

操作系统实验报告

姓名	王硕	学号	55200423	班级	卓越工程师班
实验题目： 进程创建以及 Linux 进程通信					
实验内容： 1.掌握 fork()、pipe()系统调用的形式和功能 2.掌握通过 fork()派生的子进程之间通过 pipe 文件的通信方式					
设计原理： 用 pipe()创建一个管道文件，然后用 fork()创建 2 个生产者进程和 2 个消费者进程，实现它们之间通过 pipe 文件实现信息传递。					
实验源码 <pre>1. #include"sys/types.h" 2. #include"sys/file.h" 3. #include"unistd.h" 4. 5. char r_buf[4]; 6. char w_buf[4]; 7. int pipe_fd[2]; 8. pid_t pid1,pid2,pid3,pid4; 9. 10. int producer(int id); 11. int consumer(int id); 12. 13. int main(int argc,char **argv) 14. { 15. if(pipe(pipe_fd) < 0) 16. { 17. printf("pipe create erro"); 18. exit(-1); 19. } 20. else 21. { 22. printf("pipe is created successfully!\n");</pre>					

```

23.         if((pid1 = fork()) == 0 )
24.             producer(1);
25.         if((pid2 = fork()) == 0 )
26.             producer(2);
27.         if((pid3 = fork()) == 0 )
28.             consumer(1);
29.         if((pid4 = fork()) == 0 )
30.             consumer(2);
31.     }
32.     close(pipe_fd[0]);
33.     close(pipe_fd[1]);
34.
35.
36.     int i, pid,status;
37.     for( i = 0 ; i < 4;i++)
38.         pid = wait(&status);
39.     exit(0);
40. }
41.
42. int producer(int id)
43. {
44.     printf("producer %d is running! \n",id);
45.     close(pipe_fd[0]);
46.     int i = 0;
47.     for( i = 1 ;i < 30 ;i++)
48.     {
49.         sleep(3);
50.         if(id == 1)
51.             strcpy(w_buf, "aaa\0");
52.         else
53.             strcpy(w_buf, "bbb\0");
54.         if(write(pipe_fd[1],w_buf,4) == -1)
55.             printf("write to pipe error\n");
56.     }
57.     close(pipe_fd[1]);
58.     printf(" producer %d is over! \n",id);
59.     exit(id);
60. }
61. int consumer( int id)
62. {
63.
64.     close(pipe_fd[1]);
65.     printf("consumer %d is running\n",id);
66.     if( id == 1)

```

```

67.         strcpy(w_buf , "ccc\0");
68.     else
69.         strcpy(w_buf , "ddd\0");
70.     while(1)
71.     {
72.         sleep(1);
73.         strcpy(r_buf, "eee\0");
74.         if(read(pipe_fd[0] , r_buf, 4) == 0)
75.             break;
76.         printf("consumer %d get %s ,while the w_buf is %s\n", id, r_buf, w_buf);
77.     }
78.     close(pipe_fd[0]);
79.     printf("consumer %d is over !\n", id);
80.     exit(id);
81. }

```

2.实验结果

```

ws@192:~/home/OSsoftWare/semid
[ws@192 semid]$ ./pac
pipe is created successfully!
producer 2 is running!
consumer 1 is running
consumer 2 is running
producer 1 is running!
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
consumer 2 get aaa, while the w_buf is ddd
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
consumer 2 get aaa, while the w_buf is ddd
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
consumer 2 get aaa, while the w_buf is ddd
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
consumer 2 get aaa, while the w_buf is ddd
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
consumer 2 get aaa, while the w_buf is ddd
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
consumer 2 get aaa, while the w_buf is ddd
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
consumer 2 get aaa, while the w_buf is ddd
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
consumer 2 get aaa, while the w_buf is ddd
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 1 get bbb, while the w_buf is ccc
producer 2 is over!
consumer 1 is over!
consumer 2 get bbb, while the w_buf is ddd
consumer 1 get aaa, while the w_buf is ccc
consumer 2 is over !
consumer 1 is over !
[ws@192 semid]$

```

问题解析 1.参考代码 main 中两个 close(pipe_fd[0])和 close(pipe_fd[1])的作用

这两行代码的作用是关闭父进程对管道文件的读写操作，防止其影响模拟生产者、消费者的四个子进程间对管道的数据读写操作。如果 main 函数里没有这两行代码，在不同情况下会有生产者或则消费者永远等待。

学生通过修改实验指导手册的源码得到了 main 函数未写 close(pipe_fd[0]) 和

close(pipe_fd[1])时，生产者、消费者分别无限期等待的两种情况，以下分别展开分析。

一、消费者无限等待

1.修改后代码：在源代码基础上删去 main 函数中的 close(pipe_fd[0])和 close(pipe_fd[1])两行代码即可。

2.结果展示

```
[ws@92 semid]$ ./pac
pipe is created successfully!
producer 2 is running!
consumer 1 is running
consumer 2 is running
producer 1 is running!
consumer 2 get bbb ,while the w_buf is ddd
consumer 1 get aaa ,while the w_buf is ccc
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 2 get bbb ,while the w_buf is ddd
consumer 1 get aaa ,while the w_buf is ccc
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 2 get bbb ,while the w_buf is ddd
consumer 1 get aaa ,while the w_buf is ccc
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get bbb ,while the w_buf is ddd
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get aaa ,while the w_buf is ccc
consumer 2 get bbb ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get aaa ,while the w_buf is ccc
consumer 2 get bbb ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get aaa ,while the w_buf is ccc
consumer 2 get bbb ,while the w_buf is ddd
producer 1 is over!
producer 2 is over!
consumer 2 get bbb ,while the w_buf is ddd
consumer 1 get aaa ,while the w_buf is ccc
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 is over !
consumer 1 is over !
[ws@92 semid]$
```

```
[ws@92 semid]$ ./pac
pipe is created successfully!
producer 1 is running!
producer 2 is running!
consumer 1 is running
consumer 2 is running
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 1 get aaa ,while the w_buf is ccc
consumer 2 get bbb ,while the w_buf is ddd
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 1 get aaa ,while the w_buf is ccc
consumer 2 get bbb ,while the w_buf is ddd
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 1 get aaa ,while the w_buf is ccc
consumer 2 get bbb ,while the w_buf is ddd
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 1 get aaa ,while the w_buf is ccc
consumer 2 get bbb ,while the w_buf is ddd
consumer 2 get aaa ,while the w_buf is ddd
producer 1 is over!
producer 2 is over!
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 is over !
consumer 1 is over !
[ws@92 semid]$
```

图 1: consumer 无限等待 图左，进程正常结束 图右

3.结果分析

如果父进程没有关闭文件读写描述符，当生产者进程全部生产完物品（数据）后关闭了他自己的写文件描述符 pipe_fd[1]，消费者一直在运行，消耗物品（数据），直到管道缓冲区内数据全部消耗完，此时因为父进程的写文件描述符 pipe_fd[1]未关闭，消费者进程在执行 read()操作时会被阻塞，等待父进程往管道缓冲区写数据，但是父进程没有 write()操作，因此消费者进程进入了无限期等待。

如果父进程的写文件描述符 pipe_fd[1]关闭，即此时的管道文件 Inode 的 i 写文件描述符的 count 值为 0，所有进程都关闭了写端，系统认为已经读到了数据的末尾, read()函数返回的读出字节数为 0,跳出循环，所以 consumer 不会被阻塞,无限等待。

二、生产者无限等待

1.修改后的代码

```
1. int producer(int id)
2. {
3.     printf("producer %d is running! \n",id);
4.     close(pipe_fd[0]);
5.     int i = 0;
6.     while(1)
```

```

7.      {
8.          sleep(1);
9.          if(id == 1)
10.             strcpy(w_buf, "aaa\0");
11.         else
12.             strcpy(w_buf, "bbb\0");
13.         if(write(pipe_fd[1],w_buf,4) == -1)
14.             {
15.                 printf("write to pipe error\n");
16.                 break;
17.             }
18.     }
19.     close(pipe_fd[1]);
20.     printf(" producer %d is over! \n",id);
21.     exit(id);
22. }
23. int consumer( int id)
24. {
25.
26.     close(pipe_fd[1]);
27.     printf("consumer %d is running\n",id);
28.     if( id == 1)
29.         strcpy(w_buf , "ccc\0");
30.     else
31.         strcpy(w_buf , "ddd\0");
32.
33.     int i ;
34.     for( i = 1 ;i < 10 ;i++)
35.     {
36.         sleep(3);
37.         strcpy(r_buf, "eee\0");
38.         if(read(pipe_fd[0] ,r_buf,4) == 0)
39.             break;
40.         printf("consumer %d get %s ,while the w_buf is %s\n",id, r_buf,w_buf);
41.     }
42.     close(pipe_fd[0]);
43.     printf("consumer %d is over !\n",id);
44.     exit(id);
45. }

```

2.结果展示

```
[ws@192 semid]$ ./pacf
pipe is created successfully!
producer 2 is running!
consumer 1 is running
consumer 2 is running
producer 1 is running!
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 is over !
consumer 2 get aaa ,while the w_buf is ddd
consumer 2 is over !
```

```
[ws@192 semid]$ ./pacf
pipe is created successfully!
producer 2 is running!
consumer 1 is running
consumer 2 is running
producer 1 is running!
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 get bbb ,while the w_buf is ccc
consumer 2 get aaa ,while the w_buf is ddd
consumer 1 is over !
consumer 2 get aaa ,while the w_buf is ddd
consumer 2 is over !
[ws@192 semid]$
```

图 2: producer 无限等待图左, 进程正常结束图右

3.结果分析

如果父进程没有关闭文件读写描述符, 当消费者进程消耗完指定数量的物品(数据)后关闭了他自己的读文件描述符 `pipe_fd[0]`, 此时生产者一直在运行, 生产物品(数据), 直到管道缓冲区内空间全部被占满, 此时因为父进程的读文件描述符 `pipe_fd[0]`未关闭, 生产者进程在执行 `write()`操作时会被阻塞, 等待父进程从管道缓冲区读数据, 但是父进程没有 `read()`操作, 因此生产者进程被阻塞, 进入了无限期等待。

如果父进程的读文件描述符 `pipe_fd[0]`关闭, 即此时的管道文件 `Inode` 的读文件描述符的 `count` 值 0, 所有读进程都关闭了写端, 生产者执行 `write()`函数时将收到内核传来的 `SIGPIPE` 信号, 应用程序可以处理该信号, 也可以忽略(默认动作则是应用程序终止)。跳出循环, 所以 `producer` 不会被阻塞,无限等待。

问题解析 2.参考代码中

```
for(i=0;i<4;i++)
    pid=wait(&ststus);
```

的作用, 如果没有会是何种结果, 为什么?

`Wait()`函数起阻塞父进程进一步执行的作用, 父进程调用 `wait()`函数后会寻找在父进程前结束的变成僵尸进程的子进程, 从而释放子进程所占用的资源。

此代码用于父进程查询等待其 `fork()`派生的所有子进程是否终止, 在此问题中, 创建两个生产者进程和两个消费者进程工作均结束后, 通过 `exit(id)`向父进程传递该子任务结束的信息, 并由 `wait(&ststus)`接收, 由于有四个子进程所以需要循环四次。最后父进程和子进程所占的所有资源被回收。

```

[ws@92 semid]$ ./pac
pipe is created successfully!
producer 2 is running!
consumer 1 is running
consumer 2 is running
producer 1 is running!
consumer 2 get bbb ,while the w_buf is ddd
fahterID:5876
consumer 1 get aaa ,while the w_buf is ccc
fahterID:5876
consumer 1 get bbb ,while the w_buf is ccc
fahterID:5876
consumer 2 get aaa ,while the w_buf is ddd
fahterID:5876
consumer 2 get bbb ,while the w_buf is ddd
fahterID:5876
consumer 1 get aaa ,while the w_buf is ccc
fahterID:5876
consumer 1 get bbb ,while the w_buf is ccc
fahterID:5876
consumer 2 get aaa ,while the w_buf is ddd
fahterID:5876
consumer 1 get bbb ,while the w_buf is ccc
fahterID:5876
consumer 2 get aaa ,while the w_buf is ddd
fahterID:5876
producer 2 is over!
producer 1 is over!
consumer 2 get bbb ,while the w_buf is ddd
fahterID:5876
consumer 1 get aaa ,while the w_buf is ccc
fahterID:5876
consumer 2 is over !
consumer 1 is over !
[ws@92 semid]$ a

```

图三：父进程等待子进程结束，回收子进程资源

如果没有上述两行代码（即没有 `wait()` 函数），会出现两种情况。

一、由于本程序父进程指令少，很可能在四个子进程结束前提前结束，这四个进程就变成了孤儿进程，此时由进程号为 1 的 `init` 进程接管，待四个进程指令执行完毕后由 `init` 进程回收，这种情况是 OS 可以正常处理的情况，不会带来资源浪费的情况。

```

[ws@92 semid]$ ./pac
pipe is created successfully!
producer 1 is running!
producer 2 is running!
consumer 1 is running
consumer 2 is running
[ws@92 semid]$ consumer 2 get aaa ,while the w_buf is ddd
fahterID:1
consumer 1 get bbb ,while the w_buf is ccc
fahterID:1
consumer 1 get aaa ,while the w_buf is ccc
fahterID:1
consumer 2 get bbb ,while the w_buf is ddd
fahterID:1
consumer 2 get aaa ,while the w_buf is ddd
fahterID:1
consumer 1 get bbb ,while the w_buf is ccc
fahterID:1
consumer 1 get aaa ,while the w_buf is ccc
fahterID:1
consumer 2 get bbb ,while the w_buf is ddd
fahterID:1
consumer 2 get aaa ,while the w_buf is ddd
fahterID:1
consumer 1 get bbb ,while the w_buf is ccc
fahterID:1
consumer 1 get aaa ,while the w_buf is ccc
fahterID:1
consumer 2 get bbb ,while the w_buf is ddd
fahterID:1
producer 1 is over!
producer 2 is over!
consumer 2 get aaa ,while the w_buf is ddd
fahterID:1
consumer 1 get bbb ,while the w_buf is ccc
fahterID:1
consumer 2 is over !
consumer 1 is over !
█

```

图 4 父进程提前结束，`init` 接管子进程

二、如果有子进程在父进程结束前提前结束，该子进程就变成了僵尸进程：父进程由于没有 `wait()` 函数来等待它，如果该父进程一直运行，该子进程所占有的某些资源未被释放，当系统中有大量僵尸进程时，会阻碍新进程的产生，这是我们应该避免的。

问题解析 3. 参考代码 `producer()` 和 `consumer()` 中的 `sleep` 的作用

如上述代码所见，`producer()` 和 `consumer()` 每执行一次生产物品或消费物品的操作时，会调用一次 `sleep(n)` 操作。

