Q1

进程是一个正在运行的程序。Linux操作系统通过PCB（进程控制块）来描述、管理运行中的程序。

Q2

A: 父进程创造子进程，子进程还会再创造子进程，最终总共产生32个进程。

B: 父进程创造子进程，子进程不再创造子进程，最终总共产生6个进程。

Q3

子进程先退出，父进程后退出（父进程先退出则子进程由init接管形成孤儿进程），且父进程对子进程不管不顾（没有调用wait/waitpid），则子进程变成一个僵尸进程。

调用wait()，父进程暂停执行，直到任意一个子进程结束。返回已结束子进程的pid，参数使父进程能够获取子进程的退出状态。

Q4

主要用于进程与进程之间的通讯，用来通知某个进程发生了什么事件。

Q5

一个已经产生但是没有传递给任何进程的信号，它的状态就是未决状态。被阻塞的信号产生时将保持在未决状态，直到进程解除对此信号的阻塞，才执行递达的动作。

使用函数sigprocmask（）阻塞信号的传递，只是延迟信号的到达。信号会在解除阻塞后继续传递。这种情况往往需要在信号程序和其它程序共享全局变量时，如果全局变量的类型不是sig\_atomic\_t类型，当一部分程序恰好读、写到变量的一半发生信号，而信号程序里会改变该信号，那么就会产生混乱。为了避免这种混乱，提供程序的可靠性，你必须在操作这类变量前阻塞信号，操作完成后恢复信号的传递。

信号阻塞也用来处理必须保证连续操作的完整性方面。比如，你需要检测一个标志（可以是sig\_atomic\_t类型），该标志在信号程序中设置，当标志没有设置时可以执行某个操作。假如恰好在检测标志后发生信号，那么信号返回后，程序也会执行这个操作，即使已经设置了标志。这显然会引起程序的不稳定。最好的方法就是在检测标志到执行操作之间阻塞信号的发生。

Q6

一个进程写，一个进程读，对同一段共享内存进行读写，利用两个有名信号量，一个控制能否读，一个控制能否写，读完成后发送控制写的信号量到写进程，写完成后发送控制读的信号量到读进程。