1、Describe the actions taken by a kernel to context-switch between processes.

需要保存当前进程状态和恢复另一个进程状态。

1、内核将旧进程状态保存到其PCB之中

2、加载经调度而要执行的新进程的上下文，进程上下文采用进程PCB表示，包括CPU寄存器的值、进程状态和内存管理信息等

3、在上述两个必要步骤之外，上下文切换可能还将进行一些如刷新数据等操作。当活动进程数量超过寄存器组数时，即需要在寄存器和内存之间进行数据复制

2、Using the program shown in following , explain what will be the output at Line A.

#include

#include

#include

int value=8;

int main()

{

pid\_t pid;

       /\* fork a child process \*/

       pid = fork();

if (pid == 0) { /\* child process \*/

              value +=15;

       }

       else { /\* parent process \*/

              /\* parent will wait for the child to complete \*/

              wait(NULL);

              printf(" Parent :value= %d\n",value);/\*LINE A\*/

              exit(0);

       }

}

Line A的输出 ： Parent :value = 8

      因为linux下使用fork()创建子进程时，父进程和子进程共享相同内容的代码段、数据段和用户堆栈，就像父进程把自己克隆了一遍。父进程只复制了自己的PCB块，代码段，数据段和用户堆栈并没有复制一份，而是与子进程共享。

      在子进程不对变量值进行修改时，父子进程共享该内存；只有当子进程在运行中出现写操作时，才会产生中断，并为子进程分配独立的地址空间,即“写入时复制”策略。

      因此在子进程中更改一个全局变量的值时，父进程中相应的变量值不会被改变，两者的地址空间不同，不共享该值

3、ch3-3 Describe the differences among short-term, medium-term, and longterm-scheduling.

1、长期调度：从缓冲池中选择进程，加载到内存以便于执行，控制着多道程序度

2、短期调度：从上述准备执行的进程中选择进程，并分配CPU

3、中期调度：在分时系统中，将选中的进程从内存或CPU中移出，从而降低多道程序度。可进行换出和换入操作

短期调度和长期调度的主要不同之处在于执行频率的不同，长期调度通常不被频繁调用，而短期调度经常被调用。中期调度多用在分时系统之中