**1论述长期调度、中期调度和短期调度的差异。**

长期调度频率最低，负责从缓冲池中选择进程加到内存中，应该选择IO密集型和CPU密集型的合理进程组和；短期调度执行频率最高，必须快速，需要从准备执行的进程中选择进程分配CPU；中期调度程序是将进程从内存(或CPU竞争)中移出，从而降低多道程序程度，通过中期调度程序，进程可换出，并在后来可换入，这是为了改善进程组合，或由于过度使用内存从而需要释放内存。 **2内核采取一些动作以便在两个进程之间进行上下文切换，请描述一下。**

切换上下文需要保存当前进程状态和恢复另一个进程的状态，内核将当前进程状态保存到其PCB中，然后加载经调度而要执行的新进程的状态。

**3 给出Scheduler()和Interrupt()原语的处理流程。**

Scheduler()：将进程弹出就绪队列->压入新进程的寄存器，PC 等信息->CPU 开始运行该进程

Interrupt()：挂起进程；OS 保存该进程寄存器的值，PC 等信息；将该进程放入就绪队列 **4 编程题：通过系统调用fork()创建子进程，通过exec()系列中的某个执行另一个独立的程序，场景自设。**

o1.cpp:计算2^29的因数数量。 o2.cpp: 计时。

o1.cpp

1. #include <iostream>
2. #include <unistd.h>
3. #include <signal.h>
5. **int** main()
6. {
7. pid\_t pid = fork();
8. **if**(pid == 0)
9. {
10. execl("o2","o2",NULL);
11. }
12. **else** **if**(pid > 0)
13. {
14. **long** **long**  t = 0;
15. t =  2 << 29;
16. **long** **long** p = t;
17. **long** **long** s = 0;
18. **while**(p != 1)
19. {
20. **long** **long** c = t%p;
21. **if**( c == 0)
22. s++;
23. p--;
24. }
25. kill(pid,SIGTERM);
26. exit(0);
27. }
28. **else**
29. {
30. std::cout << "failed to create a new process."<< std::endl;
31. }
32. **return** 0;
34. }

o2.cpp

1. #include <iostream>
2. #include <unistd.h>
4. **int** main()
5. {
6. **int** t = 0;
7. **while** ( **true** )
8. {
9. sleep(1);
10. t++;
11. std::cout<<"child has lived for " << t << " seconds." << std::endl;
12. }
13. **return** 0;
14. }

输出结果为2s。

**5 有一台应用服务器以Socket编程方式向客户端提供英汉互译词典的服务。请完成服务器端程序编码和一个客户端应用，并分别部署到两台计算机上进行测试。说明：不考虑服务器端词典的容量，用数据库或使用Map在内存中保存少量的英汉词汇对应表。**

使用java实现，文件打包。