实验内容

（1）学习 man 命令的用法，通过它查看管道创建、同步互斥系统调用的在线帮助，并阅读参考资料。

（2）根据流程图（如图 2.2 所示）和所给管道通信程序，按照注释里的要求把代码补充完整，运行程序，体会互斥锁的作用，比较有锁和无锁程序的运行结果，分析管道通信是如何实现同步与互斥的。

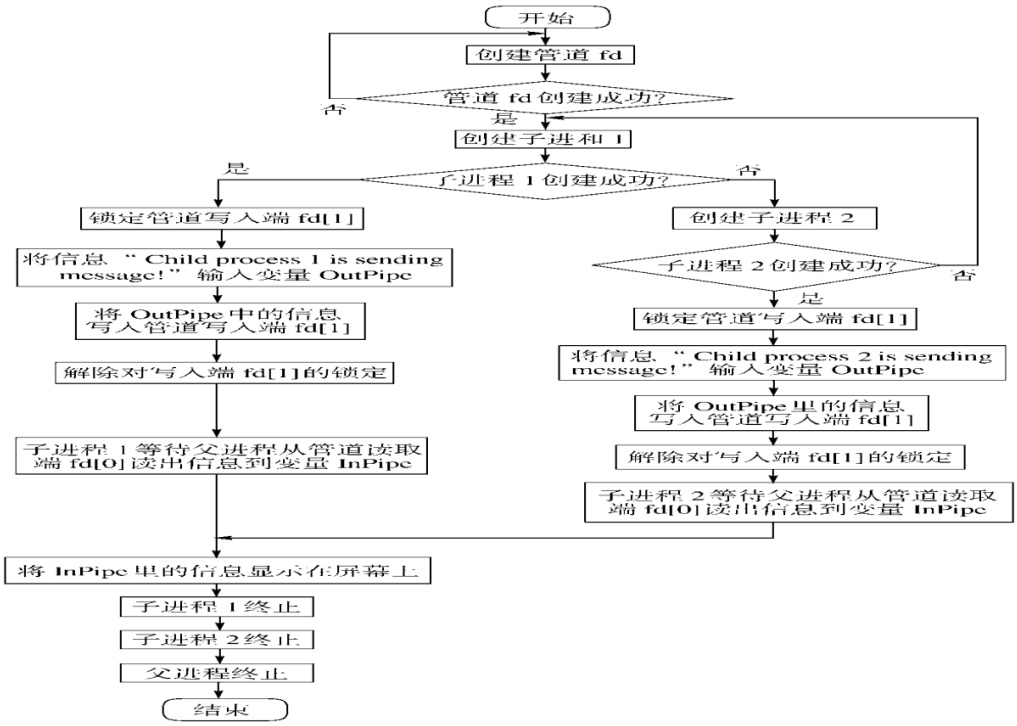


图 2.2 管道通信程序流程图

残缺代码：

```
/*管道通信实验程序残缺版 */
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
int pid1, pid2;  // 定义两个进程变量
```

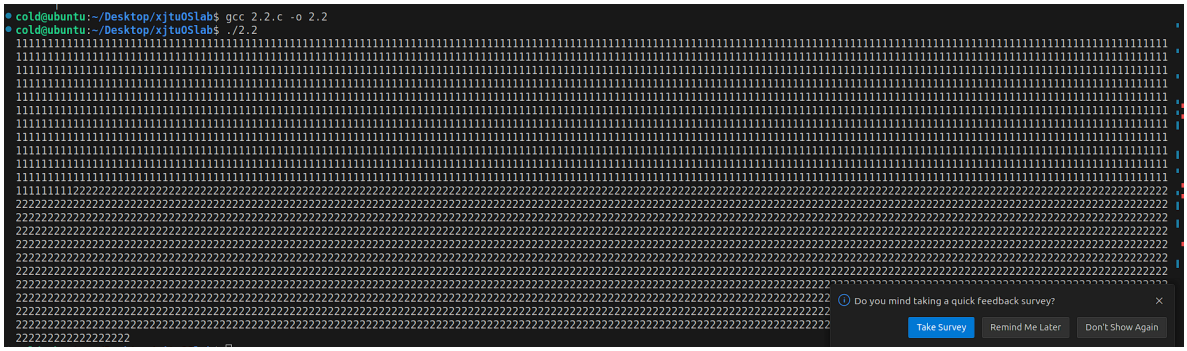
```
int main() {
  int fd[2];
  char InPipe[1000];  // 定义读缓冲区
  char c1 = '1', c2 = '2';
  pipe(fd);  // 创建管道
  while ((pid1 = fork()) == -1)
    ;  // 如果进程 1 创建不成功,则空循环
       // 如果子进程 1 创建成功,pid1 为进程号
  if (pid1 == 0) {
    // 补充:锁定管道
    // 补充:分 2000 次每次向管道写入字符'1'
    sleep(5);  // 等待读进程读出数据
    // 补充:解除管道的锁定
    exit(0);  // 结束进程 1
  } else {
    while ((pid2 = fork()) == -1)
      ;  // 若进程 2 创建不成功,则空循环
    if (pid2 == 0) {
      lockf(fd[1], 1, 0);
      // 补充:分 2000 次每次向管道写入字符'2'
      sleep(5);
      lockf(fd[1], 0, 0);
      exit(0);
    } else {
      // 补充:等待子进程 1 结束
      wait(0);  // 等待子进程 2 结束
      // 补充:从管道中读出 4000 个字符
      // 补充:加字符串结束符
      printf("%s\n", InPipe);  // 显示读出的
数据
```

```
      exit(0);                    // 父进程结束
    }
  }
```

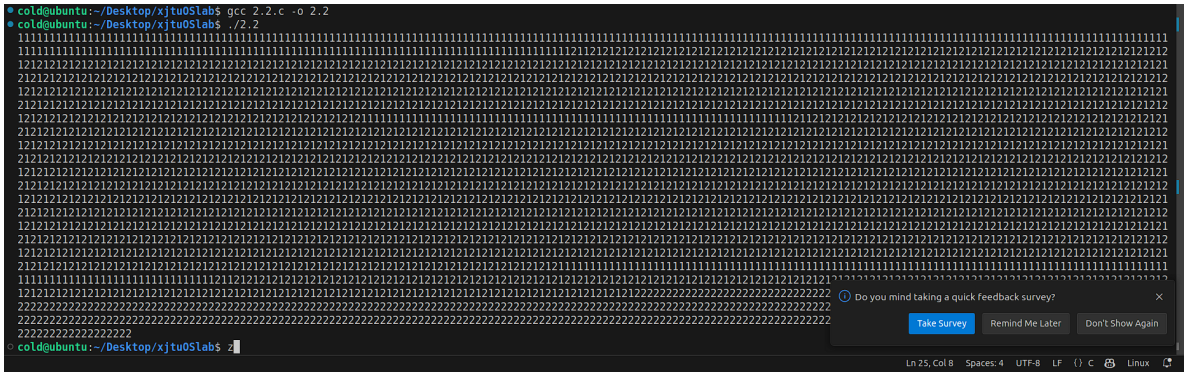
完善后的结果为：

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int pid1, pid2;

int main() {
  int fd[2];
  char InPipe[1000];
  char c1 = '1', c2 = '2';
  pipe(fd);

  while ((pid1 = fork()) == -1)
    ;  // 如果进程 1 创建不成功,则空循环
       // 如果子进程 1 创建成功,pid1 为进程号
  if (pid1 == 0) {
    lockf(fd[1], 1, 0);  // 锁定管道 0 para
means lock until the end
    for (int i = 0; i < 2000; i++) {
      write(fd[1], &c1, 1);
    }                     // 分2000 次每次向管
道写入字符'1'
```

```
    sleep(5);                  // 等待读进程读出数据
    lockf(fd[1], 0, 0);        // 解除管道的锁定
    exit(0);                   // 结束进程 1
  } else {
    while ((pid2 = fork()) == -1)
      ;  // 若进程 2 创建不成功,则空循环
    if (pid2 == 0) {
      lockf(fd[1], 1, 0);
      for (int i = 0; i < 2000; i++) {
        write(fd[1], &c2, 1);
      }  // 分2000 次每次向管道写入字符'2'
      sleep(5);
      lockf(fd[1], 0, 0);
      exit(0);
    } else {
      wait(NULL);                        // 等待子
进程 1 结束
      wait(0);                           // 等待子
进程 2 结束
      read(fd[0], InPipe, 4000);  // 从管道
中读出 4000 个字符
      InPipe[4000] = '\0';               // 加字符
串结束符
      printf("%s\n", InPipe);      // 显示读
出的数据
      exit(0);                           // 父进程
结束
    }
  }
}
```


cold@ubuntu:~/Desktop/xjtuOSlab$ gcc 2.2.c -o 2.2
cold@ubuntu:~/Desktop/xjtuOSlab$ ./2.2
Do you mind taking a quick feedback survey?
Take Survey
Remind Me Later
Don't Show Again

如果注释掉锁：


cold@ubuntu:~/Desktop/xjtuOSlab$ gcc 2.2.c -o 2.2
cold@ubuntu:~/Desktop/xjtuOSlab$ ./2.2
Do you mind taking a quick feedback survey?
Take Survey
Remind Me Later
Don't Show Again
cold@ubuntu:~/Desktop/xjtuOSlab$ z
Ln 25, Col 8
Spaces: 4
UTF-8
LF
C
Linux