1.

进程是正在执行的一个程序或命令，每个进程都是一个运行的实体，都有自己的地址空间，并占用一定的系统资源。由于UNIX系统是一个多进程的操作系统，所以每一个进程都是独立的，都有自己的权限及任务，所以当某一进程失败时并不会导致别的进程失败。对进程进行管理和控制可以判断服务器的健康状态，某个程序或命令的运行情况，杀死无法正常关闭的进程，并且查看所有的进程。

2.

A产生了32个进程；B只产生了6个进程。

代码段A在执行过程中，fork（）所产生的子程序还能继续进行for循环，继续产生子程序，以此类推直到循环结束。而代码段B只有父进程能够继续调用fork（）产生子进程，子进程则直接结束循环不再产生新的子进程。原因在于fork（）在返回时会返回两个值，一个大于0，代表父进程；另一个等于0，代表子进程，所以子进程直接break；跳出循环。

3.

僵尸进程是当子进程比父进程先结束，而父进程又没有回收子进程，释放子进程占用的资源，此时子进程将成为一个僵尸进程。如果父进程先退出 ，子进程被init接管，子进程退出后init会回收其占用的相关资源。

wait（）函数原型为：pid\_t wait(int\* status)，是一个阻塞函数。在正常情况下会被挂起，直到检测到一个僵尸子进程，wait（）会将其回收。如果返回值大于0，返回的是子进程对应的pid，回收成功；如果返回值为-1，回收失败，已经没有子进程可以回收了。

4.

信号能将信息传入进程中，控制进程的执行情况，并且可以在软件层面上模拟中断的进行，使其运行更为正常。

5.

信号产生和传递之间的时间间隔内，称此信号是未决的。一个已经产生的信号，但是还没有传递给任何进程，此时该信号的状态就称为未决状态。用于解决信号在传递到进程太过密集时，会由于进程正在处理其它信号而导致某些信号未被处理的问题；因为信号的未决状态给了信号足够长的等待时间，使其能够在进程停止阻塞状态后再控制进程。

6.

打开输入文件并初始化帧;创建并初始化共享内存和信号量;然后每次读取一帧，用信号量获取共享内存的权限后，将读取的帧写入共享内存，再释放共享内存的权限。当处理完所有的帧后，设置结束标志，并释放相关的资源。打开输出文件并初始化帧;获取共享内存和信号量;每次得到共享内存的权限后，从共享内存中读取一帧并获得结束标志进行帧处理，释放共享内存的权限。直到结束标志为真。最后释放相关的资源。reader和writer通过信号通信必须获取对方的进程号，可利用共享内存保存双方的进程号。reader和writer运行的顺序不确定，可约定先运行的进程创建共享内存并初始化。利用pause, kill, signal等函数可以实现该程序。