E02：并发进程（UNIX 进程与中断）

参考答案与说明

1. B
2. B
3. A
4. B
5. A

6. 【参考答案】

如果已知一个进程的 ID 号,可根据该 ID 号在 proc 表中找到该进程的 proc 结构,然后：

（1）根据 proc 结构中 p_flag 是否具有 SLOAD 标志位可判断该进程图象的可交换部分是否在内存。如果在内存，则 p_addr 为进程图像可交换部分在内存的起始地址，如果在盘交换区，则 p_addr 为进程图像可交换部分在盘交换区上的盘块号。

（2）根据 proc 结构中 p_texp 找到该进程代码段的 TEXT 结构，并根据其中 x_ccount 的值，判断代码段是否在内存。如果 x_ccount>=1，则代码段在内存，x_caddr 为其内存起始地址；如果 x_ccount==0，则代码段在盘交换区，x_daddr 为其在盘交换区的起始盘块号。

如果当前进程图象的可交换部分在内存，且需要换到盘交换区，则：

（1）如果 x_ccount-1=0，则将代码段同时换出；

（2）如果 x_ccount-1>0，则将代码段不换出。

7. 【参考答案】

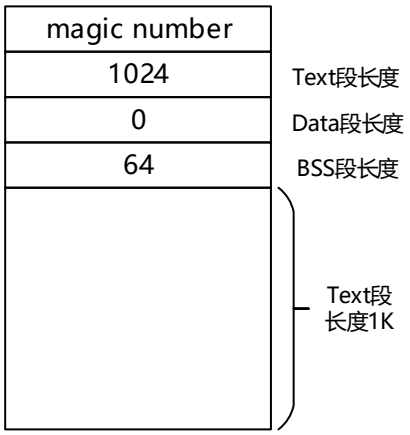
多道计算机系统中需要中断的原因主要有：（1）保证了 CPU 和设备之间的并行操作；（2）提供了进程执行内核代码的机会；（3）多道程序并发的硬件基础。

8. 【参考答案】

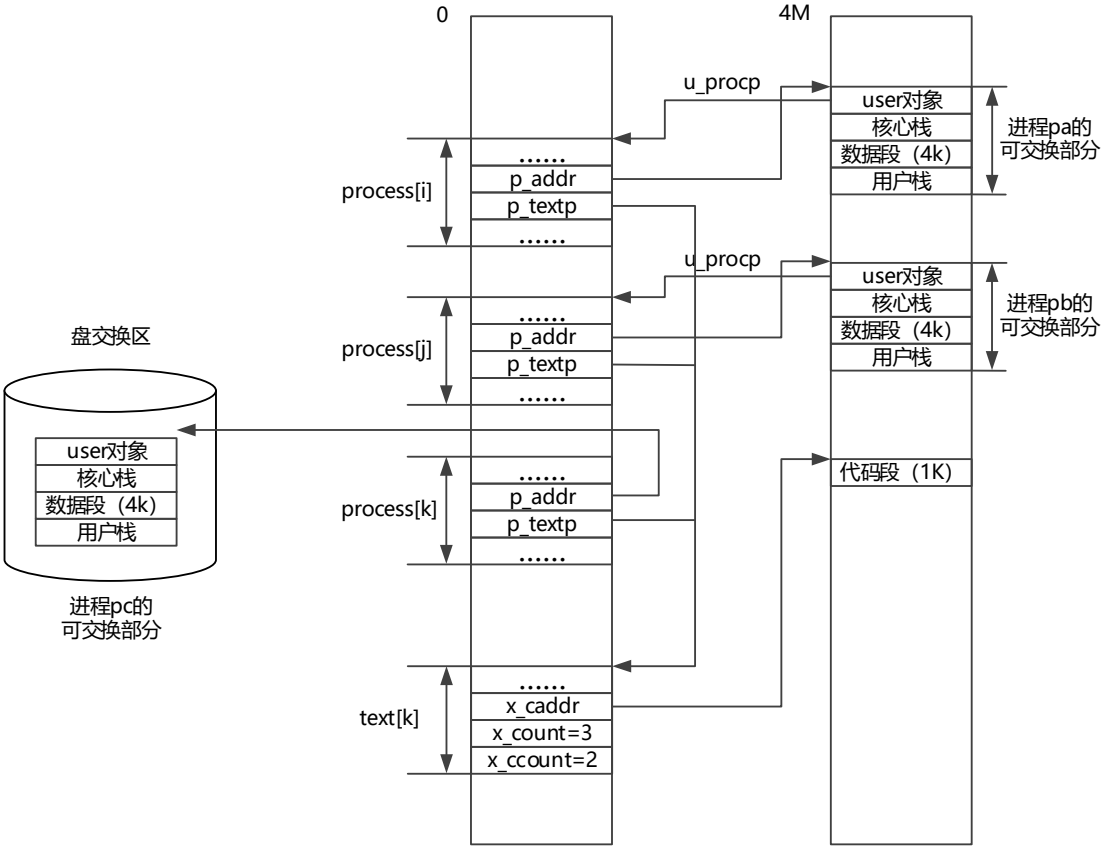
反应了 UNIX 进程管理中，内核不可抢占的思想。

9. 【参考答案】

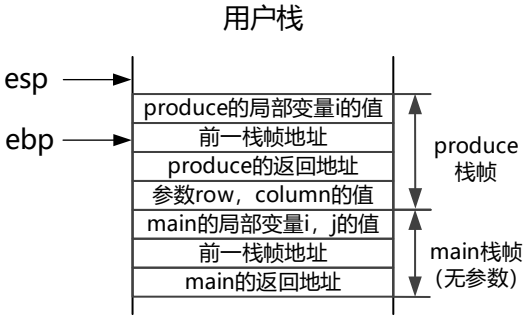
（1）可执行文件的结构如下图所示：



（2）进程 pa，pb，pc 的图象如下图所示：



（3）当 pa 执行到 produce 函数的 for 语句时，用户栈的构成如下图所示：

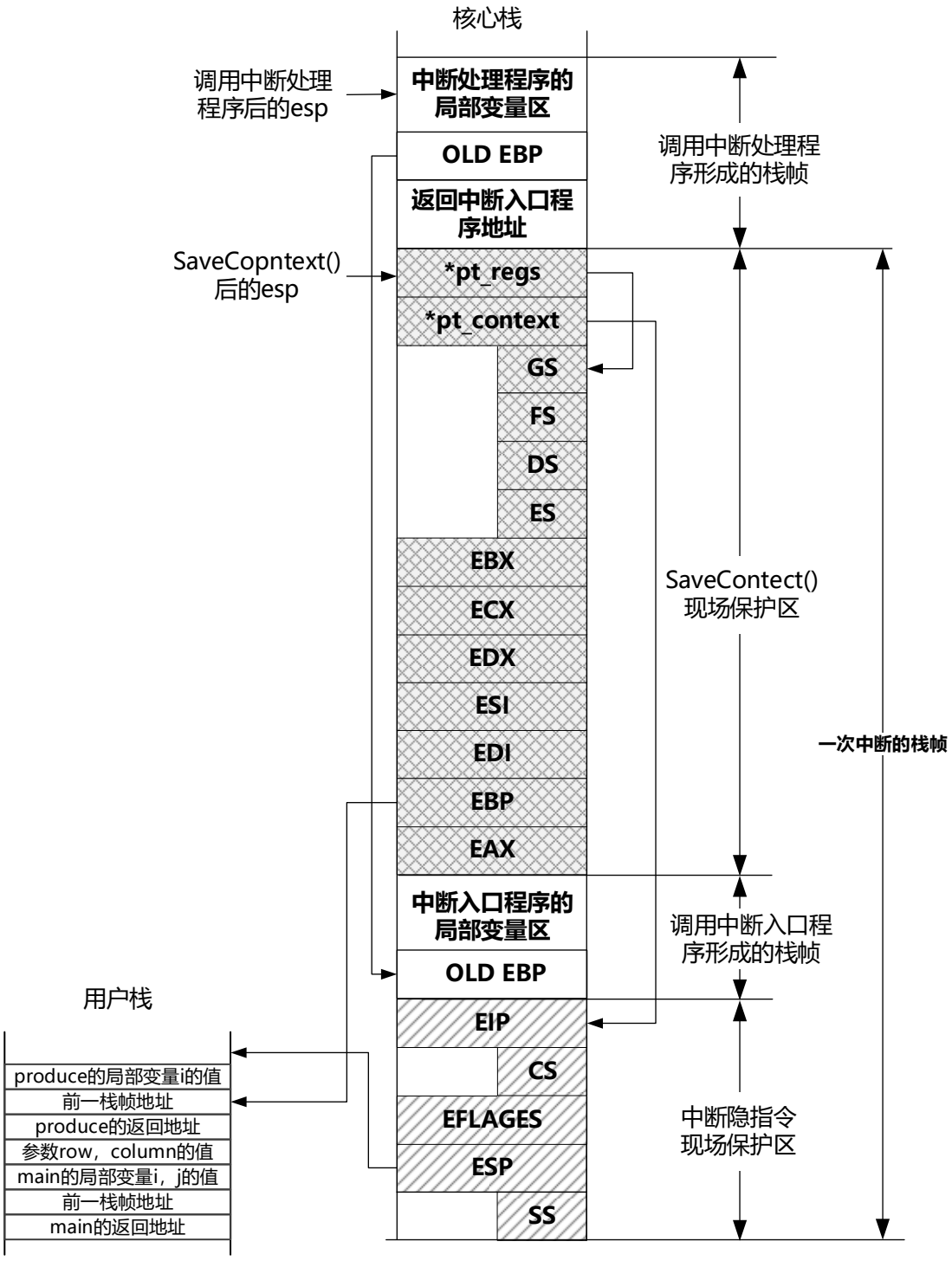


因为此时进程未进入核心态执行，所以核心态为空。

（4）此后进程 pa 响应中断，处理中断和中断返回后，核心栈的变化情况如下：

响应中断前，核心栈为空；

响应中断后，核心栈的构成如下图所示，图中示出了每一步操作过程中核心栈的变化。中断返回后，核心栈为空。


核心栈
调用中断处理程序后的esp
中断处理程序的局部变量区
OLD EBP
返回中断入口程序地址
调用中断处理程序形成的栈帧
SaveCopntext()后的esp
*pt_regs
*pt_context
GS
FS
DS
ES
EBX
ECX
EDX
ESI
EDI
EBP
EAX
SaveContect()现场保护区
一次中断的栈帧
中断入口程序的局部变量区
OLD EBP
调用中断入口程序形成的栈帧
EIP
CS
EFLAGES
ESP
SS
中断隐指令现场保护区
用户栈
produce的局部变量i的值
前一栈帧地址
produce的返回地址
参数row，column的值
main的局部变量i，j的值
前一栈帧地址
main的返回地址