**实验1　Linux下进程管理**

**一、实验目的**

1.掌握vim编译器

2.掌握gcc编译器的使用

3.了解fork()程序

**二、实验工具与设备**

1．实验设备：计算机（带CD-ROM）一台。

**三、实验预备知识**

**1. vim的编辑器**

使用语法：Vim <被编辑的文件名>

Vim有三种模式：命令模式、插入模式、最后行模式。

命令模式：

刚启动Vim后，就处于该模式。在此模式下，允许用Vim的子命

令来编辑文件或转移到其它模式。如：

x命令：删除光标上面的字符。

方向键：移动光标。

i命令：进入插入模式，可在当前光标处插入字符。

a命令：进入插入模式，可在当前光标后插入字符。

R命令：从当前光标处开始替换文本。

r命令：替换当前光标处字符。

~命令：对当前光标处字母进行大小写转换。

dd命令：删除光标所在行。

dw命令：删除光标所在处字。

h命令：光标左移。

l命令：光标右移。

k命令：光标上移。

j命令：光标下移。

：命令：进入最后一行模式。

插入模式：

此模式下允许输入文本，用回车键换行，Esc 进入命令模式。

最后一行模式：

w命令：将文件存盘，但不退出。

wq命令：将文件存盘，并退出。

q! 命令：不保存文件并退出。

r命令：将另一个文件内容插入当前光标处。

**2.gcc编译器**

Unix 上使用的C 语言编译器cc，在Linux上的派生就是gcc。在使用vim编写完源程

序之后，返回到shell下面，使用gcc对源程序进行编译的命令是：

gcc 源程序

其中，“源程序”即为你编写的以.c 为扩展名的C 语言源代码文件。

如果源代码没有语法错误，使用以上命令编译，会在当前目录下生成一个名为a.out

的可执行文件。如果源代码有语法错误，则不会生成任何文件，gcc编译器会在shell中

提示你错误的地点和类型。

也可以使用以下方法编译源代码文件，生成自命名的可执行文件：

gcc 源文件–o 自命名的文件名

执行当前目录下的编译生成的可执行文件，使用以下格式：

./可执行文件名

当使用gcc编译你写的程序源代码的时候，可能会因为源代码存在语法错误，编译

无法进行下去，这时候，就可以使用调试器gdb来对程序进行调试。

**3. gdb 简介**

Linux 包含了一个叫gdb 的GNU 调试程序。gdb 是一个用来调试C 和C++ 程序

的强力调试器。它使你能在程序运行时观察程序的内部结构和内存的使用情况。以下是

gdb 所提供的一些功能：

能监视你程序中变量的值。

能设置断点以使程序在指定的代码行上停止执行。

能一行行的执行你的代码。

当启动gdb 后，能在命令行上指定很多的选项。你也可以以下面的方式来运行 gdb：

gdb <fname>

当你用这种方式运行gdb ，就能直接指定想要调试的程序。这将告诉gdb 装入名为

fname 的可执行文件。你也可以用gdb 去检查一个因程序异常终止而产生的core 文

件，或者与一个正在运行的程序相连。你可以参考gdb 指南页或在命令行上键入gdb

-h 得到一个有关这些选项的说明的简单列表。

为调试编译代码(Compiling Code for Debugging)。为了使gdb 正常工作, 你必须

使你的程序在编译时包含调试信息。调试信息包含你程序里的每个变量的类型和在可执

行文件里的地址映射以及源代码的行号。gdb 利用这些信息使源代码和机器码相关联。

在编译时用-g 选项打开调试选项。

**4.fork() 函数说明**

pid\_t fork(void)

fork()会产生一个新的子进程。该函数包含于头文件unistd.h 中。其子进程会复制

父进程的数据与堆栈空间，并继承父进程的用户代码、组代码、环境变量、已打开的文

件代码、工作目录和资源限制等。Linux 使用copy-on-write(COW)技术，只有当其中一

进程试图修改欲复制的空间时才会做真正的复制动作，由于这些继承的信息是复制而

来，并非指相同的内存空间，因此子进程对这些变量的修改和父进程并不会同步。此

外，子进程不会继承父进程的文件锁定和未处理的信号。注意，Linux不保证子进程会比

父进程先执行或晚执行，因此编写程序时要留意死锁或竞争条件的发生。

返回值，如果fork()调用成功则在父进程会返回新建立的子进程代码(PID)，而在新

建立的子进程中则返回0。如果fork() 失败则直接返回-1，失败原因存于errno中。失

败的原因有三个：

1) 系统内存不够；

2) 进程表满（容量一般为200~400）；

3) 用户的子进程太多（一般不超过25个）。

错误代码：EAGAIN 内存不足；ENOMEM 内存不足，无法配置核心所需的数据结构空

间。

**四、实验内容和步骤**

**实验一：**

在Linux环境下，用c语言编程，使用系统调用fork创建进程多个子进程。

（1）在终端里输入vim test.c,启动vi.

（2）按a或者i进入插入模式，在里面输入以下代码。

#include <unistd.h>

#include <stdio.h>

int main ()

{

pid\_t fpid; //fpid表示fork函数返回的值

fpid=fork();

printf(" you print me\n");

return 0;

}

（3）按Esc键进入命令模式，输入：和wq 进行保存退出vim.

（4）用gcc编译器编译该程序： gcc –o test test.c

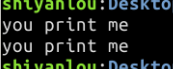
（5）运行刚刚编译的程序：./test

或者进行调式运行

用gcc编译器编译该程序： gcc –o test test.c –ggdb

调试刚刚编译的程序：gdb ./test

要求：



1. 请说出执行这个程序后，将一共运行几个进程。
2. 观察运行结果，并给出分析与解释。

**实验二：**

#include <unistd.h>

#include <stdio.h>

int main ()

{

pid\_t fpid; //fpid表示fork函数返回的值

int count=0;

fpid=fork();

if (fpid < 0)

printf("error in fork!");

else if (fpid == 0) {

printf("i am the child process, my process id is %d\n",getpid());

count++;

}

else {

printf("i am the parent process, my process id is %d\n",getpid());

count++;

}

printf("统计结果是: %d\n",count);

return 0;

}

要求：

观察运行结果，并给出分析与解释

**实验三：**

#include <unistd.h>

#include <stdio.h>

int main ()

{

pid\_t pid1;

pid\_t pid2;

pid1=fork();

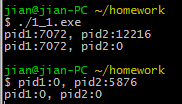
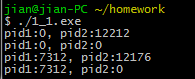
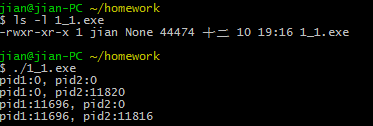
pid2=fork();

printf("pid1:%d,pid2:%d\n",pid1,pid2);

return 0;

}

这个程序运行后，一共将运行4个进程。这4个进程并没有严格的区分先后顺序。



**实验四：**

首先分析一下代码运行时其输出结果有哪几种可能性，按照实验1步骤编译调试观察其实际输出情况，比较两者的差异，分析其中的原因。

#include <unistd.h>

#include <stdio.h>

void main ()

{ int x=5;

if( fork( ) )

{

x+=30;

printf ("%d\n",x);

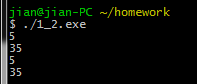
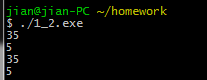
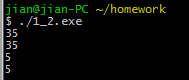
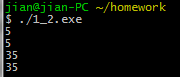
}

else

printf("%d\n",x);

printf("%d\n",x);

}（写出可能的实验结果及其分析说明）



**实验五：**

#include <unistd.h>

#include <stdio.h>

int main()

{ int size=1;

int i=0;

printf("i son/pa ppid pid fpid/n");

//ppid指当前进程的父进程pid

//pid指当前进程的pid,

//fpid指fork返回给当前进程的值

for(i=0;i<size;i++){

pid\_t fpid=fork();

if(fpid==0)

printf("%d child %4d %4d %4d/n",i,getppid(),getpid(),fpid);

else

printf("%d parent %4d %4d %4d/n",i,getppid(),getpid(),fpid);

}

return 0;

}

要求：

1、观察运行结果，并给出分析与解释。

2、如果size=2的时候会有什么结果，为什么。

**五、实验总结**

1．写出实验报告。

2. 利用vim对linux文本文件进行编辑。

3.编写fork()程序，并利用gcc和gdb进行编译、调试和运行。