进程

1. 函数接口

1.1 回收进程

```
pid_t waitpid(pid_t pid, int *status, int options);
功能：回收子进程资源
参数：
    pid：>0      指定子进程进程号
         =-1    任意子进程
         =0     等待其组 ID 等于调用进程的组 ID 的任一子进程
         <-1    等待其组 ID 等于 pid 的绝对值的任一子进程
    status：子进程退出状态
    options：0：阻塞
        WNOHANG：非阻塞
返回值：正常：结束的子进程的进程号
       当使用选项 WNOHANG 且没有子进程结束时：0
```

1.2 结束进程

```
void exit(int status);
功能：结束进程，刷新缓存
void _exit(int status);
功能：结束进程，不刷新缓存
参数：status 是一个整型的参数，可以利用这个参数传递进程结束时的状态。
    通常 0 表示正常结束；
其他的数值表示出现了错误，进程非正常结束
```

exit 和 return 区别：

exit：不管在子函数还是主函数，都可以结束进程

return：当子函数中有 return 时返回到函数调用位置，并不结束进程

b. 半双工的通信模式，具有固定的读端和写端

c. 管道可以看成是一种特殊的文件，对于它的读写可以使用文件 IO 如 read、write 函数.

d. 管道是基于文件描述符的通信方式。当一个管道建立时，它会创建两个文件描述符 fd[0]和 fd[1]。其中 fd[0]固定用于读管道，而 fd[1]固定用于写管道。

3.2 函数接口

```
int pipe(int fd[2])
功能：创建无名管道
参数：文件描述符 fd[0]：读端  fd[1]:写端
返回值：成功 0
    失败 -1
```

3.3 注意事项

a. 当管道中无数据时，读操作会阻塞；

管道中无数据时，将写端关闭，读操作会立即返回

b. 管道中装满（管道大小 64K）数据写阻塞，一旦有 4k 空间，写继续

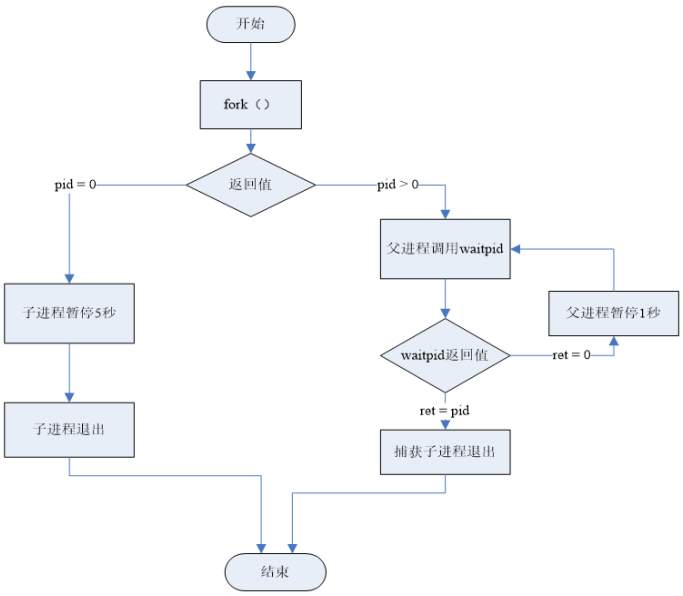
c. 只有在管道的读端存在时，向管道中写入数据才有意义。否则，会导致管道破裂，向管道中写入数据的进程将收到内核传来的 SIGPIPE 信号（通常 Broken pipe 错误)。

练习：父子进程通信。

父进程循环从终端输入字符串，子进程将字符串循环输出，当输入 quit 时，程序退出。

作业：

1. 梳理今天内容
2. 根据下面流程图编写程序


开始
fork（）
返回值
pid = 0
pid > 0
子进程暂停5秒
子进程退出
父进程调用waitpid
父进程暂停1秒
waitpid返回值
ret = 0
ret = pid
捕获子进程退出
结束