**linux 程序加载过程**

2009年02月02日 星期一 21:21

|  |  |
| --- | --- |
| |  | | --- | | 一个进程在内存中主要占用了以下几个部分，分别是代码段、数据段、BSS，栈，堆，等参数。其中，代码、数据、BSS的内容是可执行文件中对应的内容，加载程序并不是把它们的内容从可执行程序中填充到内存中，而是将它们的信息（基地址、长度等）更新到进程控制块（task\_struct）中，当CPU第 一次实际寻址执行的时候，就会引起缺页中断，操作系统再将实际的内容从可执行文件中复制内容到物理内存中。     堆的内容是程序执行中动态分配的，所以加载程序 只是将它的起始地址更新到进程控制块中，执行过程中遇到动态分配内存的操作的时候再在物理内存分配实际的页。参数区在新进程加载的时候要存入环境变量和命令行参数列表。栈在程序加载时候存入的内容就是环境参数列表和命令行参数列表的指针和命令行参数的个数。  1） 申请物理页，填充参数  参数的填充要分两种情况，一种是文件是Shell脚本文件，这个时候不但要把shell文件的命令行参数填充进去，还要解析shell文件头信息，将要传入shell程序的参数也要填充到某部分物理内存内，这样参数区中命令行参数列表的长度就加长了：shell程序参数列表。之后，把shell程序作为新的可执行程序，重新启动加载过程。另一种情况就是二进制的可执行文件，这个时候就把环境参数和命令行参数填充到某部分物理内存就行了。这步之后，就会得到参数页的页指针（这些页在物理内存中）一个二进制文件（包括shell程序本身）的内存I节点。  2） 清理工作：           关闭指定的打开的文件；复位协处理器标志；释放页目录项和页表项,这里需要说明的是和windows很不相同.如果是windows,则会一直保护打开或者运行的文件.直到退出.而linux在加载进参数选项以后,就是放弃保护,可以到其修改和删除.大家可以做个实验.  3） 更新LDT：           更新ldt中代码段的限长（可执行文件头中记录了代码段的长度），更新数据段的限长（64M），更新代码段和数据段的基地址（其实不用更新，在fork中已经将它俩的基地址设置为该进程的起始地址：64×nr（进程号），这里顺便说一句，堆栈段的段基地址也是这儿，只不过ss和ds指向ldt中的同一个段描述符，所以就一起更新了）。  4） 建立参数列表用的物理页与逻辑地址的对应关系          就是将储存参数列表的那些物理页利用页表映射到进程逻辑地址空间的最上面的部分，具体做法就是根据逻辑地址计算出页目录项表中页表项的位置，从而找到（新建）页表，再在页表中建立页与物理地址的对应关系。          建立了对应关系，就相当于把参数列表填充到了进程地址空间中了。  5） 在栈（用户栈）中建立参数表指针表：  进程空间的最上面是参数列表，在它的下面就是栈，栈中要存三个数据：环境参数列表指针、用户参数列表指针、命令行参数个数。  6） 更新task\_struct中代码段、数据段、堆段的尾值；更新堆栈开始指针值；重新设置用户id和组id。  7） 更改上层进程（调用exec的进程）的堆栈内容，使ip指向新进程的代码开始处：  ex.a\_entry；使栈指针指向新进程的栈顶，也就是存放命令行参数的地方。由于新进程的代码段、数据段、堆栈段的段选择符和原来进程都是一样的，都是指向ldt[2]处，所以不必更改栈中cs、fs、ss的值。     以上就是exec的具体工作过程，当它返回的时候，从栈中弹处的新进程的代码地址和堆栈地址，也就转向了新程序进行执行，当然，前面说过，此时代码段核数据段中是没有内容的，cpu是采用“按需加载”的机制。  附:可以看出，这里的“加载”是广义的，不但有“复制”的意思，也有“设置”的意思。参数是从内存（命令行参数、环境参数）或者可执行文件中（shell 脚本中的shell参数）复制到内存中的；程序的代码段、数据段则是在段描述符中设置它们的段信息，并没有实际把能内容复制进来。 | |