

课程名称	软硬件接口程序设计	课程编号	A2130731
实验地点	重庆邮电大学综合实验楼	实验时间	2022. 11. 17
实验名称	实验七 关于进程和管道通信的练习		

一、 实验目的

练习并理解进程操作以及管道通信。

二、 实验内容

任务 1:

1. 通过给出的例子，分析输出内容。

2. 通过给出的例子，分析 `fork()`的返回值，`fork()`返回 0 的意义以及通过哪个函数来获取父进程的 PID。

3. 通过给出的例子，分析 `wait` 函数的作用，`wait()`函数的参数是什么，`WEXITSTATUS()`的返回值是什么。

4. 通过给出的例子，分析例子中标亮的 `fprintf` 语句是否执行，`fork()`和 `exec()` 的区别在哪。

5. 编译并运行给出的程序，然后按 `Ctrl-C`。分析发生了什么？哪个信号被送到了进程中？

6. 编译并运行给出的程序，然后按 `Ctrl-C`。分析发生了什么？为什么它与以前的程序不同？

任务 2: 编写一个计算机程序，提供以下功能。

创建两个 C 程序：`p1.c` 和 `p2.c`。

1. `p1` 调用 `fork` 来创建一个子进程。这个子进程必须执行 `p2`，一个不同的程序。

2. `p1` 和 `p2` 必须能够向自己和对方发送信号（`SIGUSR1`, `SIGUSR2`）。

这些信号必须被各自的进程正确地捕捉和处理。

(1) 在信号处理程序中，必须打印进程的 **pid** 和信号信息。

3. **p1** 和 **p2** 必须能够通过 **FIFO** 传输数据。

(1) **p1** 写到一个 **FIFO**。任何字符都可以由 **p1** 写到 **FIFO** 中。

(2) **p2** 从同一个 **FIFO** 中读取。如果要读的字节是一个数字，则打印'**N**'; 如果要读的字节是一个字母，打印'**L**'; 否则，打印'**O**'。

编译这两个程序，并命名每个可执行程序。运行 **p1** 的可执行程序，验证结果。

三、实验过程原始记录(数据、图表等)

任务 1:

1:

```
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./task1_1
Hello world!
Hello world!
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./task1_1
Hello world!
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# Hello world!
./task1_1
Hello world!
Hello world!
```

图 1 task1_1 运行结果

2:

```
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./task1_2
PARENT: This is the parent process!
PARENT: My PID is 3561
PARENT: My child's PID is 3562
CHILD: This is the child process!
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# CHILD: My PID is 3562
CHILD: My parent's PID is 959
./task1_2
PARENT: This is the parent process!
PARENT: My PID is 3563
PARENT: My child's PID is 3564
CHILD: This is the child process!
CHILD: My PID is 3564
CHILD: My parent's PID is 3563
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./task1_2
PARENT: This is the parent process!
PARENT: My PID is 3565
PARENT: My child's PID is 3566
CHILD: This is the child process!
CHILD: My PID is 3566
CHILD: My parent's PID is 3565
```

图 2 task1_2 运行结果

3:

```
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./task1_3
PARENT: This is the parent process!
PARENT: My PID is 3841
PARENT: My child's PID is 3842
PARENT: I'm now waiting for my child to exit()...
CHILD: This is the child process!
CHILD: My PID is 3842
CHILD: My parent's PID is 3841
CHILD: Enter my exit status (make it small): 0
CHILD: I'm outta here!
PARENT: My child's exit status is: 0
PARENT: I'm outta here!
```

图 3 task1_3 运行结果

4:

```
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./task1_4
Child: PID of Child = 3839
Press Enter to continue

Parent: Child 3839 exited with status = 0
```

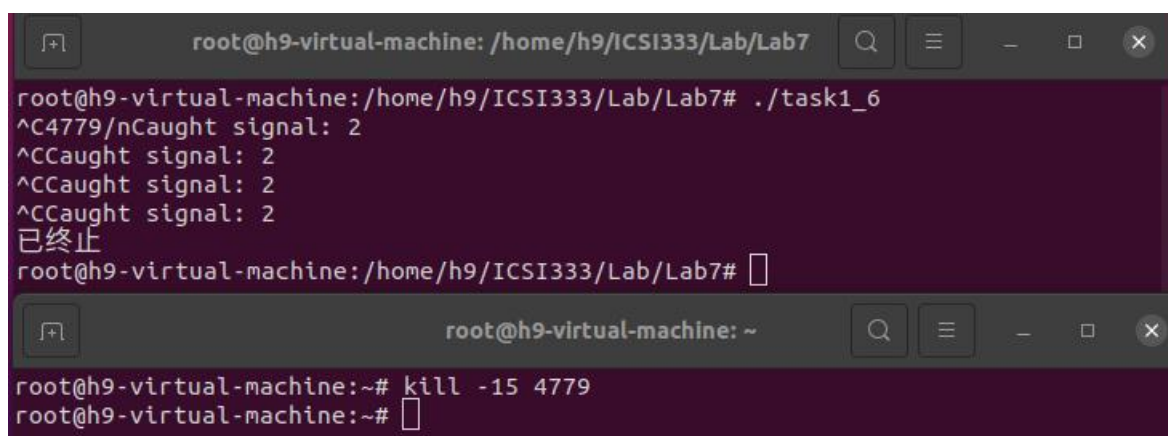
图 4 task1_4 运行结果

5:

```
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./task1_5
hello world
hello world
hello world
^C
```

图 5 task1_5 运行结果

6:



```
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./task1_6
^C4779/nCaught signal: 2
^CCaught signal: 2
^CCaught signal: 2
^CCaught signal: 2
已终止
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7#

root@h9-virtual-machine:~# kill -15 4779
root@h9-virtual-machine:~#
```

图 6 task1_6 运行结果

任务 2:

```
root@h9-virtual-machine:/home/h9/ICSI333/Lab/Lab7# ./p1

P1 send SIGUSR1 to self

P1: Handling SIGUSR1 in p1!
Pid: 3583
Signal information: 10

P1: Please put five charcaters
eqwe2

P1 send SIGUSR2 to p2

P2: Handling SIGUSR2 in p2!
Pid: 3584
Signal information: 12

P2 send SIGUSR1 to self

P2: Handling SIGUSR1 in p2!
Pid: 3584
Signal information: 10

P2:Tansfer characters
LLLLN

P2 send SIGUSR2 to self

P1: Handling SIGUSR2 in p1!
Pid: 3583
Signal information: 12
```

图 7 task2 运行结果

四、实验结果及分析

实验结果可见原始数据记录。相应分析如下：

Task1:

1. 输出结果为两次 “Hello world!”, 因为 `fork()` 创建了子进程。但易产生孤儿进程, 因为父进程可能先于子进程结束。
2. `fork()` 返回值有 >0 , $=0$, -1 三种情况。0 代表当前进程为子进程。子进程中可以通过 `getppid()` 获得父进程 `pid`, 父进程通过 `getpid()` 获得自身 `pid`。
3. `Wait()` 使调用该函数的进程等待其子进程状态改变后再继续执行。参数中的指针储存子进程的结束状态。`WEXITSTATUS()` 返回子进程的结束状态, 此例子中返回 0, 代表子进程自然结束。
4. 高亮语句不会执行, `execlp()` 使子进程执行其他程序。`Fork()` 是通过复制调用进

程来产生一个新进程，新进程是调用进程的精确副本。**Exec** 使当前进程替换为新的进程映像。

5. 按下 **Ctrl+c** 后，进程终止，即停止打印。**SIGINT** 信号发送到了当前进程。

6. 此次进程不会直接终止，而会处理信号，等待 **while** 循环结束后终止。

Task2:

1. 首先编写 **p1.c**, **main** 函数中首先调用 **fork()** 创建子进程，子进程中直接使用 **execl** 调用 **p2**。父进程中，首先使用 **kill** 将 **SIGUSR1** 发送给自身，并完成相应处理，再建立 **fifo** 存储五个字符，后将 **SIGUSR2** 发送给予进程，等待子进程结束。

2. 编写 **p2**。首先通过 **sleep(5)** 等待 **p1** 中完成字符输入，之后 **SIGUSR2** 信号已发送至 **p2**，故 **p2** 处理 **SIGUSR2**。**P2** 向自身发送 **SIGUSR1**，并完成相应处理，再读出字符，并使用判断功能函数按要求输出。之后，再向 **P1** 发送 **SIGUSR2**。

3. **P2** 结束后，**P1** 收到信号 **SIGUSR2**，完成相应处理后结束。

五、实验心得体会

韩乐陶：学习并掌握了 **fork()**, **wait()**, **pipe()**, **fifo()** 等函数，以及子进程、父进程，管道传输的两种类型，信号发送与处理，**exec** 函数集，以及它们之间的结合应用。

夏泽昊：学习并理解了操作系统中进程的概念，学会了使用 **fork()** 函数创建子进程，并使用 **exec** 函数族在进程中加载并运行一个新的程序，理解了进程和程序两个概念的区别。学会了如何使用信号中断进程并发送信号。理解了管道的概念，学会了如何使用无名管道和有名管道在进程间实现管道通信。

姚棋舰：**fork()** 所代表的父进程和子进程的使用可以帮助我们更好的使用代码，提升一些情况下的代码可读性和可写性，对于日后学习也有一定意义