

struct TASK tasks0[MAX TASKS];//1000个任务

多任务: 计算机同时执行多段代码 #define MAX_TASKS 1000 /* 最大任务数量 */ #define TASK_GDT0 3 /* 任务块在GDT中的初始位置:从GDT的3号段开始*/ struct TSS32 {//任务状态段,这个在前面已经提到过,这里再介绍一下 // 26个int成员,104字节 int backlink, esp0, ss0, esp1, ss1, esp2, ss2, cr3; //与任务设置相关的信息(任务切换时,除backlink,都不会被写入) int eip, eflags, eax, ecx, edx, ebx, esp, ebp, esi, edi;//32位寄存器;eip任务返回时,找到返回的地址 int es, cs, ss, ds, fs, gs;//16位寄存器 //有关任务设置的信息。任务切换时CPU不写;ldtr = 0; iomap = 0x4000_0000 struct TASK {//任务结构体 //sel用来存储GDT中的编号 //flag表示任务的状态 //tss是任务状态段 int sel, flags; struct TSS32 tss; //任务管理结构体 struct TASKCTL { /* 表示正在运行中的任务数量 */ int running; /* 记录当前正在运行的是哪一个任务 */ struct TASK *tasks[MAX_TASKS];//记录正在运行中的任务的地址

int priority; //这里的优先级实际上是设定了任务切换的时间 多任务的一般方法是运行第一个程序的一段代码,保存工作环境;再运行第二个程序的一段代码,保存环 境; ……恢复第一个程序的工作环境,执行第一个程序的下一段代码……现代的多任务,每个程序的时间分配相

计算机的早期,多任务被称作多道程序(multiprogramming)。多道程序是令CPU一次读取多个程序放入内存,先 运行第一个程序直到它出现了IO操作。因为IO操作慢,CPU需要等待。为了提高CPU利用率,此时运行第二个程 序。即,第n+1个程序得以执行的条件是第n个程序进行IO操作或已经运行完毕。这种方式每个程序的时间分配是 不均等的,很可能第一个程序运行了几个小时而不出现IO操作,故第二个程序没有运行。

在当初,这种情况是令人接受的。人们一次指定运行多个程序,过几个小时或一天后来看运行结果或拿走打印出 来的文件。人们不需要实时获得每个程序的运行情况,只关心运行结果。