Q1

> 每个概念被创造都有其意义，请简述“进程”这个概念在Linux系统中有什么用途。

- 进程是描述程序执行过程和资源共享的基本单位
- 主要目的：控制和协调程序的执行

进程是一个二进制程序的执行过程。在Linux操作系统中，向命令行输入一条命令，按下回车键，便会有一个进程被启动。但进程并不是程序。进程存在于内存中，占用系统资源，是抽象的。当一次程序执行结束之后，进程随之消失，进程所用的资源被系统回收。

Q2

```
//代码段A
int i;
for(i=0;i<5;i++){
    pid=fork();
}
```

```
//代码段B
int i;
for(i=0;i<5;i++){
    if(pid=fork()==0)
        break;
}
```

> 阅读以上代码段，回答代码段A和代码段B的执行结果有什么区别？并解释为什么会有这样的区别。

为了便于观察父子进程的差别，分别打印father和son，运行结果如下：

- 代码段A测试：

```
#include<stdio.h>
#include<unistd.h>
int main(){
    int i,pid;
    for(i=0;i<5;i++){
        pid=fork();
        if(pid==0) printf("son\n");
        else if(pid>0) printf("father\n");
        else printf("error\n");
    }
}
```

输出：

```
father
son
father
father
father
```

```
father
father
father
father
father
son
son
son
son
father
father
son
son
father
father
son
father
son
father
son
son
father
father
son
son
son
son
son
father
father
father
son
father
father
father
son
son
father
son
father
son
father
son
son
son
father
father
son
father
son
son
son
father
father
son
son
son
```

可以发现，生成了31个父进程、31个子进程

- 代码段B测试：

```
#include<stdio.h>
#include<unistd.h>
int main(){
    int i,pid;
    for(i=0;i<5;i++){
        pid=fork();
        if(pid==0) {printf("son\n"); break;}
        else if(pid>0) printf("father\n");
        else printf("error\n");
    }
}
```

输出：

```
father
father
father
father
father
son
son
son
son
son
```

生成了5个父进程，5个子进程

- 区别的原因：代码段B每当`fork()`生成一次子进程，便结束循环，不继续进行，产生的进程数即为循环的次数，即5。而代码段A中子进程创建后，还会继续当前循环，直至`i=5`跳出循环，故产生的进程数为1+2+4+8+16=31。

Q3

> 用自己的话阐述什么是僵尸进程，并描述进程通过调用`wait()`捕获僵尸态的子进程的过程。

僵尸进程：已经执行完毕、但未被父进程处理回收的子进程。

如当进程调用了`exit()`函数后，该进程并不是马上消失，而是留下一个僵尸进程的数据结构——它几乎放弃进程退出前占用的所有内存，既没有可执行代码也不能被调度，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。若父进程中没有回收子进程的代码，子进程将会一直处于僵尸态。

调用`wait()`函数的进程会被挂起，进入阻塞状态，直到子进程变为僵尸态，`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；

若没有变为僵尸态的子进程，`wait()`函数会让进程一直阻塞。若当前进程有多个子进程，只要捕获到一个变为僵尸态的子进程的信息，`wait()`函数就会返回并使进程恢复执行。

Q4

> 请简述信号在Linux系统中的作用。

- 信号是一种异步的通知机制，用来提醒进程一个事件已经发生。当一个信号发送给一个进程时，操作系统会中断进程正常的控制流程，此时，任何非原子操作都将被中断，进程转而去执行相应的信号处理程序。
- 信号被应用于进程间通信

Q5

> 请简述信号什么时候处于未决状态，并简述信号存在未决状态的作用。

- 未决状态：发送的信号被阻塞，无法到达进程，内核就会将该信号的状态设置为未决。
- 有的时候信号可能因为当前的信号被阻塞，而不能及时地被处理，这时候系统会将该信号保存起来，如在调用`sigpending()`时，会有未决状态信号`SIGQUIT`。直到进程解除对此信号的阻塞，才执行递达的动作。
- 信号需要存在未决状态是因为这样能更完整地表示出信号在产生、递达之间的状态。

Q6

> 请设计一种通过信号量来实现共享内存读写操作同步的方式，文字阐述即可，不需要代码实现。（提示：在写进程操作未完成时，需要防止其他进程从共享内存中读取数据）

对于读者和写者两组并发进程，共享一个文件，当两个或以上的读进程同时访问共享数据时不会产生副作用，但若某个写进程和其他进程（读进程或写进程）同时访问共享数据时则可能导致数据不一致的错误。因此为了解决这个问题，大致思路是：

1. 允许多个读者可以同时对文件执行读操作；
2. 只允许一个写者往文件中写信息；
3. 任一写者在完成写操作之前不允许其他读者或写者工作；
4. 写者执行写操作前，应让已有的读者和写者全部退出。

对于代码，我们可以设置一个公共变量count = 0，表示当前正在读的进程的数量。并设置两个信号量：w = 1，用于写时互斥；r = 1，用于在修改count时互斥。

对于写进程，首先执行P（w），之后执行写操作，最后执行V（w）。

对于读进程，首先，更新count变量：执行P（r），count++，执行V（r）。在count++之前，如果count==0，则执行P（w），阻塞写进程。之后，执行读操作。最后，更新count变量：执行P（r），count--，执行V（r）。在count--之后，如果count==0，则执行V（w），允许写进程。