1.进程可以在复杂的操作系统中区分各个运行程序，使得操作系统能够协调好各个程序间的关系，提高整体的工作效率，也方便对错误进行针对调试与保存。

2.区别：代码段A执行时所创建的子进程数要比代码段B中多得多。

解释：代码段B的每次循环中，都通过fork()创建子进程，子进程从这一语句起继续运行，由于fork()函数对子进程的返回值是0，子进程的循环在此时通过break语句结束，然后子进程结束；父进程继续进行循环。而在代码段A中，父进程和每个通过fork()函数创建出的子进程都会在pid=fork()语句之后继续执行循环，直到循环结束后结束进程，在这一过程中，每个子进程在新的循环中也会通过fork()创建子进程，新的子进程同理，因而代码段A在执行时会产生数量相当多的进程。

3.已经执行完毕，但是仍然占据空间，没有被释放的进程是僵尸进程。

调用wait()函数的进程会被挂起，成为阻塞态。直到捕捉到（一个）成为僵尸进程的子进程，此时wait()函数会回收僵尸子进程占用的资源，该子进程彻底结束，然后wait()函数返回，原进程转为运行态。

4.信号是在软件层次上对硬件中断的一种模拟，用于提醒进程某件事情已经发生。信号被用于进程间通信，进程会根据信号执行一定的操作。

5.信号刚刚产生，还没有到达进程，或者由于某种原因被阻塞而（暂时）无法到达进程，进程就为“未决”状态。

信号的未决状态在其产生、传递、送达和最终处理的过程中提供了缓冲，也为不同信号的优先级提供了设计逻辑。让系统能够协调不同信号的先后处理，提高整体的运行效率。

6.可在共享内存的一段空间中保存两个信号量W和R，W表示内存空间是否可写，R表示内存空间是否可读。通过W和R的搭配表示该内存空间的状态：

W1R1表示既可读又可写，即没有进程正在读或写这一空间

W0R1表示可读不可写，即有进程正在读这一内存空间

W0R0表示既不可读又不可写，即有进程正在写这一内存空间

当进程发出读内存的请求时，若R为1，则同意进程的读请求，进程开始读取内存空间，同时将W置0，此时别的进程可以读这一空间但不可写入。读操作结束后，恢复W为1。

当进程发出写内存的请求时，若W为1，则同意进程的写请求，进程开始向内存空间写入，同时将W和R均置为0，别的进程不能读也不能写这一空间。写操作结束后，恢复W和R为1。

当W或（和）R为0时，若有别的进程向内存发出写或读请求，则将这一进程的请求挂起，该进程成为阻塞状态，直到正在读或写内存的进程结束操作后，W和R恢复为1，然后按照上述规则，依次处理被挂起的请求。