进程等待:处理僵尸进程

1.wait(int\* status)

阻塞式的等待子进程

成功退出后返回等待进程pid

失败返回-1

2.waitpid(pid\_t pp,int\* status,int options)

能够指定等待子进程, 成功退出后返回等待进程pid

既能够阻塞等待 又能够非阻塞等待

pid\_t /:需要等待的进程的pid

int\* status:正常终止子进程时返回值大于0 异常终止子进程可提取退出码

int options:WNOHANG -->非阻塞等待

wait(-1, ,0);第一个参数为-1任何一个子进程结束 都会被阻塞式等待

status(1.2方法的参数):能够反馈退出信息

正常退出:8-15比特位为保存退出信息 pp & 0x7f (7个1)

异常退出: (pp>>8)&0xff (8个1)

进程程序替换

替换的是程序代码和数据--若替换成功无返回值,替换失败返回-1

替换函数:(若有多个替换函数,只能执行一个替换函数就不能继续执行后面了)

execl(l 可变参数列表) --->库函数

execlp(p 环境变量path) --->库函数

execle(e 环境变量) --->库函数

execv(v 表示vector向量-->数组) --->库函数

execvp --->库函数

execve --->系统调用

要用NULL作为环境变量参数的结尾

shell运行原理

1. 等待用户输入指令
2. 解析用户输入的指令(可执行程序是什么,以及对应的参数是什么)
3. shell进程创建子进程
4. 子进程进行程序替换,替换成刚才输入的可执行文件
5. 父进程进行进程等待,等待子进程运行结束
6. shell重复执行1

ls -l /

stork(NULL,” ”)

第一次调用:返回结果是指向l的指针，同时会把s后面的空格设置为 \0

第二次调用：参数要填NULL，表示根据上次切分的结果，决定这次从哪个地方开始切分，返回结果就是指向 - 的指针了，并且把l后面的空格设置为 \0

第三次调用：参数还是填Null，返回结果就是指向/的指针，并且/后面的字符设置为0；

第四次调用：返回NULL

如何让shell支持重定向:

1. 解析指令时判断有没有 > 符号
2. 如果有 > 前面的部分就是要执行的指令 > 后面的部分就是要重定向的部分
3. 打开文件
4. 在子进程程序替换之前,对子进程进行重定向 把1重定向到刚才 > 打开的文件中
5. 程序指令执行完毕 关闭文件

头文件顺序

1. 标准库的文件
2. 系统文件
3. 第三方库文件
4. 当前项目中的其他头文件

myshell:父进程自身的工作目录没有改变

cd:修改了当前进程的工作目录

myshell缺陷:

1. 不支持内建指令(cd history builtin)
2. 不支持重定向(cat test1.c > test2.c)
3. 不支持管道符
4. 能够动态的获取到用户名,主机名,以及当前的工作目录