## 实验五：管道通信

1．阅读以下程序：

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include "apue.h"

main()

{

int filedes[2];

char buffer[80];

if (pipe(filedes) < 0)

err\_quit("pipe error");

if (fork() > 0)

{

char s[] = "hello!\n";

close(filedes[0]);

write(filedes[1], s, sizeof(s));

close(filedes[1]);

}

else

{

close(filedes[1]);

read(filedes[0], buffer, 80);

printf("%s", buffer);

close(filedes[0]);

}

}

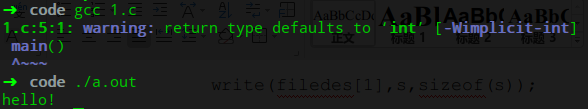
编译并运行程序，分析程序执行过程和结果，注释程序主要语句。

答：

程序代码注释如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #include <unistd.h>  #include <stdio.h>  #include <stdlib.h>  #include "apue.h"  main()  {  int filedes[2]; //filedes[0]为管道里的读取端，filedes[1]为管道的写入端  char buffer[80];  if (pipe(filedes) < 0) //成功返回0，失败返回-1，错误原因存于errno中  err\_quit("pipe error");  if (fork() > 0) //父进程  {  char s[] = "hello!\n";  close(filedes[0]); //关闭父进程读取端  write(filedes[1], s, sizeof(s)); //写入数据  close(filedes[1]); //关闭父进程写入端  }  else //子进程  {  close(filedes[1]); //关闭子进程写入端  read(filedes[0], buffer, 80); //读取数据  printf("%s", buffer); //输出读取的内容  close(filedes[0]); //关闭子进程读取端  }  } |

这段代码代码使用无名管道，通过从父进程中写入字符串数据”hello!\n”，并在子进程中在屏幕上打印接收到的数据，用于测试管道通信，其执行效果如下：



2．阅读以下程序：

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

main()

{

char buffer[80];

int fd;

unlink("FIFO");

mkfifo("FIFO", 0666); //FIFO是管道名，0666是权限

if (fork() > 0)

{

char s[] = "hello !\n";

fd = open("FIFO", O\_WRONLY);

write(fd, s, sizeof(s));

close(fd);

}

else

{

fd = open("FIFO", O\_RDONLY);

read(fd, buffer, 80);

printf("%s", buffer);

close(fd);

}

}

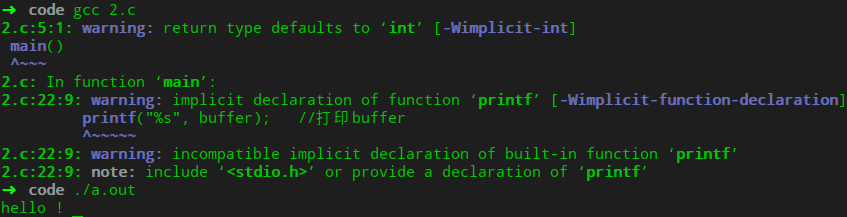
编译并运行程序，分析程序执行过程和结果，注释程序主要语句。

答：

程序代码注释如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #include <sys/types.h>  #include <sys/stat.h>  #include <fcntl.h>  #include <unistd.h>  main()  {  char buffer[80];  int fd;  unlink("FIFO"); //如果已经存在FIFO文件则先删除  mkfifo("FIFO", 0666); //创建权限为0666的名称为FIFO的命名管道  if (fork() > 0) //父进程  {  char s[] = "hello !\n";  fd = open("FIFO", O\_WRONLY); //以只写方式打开FIFO命名管道  write(fd, s, sizeof(s)); //写入数据s  close(fd); //关闭命名管道  }  else //子进程  {  fd = open("FIFO", O\_RDONLY); //以只读方式打开FIFO命名管道  read(fd, buffer, 80); //读出数据存入buffer  printf("%s", buffer); //打印buffer  close(fd); //关闭命名管道  }  } |

这段代码代码使用有名管道，创建了一个名为FIFO的命名管道，通过从父进程中写入字符串数据”hello !\n”，并在子进程中在屏幕上打印接收到的数据，用于测试管道通信，其执行效果如下：



3．阅读以下程序：

#include<stdio.h>

main()

{

FILE \* fp;

char buffer[80];

fp=popen("cat /etc/passwd","r");

fgets(buffer,sizeof(buffer),fp);

printf("%s",buffer);

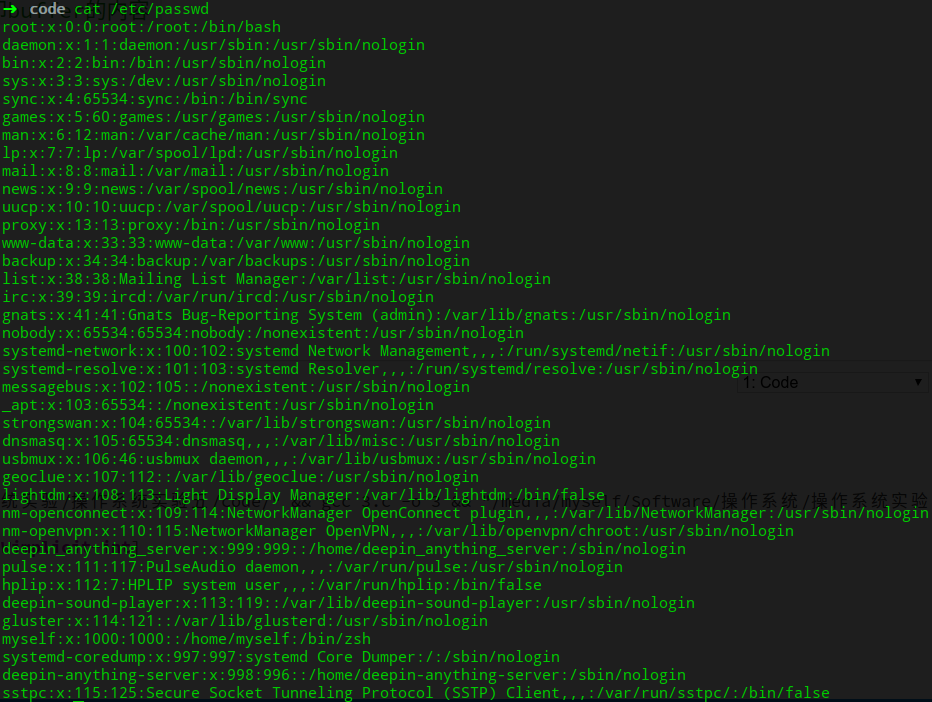
pclose(fp);

}

编译并运行程序，分析程序执行过程和结果，注释程序主要语句。

答：

Shell命令cat /etc/passwd的输出结果如下：

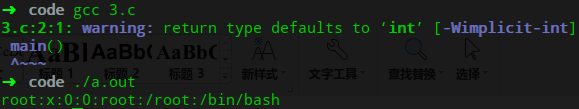


程序代码注释如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | #include<stdio.h>  main()  {  FILE \* fp;  char buffer[80];  fp=popen("cat /etc/passwd","r"); //子进程执行cat /etc/passwd，建立管道I/O连接到子进程标准输出设备  fgets(buffer,sizeof(buffer),fp); //读取一行内容  printf("%s",buffer); //打印buffer的内容  pclose(fp); //关闭管道I/O  } |

这段程序使用popen建立管道I/O连接到执行cat /etc/passwd命令的子进程的标准输出设备，并使用fgets函数从中读取一行内容打印在屏幕上。

程序的输出结果如下：



4．编写一个程序，读取一个数据文件，对每一个数据进行某种运算，再在屏幕输出计算结果。要求以上工作用两个进程实现，父进程负责读文件和显示，子进程进行计算，进程间通信使用无名管道。（使用系统调用）

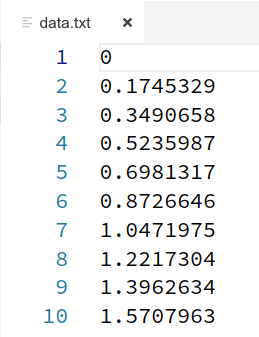
答：

程序代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | #include <stdio.h>  #include <unistd.h>  #include <math.h>  #include <sys/wait.h>  #include "apue.h"  int main(void)  {  int fda[2], fdb[2];  int pid, i;  if(pipe(fda) < 0 || pipe(fdb) < 0) //建立无名管道  err\_quit("pipe error");  while((pid = fork()) == -1) ; //创建子进程    if(pid > 0) //父进程  {  FILE \*fpread = fopen("data.txt", "r"); //打开文件data.txt  float data, result;  if(fpread == NULL) //打开文件失败  {  perror("open file error!\n");  exit(1);  }  close(fda[0]); //关闭父进程管道a读取端  close(fdb[1]); //关闭父进程管道b写入端  for(i = 0; i < 10; i++)  {  fscanf(fpread, "%f", &data); //读入一个数据存入data  write(fda[1], &data, sizeof(data)); //data通过管道a发送给子进程  read(fdb[0], &result, sizeof(result)); //通过管道b读计算结果到result  printf("sin(%f)=%f\n", data, result);  }  close(fdb[0]); //关闭父进程管道b读取端  close(fda[1]); //关闭父进程管道a写入端  fclose(fpread); //关闭文件data.txt  }  else //子进程  {  float data, result;  close(fda[1]); //关闭子进程管道a写入端  close(fdb[0]); //关闭子进程管道b读取端  for(i = 0; i < 10; i++)  {  read(fda[0], &data, sizeof(data));  result = sinf(data);  write(fdb[1], &result, sizeof(result));  }  close(fdb[1]); //关闭子进程管道b写入端  close(fda[0]); //关闭子进程管道a读取端  }  return 0;  } |

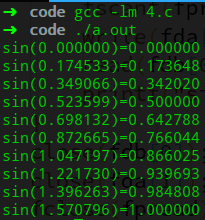
这段代码建立了2个无名管道，由于无名管道是半双工的，数据只能在一个方向传送，因此2个无名管道分别对应父进程写数据子进程读数据（fda）和子进程写数据父进程读数据（fdb）。

data.txt文件中存储了10条数据，如下图所示：



父进程和子进程中均循环了10次，每次循环父进程从文件中读取一行浮点型数据，使用fda发送给子进程，子进程接收到这个浮点数对其求正弦值并将计算结果发回给父进程并显示在屏幕上。

程序的输出结果如下：



5．编写两个程序，一个创建一个FIFO，并读管道，并显示在屏幕上，另一个每过一段时间向该管道写数据（进程PID）。运行多个写程序和一个读程序，观察运行结果。（使用系统调用）

答：

读程序的代码如下：

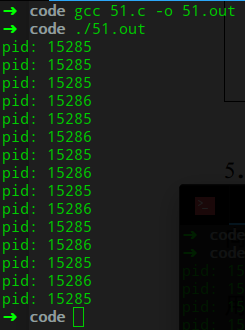
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #include <stdio.h>  #include <unistd.h>  #include <sys/stat.h>  #include <fcntl.h>  int main(void)  {  int fd, pid;  unlink("FIFO"); //如果已经存在FIFO文件则先删除  mkfifo("FIFO", 0666); //创建权限为0666的名称为FIFO的命名管道  fd = open("FIFO", O\_RDONLY); //以只读方式打开FIFO命名管道  while(read(fd, &pid, sizeof(pid)))  {  printf("pid: %d\n", pid);  }  close(fd);  return 0;  } |

写程序的代码如下：

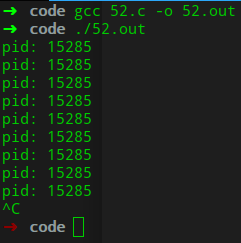
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | #include <stdio.h>  #include <unistd.h>  #include <fcntl.h>  int main(void)  {  int pid, fd;  pid = getpid(); //获得当前进程的pid  fd = open("FIFO", O\_WRONLY); //以只写方式打开FIFO命名管道  while(write(fd, &pid, sizeof(pid))) //持续写数据  {  printf("pid: %d\n", pid);  sleep(3); //等待3秒  }  close(fd);  return 0;  } |

读程序和写程序中都创建了一个名为FIFO的命名管道，读程序连续不断的从FIFO中读取数据并打印，而写程序则写入自己的pid，最终执行写程序一段时间后通过Ctrl+C停止两个写程序的执行，读程序也随之停止执行。读程序输出了两个写程序的PID，输出的数量与两个写程序写入的次数之和一致。

读程序的执行效果如下：



第一个写程序的执行效果如下：



第二个写程序的执行效果如下：

