### **进程退出和等待进程**

当进程执行完它的工作后，就需要执行退出操作，释放进程占用的资源。ucore分了两步来完成这个工作，首先由进程本身完成大部分资源的占用内存回收工作，然后由此进程的父进程完成剩余资源占用内存的回收工作。为何不让进程本身完成所有的资源回收工作呢？这是因为进程要执行回收操作，就表明此进程还存在，还在执行指令，这就需要内核栈的空间不能释放，且表示进程存在的进程控制块不能释放。所以需要父进程来帮忙释放子进程无法完成的这两个资源回收工作。

为此在用户态的函数库中提供了exit函数，此函数最终访问sys\_exit系统调用接口让操作系统来帮助当前进程执行退出过程中的部分资源回收。我们来看看ucore是如何做进程退出工作的。

首先，exit函数会把一个退出码error\_code传递给ucore，ucore通过执行内核函数do\_exit来完成对当前进程的退出处理，主要工作简单地说就是回收当前进程所占的大部分内存资源，并通知父进程完成最后的回收工作，具体流程如下：

****1.**** 如果current->mm != NULL，表示是用户进程，则开始回收此用户进程所占用的用户态虚拟内存空间；

a) 首先执行“lcr3(boot\_cr3)”，切换到内核态的页表上，这样当前用户进程目前只能在内核虚拟地址空间执行了，这是为了确保后续释放用户态内存和进程页表的工作能够正常执行；

b) 如果当前进程控制块的成员变量mm的成员变量mm\_count减1后为0（表明这个mm没有再被其他进程共享，可以彻底释放进程所占的用户虚拟空间了。），则开始回收用户进程所占的内存资源：

i. 调用exit\_mmap函数释放current->mm->vma链表中每个vma描述的进程合法空间中实际分配的内存，然后把对应的页表项内容清空，最后还把页表所占用的空间释放并把对应的页目录表项清空；

ii. 调用put\_pgdir函数释放当前进程的页目录所占的内存；

iii. 调用mm\_destroy函数释放mm中的vma所占内存，最后释放mm所占内存；

c) 此时设置current->mm为NULL，表示与当前进程相关的用户虚拟内存空间和对应的内存管理成员变量所占的内核虚拟内存空间已经回收完毕；

****2.**** 这时，设置当前进程的执行状态current->state=PROC\_ZOMBIE，当前进程的退出码current->exit\_code=error\_code。此时当前进程已经不能被调度了，需要此进程的父进程来做最后的回收工作（即回收描述此进程的内核栈和进程控制块）；

****3.**** 如果当前进程的父进程current->parent处于等待子进程状态：

current->parent->wait\_state==WT\_CHILD，

则唤醒父进程（即执行“wakup\_proc(current->parent)”），让父进程帮助自己完成最后的资源回收；

****4.**** 如果当前进程还有子进程，则需要把这些子进程的父进程指针设置为内核线程initproc，且各个子进程指针需要插入到initproc的子进程链表中。如果某个子进程的执行状态是PROC\_ZOMBIE，则需要唤醒initproc来完成对此子进程的最后回收工作。

****5.**** 执行schedule()函数，选择新的进程执行。

那么父进程如何完成对子进程的最后回收工作呢？这要求父进程要执行wait用户函数或wait\_pid用户函数，这两个函数的区别是，wait函数等待任意子进程的结束通知，而wait\_pid函数等待进程id号为pid的子进程结束通知。这两个函数最终访问sys\_wait系统调用接口让ucore来完成对子进程的最后回收工作，即回收子进程的内核栈和进程控制块所占内存空间，具体流程如下：

****1.**** 如果pid!=0，表示只找一个进程id号为pid的退出状态的子进程，否则找任意一个处于退出状态的子进程；

****2.**** 如果此子进程的执行状态不为PROC\_ZOMBIE，表明此子进程还没有退出，则当前进程只好设置自己的执行状态为PROC\_SLEEPING，睡眠原因为WT\_CHILD（即等待子进程退出），调用schedule()函数选择新的进程执行，自己睡眠等待，如果被唤醒，则重复跳回步骤1处执行；

****3.**** 如果此子进程的执行状态为PROC\_ZOMBIE，表明此子进程处于退出状态，需要当前进程（即子进程的父进程）完成对子进程的最终回收工作，即首先把子进程控制块从两个进程队列proc\_list和hash\_list中删除，并释放子进程的内核堆栈和进程控制块。自此，子进程才彻底地结束了它的执行过程，消除了它所占用的所有资源。