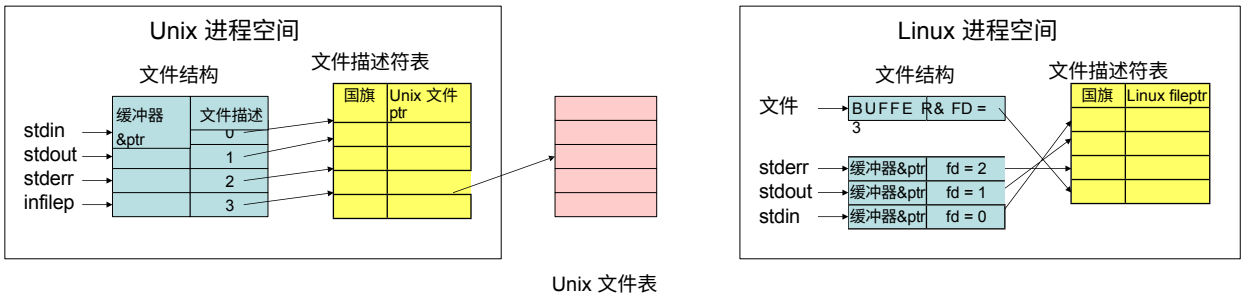
使用管道进行进程间通信	实验室	六
目标		
1. 文件和数据流 2. 通过管道进行通信		

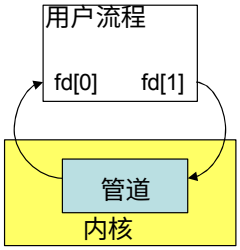
文件和数据流

在 Unix/Linux 中，几乎所有东西都被视为***文件，***包括*设备*、*管道*和*套接字*。回想一下，打开一个文件后，会返回一个 **`FILE*`** 类型的*文件句柄*，指向进程用户区中一个名为*文件结构的*数据结构。它包含一个*缓冲区*和一个*文件描述符*。文件由*文件句柄*或*文件描述符*标识。管道和套接字用于进程之间的通信，我们称之为*流*。这是因为，与程序可以在文件中跳转（但并非总是如此）的文件不同，程序只能从输入流中获取（读取）输入，并*按顺序*将输出（写入）到输出流中。因此，在 Unix/Linux 中，每个流都被视为一个*文件*，由*文件描述符通过文件句柄*间接标识。回想一下与每个用户进程相关联的文件结构和文件描述符表的结构，以及指向文件实际存储空间的共享 Unix/Linux 文件表。



通过管道进行通信

在 Unix/Linux 中，进程可以通过临时文件进行通信，这样一个进程可以向文件写入，另一个进程可以从文件读取。使用临时文件并不是一种干净的编程方式，因为其他进程和用户可能会看到临时文件的存在，并可能破坏它。更好的方法是使用*管道*进行通信。**管道**是 Unix/Linux 中最简单的*进程间通信*（IPC）机制，是一种*基于消息的直接通信机制*（参见第 3、4 和 5 讲）。*文件*由一个文件描述符表示，*管道*由***一对****文件描述符表示*；第一个*文件描述符*用于读取，第二个*文件描述符*用于写入。管道由系统调用 `pipe()` 创建。系统调用的参数是一个大小为 2 的整数*数组，*用于返回一对文件描述符。



如图所示，**pipe()** 系统调用将接受一个包含 2 个文件描述符（整数）的数组，并在内核空间创建一个数据结构。我们可以将管道结构视为一个队列。我们可以通过向管道写入元素来插入队列的末尾，也可以通过从管道读取元素来移除队列前端的元素。下面的 **lab6A.c** 程序创建了一个管道，其标识为

fd[0] 和 **fd[1]**。顾名思义，**fd[0]** 用于读取（**stdin** 指的是数据流 0），fd**[1]** 用于写入（**stdout 指的是**数据流 1）。进程将从键盘读取的数据写入管道，然后从管道中读取使用另一个变量写入的信息。键盘输入由 **<Ctrl-D>** 终止。请注意，如果在程序中输入*空行，会*有一个*隐藏的错误*。你能*调试*一下吗?

```
// 实验室 6A
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main()
{
  intfd [2]; // 用于管道
  char buf[80]、buf2[80];
  n     n;

  if (pipe(fd) < 0) {
     printf("Pipe creation error\n");
     exit(1);
  }
  // 重复循环，向管道中写入内容，并从同一管道中读取内容 while (1) {
       printf("Please input a line\n");
       n = read(STDIN_FILENO, buf, 80); // 从 stdin 读一行 if (n <=
       0) break; // EOF 或错误
       buf[--n] = 0; // 删除换行符 printf("%d char in input
       line: [%s]\n", n, buf);
       write(fd[1], buf, n); // write to pipe
       printf("Input line [%s] written to pipe\n",buf);
       n = read (fd[0], buf2, 80) ; // 从管道读取数据
       buf2[n] = 0;
       printf("%d char read from pipe: [%s]\n", n, buf2);
  }
  printf("bye bye\n");
  close(fd[0]);
  close(fd[1]);
  exit(0);
}
```

请注意，Unix/Linux 中的管道就像*水管*。不管是倒入两小杯水还是倒入一大杯水，只要容量相同，就没有区别。在 **lab6B.c** 中试试看。在这里，你将输入两行并写入管道。然后从管道中读出单行。您将看到两行合并为一条信息。现在，尝试输入两行长信息，然后再输入一些短信息。你观察到了什么？您能得出什么结论?

您是否发现程序中没有收到*最后一行奇数行*（如果有的话）？在程序结束前*清理*管道是一种好的做法。您可以尝试修改程序，以便在程序结束前收到最后一行奇数行。

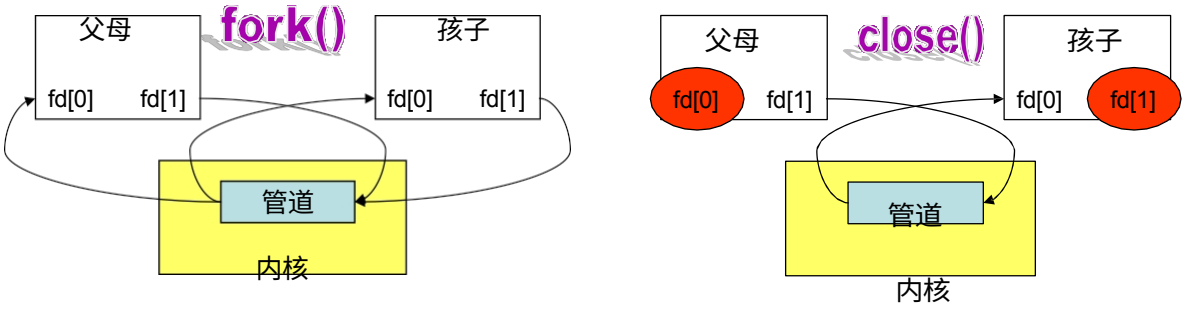
```
//  实 验 室    6B
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main()
{
  intfd [2]; // 管道 char
  buf[80], buf2[80];
  inteven      , n;
  if (pipe(fd) < 0) {
```

```
    printf("Pipe creation error\n");
    exit(1);
}
偶数 = 1;
// 重复循环，向管道中写入内容，并从同一管道中读取内容 while (1) {
    even = 1 - even; // 切换偶数变量
```

```
        如果
           printf("Please input an even line\n");
        else
           printf("Please input an odd line\n");
        n = read(STDIN_FILENO, buf, 80); // 从 stdin 读一行 if (n <=
        0) break; // EOF 或错误
        buf[--n] = 0; // 删除换行符 printf("%d char in input
        line: [%s]\n", n, buf);
        write(fd[1], buf, n); // write to pipe
        printf("Input line [%s] written to pipe\n",buf);
        if (even) { // 只用偶数循环读取
           n = read (fd[0], buf2, 80) ;
           buf2[n] = 0;
           printf("%d char read from pipe: [%s]\n", n, buf2);
        }
  }
  printf("bye bye\n");
  close(fd[0]);
  close(fd[1]);
  exit(0);
}
```

带有子进程的管道

*管道*是一种*通信机制*。如果一个进程将数据写入一个它正在读取的管道，那就毫无意义了。管道的一个自然应用就是使用多个进程。在 Unix/Linux 中，标准的通信方式是父进程创建*管道*，然后执行 **fork()**。然后，父进程和子进程都知道管道，因为它们共享管道的文件描述符。然后，它们会**关闭（）管道***过长的两端*，以避免意外和错误地访问管道。它们使用管道的*剩余两端*进行通信，传递数据。管道的实际缓冲区位于系统内核空间，程序员不可能看到。你还可以尝试在用**lab6C.c**中的**pipe()**创建管道之前执行fork()。



在下面这个简单的加密程序中，父程序创建了一个*管道*，然后*分叉*给子程序。两者都将*关闭管道*的*两端*，然后父进程将使用 **write()** 向子进程发送数据。子程序使用 **read()** 从父程序接收数据。**read()** **的**返回值是从管道读入缓冲区的字节数。如果出现错误，则返回负值。请注意，父节点和子节点之间传递的所有数据都是*连续的字节*序列，没有任何预定边界。如果读取器在管道另一端关闭时试图从管道中读取数据，或写入器在管道另一端关闭时试图向管道中写入数据，都会发生错误。检查*返回值*是否为*负数*，以确定*管道的使用是否完成*。

```
// 实验室 6C
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main()
{
  char mapl[] = "qwertyuiopasdfghjklzxcvbnm"; // 用于编码字母 char mapd[]
  = "1357924680"; // 用于编码数字
  intfd [2]; // 用于管道
```

```
  char buf[80];
  Inti  , n, childid;
  if (pipe(fd) < 0) {
     printf("Pipe creation error\n");
     exit(1);
  }
  childid = fork();
  if (childid < 0) {
     printf("Fork failed\n");
     exit(1);
  } else if (childid == 0) { // child
     close(fd[1]); // close child out
     while ((n = read(fd[0],buf,80)) > 0) { // 从管道读取数据
          buf[n] = 0;
          printf("<child> message [%s] of size %d bytes received\n",buf,n);
     }
     close(fd[0]);
     printf("<child> I have completed!\n");
  } else { // parent
     close(fd[0]); // close parent in
     while (1) {
          printf("<parent> please enter a message\n");
          n = read(STDIN_FILENO,buf,80); // read a line
          if (n <= 0) break; // EOF or error
          buf[-n] = 0;
          printf("<parent> message [%s] is of length %d\n",buf,n);
          for (i = 0; i < n; i++) // encrypting
              if (buf[i] >= 'a' && buf[i] <= 'z')
                 buf[i] = mapl[buf[i]-'a'];
              else if (buf[i] >= 'A' && buf[i] <= 'Z')
                 buf[i] = mapl[buf[i]-'A']-('a'-'A');
              else if (buf[i] >= '0' && buf[i] <= '9')
                 buf[i] = mapd[buf[i]-'0'];
          printf("<parent> sending encrypted message [%s] to child\n",buf);
          write(fd[1],buf,n); // 发送加密字符串
     }
     close(fd[1]);
     wait(NULL);
     printf("<parent> I have completed!\n");
  }
  exit(0);
}
```

请注意，在 **lab6C.c** 中，只有父节点可以向子节点传递数据，因为只有 *一个管道*，管道的*两端*（作为数组的两个文件描述符）。在传统的 Unix 系统中，如果子进程需要将数据返回给父进程，则需要*另一条管道*（另有两个文件描述符），而且第二条管道中的通信方向将是*相反的*。这种传统管道称为*非双工管道*。要了解***关闭***未使用管道的重要性，请看 **lab6D.c**，其中子代和父代***都***向同一管道写入文件。换句话说，子代***忘记***了*关闭*其输出管道的一端。试着反复输入并观察输出。现在，阅读器读取的将是*一团糟*，来自父代和子代的数据混杂在一起！

```
// 实验室 6D
.. .
int main()
{
  .. .
  } else if (childid == 0) { // child
     // close(fd[1]); 孩子忘记关闭了
     while ((n = read(fd[0],buf,80)) > 0) { // read from pipe
          buf[n] = 0;
          printf("<child> message [%s] of size %d bytes received\n",buf,n);
          sleep(3); // 将这一行添加到延迟子程序中
          for (i = 0, j = 0; i < n; i = i+2, j++) // 跳过奇数字符 buf2[j] =
              buf[i];
```

```
            write(fd[1],buf2,j); // 意外地通过 fd[1] 回传新字符串
        }
        close(fd[0]);
        printf("<child> I have completed!\n");
    } else { // parent
        .. .
}
```

实验室练习

在**实验 4** 中，您从父进程创建了 *n* 个子进程，并通过让子进程从参数列表或变量中获取必要的数据，将模拟任务分配给这些子进程。在实际应用中，父进程*可能无法*预先*知道*数据，只能由用户提供。父进程也可能需要*根据需要*决定将哪些数据传递给哪个子进程、

例如，在有许多任务需要分配的情况下，将下一个可用任务传递给刚刚完成当前任务的子进程。在**实验室 4 的**模拟研究中，当任务很多但子进程数量有限时，这样做会更有效。因此，父进程必须能够***在****子进程*创建***后****将数据传递给子进程，这一点*与使用**管道的 lab6C.c** 类似。更重要的是，子进程通常需要将*结果反馈*给父进程，因为父进程创建子进程的目的通常是帮助处理父进程的部分工作量。*退出状态*只能返回 0 到 255 之间的一个值，而且只能在子进程终止时返回。因此，需要在子进程仍在执行时将数据传回父进程。

您需要扩展程序 **lab6C.c，**以便玩一个简单版本的**红心**游戏，并允许父进程和子进程之间进行*通信*。父进程首先*创建*所有必要的*管道*（每个子进程***两个管道***，一个从父进程到子进程，另一个从子*进程*到父*进程*），*然后分叉子进程*。请注意，每个管道都与有两个文件描述符，一个用于读取，一个用于写入。父进程和每个子进程都将***关闭***各自管道的过度端。这对程序开发非常重要。如果没有关闭过多的管道，往往会导致程序不正确。进程将通过管道进行通信，以执行必要的计算。最后，所有人都关闭所有管道，父进程将*等待*所有子进程在游戏结束时终止。实际上，我们采用的是星形通信拓扑结构，父进程扮演着关键角色。

您将创建 **4** 个子进程来玩 "**红心** "游戏。输入是从*键盘*接收的，但预期输入会从文件*重定向*，因此输入应采用文件*结束测试*。输入文件包含一组牌。每张牌的形式为 <花色等级>，其中花色为 **S**（♠ 或*黑桃*）、**H**（♥ 或*红心*）、**C**（♣ 或*梅花*）或 **D**（♦ 或*方块*），等级从 **{A, K, Q, J, T, 9, 8, 7, 6, 5, 4, 3, 2} 中**按降序抽取，其中 **T** 表示 **10**。可以假设输入行中没有错误。父进程读入纸牌，直到 *EOF*。然后以*循环方式*将牌发给子进程，游戏开始。

在**红心**游戏中，*起始玩家*出*一张牌*。这张牌称为*先手牌*。每位玩家轮流出另一张牌。以下是游戏规则。**规则 1**：每位玩家***必须出***一张与主牌*花色相同的牌*，除非该玩家没有同花色的牌。在这种情况下，他/她可以出*任何牌*。我们称这种情况为 "***弃牌***"。**规则 2**：打出与主牌花色*相同*的*最高牌的*玩家将 "***赢得***"本轮打出的所有 4 张牌。与主牌花色不同的牌总是***输***，即使它是 **A**。例如，如果主牌是 ♦**2**，♠A 这张牌仍然会输给它。**规则 3**：一轮的赢家将是下一轮的开始玩家。

红心的计分规则是*非常规的*。赢得一张 ♥ 的玩家将得到 1 分。赢得♠Q 这张特殊牌的玩家将得到 13 分。因此一手牌打完后，所有玩家通常会得到 26 点。有一个非常重要的***例外情况***。如果某位玩家赢了所有 13 张 ♥ 牌和 **♠Q**，那么他/她就是***大赢家，***其他三位玩家都会得到 26 点（因此总共会得到 78 点）。游戏进行到预定次数后，***积分最少的***玩家将成为冠军。

玩牌的策略是*避免*赢得一轮，并尽量不要意外赢得 ♠ Q。因此简单的方法是尽可能打出低牌。需要注意的是，如果已经打出了一张大牌，玩家可以尝试打 出 一张比最高牌*小的*大牌，以降低在接下来的花色中获胜的风险 (***S1***)。如果玩家是最后一名玩家，即使打出最低的牌也能赢牌，那么他/她就会打出最高的牌来赢牌，从而保留手中的 低牌以备将来使用（***S2***）。不过，为了简化程序逻辑，我们不会实施这些策略。换句话说，你只需打出领头牌所需花色中的最低牌即可。将来，你可以考虑在程序中引入人工智能，让它像一个合格的玩家一样玩牌。

我们假设每个子进程在玩游戏时都很幼稚。起始玩家总是会选择手中最低的牌来玩。由于没人想得到分数，每个玩家都会尽量避免赢得 ♠Q 和 ♥ 牌。因此，一个人总是会打出领头牌花色中的最低牌，以避免赢牌，或至少减少赢牌的机会。如果在游戏过程中有机会 "弃牌"（即没有与先导牌花色相同的牌），♠Q 将被选作弃牌。如果没有 **♠Q**，则弃最高的 ♥ 牌。 如果没有 ♥ 牌，则弃掉剩余牌中最高的牌。如果有多于一张不同花色的最高牌可弃，则只需先弃 ♠ 后弃 ♣ 再弃 ♦ （简单的平分规则）。

在这个程序中，家长就像一张放牌的桌子，同时也是一个*仲裁者，*告诉每个孩子出了什么牌。一个孩子出牌时，会通过*管道将*自己的牌发送给家长，然后家长会通过*管道*将这一轮出的牌转发给下一个孩子，供其考虑。父进程实际上是整个程序的*总控制器*，它负责提示子进程出牌，并将出过的牌通知给子进程。

下面是一个简单的安排：每个子代总是试图从管道中读取父代的*请求*。父代开始游戏时，先 "写 " 给第一个子代，然后从该子代 "读 "牌。子代从父代那里 "读取 "并 "写入 "自己的牌，让父代和子代一起玩。然后，家长将这一轮玩过的所有牌 "写 "给下一个孩子。游戏结束后，父代会计算并打印每个子代的得分，因为它知道所有出的牌和每轮的 "赢家"。然后，父进程会将计算结果 "写入 " 所有子进程，并结束游戏。请注意，子进程在开始时打印信息的顺序可能不同，您不必试图控制它们的顺序。

不要忘记***等待***所有子进程完成，并在游戏结束时***关闭***管道。请提供适当的注释，并在提交前检查您的程序。您的程序必须在 **apollo** 或 **apollo2** 上运行。下面是一些执行示例，显示的是数据文件 **card.txt** **的**内容

执行示例（输入数据存储在文本文件中，如 **card.txt**）：

```
hearts < card1.txt
```

```
dk dq s4 s8  cj c2 d3 ha sk s2 ct hq st h7 ht hj c4 c8 d7cq c6 d8 h9
             s5 h4 d2 s7 c5 hk h6 h2sa sq c9 d9 dah5 s3                    ca d4
c7s 9  dt sj
```

输出示例 1（由于同时执行进程，输出行可能不按此特定顺序排列）：

```
父播放器 pid 12345：子播放器为 12346 12347 12348 12349
Child 2 pid 12347: received DQ CJ SK ST C4 C3 H9 H4 HK D6 D9 CA S9
Child 1 pid 12346: received DK H3 HA HQ HJ H8 D8 S5 C5 S6 C9 S3 CK
Child 2 pid 12347: arranged SK ST S9 HK H9 H4 CA CJ C4 C3 DQ D9 D6
Child 3 pid 12348: received S4 C2 S2 H7 C8 CQ D5 D2 H6 SA DA D4 DT
Child 1 pid 12346：已安排 S6 S5 S3 HA HQ HJ H8 H3 CK C9 C5 DK D8 儿
童 4 pid 12349：已收到 S8 D3 CT HT D7 C6 DJ S7 H2 SQ H5 C7 SJ 儿童 4
pid 12349：已安排 SQ SJ S8 S7 HT H5 H2 CT C7 C6 DJ D7 D3 儿童 3 pid
12348：已安排 SA S4 S2 H7 H6 CQ C8 C2 DA DT D5 D4 D2 家长 pid 12345
：第 1 轮儿童 1 将领先
儿童 1 pid 12346：播放 H3
父代 pid 12345：子代 1 播放 H3 子代 2 pid
12347：播放 H4
父代 pid 12345：子代 2 播放 H4 子代 3 pid
12348：播放 H6
父代 pid 12345：子代 3 播放 H6 子代 4 pid
12349：播放 H2
家长 pid 12345：孩子 4 玩 H2 家长 pid 12345
：孩子 3 赢了把戏
```

```
父级 pid 12345：第 2 轮子 3 到领先
儿童 3 pid 12348：播放 D2
父代 pid 12345：子代 3 播放 D2 子代 4 pid
12349：播放 D3
父代 pid 12345：子代 4 播放 D3 子代 1 pid
12346：播放 D8
父代 pid 12345：子代 1 播放 D8 子代 2 pid
12347：播放 D6
家长 pid 12345：孩子 2 下 D6 家长 p id 12345
：孩子 1 赢了把戏 家长 pid 12345：第 3 轮孩子
1 领先
儿童 1 pid 12346：播放 S3
父代 pid 12345：子代 1 播放 S3 子代 2 pid
12347：播放 S9
父代 pid 12345：子代 2 播放 S9 子代 3 pid
12348：播放 S2
父代 pid 12345：子代 3 播放 S2 子代 4 pid
12349：播放 S7
家长 pid 12345：孩子 4 玩 S7 家长 p id 12345
：孩子 2 赢了把戏 家长 pid 12345：第 4 轮孩子
2 领先
儿童 2 pid 12347：播放 C3
父代 pid 12345：子代 2 播放 C3 子代 3 pid
12348：播放 C2
父代 pid 12345：子代 3 播放 C2 子代 4 pid
12349：播放 C6
父代 pid 12345：子代 4 播放 C6 子代 1 pid
12346：播放 C5
家长 pid 12345：孩子 1 下 C5 家长 p id 12345
：孩子 4 赢了戏法 家长 pid 12345：第 5 轮孩子
4 领先
儿童 4 pid 12349：播放 H5
父代 pid 12345：子代 4 播放 H5 子代 1 pid
12346：播放 H8
父代 pid 12345：子代 1 播放 H8 子代 2 pid
12347：播放 H9
父代 pid 12345：子代 2 播放 H9 子代 3 pid
12348：播放 H7
家长 pid 12345：孩子 3 下 H7 家长 p id 12345
：孩子 2 赢了把戏 家长 pid 12345：第 6 轮孩子
2 领先
儿童 2 pid 12347：播放 C4
父代 pid 12345：子代 2 播放 C4 子代 3 pid
12348：播放 C8
父代 pid 12345：子代 3 播放 C8 子代 4 pid
12349：播放 C7
父代 pid 12345：子代 4 播放 C7 子代 1 pid
12346：播放 C9
家长 pid 12345：孩子 1 下 C9 家长 p id 12345
：孩子 1 赢了把戏 家长 pid 12345：第 7 轮孩子
```

```
1 领先
儿童 1 pid 12346：播放 S5
父代 pid 12345：子代 1 播放 S5 子代 2 pid
12347：播放 ST
父代 pid 12345：子代 2 播放 ST 子代 3 pid
12348：播放 S4
父代 pid 12345：子代 3 播放 S4 子代 4 pid
12349：播放 S8
家长 pid 12345：孩子 4 玩 S8 家长 pid 12345
：孩子 2 赢了戏法 家长 pid 12345：第 8 轮孩子
2 领先
儿童 2 pid 12347：播放 D9
父代 pid 12345：子代 2 播放 D9 子代 3 pid
12348：播放 D4
父代 pid 12345：子代 3 播放 D4 子代 4 pid
12349：播放 D7
父代 pid 12345：子代 4 播放 D7 子代 1 pid
12346：播放 DK
家长 pid 12345：孩子 1 玩 DK 家长 pid 12345
：孩子 1 赢了把戏 家长 pid 12345：第 9 轮孩子
1 领先
儿童 1 pid 12346：播放 S6
父代 pid 12345：子代 1 播放 S6 子代 2 pid
12347：播放 SK
父代 pid 12345：子代 2 播放 SK 子代 3 pid
12348：播放 SA
父代 pid 12345：子代 3 播放 SA
```

```
儿童 4 pid 12349: 播放 SJ
家长 pid 12345: 孩子 4 玩 SJ 家长 pid 12345
: 孩子 3 赢了把戏
父级 pid 12345: 第 10 轮子 3 到领先
儿童 3 pid 12348: 播放 D5
父代 pid 12345: 子代 3 播放 D5 子代 4 pid
12349: 播放 DJ
父代 pid 12345: 子代 4 播放 DJ 子代 1 pid
12346: 播放 HA
父代 pid 12345: 子代 1 下 HA                    #，注意这是第一次弃子 子代 2 pid 12347:
下 DQ
家长 pid 12345: 孩子 2 玩 DQ 家长 pid 12345
: 孩子 2 赢了把戏
父级 pid 12345: 第 11 轮子 2 到领先
儿童 2 pid 12347: 播放 CJ
父代 pid 12345: 子代 2 播放 CJ 子代 3 pid
12348: 播放 CQ
父代 pid 12345: 子代 3 播放 CQ 子代 4 pid
12349: 播放 CT
父代 pid 12345: 子代 4 播放 CT 子代 1 pid
12346: 播放 CK
家长 pid 12345: 孩子 1 玩 CK 家长 pid 12345
: 孩子 1 赢了把戏
父级 pid 12345: 第 12 轮子 1 到领先
儿童 1 pid 12346: 播放 HJ
父代 pid 12345: 子代 1 播放 HJ 子代 2 pid
12347: 播放 HK
父代 pid 12345: 子代 2 播放 HK 子代 3 pid
12348: 播放 DA
父代 pid 12345: 子代 3 播放 DA 子代 4 pid
12349: 播放 HT
家长 pid 12345: 孩子 4 玩 HT 家长 pid 12345
: 孩子 2 赢了把戏
父级 pid 12345: 第 13 轮子 2 到领先
儿童 2 pid 12347: 播放 CA
父代 pid 12345: 子代 2 播放 CA 子代 3 pid
12348: 播放 DT
父代 pid 12345: 子代 3 播放 DT 子代 4 pid
12349: 播放 SQ
父代 pid 12345: 子代 4 播放 SQ 子代 1 pid
12346: 播放 HQ
家长 pid 12345: 孩子 1 玩 HQ 家长 pid 12345
: 孩子 2 赢了游戏 家长 pid 12345: 游戏结束
父级 pid 12345: 得分 = <0 22 4 0>
```

hearts < card2.txt

```
h3 cj c2 d3 ha sk s2 ct hq st h7 ht hj c4 c8 d7 h8 c3 cq c6 d8 h9 d5 dq s5 h4 d2 s7 c5 hk h6 h2
s6 d6 sa sq c9 d9 da h5 s3 ca d4 c7 ck s9 dt sj dk dj s4 s8
```

输出示例 2：

```
父播放器 pid 12355：子播放器为 12356 12357 12358 12359
Child 1 pid 12356: received H3 HA HQ HJ H8 D8 S5 C5 S6 C9 S3 CK DK
Child 1 pid 12356: arranged S6 S5 S3 HA HQ HJ H8 H3 CK C9 C5 DK D8
Child 2 pid 12357: received CJ SK ST C4 C3 H9 H4 HK D6 D9 CA S9 DJ
Child 2 pid 12357: arranged SK ST S9 HK H9 H4 CA CJ C4 C3 DJ D9 D6
Child 3 pid 12358：收到 C2 S2 H7 C8 CQ D5 D2 H6 SA DA D4 DT S4
Child 3 pid 12358: arranged SA S4 S2 H7 H6 CQ C8 C2 DA DT D5 D4 D2
Child 4 pid 12359: received D3 CT HT D7 C6 DQ S7 H2 SQ H5 C7 SJ S8
Child 4 pid 12359: arranged SQ SJ S8 S7 HT H5 H2 CT C7 C6 DQ D7 D3
Parent pid 12355: round 1 child 1 to lead（第 1 轮第 1 个孩子领先）。
儿童 1 pid 12356：播放 H3
父代 pid 12355：子代 1 播放 H3 子代 2 pid
12357：播放 H4
父代 pid 12355：子代 2 播放 H4 子代 3 pid
12358：播放 H6
父代 pid 12355：子代 3 播放 H6 子代 4 pid
12359：播放 H2
家长 pid 12355：孩子 4 玩 H2 家长 p id 12355
：孩子 3 赢了把戏 家长 pid 12355：第 2 轮孩子
3 领先
儿童 3 pid 12358：播放 D2
父代 pid 12355：子代 3 播放 D2 子代 4 pid
12359：播放 D3
父代 pid 12355：子代 4 播放 D3 子代 1 pid
12356：播放 D8
```

```
父代 pid 12355：子代 1 播放 D8 子代 2 pid
12357：播放 D6
家长 pid 12355：孩子 2 下 D6 家长 pid
12355：孩子 1 赢了把戏 家长 pid 12355：第 3
轮孩子 1 领先
儿童 1 pid 12356：播放 S3
.. .
家长 pid 12355：孩子 2 赢了把戏 家长 pid 12355：第
13 轮孩子 2 领先
儿童 2 pid 12357：播放 CA
父代 pid 12355：子代 2 播放 CA 子代 3 pid
12358：播放 DT
父代 pid 12355：子代 3 播放 DT 子代 4 pid
12359：播放 SQ
父代 pid 12355：子代 4 播放 SQ 子代 1 pid
12356：播放 HQ
家长 pid 12355：孩子 1 玩 HQ 家长 pid 12355
：孩子 2 赢了把戏 家长 pid 12355：游戏结
束
父 pid 12355：得分 = <0 21 4 1>
```

hearts < card3.txt

```
DK DQ S4 S8 H3 CJ C2 D3 HA SK S2 CT H8 ST H7 HT HJ C4 C8 D7 H4 C3 CQ C6 D8 H9 D5 DJ S5 HQ D2 S7
C5 HK H6 H2 S6 D6 SA SQ C9 D9 DA H5 S3 CA D4 C7 CK S9 DT SJ
```

输出示例 3：

```
父 pid 12432：子播放器为 12433 12434 12435 12436
Child 1 pid 12433： received DK H3 HA H8 HJ H4 D8 S5 C5 S6 C9 S3
CK Child 1 pid 12433： arranged S6 S5 S3 HA HJ H8 H4 H3 CK C9 C5
DK D8 Child 2 pid 12434： received DQ CJ SK ST C4 C3 H9 HQ HK D6
D9 CA S9 Child 2 pid 12434： arranged SK ST S9 HK HQ H9 CA CJ C4
C3 DQ D9 D6 Child 3 pid 12435：收到 S4 C2 S2 H7 C8 CQ D5 D2 H6 SA
DA D4 DT 儿童 3 pid 12435：已安排 SA S4 S2 H7 H6 CQ C8 C2 DA DT D5
D4 D2 儿童 4 pid 12436：已收到 S8 D3 CT HT D7 C6 DJ S7 H2 SQ H5 C7
SJ 儿童 4 pid 12436：已安排 SQ SJ S8 S7 HT H5 H2 CT C7 C6 DJ D7 D3
家长 pid 12432：第 1 轮儿童 1 将领先
儿童 1 pid 12433：播放 H3
父代 pid 12432：子代 1 播放 H3 子代 2 pid
12434：播放 H9
父代 pid 12432：子代 2 播放 H9 子代 3 pid
12435：播放 H6
父代 pid 12432：子代 3 播放 H6 子代 4 pid
12436：播放 H2
家长 pid 12432：孩子 4 玩 H2 家长 p id 12432
：孩子 2 赢了把戏 家长 pid 12432：第 2 轮孩子
2 领先
儿童 2 pid 12434：播放 C3
父代 pid 12432：子代 2 播放 C3 子代 3 pid
12435：播放 C2
父代 pid 12432：子代 3 播放 C2 子代 4 pid
12436：播放 C6
父代 pid 12432：子代 4 播放 C6 子代 1 pid
12433：播放 C5
家长 pid 12432：孩子 1 下 C5 家长 p id 12432
：孩子 4 赢了戏法 家长 pid 12432：第 3 轮孩子
4 领先
儿童 4 pid 12436：播放 D3
.. .
家长 pid 12432：孩子 2 赢了把戏 家长 pid 12432：第
13 轮孩子 2 领先
儿童 2 pid 12434：播放 CA
父代 pid 12432：子代 2 播放 CA 子代 3 pid
12435：播放 DT
父代 pid 12432：子代 3 播放 DT 子代 4 pid
12436：播放 SQ
父代 pid 12432：子代 4 播放 SQ 子代 1 pid
12433：播放 HJ
父级 pid 12432：孩子 1 玩 HJ 父级 pid 12432
：孩子 2 赢了把戏 父级 pid 12432：游戏结束
家长 pid 12432: score = <26 0 26 26>                    # 红心和 SQ 的大赢家孩子 2
```

```
hearts < card4.txt
```

```
c4 ht h9 hj d4 st dt ck cj ha d9 c9 d7 d5 d2 sa c8 sj s6 sk h5 h2 s4 dq dk c2 d8 h6 sq s7 s3 ct
hk s2 ca h4 da c7 h8 hq cq c3 d3 s8 dj h7 c5 d6 s5 s9 c6 h3
```

输出示例 4：

```
父播放器 pid 12456：子播放器为 12457 12458 12459 12460
Child 1 pid 12457: received C4 D4 CJ D7 C8 H5 DK SQ HK DA CQ DJ S5
Child 1 pid 12457: arranged SQ S5 HK H5 CQ CJ C8 C4 DA DK DJ D7 D4
Child 2 pid 12458: received HT ST HA D5 SJ H2 C2 S7 S2 C7 C3 H7 S9
Child 2 pid 12458: arranged SJ ST S9 S7 S2 HA HT H7 H2 C7 C3 C2 D5
Child 3 pid 12459：接收到 H9 DT D9 D2 S6 S4 D8 S3 CA H8 D3 C5 C6 儿
童 3 pid 12459：已安排 S6 S4 S3 H9 H8 CA C6 C5 DT D9 D8 D3 D2 儿童 4
pid 12460：接收到 HJ CK C9 SA SK DQ H6 CT H4 HQ S8 D6 H3 儿童 4 pid
12460：已安排 SA SK S8 HQ HJ H6 H4 H3 CK CT C9 DQ D6 家长 pid 12456
：第 1 轮儿童 1 到领先地位
儿童 1 pid 12457：播放 D4
父代 pid 12456：子代 1 播放 D4 子代 2 pid
12458：播放 D5
父代 pid 12456：子代 2 播放 D5 子代 3 pid
12459：播放 D2
父代 pid 12456：子代 3 播放 D2 子代 4 pid
12460：播放 D6
家长 pid 12456：孩子 4 下 D6 家长 p i d
12456：孩子 4 赢了把戏 家长 pid 12456：第 2
轮孩子 4 领先
儿童 4 pid 12460：播放 H3
父代 pid 12456：子代 4 播放 H3 子代 1 pid
12457：播放 H5
父代 pid 12456：子代 1 播放 H5 子代 2 pid
12458：播放 H2
父代 pid 12456：子代 2 播放 H2 子代 3 pid
12459：播放 H8
父方 pid 12456：子 3 下 H8 父方 p i d  12456
：子 3 赢牌 父方 pid 12456：第 3 轮子 3 领先
儿童 3 pid 12459：播放 D3
父节点 12456：子节点 3 播放 D3 子节点 4
12460：播放 DQ
父代 pid 12456：子代 4 播放 DQ 子代 1 pid
12457：播放 D7
父代 pid 12456：子代 1 播放 D7 子代 2 pid
12458：播放 HA
父级 pid 12456：子 2                                    下     HA#
弃子 父级 pid 12456：子 4 赢棋
父级 pid 12456：第 4 轮子 4 到领先
儿童 4 pid 12460：播放 H4
父代 pid 12456：子代 4 播放 H4 子代 1 pid
12457：播放 HK
父代 pid 12456：子代 1 播放 HK 子代 2 pid
12458：播放 H7
父代 pid 12456：子代 2 播放 H7 子代 3 pid
12459：播放 H9
家长 pid 12456：孩子 3 玩 H9 家长 p i d
12456：孩子 1 赢了把戏 家长 pid 12456：第 5
```

```
轮孩子 1 领先
儿童 1 pid 12457：播放 C4
父代 pid 12456：子代 1 播放 C4 子代 2 pid
12458：播放 C2
父代 pid 12456：子代 2 播放 C2 子代 3 pid
12459：播放 C5
父代 pid 12456：子代 3 播放 C5 子代 4 pid
12460：播放 C9
家长 pid 12456：孩子 4 下 C9 家长 pid
12456：孩子 4 赢了戏法 家长 pid 12456：第 6
轮孩子 4 领先
儿童 4 pid 12460：播放 H6
父代 pid 12456：子代 4 播放 H6
孩子 1 pid 12457：播放                                SQ# 尽可能早地丢弃 SQ 家长 pid 12456：孩子 1
播放 SQ
儿童 2 pid 12458：播放 HT
父代 pid 12456：子代 2 播放 HT 子代 3 pid
12459：播放 CA
家长 pid 12456：孩子 3 玩 CA 家长 pid 12456
：孩子 2 赢了把戏 家长 pid 12456：第 7 轮孩子
2 领先
儿童 2 pid 12458：播放 S2
. . .
父级 pid 12456：子级 4 赢了把戏
```

```
父级 pid 12456：第 13 轮子 4 到领先
儿童 4 pid 12460：播放 SA
父代 pid 12456：子代 4 播放 SA 子代 1 pid
12457：播放 DJ
父节点 12456：子节点 1 播放 DJ 子节点 2
12458：播放 S9
父代 pid 12456：子代 2 播放 S9 子代 3 pid
12459：播放 S6
父级 pid 12456：孩子 3 玩 S6 父级 pid 12456
：孩子 4 赢了把戏 父级 pid 12456：游戏结束
父级 pid 12456：得分 = <4 15 4 3>
```

hearts < card5.txt

```
ca cq d3 sk c3 s2 s4 dk cj h2 dq sq s3 s5 c2 d4 da c9 st h4 d7 s8 d6 sj hq d9
h9 ct c4 s6 h7 c5 ht c8 hk d5 c6 h6 h3 c7 h8 dt hj d2 d8 ha h5 sa s9 ck s7 dj
```

输出示例 5：

```
父节点 12578：子节点 12579 12580 12581 12582
Child 1 pid 12579: received CA C3 CJ S3 DA D7 HQ C4 HT C6 H8 D8 S9
Child 1 pid 12579: arranged S9 S3 HQ HT H8 CA CJ C6 C4 C3 DA D8 D7
Child 2 pid 12580: received CQ S2 H2 S5 C9 S8 D9 S6 C8 H6 DT HA CK
Child 2 pid 12580: arranged S8 S6 S5 S2 HA H6 H2 CK CQ C9 C8 DT D9
Child 3 pid 12581：收到 D3 S4 DQ C2 ST D6 H9 H7 HK H3 HJ H5 S7 儿童
3 pid 12581：已安排 ST S7 S4 HK HJ H9 H7 H5 H3 C2 DQ D6 D3 儿童 4
pid 12582：已收到 SK DK SQ D4 H4 SJ CT C5 D5 C7 D2 SA DJ 儿童 4 pid
12582：已安排 SA SK SQ SJ H4 CT C7 C5 DK DJ D5 D4 D2 家长 pid 12578
：第 1 轮第 1 个孩子领先
儿童 1 pid 12579：播放 C3
父代 pid 12578：子代 1 播放 C3 子代 2 pid
12580：播放 C8
父节点 12578：子节点 2 播放 C8 子节点 3
12581：播放 C2
父代 pid 12578：子代 3 播放 C2 子代 4 pid
12582：播放 C5
家长 pid 12578：孩子 4 下 C5 家长 p i d
12578：孩子 2 赢了戏法 家长 pid 12578：第 2
轮孩子 2 领先
儿童 2 pid 12580：播放 H2
父代 pid 12578：子代 2 播放 H2 子代 3 pid
12581：播放 H3
父代 pid 12578：子代 3 播放 H3 子代 4 pid
12582：播放 H4
父代 pid 12578：子代 4 播放 H4 子代 1 pid
12579：播放 H8
家长 pid 12578：孩子 1 下 H8 家长 p i d
12578：孩子 1 赢了把戏 家长 pid 12578：第 3
轮孩子 1 领先
儿童 1 pid 12579：播放 S3
父代 pid 12578：子代 1 播放 S3 子代 2 pid
12580：播放 S2
父代 pid 12578：子代 2 播放 S2 子代 3 pid
12581：播放 S4
父代 pid 12578：子代 3 播放 S4 子代 4 pid
12582：播放 SJ
家长 pid 12578：孩子 4 玩 SJ 家长 pid 12578
：孩子 4 赢了把戏 家长 pid 12578：第 4 轮孩子
4 领先
儿童 4 pid 12582：播放 D2
父节点 12578：子节点 4 播放 D2 子节点 1
12579：播放 D7
父代 pid 12578：子代 1 播放 D7 子代 2 pid
12580：播放 D9
父代 pid 12578：子代 2 播放 D9 子代 3 pid
12581：播放 D3
家长 pid 12578：孩子 3 下 D3 家长 p i d
12578：孩子 2 赢了戏法 家长 pid 12578：第 5
轮孩子 2 领先
儿童 2 pid 12580：播放 S5
```

```
父代 pid 12578：子代 2 播放 S5 子代 3 pid
12581：播放 S7
父节点 12578：子节点 3 播放 S7 子节点 4
12582：播放 SQ
父节点 12578：子节点 4 播放 SQ 子节点 1
12579：播放 S9
```

```
家长 pid 12578：孩子 1 下 S9 家长 pid
12578：孩子 4 赢了戏法 家长 pid 12578：第 6
轮孩子 4 领先
儿童 4 pid 12582：播放 D4
.. .
家长 pid 12578：孩子 2 赢了把戏 家长 pid 12578：第
13 轮孩子 2 领先
儿童 2 pid 12580：播放 CK
父节点 12578：子节点 2 播放 CK 子节点 3
12581：播放 H7
父节点 12578：子节点 3 播放 H7 子节点 4
12582：播放 DJ
父代 pid 12578：子代 4 播放 DJ 子代 1 pid
12579：播放 CJ
家长 pid 12578：孩子 1 玩 CJ 家长 pid 12578
：孩子 2 赢了游戏 家长 pid 12578：游戏结束
父级 pid 12578：得分 = <8 4 0 14>
```

1 级要求：创建 4 个子进程，并能将收到的卡片打印出来，***同时***使用 I/O 重定向（即使用 C 程序从 **stdin** 或键盘读取数据）。在这种简单的要求下，子进程只需根据自己的位置从输入中选出自己的卡片，*而无需使用管道*。

第 2 级要求：子进程能从父进程处收到自己的牌，并能通过管道与父进程进行通信以进行游戏，而且能在没有弃牌的情况下正确地进行第一轮游戏。

第 3 级要求：在没有弃子的情况下，棋局下得正确。

第 4 级要求：在任何情况下都能正确地进行比赛，并计算出正确的分数。

奖励关：在子进程中实施两个较好的策略（上面的 **S1** 和 **S2**），以便打出的最高牌刚好低于已打出的最高牌，从而降低将来获胜的风险，并在被迫作为最后玩家获胜时用最高牌获胜。或者，你也可以实施另一种更聪明的策略 **S3**，并用实例来解释它是如何工作的。

将程序命名为 **hearts.c**，并**在 2024 年 3 月 25 日或之前通过 BlackBoard 提交**。