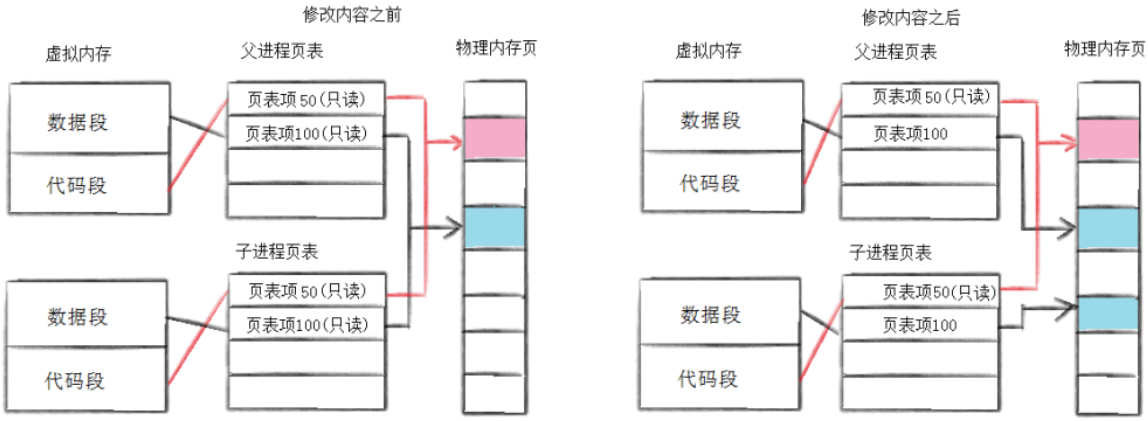
进程控制：

1. 创建 fork

> 操作系统OS 在创建进程的时候
>
> 1. 进程pcb
> 2. 地址空间
> 3. 页表
> 4. 加载程序的代码和数据进入内存
> 5. 开始调度

写时拷贝 and fork



fork常规用法

> - 一个父进程希望复制自己，使父子进程同时执行不同的代码段。例如，父进程等待客户端请求，生成子进程来处理请求。
> - 一个进程要执行一个不同的程序。例如子进程从fork返回后，调用exec函数。

fork调用失败的原因

> - 系统中有太多的进程
> - 实际用户的进程数超过了限制

2. 终止 exit

> 进程的退出分为三种情况
>
> 1. 代码执行完毕，结果正确
> 2. 代码执行完毕，结果错误，错误的原因->进程的退出码（退出信息）strerror
> 3. 代码未执行完毕，进程出现了异常 ->进程收到了异常信号 -> 异常信号具有编号->编号能确定异常原因

使用一个进程的目的是让进程完成任务，当进程结束了，我们需要知道进程的退出情况，无论是正常退出还是异常退出，【**退出码，退出信号**】帮助我们快速了解进程退出的情况。

3. 等待 wait wait_pid

> 1. 等待的目的： 避免进程变成 僵尸进程， 等待回收子进程的资源
> 2. 查看 子进程退出的信息情况

```
pid_t wait(int *wstatus); //返回值 pid > 0 成功回收的子进程的pid
pid_t waitpid(pid_t pid, int *wstatus, int options);
//参数         pid -1 全部   位图输出型参数   是否阻塞
              //指定pid
```

4. 替换