1.

进程是一个二进制程序的执行过程，是系统进行资源分配和调度的基本单位。进程可以使多个程序并发的执行，提高了系统的资源利用效率，并且可以对程序进行有效的管理和调度。

2.

第一个程序最终的结果是：产生了31个子进程。第二个程序的结果是：产生了5个子进程。

原因：在子进程中，fork函数的返回值是0.第二个程序中的判断语句说明如果是子进程，则直接会跳出循环，不会再执行fork函数，产生新的子进程。故只有父进程才能产生新的子进程。第一个程序则没有这样的约束，父进程和其产生的子进程都可以产生新的子进程。

3.

我认为僵尸进程是已终止，但是没有被正确清除，仍然占有系统一部分资源的进程。其只在进程列表中保留一个位置，记载进程的退出状态等信息供父进程收集。当进程调用wait()函数时，其会进入阻塞状态直到子进程变为僵尸进程。wait()函数捕获到该子进程的退出信息时，才会转为运行态，回收子进程资源并返回。若子进程没有变为僵尸进程，wait()函数会使进程一直阻塞。若当前进程有多个子进程，只要捕获到一个子进程的退出信息，wait()函数就会返回，并使进程恢复运行。

4.

信号是一种进程通讯机制。其本质上是软件层次对中断机制的一种模拟，用于提醒进程某件事情已经发生。信号是发送给进程的特殊异步消息。当进程接受到信号时会立即处理，此时并不需要完成当前函数调用甚至当前代码行，这些可能会被立即中断，转而去处理信号。

5.

未决状态：当发送的信号被阻塞，无法到达进程，内核就会该信号的状态设置为未决。

对进程来说，若信号发送的过于密集，即在信号处理的过程中收到其他信号，那么进程会将收到的信号丢弃。对于信号的发送方来说，应该发送的信号已经发送，自然不会再发送信号。对于信号的接受方来说，其未对信号做出应有的处理。未决信号和阻塞就是来解决信号不可靠问题的。

6.

我的方法是：首先设置两个信号量signal\_w,signal\_r。signal\_w和signal\_r分别控制是否能写与是否能读。其中signal\_w的初始值为1，signal\_r的初始值为0.在写进程中，首先将signal\_w的值减1，然后向共享内存中写入内容。写完之后，将signal\_r加1.对于读进程，其只有等到写进程执行完之后才能进入读操作，此时先将signal\_r减1，然后进行读操作，最后将signal\_w加1.