sys_call 系统调用，

read, write, exit, open, close, fork, exec, wait, pipe

其他的简单的如 getpid, sleep, 等； dup 暂时还不明白

read, write 读写

eg. ① read(fd, buf, sizeof(buf))

fd: 文件描述符　buf: 存字符串的指针　sizeof(buf): 长度

② read(p[1], "ping", 4)

open 打开文件　进程(主体)打开文件，内核分配文件描述符

(内核维护表，key为fd)

open 会分配最小的(当前)，可用的文件描述符，并返回

进程初始就打开了三个文件 (一切皆文件)

fd	文件
0	标准输入 stdin
1	标准输出 stdout
2	错误输出 stderr

可以用 close(fd) 关闭进程打开的文件，同时释放 fd.

0，1，2 也是可以被释放的，用于 重定向

exit 进程结束返回父进程一个值，只有 0 表示正常结束

fork 拷贝当前进程，得到子进程.

在父进程中，返回创建的子进程的 pid.

在子进程中，返回 0

用这个来区分父子进程 if (fork()==0)

父子进程内存空间独立. 数据相同，且同时运行

拷贝内存，拷贝了之前进程的文件描述符；

共享文件偏移量（offset）

与 shell 关系：shell 本身也是一个进程.

输入命令执行，shell会 fork 自身得到子进程，

然后用 exec 修改

↓

exec：加载指令，替代当前指令 eg：execl("echo", argv)

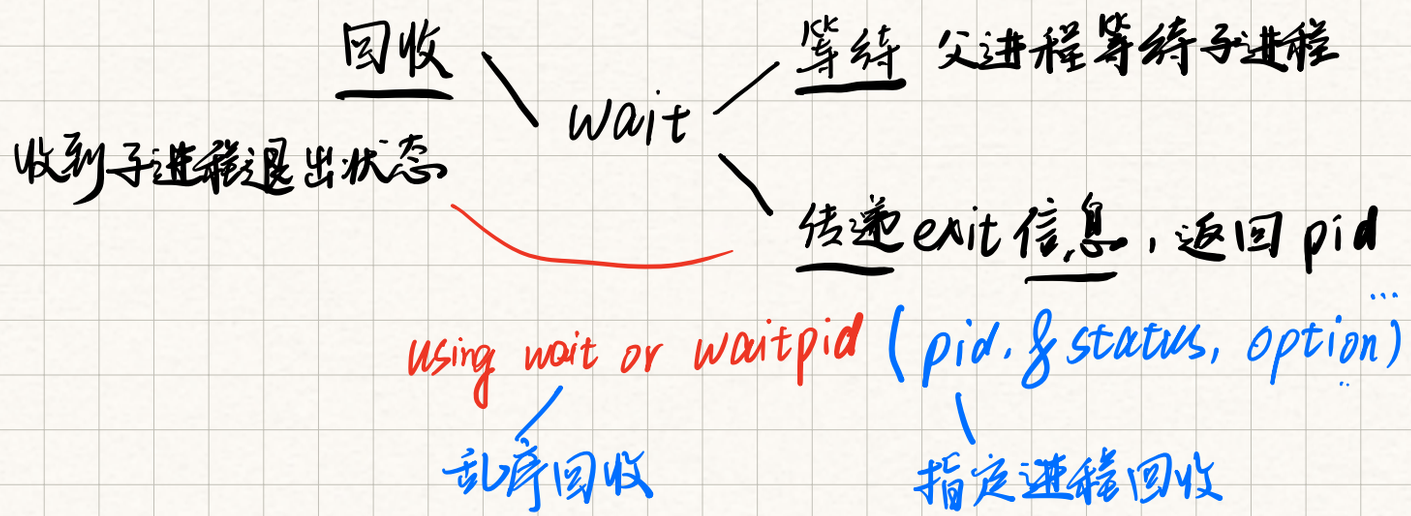
载入echo指令

若exec 正常运行，就与调用它的进程无关了。 错误就返回调用它的进程

fork+exec 组合， 1) shell 执行 fork

2) fork 产生的子进程调用exec

常见的 Unix 程序调用风格


回收
收到子进程退出状态
wait
等待 父进程等待子进程
传递exit信息，返回pid
using wait or waitpid (pid, &status, option)
乱序回收
指定进程回收

僵尸进程与孤儿进程

1. 正常的例子：父进程 fork 子进程，子进程 exit，父进程 wait 回收了，就算是圆满了。

2. 要知道的：父进程退出与否本来是不影响子进程的运行的

3. 回收进程的原因：进程中止后还消耗着系统资源 eg：返回特态、变量表等

4. 僵尸进程：子进程终止但父进程未回收子进程 → 杀死父进程，子进程将被 init 回收

5. 孤儿进程：父进程终止但未回收子进程 → 孤儿进程由 Init 进程“收养”

pipe：0)管道。一段缓冲区，用于进程间通信。

1)创建：int p[2]; pipe(p);

read 读出 ← p[0] - p[1] ← write 写入

2) p[0]为管道写入端，p[1]为管道读出端

3)

```
/*
打开文件就是 进程为文件分配一个文件描述符
那这里 返回两个文件描述符，用以描述管道的读写端
就可以认为是进程打开了管道的 读端文件和写端文件

不用的话就要关掉，用完之后也要关掉
*/
```

好

通信时一个进程一般只持有一边，所以会先关掉一个，

使用完之后，持有的那一个也关闭。

4)阻塞：read 和 write 均有堵塞。

pipe内无内容，则 read 堵塞；pipe 内满了，则 write 堵塞。

dup: 用法 eg: int fd; fd = dup(1).

参数是已有的一个文件描述符，实质是为这个文件再分配一个文件描述符

例子中就是 将标准输出有了两个文件描述符，一个1，一个fd.

共享文件偏移