实验三 Linux环境编程及GCC、Make

**一、实验目的**

1. 掌握系统调用的实现方式并能完成基本的文件读、写，进程管理操作；
2. 掌握GCC的使用方法 ；
3. 掌握MAKEFILE的编写规则，并能使用make完成大型程序的编译、管理。

**二、实验环境**

Ubuntu

**三、实验说明**

1. 实验报告以此为模板，正文中中文字体为小四宋体，英文字体为小四Times New Roman。
2. 实验报告以“Exp2-学号-姓名”方式命名，如Exp2-201401010101-张三.doc
3. 必要时使用man等帮助命令，或网上搜索。

**四、实验内容&实验结果（包括代码和运行结果截图）**

1. 使用gcc编译fork\_CPUTime.c源文件。

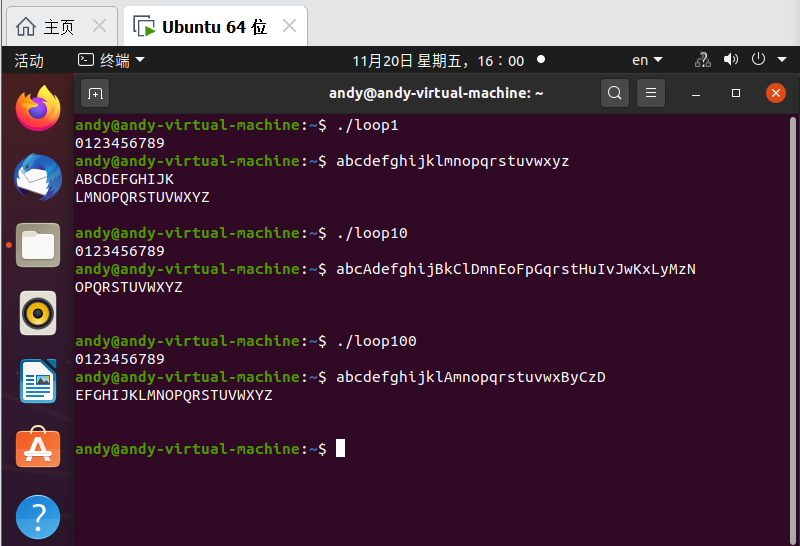
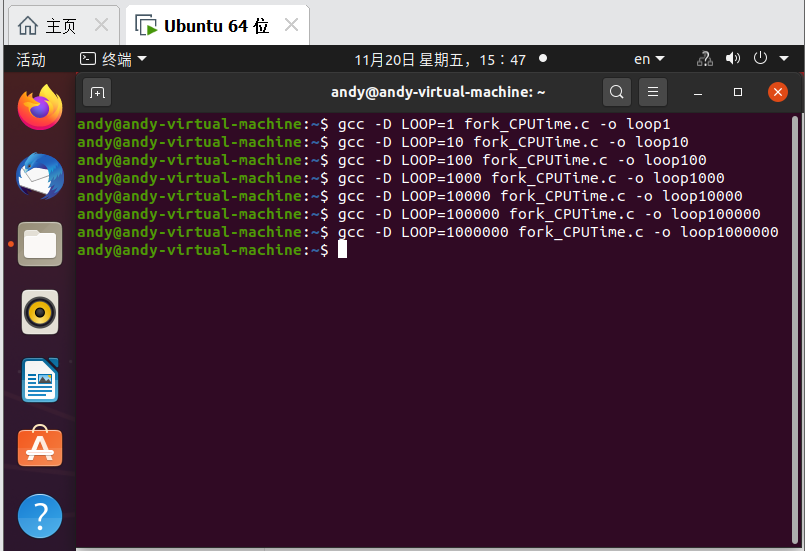
**要求：**

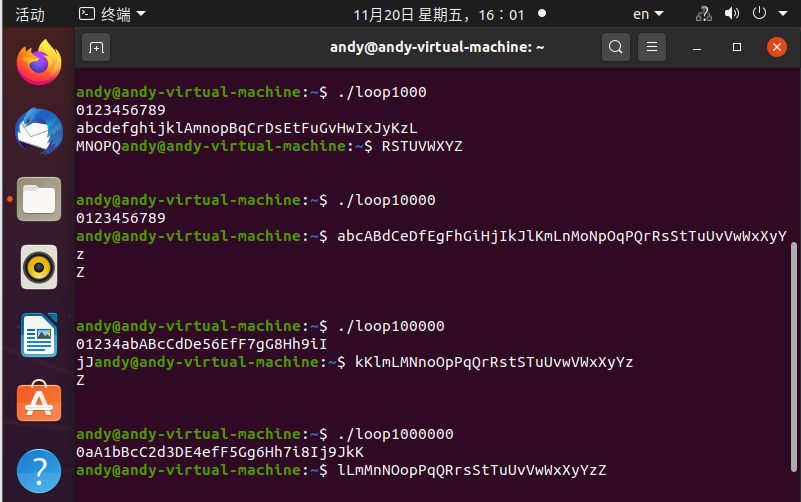
1） 通过正确的选项定义宏变量LOOP，并对其分别赋值1，10，100，1000，10000，100000，1000000；

2） 对每一次新的LOOP赋值，执行生成文件，对比结果；

3） 通过以上实例，说明Linux进程管理（CPU时间分配)的方式。

* 实验截图如下：



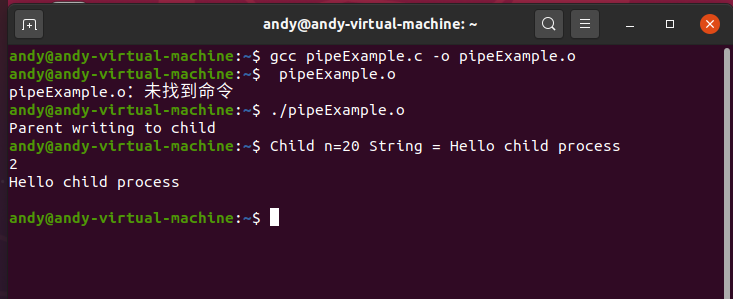


* 结果分析如下：

Linux系统内核在微观上，把CPU的运行时间分成许多时间片，然后分配给各个进程轮流运行，在宏观上看起来所有的进程仿佛同时执行。通过设置了LOOP循环的CPU延迟时间后可以发现输出结果里，不同字串序列是混在了一起，这说明是一个进程运行一会儿达到时间片限制后换另一个进程继续运行。随着LOOP值设置的不同，字串混杂的效果也不同，LOOP越大，CPU时间延迟越多，超过了所分配的时间片，，三个进程字串序列的混杂程度越高。

1. 使用gcc编译pipeExample.c源文件。运行生成文件，查看结果并说明文件描述符赋值规则。

* 实验截图如下



* 分析如下：

文件描述符为整数，每个进程启动时会自动打开3个文件：Stdin、Stdout、Stderr，其对应的文件描述符分别为0、1、2，之后打开的文件会去找当前没有被使用的最小的一个整数作为新打开文件的文件描述符。

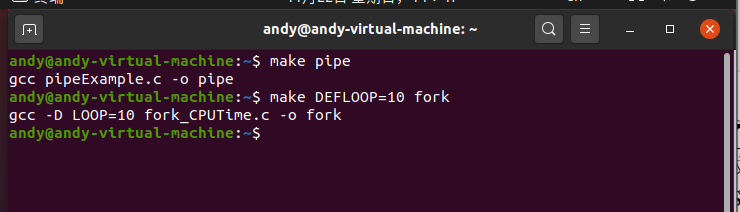
Pipe参数fd数组中fd[0]为管道读端，fd[1]为管道写端。

1. 编写一个makefile文件使得用户可以通过make选择编译不同的对象（如fork\_CPUTime.c或pipeExample.c）。

* Makefile内容如下



* 选择编译不同的对象



1. **实验心得及总结**

通过本次实验，我掌握系统调用的实现方式并能完成基本的文件读、写，进程管理操作。此外我还掌握GCC的使用方法，在第一个小实验中，我熟悉了gcc命令的使用方式，并了解了宏定义变量的传入，通过设置不同的LOOP值，令不同进程等待不同的时间，从而与CPU分配的时间片对比造成输出序列乱序。在第二个小实验中，我了解 了文件描述符和管道的使用。第三个小实验中，通过编辑makefile文件来管理程序文件的编译管理，我掌握了MAKEFILE的编写规则，并能使用make完成大型程序的编译、管理。