_info;//进程的信息
STARTUPINFO pro_start_info;//进程的启动信息

//该结构中的所有成员初始化为零，然后将 cb 成员设置为该结构的大小
memset(&pro_start_info, 0, sizeof(pro_start_info));
pro_start_info.cb = sizeof(pro_start_info);
```

(2)取出命令行中的参数：

```
char cmd[1000];
memset(cmd, 0, sizeof(cmd));
//取出命令行中的参数
for (int i = 1; i < argc; i++) {
 strcat(cmd, argv[i]);
 strcat(cmd, " ");
}
```

(3)调用 CreateProcess()函数创建进程：

```
//根据命令行中的参数创建进程
if(CreateProcess(NULL, cmd, NULL, NULL, FALSE, 0, NULL, NULL, &pro_st
art_info, &pro_info)){
 GetSystemTime(&starttime);
}
```

其中 CreateProcess 中各个参数的含义如下所示：

NULL 表示使用命令行，cmd 表示命令行的具体内容，NULL,NULL 表示均不继承进程句柄和线性句柄，FALSE 表示不继承当前进程句柄，0 表示无进程创建标志，NULL,NULL 表示使用父进程的环境和目录，&si，&pi 表示使用进程启动信息地址和进程的信息地址。

此时，如果成功打开子进程，就记录一下开始的时间。

(4)主进程调用 WaitForSingleObject 等待子进程结束，进入阻塞状态。

```
WaitForSingleObject(pro_info.hProcess, INFINITE);
```

```
GetSystemTime(&endtime);
CloseHandle(pro_info.hProcess);
```

(5)计算进程运行的时间：

```
//计算进程运行时间
int hour = endtime.wHour - starttime.wHour;
int minute = endtime.wMinute - starttime.wMinute;
int second = endtime.wSecond - starttime.wSecond;
int milliseconds = endtime.wMilliseconds - starttime.wMilliseconds;
if (minute < 0) {
 hour -= 1;
 minute += 60;
}
if (second < 0) {
 minute -= 1;
 second += 60;
}
if (milliseconds < 0) {
 second -= 1;
 milliseconds += 1000;
}
cout << hour << "小时" << minute << "分" << second << "秒
" << milliseconds << "毫秒";

return 0;
```

2. 在 Linux 系统中的实现：

(1)制定系统的时间变量，将 mytime 文件路径加入到临时环境变量中，这样即可通过./mytime program 直接调用 mytime 目录下的可执行文件：

定义系统的时间对象：

```
struct timeval t_start, t_end;
int status = -1;
```

加入临时环境变量：

```
char path[100],dic[100];
memset(path,0,sizeof(path));
memset(dic,0,sizeof(dic));
// connect string of environment path
getcwd(dic,sizeof(dic));
strcpy(path,getenv("PATH"));
strcat(path,":");
strcat(path,dic);
//set
setenv("PATH",path,1);
```

(2)创建子进程，通过 pid 值区分父子进程，fork()函数的返回值为 pid，返回值为负数时，表示 fork()函数调用错误，返回值为 0，别是目前在子进程中，其他值表示目前在父进程中。

```
//use fork() to create a process
int pid = fork();
if(pid < 0){
 // here is error
 printf("error!\n");
}
else if(pid == 0){
 // here is child process
 printf("this is child process.\n");
 execvp(argv[1],argv+1);
}
```

其中，execvp()是在子进程中运行命令行的命令，打开所调用的程序。

(3)在父进程中等待子进程的结束，调用 wait()，之后计算时间。

```
else{
 // here is parent process
 gettimeofday(&t_start,NULL);
 wait(&status);
 gettimeofday(&t_end,NULL);
 int sec = t_end.tv_sec - t_start.tv_sec;
 int usec = t_end.tv_usec - t_start.tv_usec;
 if(usec < 0){
  sec -= 1;
  usec += 1000000;
 }
 printf("The child run for %d seconds %d useconds\n",sec,usec);
}
return 0;
```

五、实验结果和分析

1. 在 Windows 系统中的运行结果：

使用 mytime.exe 打开 python：

```
D:\张驰\学习\OS课设\实验2\windows>mytime.exe notepad
0小时0分2秒703毫秒
D:\张驰\学习\OS课设\实验2\windows>mytime.exe python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
0小时0分10秒404毫秒
```

使用 mytime.exe 调用 program2.exe，使 program2.exe 运行指定的时间：

```c
#include<stdio.h>
#include<windows.h>

int main(int argc,char *argv[])
{
    int sleeptime = 0;
    sleeptime = atoi(argv[1]);
    printf("\n----------将休眠%s 秒----------\n",argv[1]);
    Sleep(sleeptime * 1000);
    return 0;
}
```

```
0小时0分10秒404毫秒
D:\张驰\学习\OS课设\实验2\windows>mytime.exe program2.exe 3

----------将休眠3秒----------
0小时0分3秒17毫秒
```

两者时间相差不多，17 毫秒，在误差允许的范围内。

2. 在 Linux 系统中的运行结果：

首先在 linux 系统中通过 mytime 来调用命令行中的 ls 指令，效果如下：

```
zc1120191600@ubuntu: ~/code
File  Edit  View  Search  Terminal  Help
zc1120191600@ubuntu:~$ cd code
zc1120191600@ubuntu:~/code$ ./mytime ls
this is child process.
a.out  mytime  mytime.c  program  program2.c  programm
The child run for 0 seconds 809 useconds
zc1120191600@ubuntu:~/code$
```

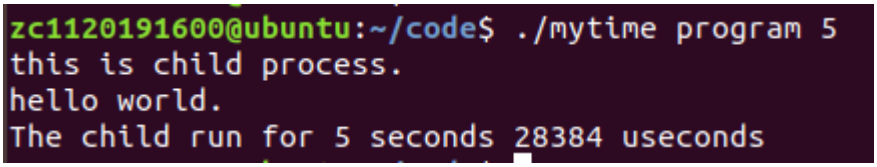
之后通过 Linux 系统运行 program，给 program 指定运行的时间：

program2.c 如下所示：

```c
#include <stdio.h>
#include <sys/time.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/time.h>
#include <unistd.h>

int main(int argc, char* argv[]){
 printf("hello world.\n");
 int sleeptime = atoi(argv[1]);
 sleep(sleeptime);
 return 0;
}
```

运行结果：



运行时间在误差允许的范围内，调用成功。

六、讨论、心得

这次实验过程较为简单，通过学习 Windows 系统与 Linux 系统下的各种 API、函数的具体用法，了解函数中参数含义，例如了解 exec 函数族各种用法的差异，也学习在 Linux 加入临时修改环境变量的方法，最终完成了实验要求。