**多进程编程**

**组名：The three stooges**

**小组成员：曾毓宁、徐锦畅、唐培津**

**一、实验目的**

**1、学会创建多进程程序，理解fork()，vfork()创建的子进程的特点。**

**2、学会使用有名管道和无名管道实现进程间通信。**

**3、学会使用信号机制实现进程间通信。**

**二、实验内容**

**1、编写fork()，vfork()多进程程序，通过输出结果比较所创建的子进程的不同之处。**

**2、编写fork()多进程程序，进程间模拟信号通信，包括信号发送、捕捉与处理。**

**3、编写fork()多进程程序，进程间实现有名管道和无名管道通信。具体要求如下： **

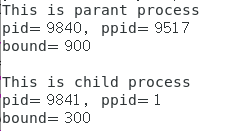
**1.编写1个服务器程序server.c和2个客户端程序client1.c,**

**client2.c ，服务器和客户端之间采用FIFO管道通信； **

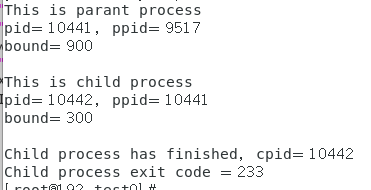
**2.服务器和客户端各自创建自己的子进程，父进程负责完成FIFO管道通信，子进程负责输出接收到的信息，父子进程间采用无名管道通信。**

**三、实验过程与结果**

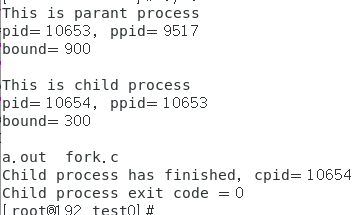
**1、编写一个fork（）多进程程序。该程序在父进程中没有调用wait（）或waitpid（），结果如下所示。可以看到子进程的ppid为1。出现这个现象的原因是父进程先于子进程结束，子进程变成孤儿进程，被init收养。**



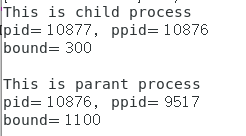
**加入wait()后，子进程ppid为其父进程pid。这里还加入了WIFEXITED（status）和WEXITSTATUS(status)宏来检查子进程的状态。**



**最后，我尝试加入exec()。Exec()和system()的作用类似，都是进入一个新进程，它们的区别是system()会返回调用进程，而exec()直接覆盖了原有进程。加入后结果如下所示，可以看到此时子进程的返回数已不是233，而是默认的0。说明exec()后的exit(233)没有执行。**



1. **编写一个vfork程序。结果如下。Vfork中父子进程共享内存，且子进程优先运行，所以父进程运行后bound结果为1100。**



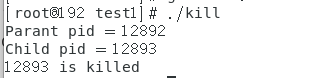
**至此，我们可以比较fork和vfork的异同。**

**相同之处是，fork和vfork在调用执行中会返回3个值（已算上错误信息），且在不调用exec()的情况下，父子进程均共享代码段。**

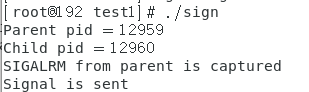
**不同之处是，fork创建子进程时会复制父进程全部资源，但不与父进程共享内存，并且父进程可以在子进程结束前结束；vfork创建子进程时只是复制了父进程的指针，父子进程共享内存，和线程区别不大，并且父进程会在子进程运行结束后运行。**

**为了数据安全，我们在工作中应谨慎使用vfork。**

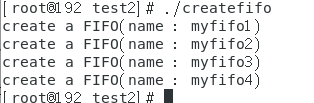
1. **编写fork()多进程程序，模拟信号通信。这里我先编写了一个使用kill()发送信号的程序，结果如下。在子进程运行之前父进程已经执行删除子进程的操作，因此不会执行子进程（这里父子进程执行顺序不固定，由系统资源统一分配）。在父进程中加入sleep(1)后，确认kill正常发送函数。**



**随后，编写含有信号发送、捕获与处理的fork程序。结果如下。SIGALRM信号发送、捕获、处理正常。**

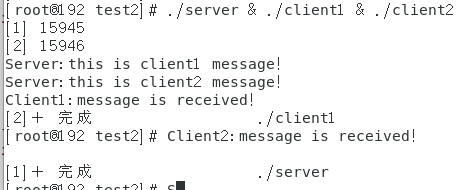


1. **首先，编写createfifo程序，创建了四个fifo管道myfifo1、myfifo2、myfifo3、myfifo4。编译运行后生成四个fifo管道。其中myfifo1、myfifo2为server向两个client发送信号的管道，myfifo3、myfifo4为两个client向server发送确认收到信号的管道。**



**接着，编写服务器程序以及两个客户端程序。两个客户端client1和client2中调用fork（）创造子进程。在父进程中完成fifo管道通信，父子进程用一个无名管道通信，父进程把从fifo收到的信号通过无名管道送给子进程，在子进程中输出收到的信号。**

**在服务器程序server中，同样创建子进程。在这两个进程之间，我创建了两个无名管道，分别把server从client1和client2中收到的确认信号通过两无名管道送给子程序，在子程序中输出收到的信号。结果如下。**



1. **实验总结**

**本次实验中，我们熟悉了多进程编程，为以后进行复杂的编程工程奠定了良好的基础。**

1. **附实验源码**

**Fork.c:**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h>**

**#include<wait.h>**

**int main()**

**{**

**pid\_t pid, cpid;**

**int status,bound = 100;**

**if((pid=fork())==-1){**

**perror("fork");**

**exit(EXIT\_FAILURE);**

**}else if(pid==0){**

**puts("This is child process");**

**bound +=200;**

**printf("pid= %d, ppid= %d\n",getpid(),getppid());**

**printf("bound= %d\n\n",bound);**

**execlp("ls","ls",NULL);**

**exit(233);**

**}else{**

**puts("This is parant process");**

**bound +=800;**

**printf("pid= %d, ppid= %d\n",getpid(),getppid());**

**printf("bound= %d\n\n",bound);**

**cpid = wait(&status);**

**printf("Child process has finished, cpid= %d\n",cpid);**

**if(WIFEXITED(status)) printf("Child process exit code = %d\n",WEXITSTATUS(status));**

**exit(EXIT\_SUCCESS);**

**}**

**}**

**Vfork.c:**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h>**

**int main()**

**{**

**pid\_t pid;**

**int bound = 100;**

**if((pid=vfork())==-1){**

**perror("vfork");**

**exit(EXIT\_FAILURE);**

**}else if(pid==0){**

**puts("This is child process");**

**bound +=200;**

**printf("pid= %d, ppid= %d\n",getpid(),getppid());**

**printf("bound= %d\n\n",bound);**

**exit(233);**

**}else{**

**puts("This is parant process");**

**bound +=800;**

**printf("pid= %d, ppid= %d\n",getpid(),getppid());**

**printf("bound= %d\n\n",bound);**

**exit(EXIT\_SUCCESS);**

**}**

**}**

**Kill.c:**

**#include<stdio.h>**

**#include<stdlib.h>**

**//#include<unistd.h>**

**#include<signal.h>**

**#include<wait.h>**

**int main()**

**{**

**pid\_t pid;**

**int flag;**

**pid = fork();**

**if(pid==-1){**

**perror("fork");**

**exit(EXIT\_FAILURE);**

**}else if(pid==0){**

**printf("Child pid = %d\n",getpid());**

**raise(SIGSTOP);**

**exit(EXIT\_SUCCESS);**

**}else{**

**printf("Parant pid = %d\n",getpid());**

**sleep(1);**

**flag = kill(pid,SIGKILL);**

**if(flag==0){**

**printf("%d is killed\n",pid);**

**}else{**

**perror("kill error");**

**exit(1);**

**}**

**exit(EXIT\_SUCCESS);**

**}**

**}**

**Sign.c:**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h>**

**#include<signal.h>**

**#include<wait.h>**

**void handler(int signum)**

**{**

**if(signum==SIGALRM) printf("SIGALRM from parent is captured\n");**

**}**

**int main()**

**{**

**pid\_t pid,cpid;**

**int flag,status;**

**if((pid=fork())==-1){**

**perror("fork");**

**exit(EXIT\_FAILURE);**

**}else if(pid==0){**

**printf("Child pid = %d\n",getpid());**

**signal(SIGALRM,handler);**

**sleep(1);**

**exit(EXIT\_SUCCESS);**

**}else{**

**printf("Parent pid = %d\n",getpid());**

**sleep(1);**

**flag = kill(pid,SIGALRM);**

**cpid=wait(&status);**

**if(flag==0){**

**printf("Signal is sent\n",pid);**

**}else{**

**perror("kill error");**

**exit(1);**

**}**

**exit(EXIT\_SUCCESS);**

**}**

**}**

**Createfifo.c:**

**#include <sys/types.h>**

**#include <sys/stat.h>**

**#include <errno.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**int main()**

**{**

**char \*pathname1 = "myfifo1";**

**char \*pathname2 = "myfifo2";**

**char \*pathname3 = "myfifo3";**

**char \*pathname4 = "myfifo4";**

**if (mkfifo(pathname1, 0666) < 0)**

**{**

**perror("mkfifo error\n");**

**exit(1);**

**}**

**else**

**{**

**printf("create a FIFO(name : %s)\n", pathname1);**

**}**

**if (mkfifo(pathname2, 0666) < 0)**

**{**

**perror("mkfifo error\n");**

**exit(1);**

**}**

**else**

**{**

**printf("create a FIFO(name : %s)\n", pathname2);**

**}**

**if (mkfifo(pathname3, 0666) < 0)**

**{**

**perror("mkfifo error\n");**

**exit(1);**

**}**

**else**

**{**

**printf("create a FIFO(name : %s)\n", pathname3);**

**}**

**if (mkfifo(pathname4, 0666) < 0)**

**{**

**perror("mkfifo error\n");**

**exit(1);**

**}**

**else**

**{**

**printf("create a FIFO(name : %s)\n", pathname4);**

**}**

**}**

**Server.c:**

**#include <sys/types.h>**

**#include <sys/stat.h>**

**#include <unistd.h>**

**#include <fcntl.h>**

**#include <stdlib.h>**

**#include <stdio.h>**

**#include <errno.h>**

**#include <string.h>**

**#define BUF\_SIZE 256**

**int main()**

**{**

**char buf1[BUF\_SIZE],buf2[BUF\_SIZE],buf3[BUF\_SIZE],buf4[BUF\_SIZE],buff1[BUF\_SIZE],buff2[BUF\_SIZE];**

**int fd1,fd2,fd3,fd4;**

**int len1,len2,len3,len4;**

**pid\_t pid;**

**int pfd[2],dty[2];**

**if(pipe(pfd)<0){**

**perror("pipe error");**

**exit(1);**

**}**

**if(pipe(dty)<0){**

**perror("pipe error");**

**exit(1);**

**}**

**pid = fork();**

**if(pid<0){**

**perror("fork error");**

**exit(1);**

**}else if(pid ==0){**

**bzero(buff1, BUF\_SIZE);**

**bzero(buff2, BUF\_SIZE);**

**close(pfd[1]);**

**sleep(1);**

**read(pfd[0],buff1,BUF\_SIZE);**

**printf("%s\n",buff1);**

**close(pfd[0]);**

**close(dty[1]);**

**sleep(1);**

**read(dty[0],buff2,BUF\_SIZE);**

**printf("%s\n",buff2);**

**close(dty[0]);**

**exit(0);**

**}else{**

**close(pfd[0]);**

**fd1 = open("myfifo1", O\_WRONLY);**

**if (fd1 < 0)**

**{**

**perror("open myfifo1 error!");**

**exit(1);**

**}**

**fd2 = open("myfifo2", O\_WRONLY);**

**if (fd2 < 0)**

**{**

**perror("open myfifo2 error!");**

**exit(1);**

**}**

**bzero(buf1, BUF\_SIZE);**

**len1 = sprintf(buf1, "Server:this is client1 message!");**

**if (write(fd1, buf1, len1) < 0)**

**{**

**perror("write fifo1 error!");**

**close(fd1);**

**exit(1);**

**}**

**sleep(2);// 休眠2秒便于观察**

**bzero(buf2, BUF\_SIZE);**

**len2 = sprintf(buf2, "Server:this is client2 message!");**

**if (write(fd2, buf2, len2) < 0)**

**{**

**perror("write fifo2 error!");**

**close(fd2);**

**exit(1);**

**}**

**sleep(2);// 休眠2秒便于观察**

**close(fd1);**

**close(fd2);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8**

**fd3 = open("myfifo3", O\_RDONLY);**

**if (fd3 < 0)**

**{**

**perror("open error!");**

**exit(1);**

**}**

**bzero(buf3, BUF\_SIZE);**

**while ((len3 = read(fd3, buf3, BUF\_SIZE)) > 0)**

**{**

**write(pfd[1],buf3,strlen(buf3));**

**}**

**close(fd3);**

**close(pfd[1]);**

**close(dty[0]);**

**fd4 = open("myfifo4", O\_RDONLY);**

**if (fd4 < 0)**

**{**

**perror("open error!");**

**exit(1);**

**}**

**bzero(buf4, BUF\_SIZE);**

**while ((len4 = read(fd4, buf4, BUF\_SIZE)) > 0)**

**{**

**write(dty[1],buf4,strlen(buf4));**

**}**

**close(fd4);**

**close(dty[1]);**

**waitpid(pid,NULL,0);**

**exit(0);**

**}**

**}**

**Client1.c:**

**#include <sys/types.h>**

**#include <sys/stat.h>**

**#include <unistd.h>**

**#include <fcntl.h>**

**#include <errno.h>**

**#include <string.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define BUF\_SIZE 512**

**int main()**

**{**

**char buf1[BUF\_SIZE],buf2[BUF\_SIZE],buff[BUF\_SIZE];**

**int fd1,fd2;**

**int len1,len2;**

**pid\_t pid;**

**int pfd[2];**

**if(pipe(pfd)<0){**

**perror("pipe error");**

**exit(1);**

**}**

**pid = fork();**

**if(pid<0){**

**perror("fork error");**

**exit(1);**

**}else if(pid ==0){**

**close(pfd[1]);**

**sleep(1);**

**read(pfd[0],buff,BUF\_SIZE);**

**printf("%s\n",buff);**

**close(pfd[0]);**

**exit(0);**

**}else{**

**close(pfd[0]);**

**fd1 = open("myfifo1", O\_RDONLY);**

**if (fd1 < 0)**

**{**

**perror("open error!");**

**exit(1);**

**}**

**bzero(buf1, BUF\_SIZE);**

**while ((len1 = read(fd1, buf1, BUF\_SIZE)) > 0)**

**{**

**write(pfd[1],buf1,strlen(buf1));**

**}**

**close(fd1);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**fd2 = open("myfifo3", O\_WRONLY);**

**if (fd2 < 0)**

**{**

**perror("open myfifo3 error!");**

**exit(1);**

**}**

**bzero(buf2, BUF\_SIZE);**

**len2 = sprintf(buf2, "Client1:message is received!");**

**if (write(fd2, buf2, len2) < 0)**

**{**

**perror("write fifo3 error!");**

**close(fd2);**

**exit(1);**

**}**

**close(fd2);**

**close(pfd[1]);**

**waitpid(pid,NULL,0);**

**exit(0);**

**}**

**}**

**Client2.c:**

**#include <sys/types.h>**

**#include <sys/stat.h>**

**#include <unistd.h>**

**#include <fcntl.h>**

**#include <errno.h>**

**#include <string.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define BUF\_SIZE 512**

**int main()**

**{**

**char buf1[BUF\_SIZE],buf2[BUF\_SIZE],buff[BUF\_SIZE];**

**int fd1,fd2;**

**int len1,len2;**

**pid\_t pid;**

**int pfd[2];**

**if(pipe(pfd)<0){**

**perror("pipe error");**

**exit(1);**

**}**

**pid = fork();**

**if(pid<0){**

**perror("fork error");**

**exit(1);**

**}else if(pid ==0){**

**close(pfd[1]);**

**sleep(1);**

**read(pfd[0],buff,BUF\_SIZE);**

**printf("%s\n",buff);**

**close(pfd[0]);**

**exit(0);**

**}else{**

**close(pfd[0]);**

**fd1 = open("myfifo2", O\_RDONLY);**

**if (fd1 < 0)**

**{**

**perror("open error!");**

**exit(1);**

**}**

**bzero(buf1, BUF\_SIZE);**

**while ((len1 = read(fd1, buf1, BUF\_SIZE)) > 0)**

**{**

**write(pfd[1],buf1,strlen(buf1));**

**}**

**close(fd1);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**fd2 = open("myfifo4", O\_WRONLY);**

**if (fd2 < 0)**

**{**

**perror("open myfifo4 error!");**

**exit(1);**

**}**

**bzero(buf2, BUF\_SIZE);**

**len2 = sprintf(buf2, "Client2:message is received!");**

**if (write(fd2, buf2, len2) < 0)**

**{**

**perror("write fifo4 error!");**

**close(fd2);**

**exit(1);**

**}**

**close(fd2);**

**close(pfd[1]);**

**waitpid(pid,NULL,0);**

**exit(0);**

**}**

**}**