第二章

并发进程

方　钰

主 要 内 容

2.1 进程基本概念

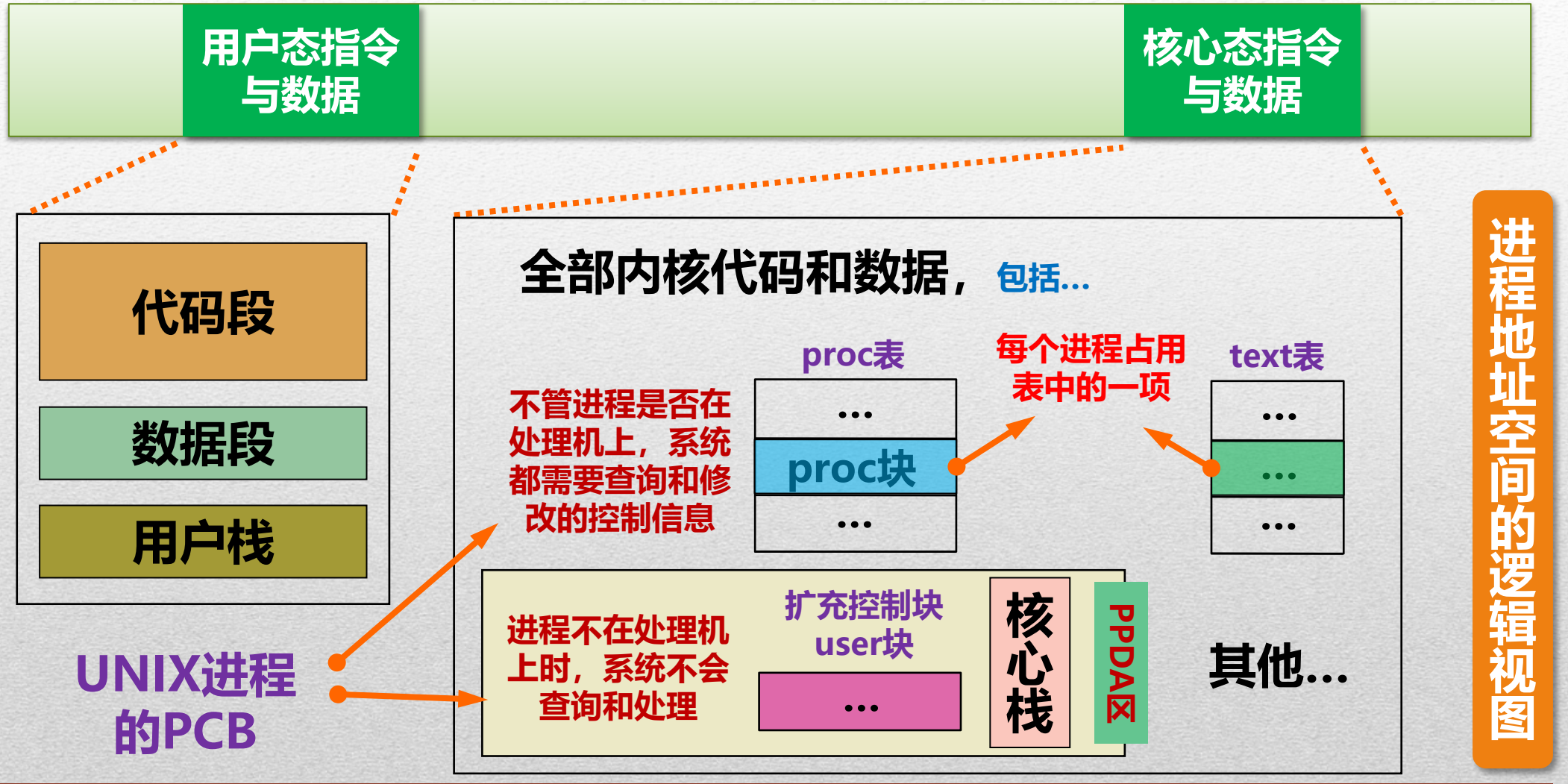
2.2 UNIX的进程（续）

2.3 中断的基本概念及UNIX中断处理

2.4 处理机调度与死锁

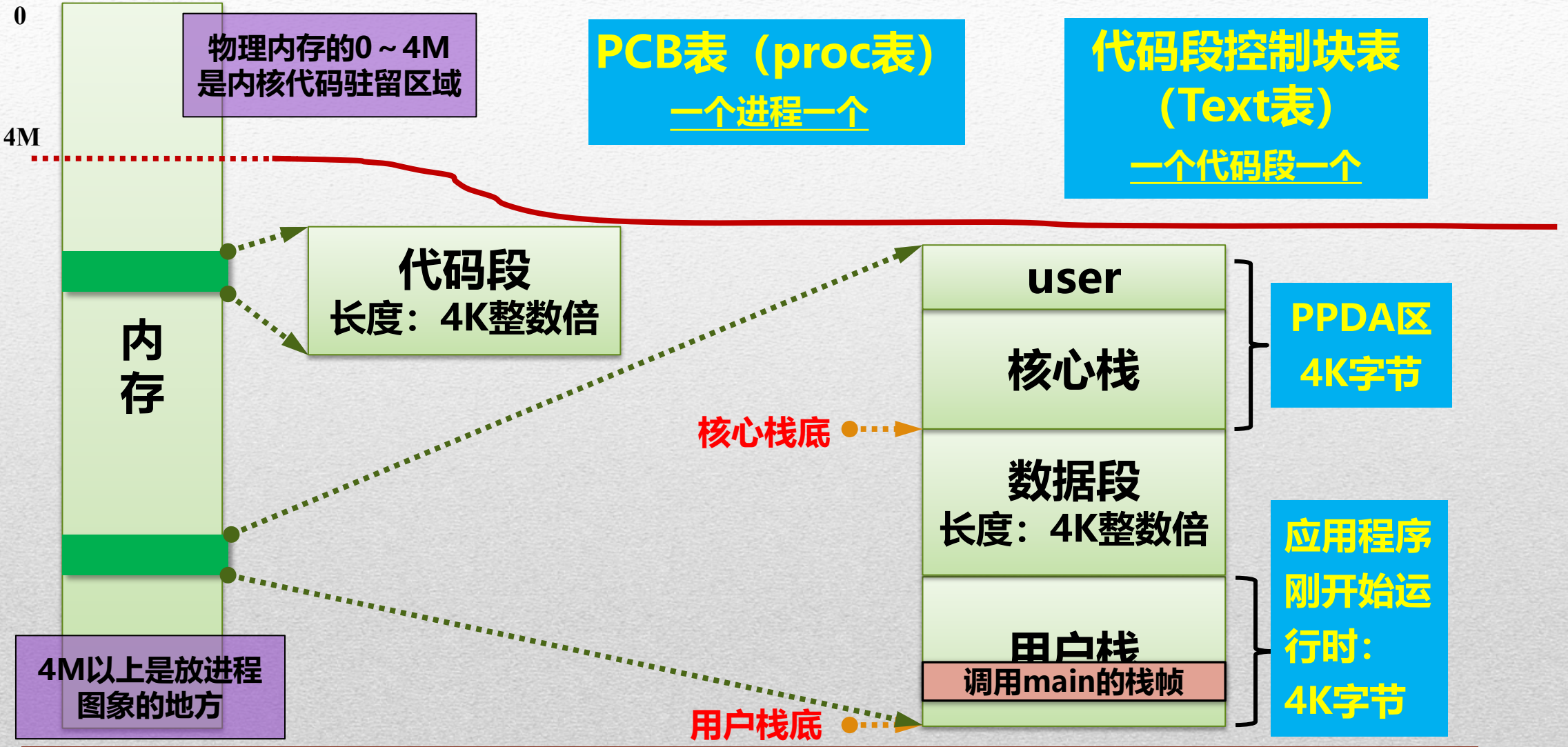
2.5 进程通信机制

UNIX中进程核心态地址空间的构成

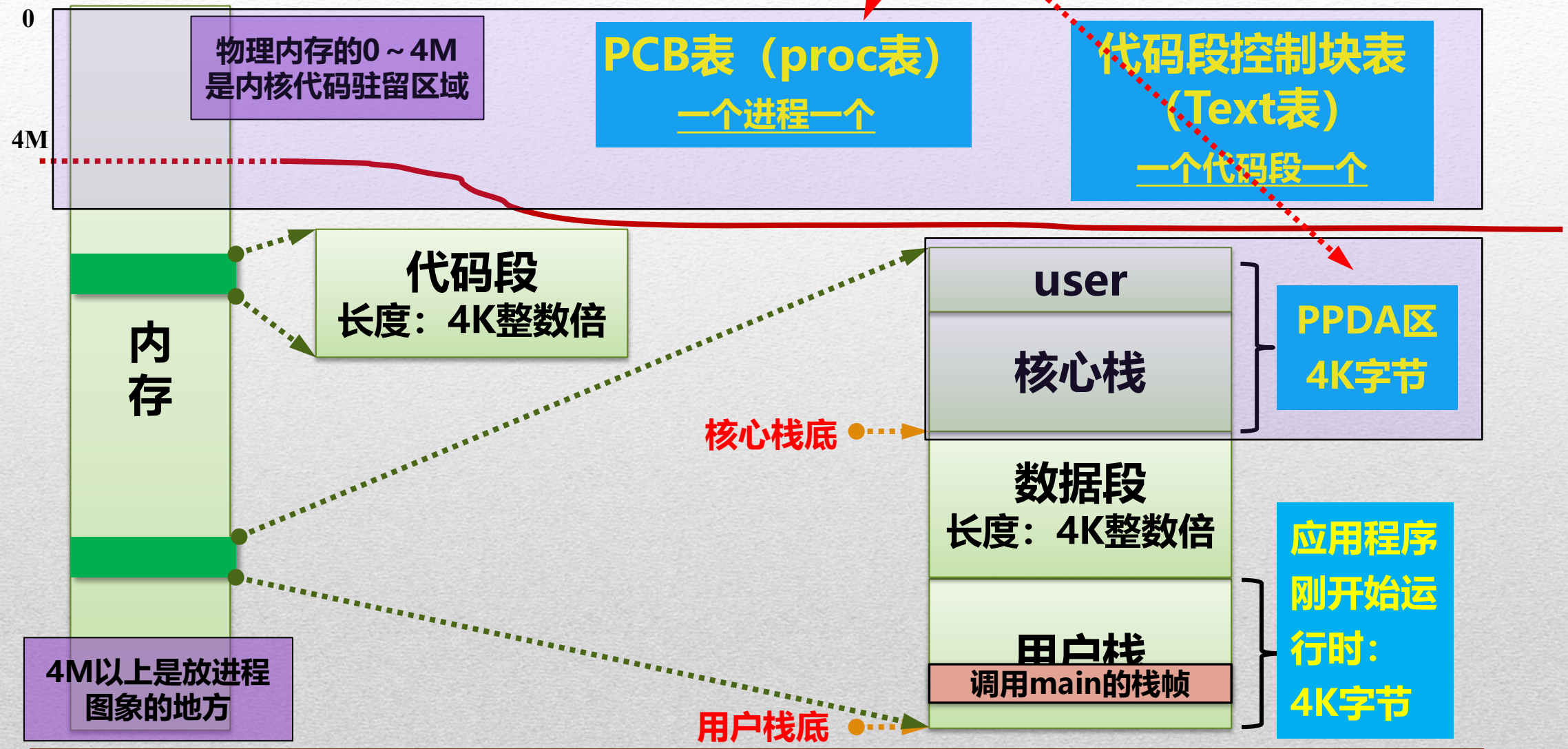


UNIX中进程的构成（进程图象）

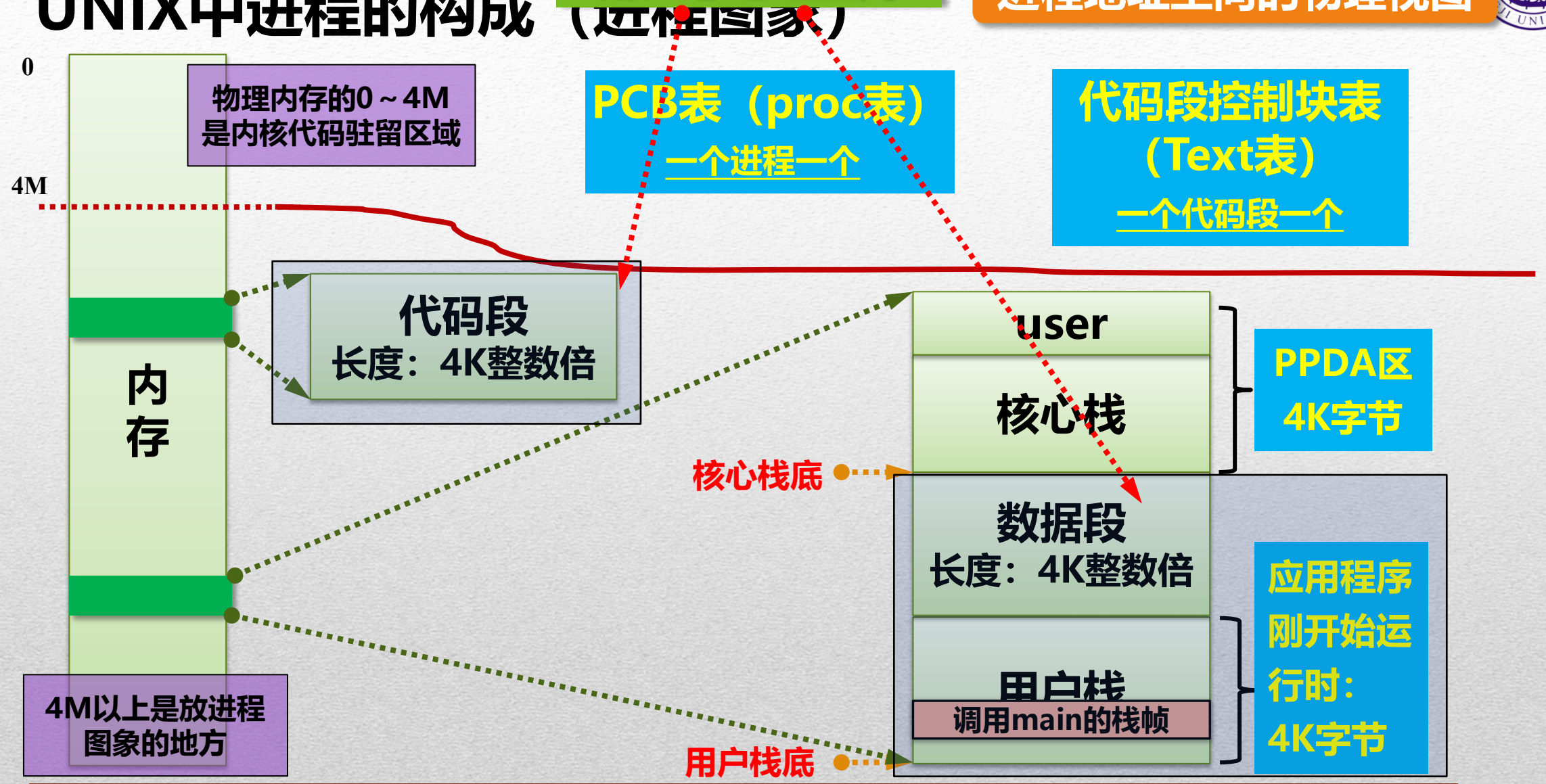
进程地址空间的物理视图



UNIX中进程的构成（进程图象）



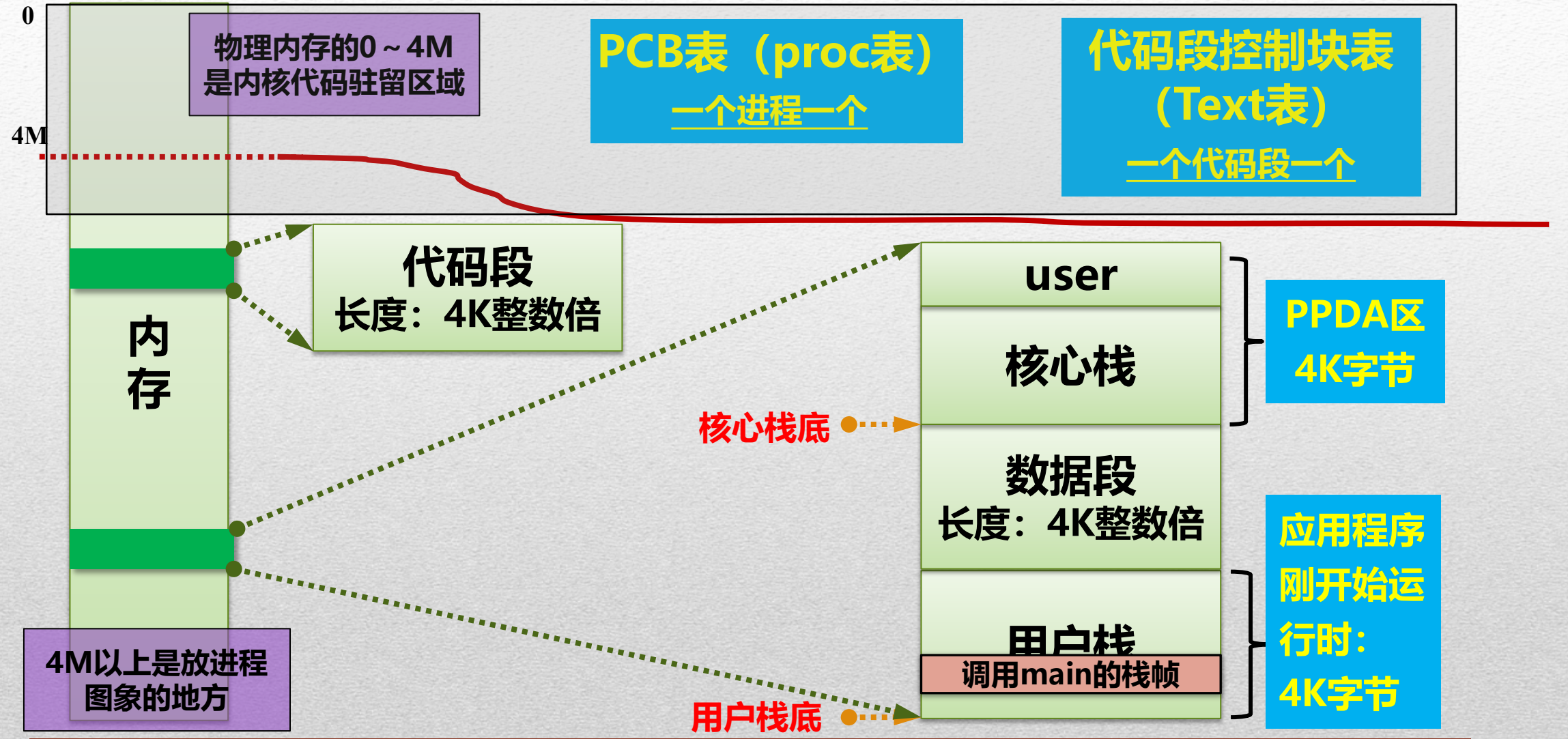
UNIX中进程的构成（进程图象）



常驻内存部分

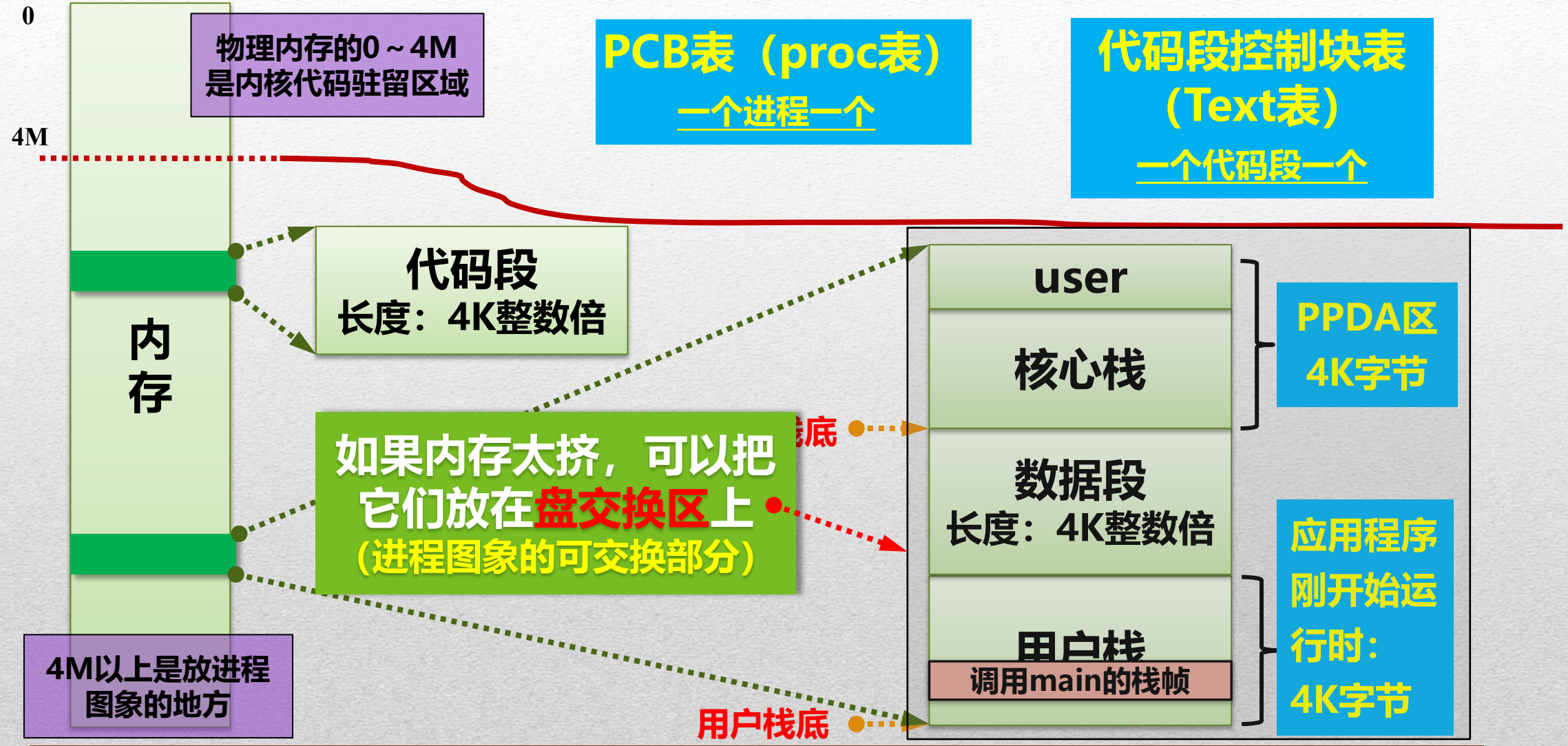
进程地址空间的物理视图

UNIX中进程的构成（进程图象）



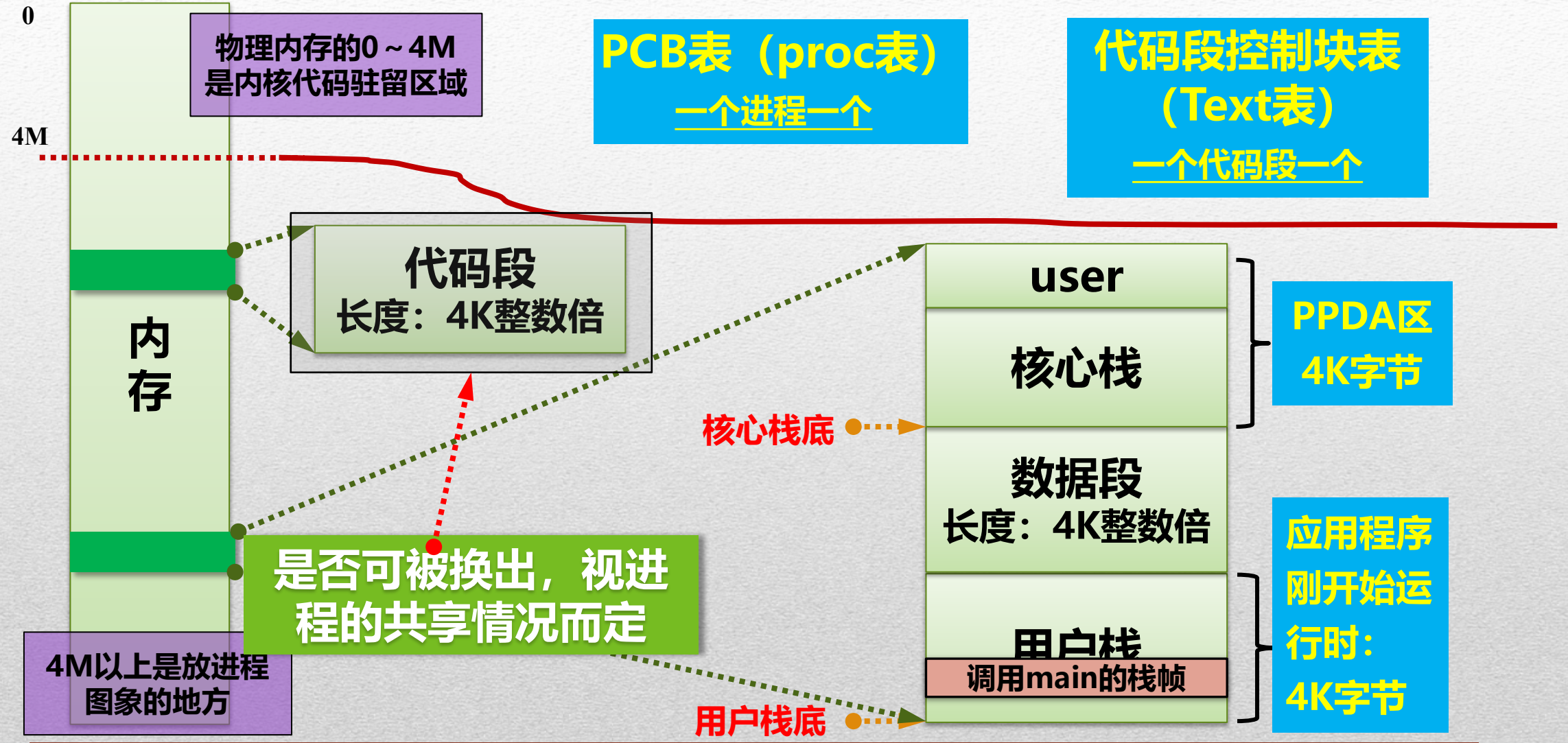
UNIX中进程的构成（进程图象）

进程地址空间的物理视图

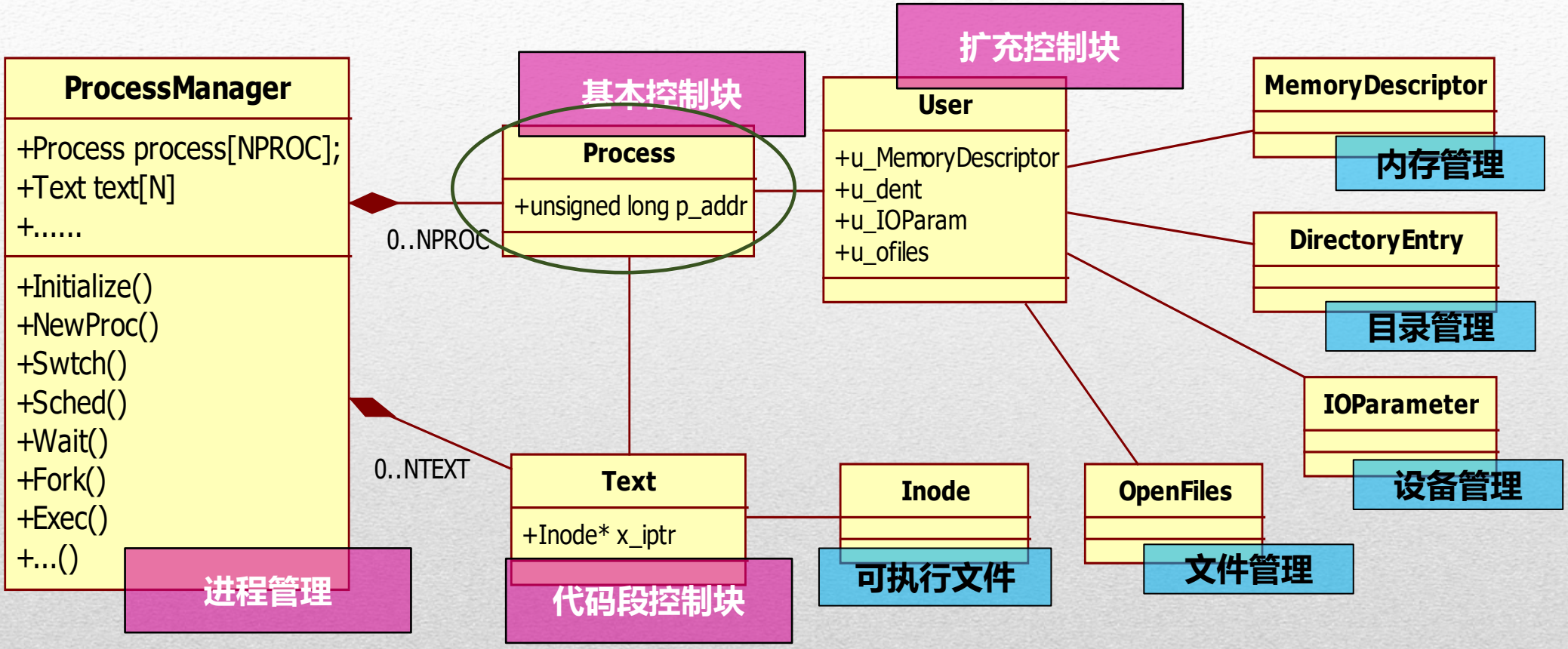


UNIX中进程的构成（进程图象）

进程地址空间的物理视图



UNIX V6++进程图象的实现



所有进程管理相关的类结构

Process类

	名称	类型	含义
进程标识	p_uid	short	用户ID
	p_pid	int	进程标识数，进程编号
	p_ppid	int	父进程标识数
进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符
进程调度相关信息	p_stat	ProcessState	进程当前的调度状态
	p_flag	int	进程标志位，可以将多个状态组合
	p_pri	int	进程优先数
	p_cpu	int	cpu值，用于计算p_pri
	p_nice	int	进程优先数微调参数
	p_time	int	进程在盘交换区上（或内存内）的驻留时间
	p_wchan	unsigned long	进程睡眠原因
信号与控制台终端	p_sig	int	进程信号
	p_ttyp	TTy*	进程tty结构地址

Process类

	名称	类型	含义
进程标识	p_uid	short	用户ID
	p_pid	int	进程标识数，进程编号
	p_ppid	int	父进程标识数
进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符
进程调度相关信息	p_stat	ProcessState	进程当前的调度状态
	p_flag	int	进程标志位，可以将多个状态组合
	p_pri	int	进程优先数
	p_cpu	int	cpu值，用于计算p_pri
	p_nice	int	进程优先数微调参数
	p_time	int	进程在盘交换区上（或内存内）的驻留时间
	p_wchan	unsigned long	进程睡眠原因
信号与控制台终端	p_sig	int	进程信号
	p_ttyp	TTy*	进程tty结构地址

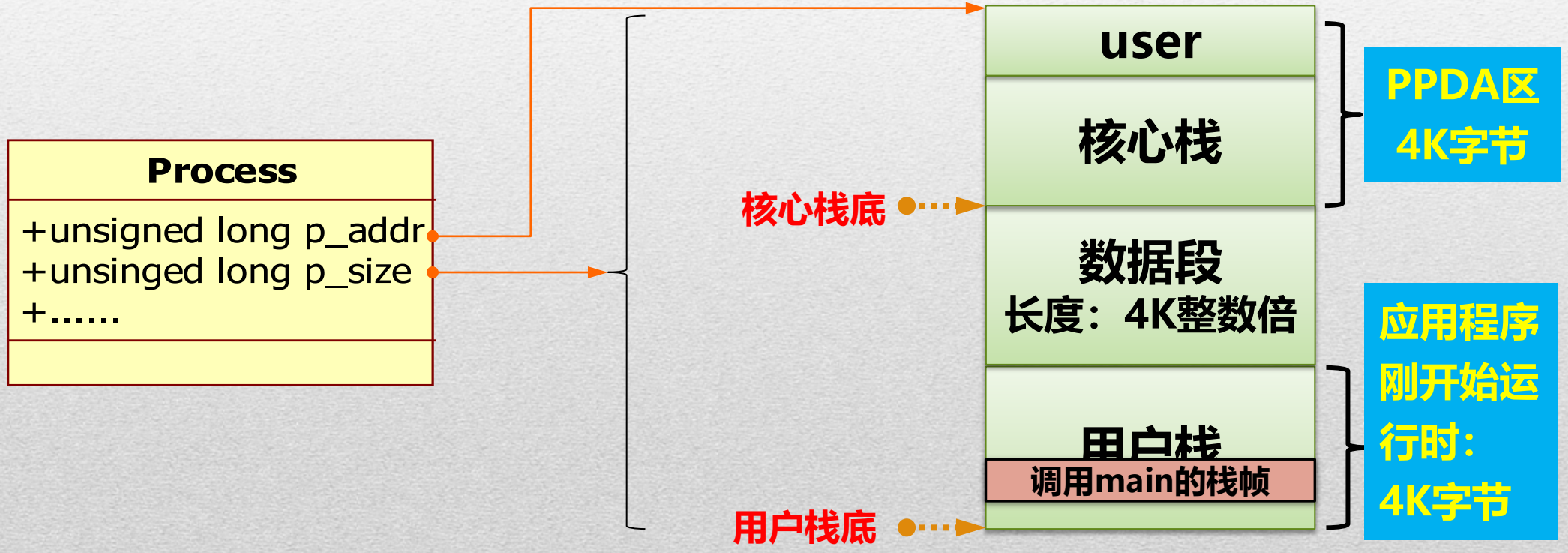
Process类

	名称	类型	含义
进程标识	p_uid	short	用户ID
	p_pid	int	进程标识数，进程编号
	p_ppid	int	父进程标识数
进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址（物理地址）
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符
进程调度相关信息	p_stat	ProcessState	进程当前的调度状态
	p_flag	int	进程标志位，可以将多个状态组合
	p_pri	int	进程优先数
	p_cpu	int	cpu值，用于计算p_pri
	p_nice	int	进程优先数微调参数
	p_time	int	进程在盘交换区上（或内存内）的驻留时间
	p_wchan	unsigned long	进程睡眠原因
信号与控制台终端	p_sig	int	进程信号
	p_ttyp	TTy*	进程tty结构地址

Process类

进程图象在内存中的位置信息

p_addr	unsigned long	ppda区在物理内存中的起始地址（物理地址）
p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
p_textp	Text *	指向该进程所运行的代码段的描述符



Process类

进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址（物理地址）
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符

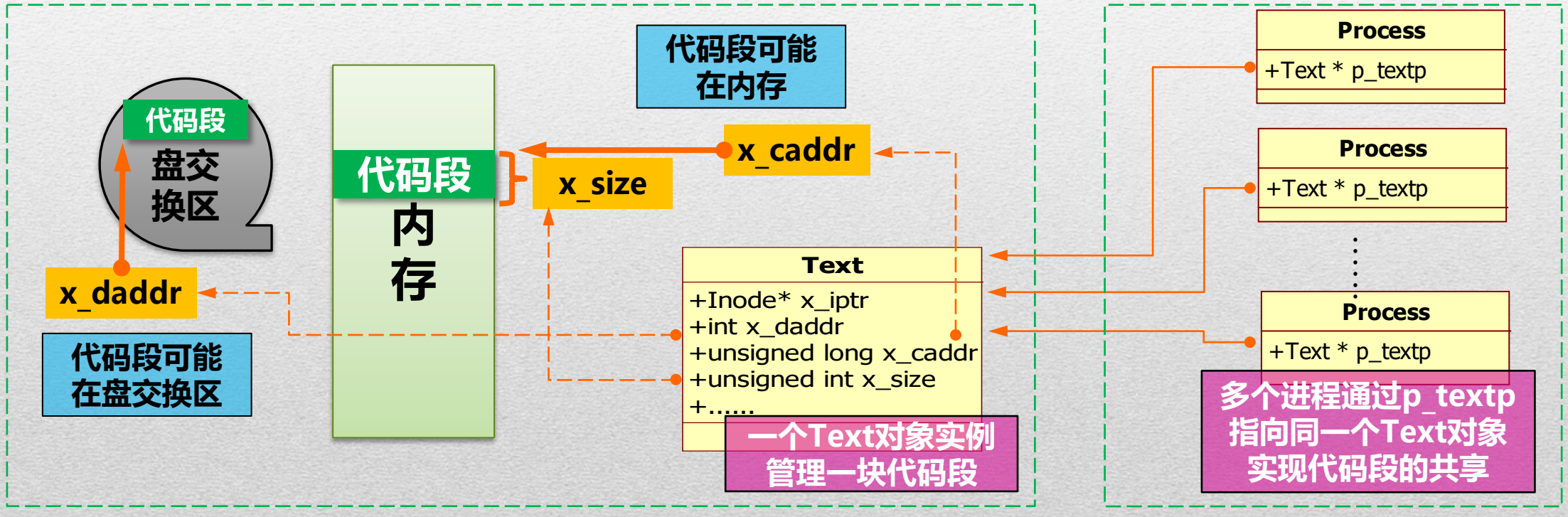
Text类

	名称	类型	含义
位置相关	x_daddr	int	代码段在盘交换区上的地址
	x_caddr	unsigned long	代码段在物理内存中的起始地址，以字节为单位（物理地址）
	x_size	unsigned int	代码段长度，以字节为单位
	*x_iptr	Inode	内存inode地址（用于相应的可执行文件的管理）
共享代码的进程数	x_count	unsigned int	共享该代码段的进程数
	x_ccount	unsigned short	共享该代码段，且图像在内存的进程数

Process类

进程图象在内存中的位置信息

p_addr	unsigned long	ppda区在物理内存中的起始地址（物理地址）
p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
p_textp	Text *	指向该进程所运行的代码段的描述符



Process类

进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址（物理地址）
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符

Text类

	名称	类型	含义
位置相关	x_daddr	int	代码段在盘交换区上的地址
	x_caddr	unsigned long	代码段在物理内存中的起始地址，以字节为单位（物理地址）
	x_size	unsigned int	代码段长度，以字节为单位
	*x_iptr	Inode	内存inode地址（用于相应的可执行文件的管理）
共享代码的进程数	x_count	unsigned int	共享该代码段的进程数
	x_ccount	unsigned short	共享该代码段，且图像在内存的进程数

Process类

	名称	类型	含义
进程标识	p_uid	short	用户ID
	p_pid	int	进程标识数，进程编号
	p_ppid	int	父进程标识数
进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符
进程调度相关信息	p_stat	ProcessState	进程当前的调度状态
	p_flag	int	进程标志位，可以将多个状态组合
	p_pri	int	进程优先数
	p_cpu	int	cpu值，用于计算p_pri
	p_nice	int	进程优先数微调参数
	p_time	int	进程在盘交换区上（或内存内）的驻留时间
	p_wchan	unsigned long	进程睡眠原因
信号与控制台终端	p_sig	int	进程信号
	p_ttyp	TTy*	进程tty结构地址

Process类

	名称	类型	含义
进程标识	p_uid	short	用户ID
	p_pid	int	进程标识数，进程编号
	p_ppid	int	父进程标识数
进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符
进程调度相关信息	p_stat	ProcessState	进程当前的调度状态
	p_flag	int	
	p_pri	int	
	p_cpu	int	
	p_nice	int	
	p_time	int	
	p_wchan	unsigned	
信号与控制台终端			

```
enum ProcessState
{
    SNULL= 0,     /* 未初始化空状态
    SSLEEP = 1,   /* 高优先权睡眠状态
    SWAIT = 2,    /* 低优先权睡眠状态
    SRUN = 3,     /* 运行、就绪状态
    SIDL = 4,     /* 进程创建时的中间状态
    SZOMB = 5,    /* 进程终止时的中间状态
    SSTOP = 6     /* 进程正被跟踪
};
```

p_stat一定为7个状态其中之一！

Process类

名称	类型	含义
进程标识	p_uid	
	p_pid	
	p_ppid	
进程图象在内存中的位置信息	p_addr	
	p_size	
	p_textp	
进程调度相关信息	p_stat	ProcessState	进程当前的调度状态
	p_flag	int	进程标志位，可以将多个状态组合
	p_pri		进程优先数
信号与控制台终端			

5	4	3	2	1	0
STWED	STRC	SSWAP	SLOCK	SSYS	SLOAD

```
p_flag = SLOAD | SLOCK → p_flag=101;  //二进制
p_flag &= ~SLOCK; → p_flag=001;
p_flag &= ~SLOAD; → p_flag=000;       //二进制
if (p_flag & SLOAD) != 0)
```

```
enum ProcessFlag  /* 进程标志位（用于进程图像换进换出）
{
    SLOAD       = 0x1,    /* 进程图像在内存中
    SSYS        = 0x2,    /* 系统进程图像，不允许被换出
    SLOCK       = 0x4,    /* 含有该标志的进程图像暂不允许换出
    SSWAP       = 0x8,    /* 该进程被创建时图像就在交换区上
    STRC        = 0x10,   /* 父子进程跟踪标志，UNIX V6++未使用到
    STWED       = 0x20    /* 父子进程跟踪标志，UNIX V6++未使用到
};
```

p_flag可以是6个值的合理组合！

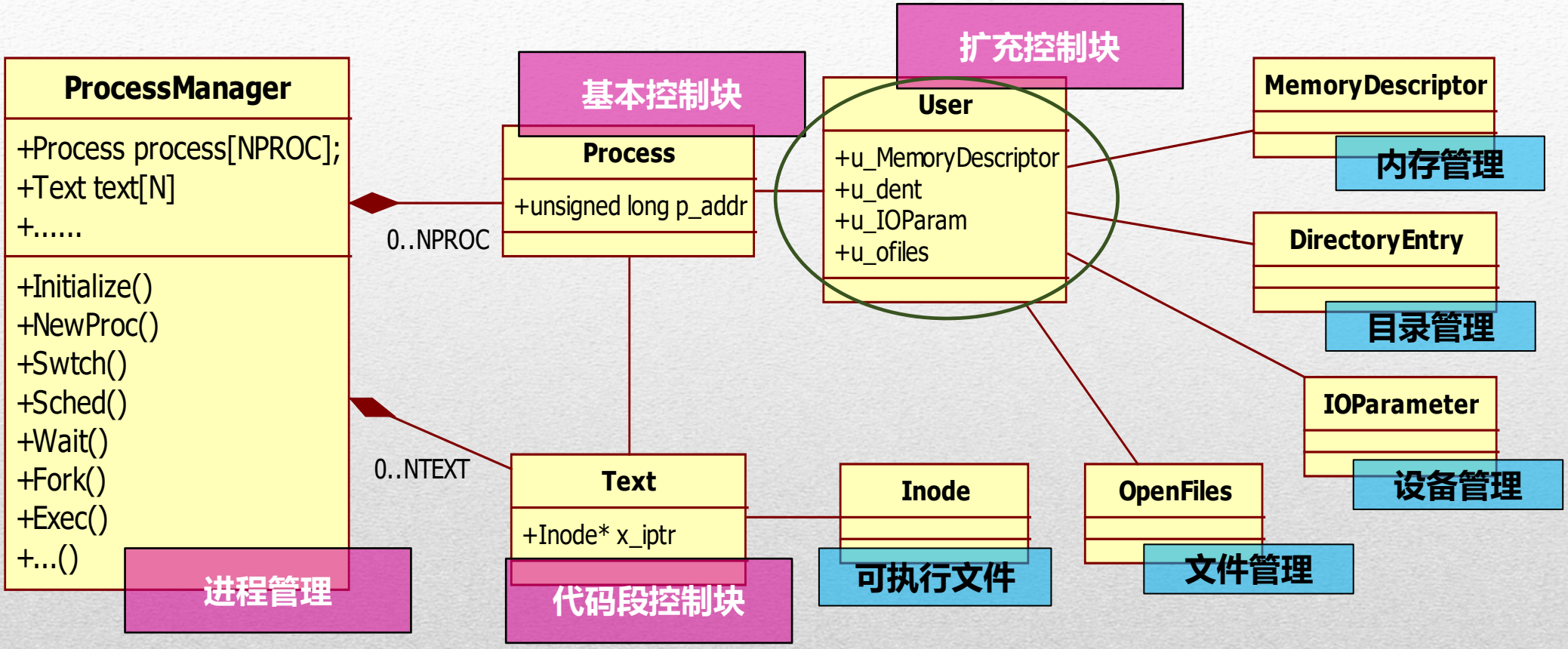
Process类

	名称	类型	含义
进程标识	p_uid	short	用户ID
	p_pid	int	进程标识数，进程编号
	p_ppid	int	父进程标识数
进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符
进程调度相关信息	p_stat	ProcessState	进程当前的调度状态
	p_flag	int	进程标志位，可以将多个状态组合
	p_pri	int	进程优先数（值越大，优先级越小）
	p_cpu	int	cpu值，用于计算p_pri
	p_nice	int	进程优先数微调参数
	p_time	int	进程在盘交换区上（或内存内）的驻留时间
	p_wchan	unsigned long	进程睡眠原因
信号与控制台终端	p_sig	int	进程信号
	p_ttyp	TTy*	进程tty结构地址

Process类

	名称	类型	含义
进程标识	p_uid	short	用户ID
	p_pid	int	进程标识数，进程编号
	p_ppid	int	父进程标识数
进程图象在内存中的位置信息	p_addr	unsigned long	ppda区在物理内存中的起始地址
	p_size	unsigned int	进程图象（除代码段以外部分）的长度，以字节单位
	p_textp	Text *	指向该进程所运行的代码段的描述符
进程调度相关信息	p_stat	ProcessState	进程当前的调度状态
	p_flag	int	进程标志位，可以将多个状态组合
	p_pri	int	进程优先数（值越大，优先级越小）
	p_cpu	int	cpu值，用于计算p_pri
	p_nice	int	进程优先数微调参数
	p_time	int	进程在盘交换区上（或内存内）的驻留时间
	p_wchan	unsigned long	进程睡眠原因
信号与控制台终端	p_sig	int	进程信号
	p_ttyp	TTy*	进程tty结构地址

UNIX V6++进程图象的实现



所有进程管理相关的类结构

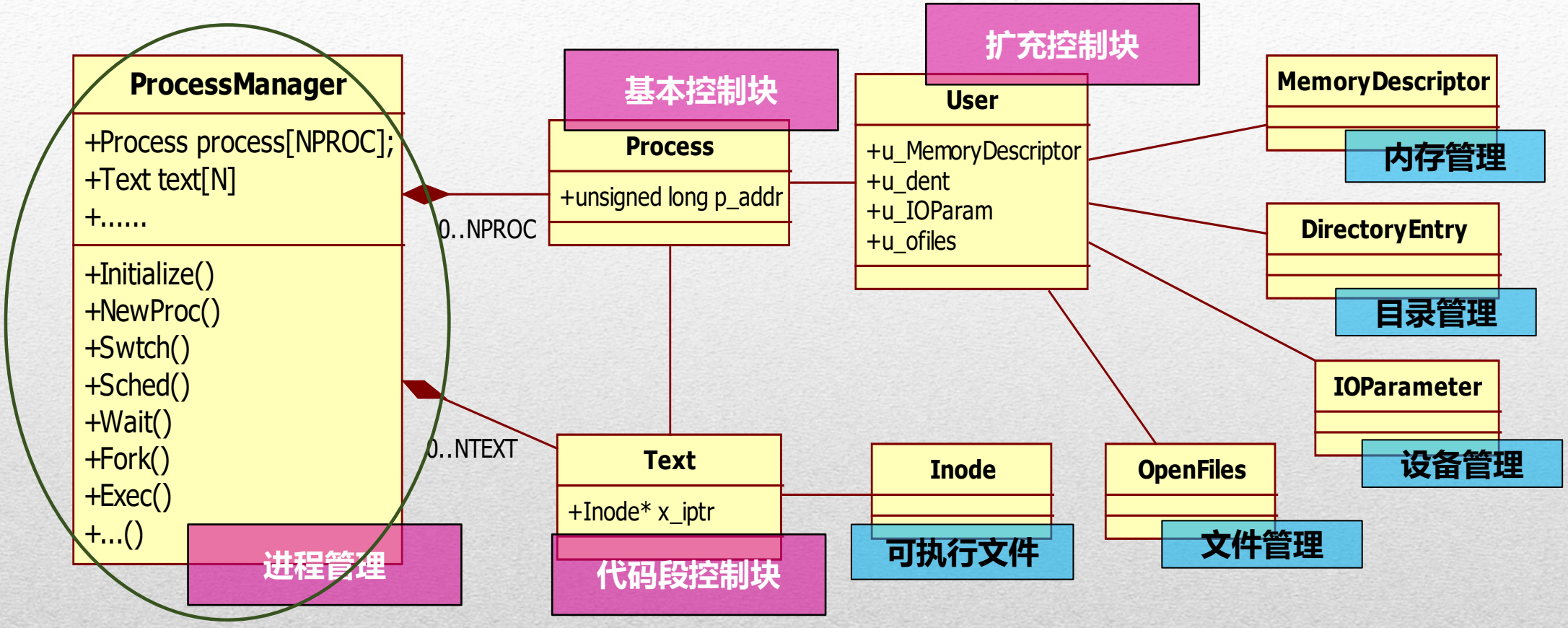
User类

	名称	类型	含义
进程的用户标识	u_uid	short	有效用户ID
	u_gid	short	有效组ID
	u_ruid	short	真实用户ID
	u_rgid	short	真实组ID
进程的时间相关	u_utime	int	进程用户态时间
	u_stime	int	进程核心态时间
	u_cutime	int	子进程用户态时间总和
	u_cstime	int	子进程核心态时间总和
现场保护相关	u_rsav[2]	unsigned long	用于保存esp与ebp指针
	u_ssav[2]	unsigned long	用于对esp和ebp指针的二次保护
内存管理相关	*u_procp	Process	指向该u结构对应的Process结构
	u_MemoryDescriptor	MemoryDescriptor	封装了进程的图象在内存中的位置、大小等信息
系统调用相关	EAX = 0	static const int	访问现场保护区中EAX寄存器的偏移量
	*u_ar0	unsigned int	指向核心栈现场保护区EAX寄存器存放的栈单元
	u_arg[5];	int	存放当前系统调用参数
	*u_dirp	char	系统调用参数（一般用于Pathname）的指针

User类

	名称	类型	含义
进程的用户标识	u_signal[NSIG]	unsigned long	信号处理表
	u_qsav[2]	unsigned long	用于接收到信号时直接从sleep跳回至Trap
	u_intflg	bool	系统调用期间是否受到信号打断
与文件操作相关	u_cdir	Inode*	指向当前目录的Inode指针
	u_pdir	Inode*	指向父目录的Inode指针
	u_dent;	DirectoryEntry	当前目录的目录项
	u_dbuf[]	char	当前路径分量
	u_curdir[128]	char	当前工作目录完整路径
	u_segflg	int	表明I/O针对用户或系统空间
	u_ofiles	OpenFiles	进程打开文件描述符表对象
	u_IOParam	IOParameter	当前读写文件偏移量，用户目标区域和剩余字节
出错	u_error	ErrorCode	存放错误码，具体数值及其含义请查阅源代码

UNIX V6++进程图象的实现



所有进程管理相关的类结构

ProcessManager类

名称	类型	含义
process[NPROC]	Process	进程基本控制块数组
text[NTEXT]	Text	代码段控制块数组
CurPri	int	现运行占用CPU时优先数
RunRun	int	强迫调度标志
RunIn	int	内存中无合适进程可以调出至盘交换区
RunOut	int	盘交换区中无进程可以调入内存
ExeCnt	int	同时进行图像改换的进程数
SwtchNum	int	系统中进程切换次数

PCB表（proc表）
一个进程一个

#define NPROC 100
意味着？

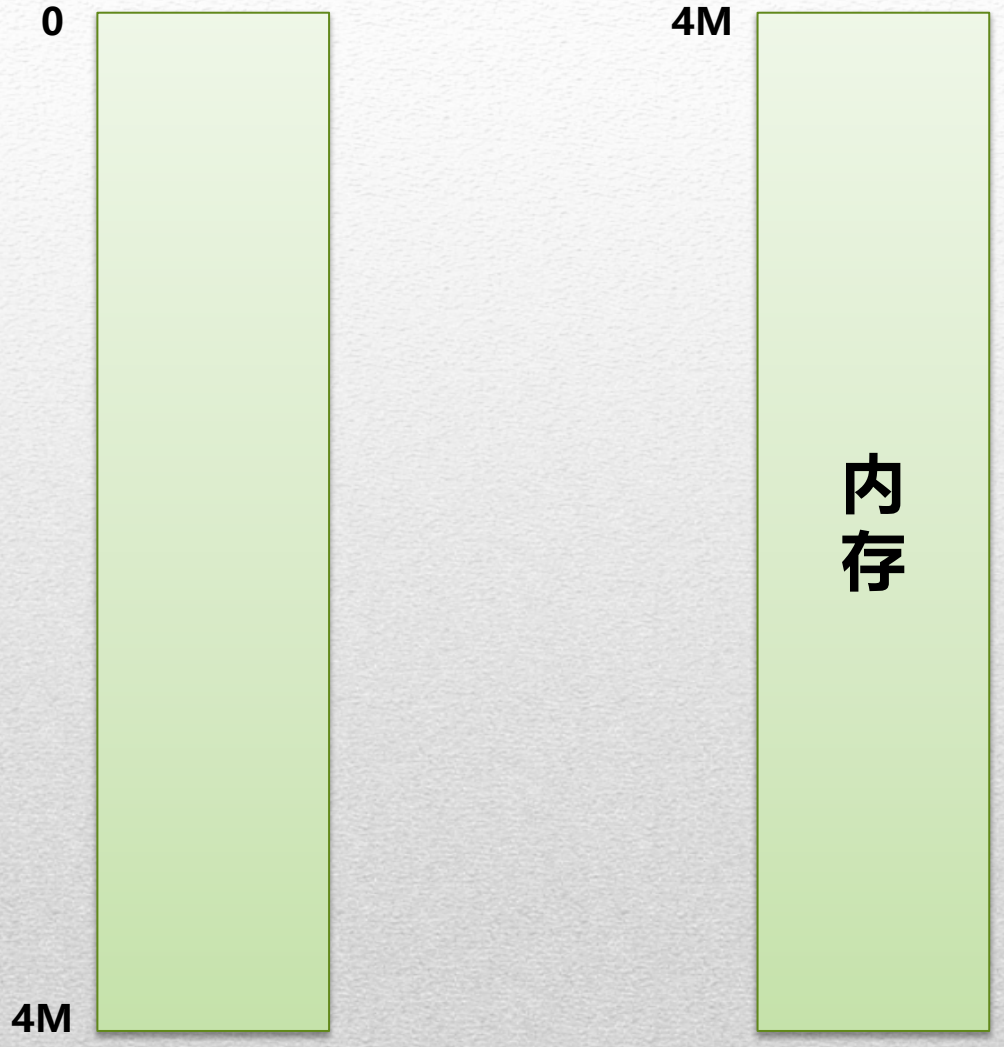
代码段控制块表（Text表）
一个代码段一个

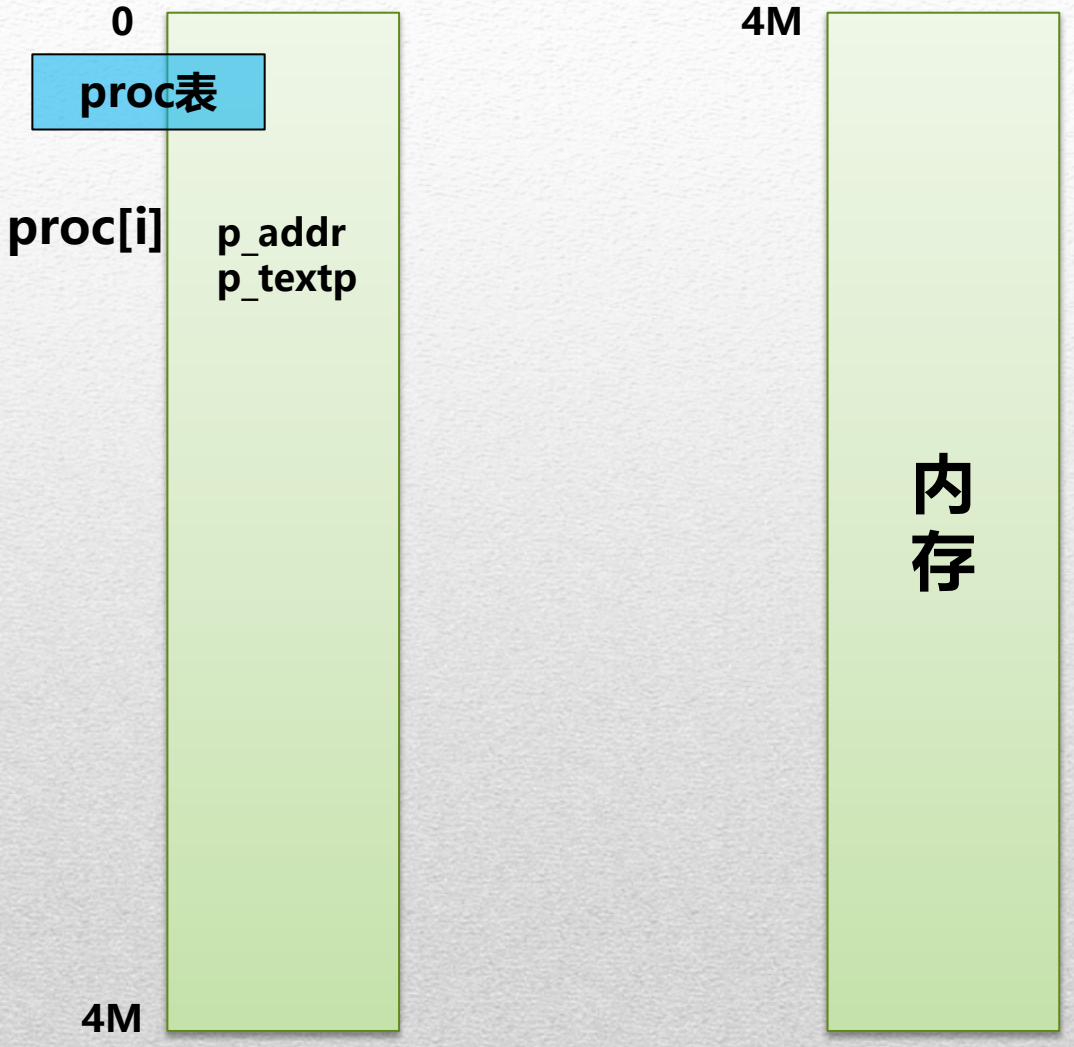
#define NTEXT 50
意味着？

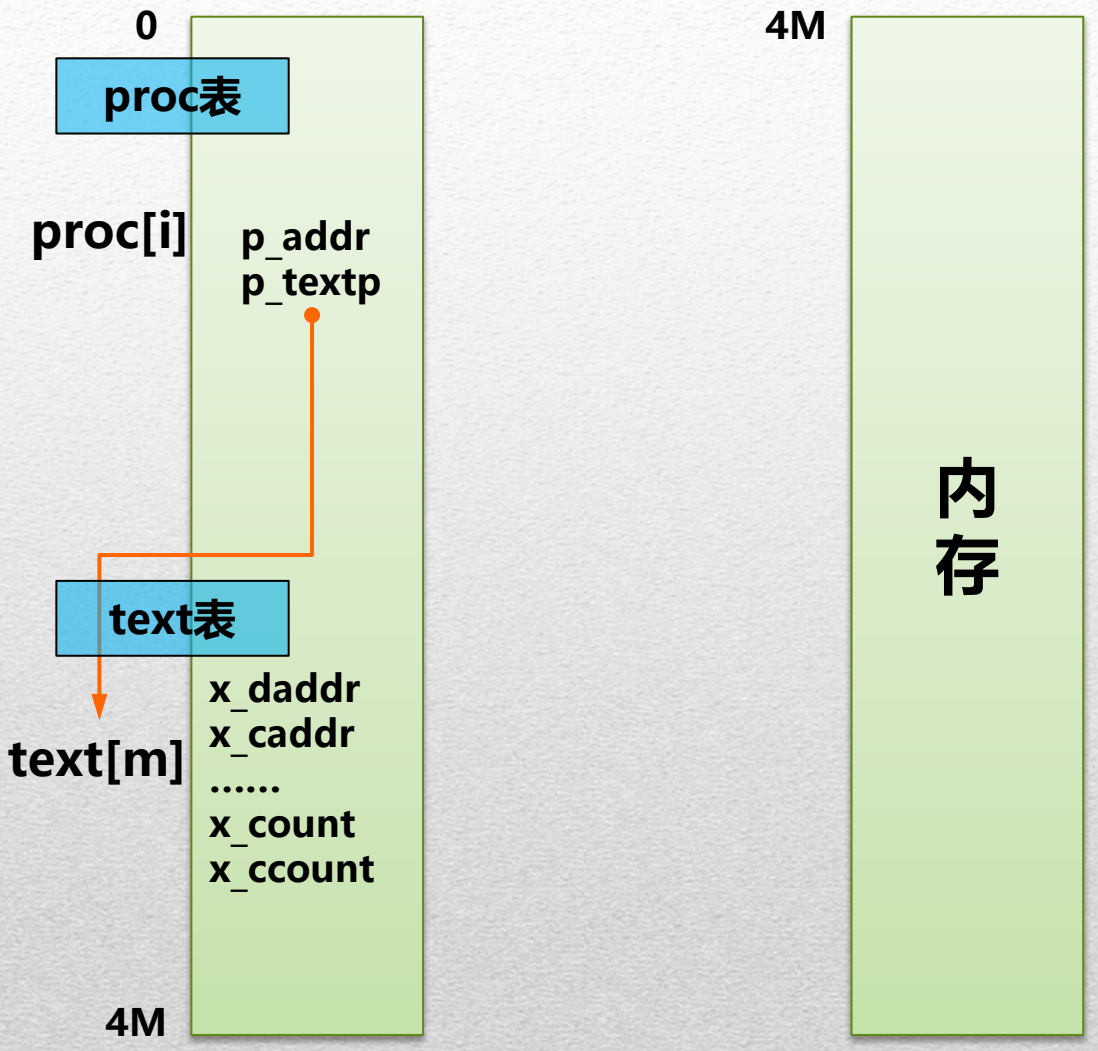
看下面这个简单的程序

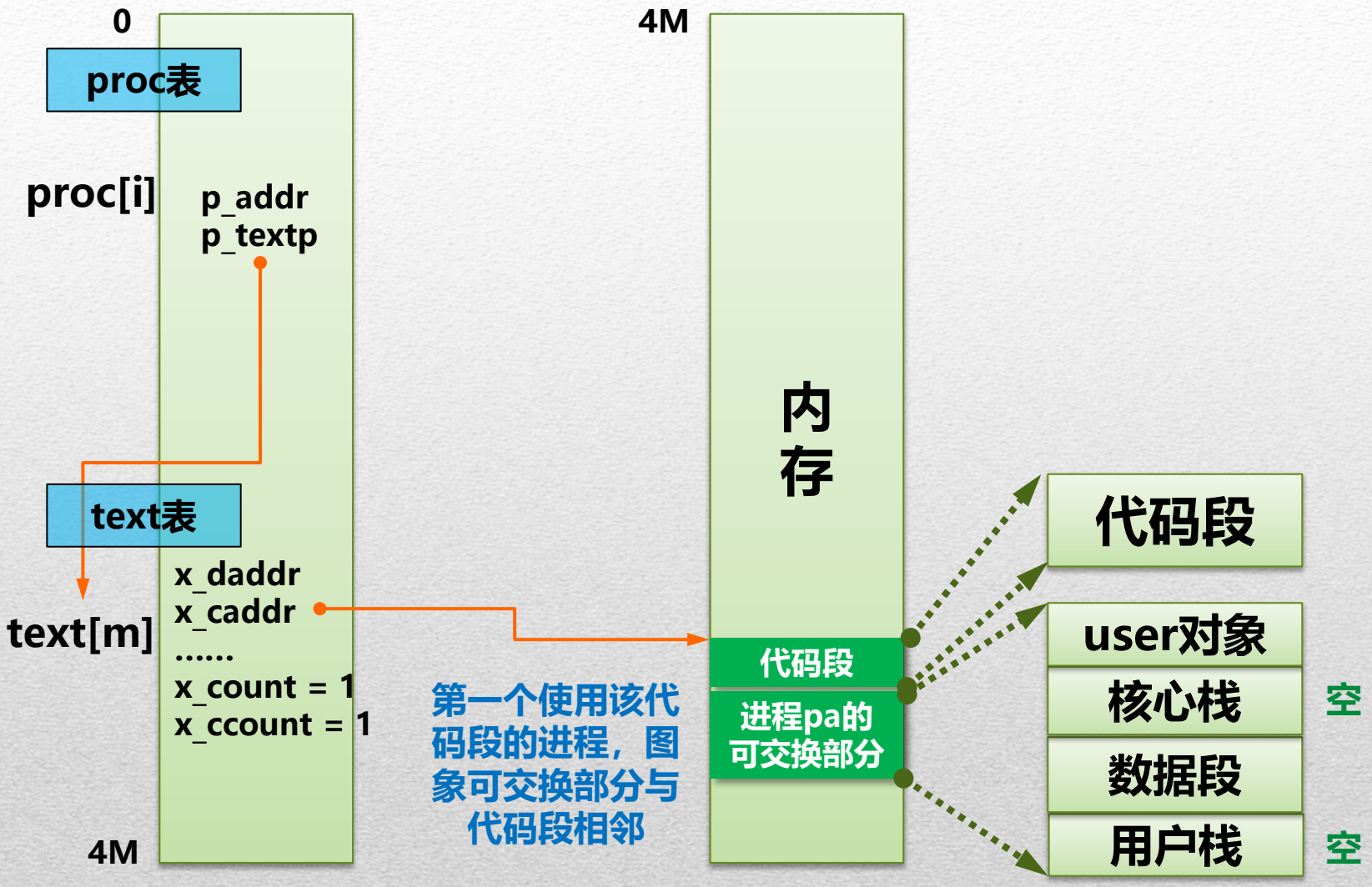
```
#include < fcntl.h>
char buffer[2048];
int version = 1;
main( argc, argv)
int   argc;
char *argv[];
{
    int a, b;
    ……;
    sum(a, b);
    exit(0);
}
int sum( var1, var2)
int var1, var2;
{
    int count;
    count = var1 +var2;
    return(count);
}
```

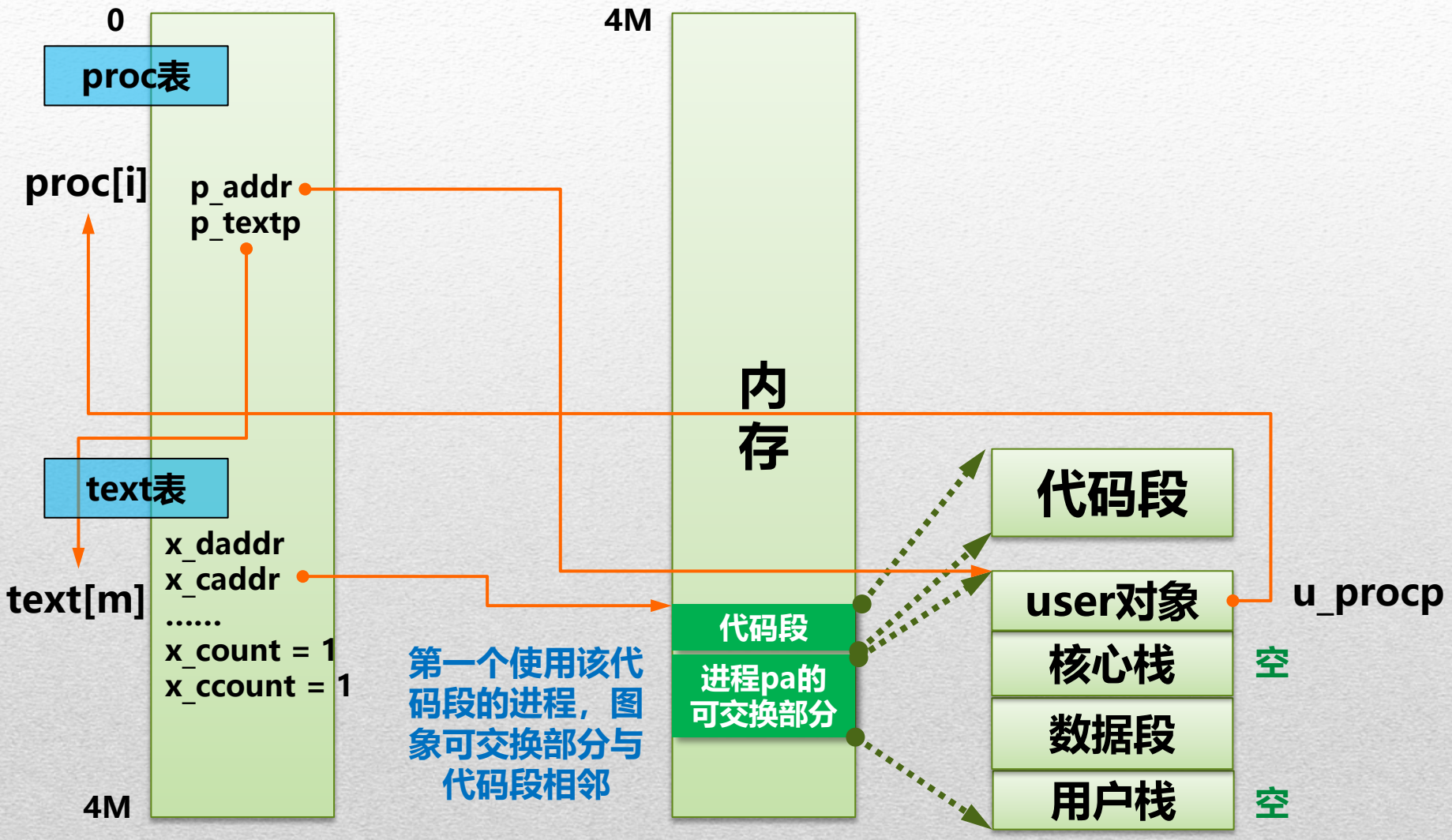
第一次执行……创建进程pa

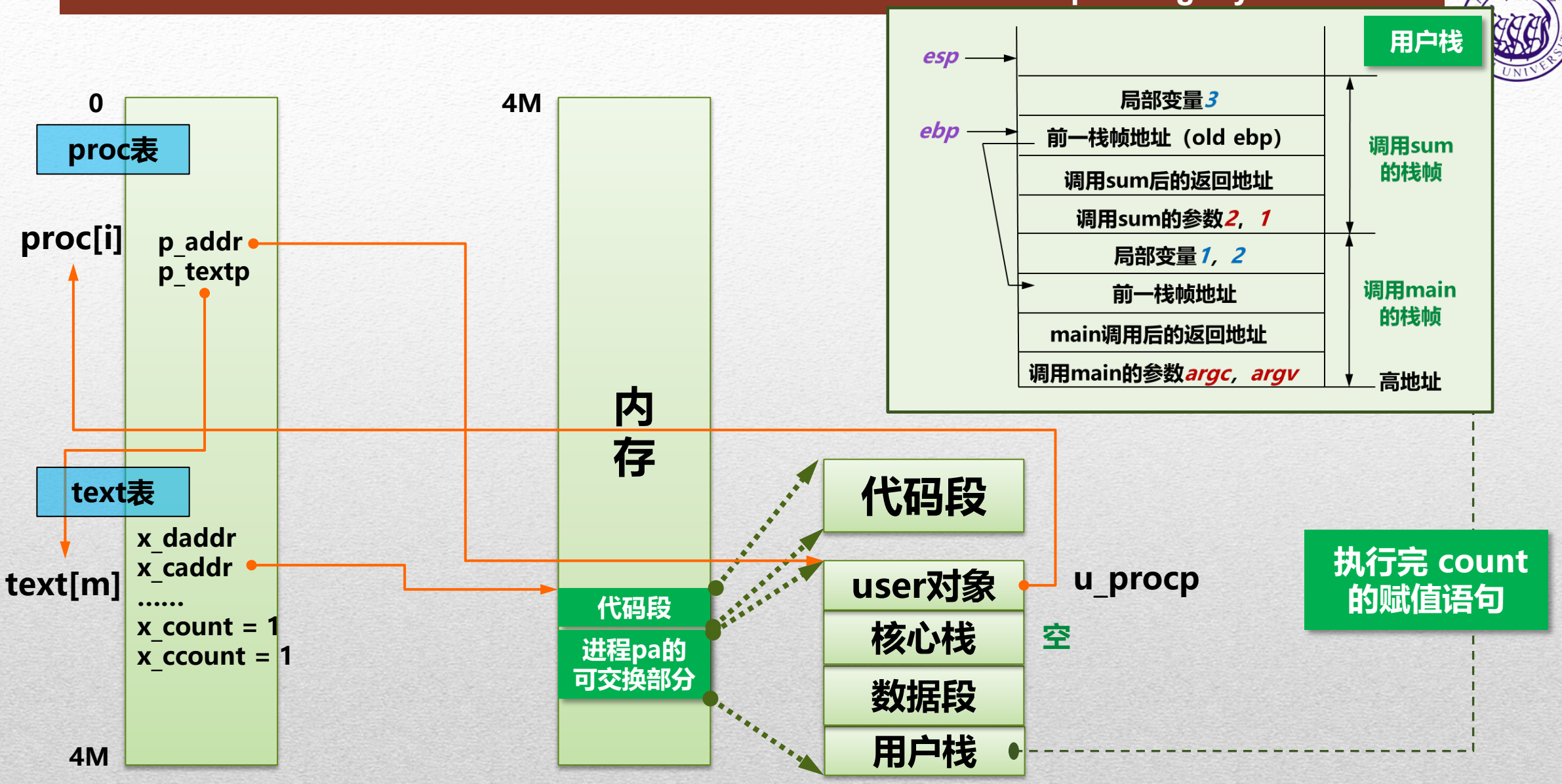

0
4M
4M
内
存


0
proc表
proc[i]
p_addr
p_textp
4M
4M
内
存


0
proc表
proc[i]
p_addr
p_textp
text表
text[m]
x_daddr
x_caddr
.....
x_count
x_ccount
4M
4M
内
存


0
proc表
proc[i]
p_addr
p_textp
text表
text[m]
x_daddr
x_caddr
.....
x_count = 1
x_ccount = 1
4M
第一个使用该代码段的进程，图象可交换部分与代码段相邻
4M
内存
代码段
进程pa的可交换部分
代码段
user对象
核心栈
空
数据段
用户栈
空


0
proc表
proc[i]
p_addr
p_textp
text表
text[m]
x_daddr
x_caddr
.....
x_count = 1
x_ccount = 1
4M
4M
内存
代码段
进程pa的可交换部分
第一个使用该代码段的进程，图象可交换部分与代码段相邻
代码段
user对象
核心栈
数据段
用户栈
空
空
u_procp

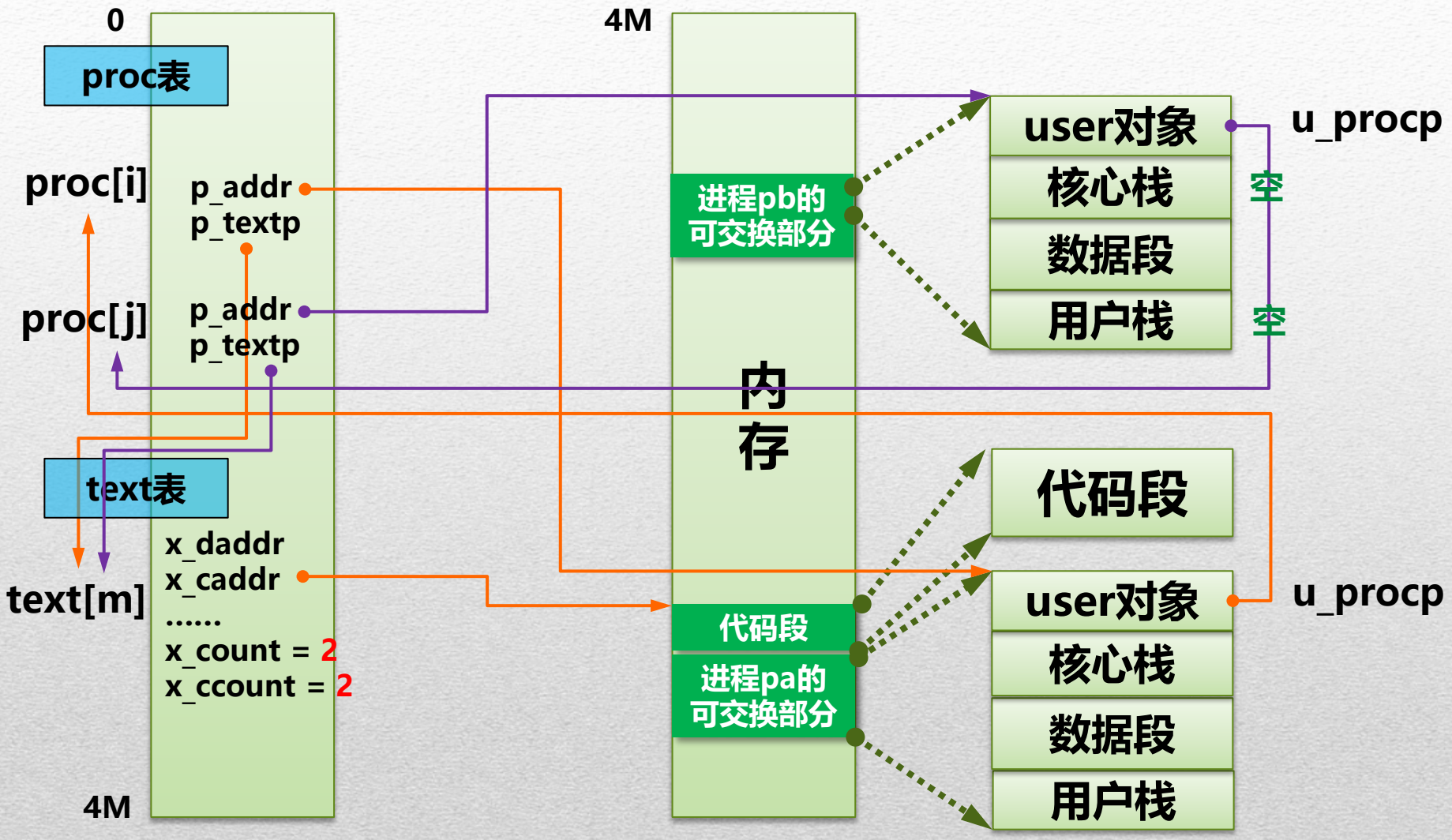

0
proc表
proc[i]
p_addr
p_textp
text表
text[m]
x_daddr
x_caddr
.....
x_count = 1
x_ccount = 1
4M
4M
内存
代码段
进程pa的可交换部分
代码段
user对象
核心栈
数据段
用户栈
u_procp
空
esp
ebp
局部变量3
前一栈帧地址（old ebp）
调用sum后的返回地址
调用sum的参数2，1
局部变量1，2
前一栈帧地址
main调用后的返回地址
调用main的参数argc，argv
用户栈
调用sum的栈帧
调用main的栈帧
高地址
执行完 count 的赋值语句

看下面这个简单的程序

```
#include < fcntl.h>
char buffer[2048];
int version = 1;
main( argc, argv)
int   argc;
char *argv[];
{
    int a, b;
    ……;
    sum(a, b);
    exit(0);
}
int sum( var1, var2)
int var1, var2;
{
    int count;
    count = var1 +var2;
    return(count);
}
```

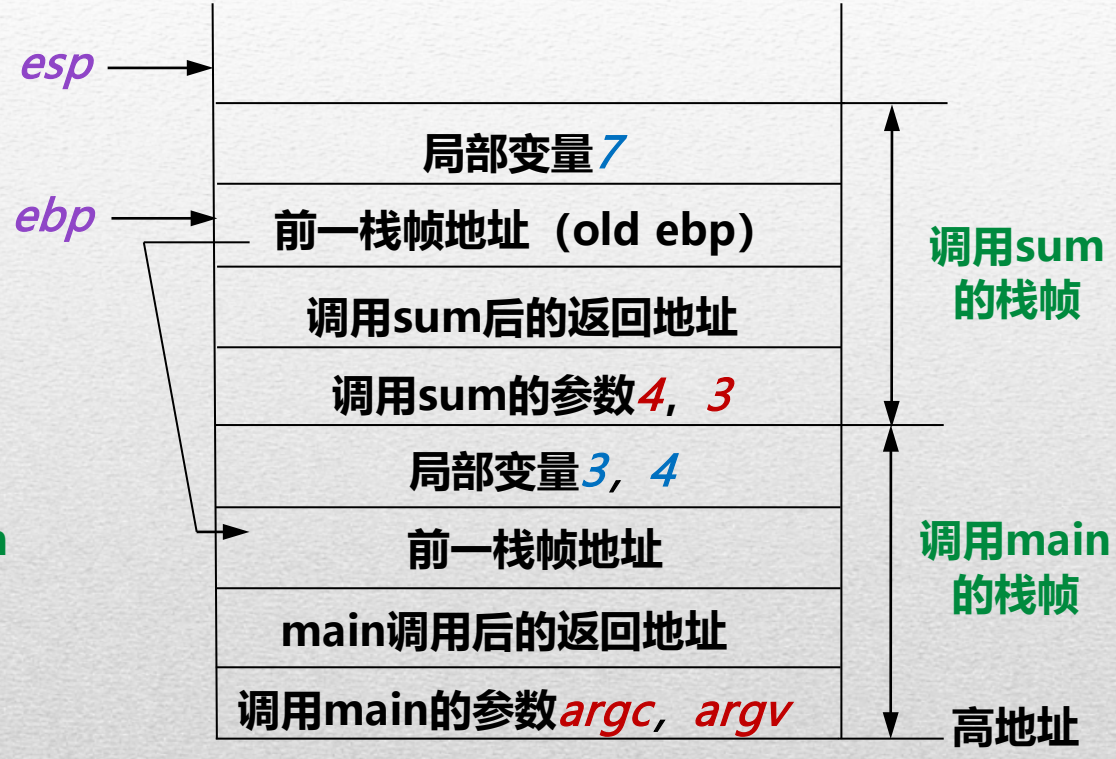
第一次执行……创建进程pa

第二次执行……创建进程pb


0
proc表
proc[i]
p_addr
p_textp
proc[j]
p_addr
p_textp
text表
text[m]
x_daddr
x_caddr
.....
x_count = 2
x_ccount = 2
4M
4M
进程pb的
可交换部分
内
存
代码段
进程pa的
可交换部分
user对象
核心栈
数据段
用户栈
u_procp
空
空
代码段
user对象
核心栈
数据段
用户栈
u_procp

执行完 count 的赋值语句 进程pa的用户栈

执行完 count 的赋值语句 进程pb的用户栈



- 每个进程有自己的用户栈，分别记录该进程当前的函数调用关系和每个函数的参数与局部变量
- 哪个进程在台上执行，esp和ebp就指向哪个进程的用户栈

看下面这个简单的程序

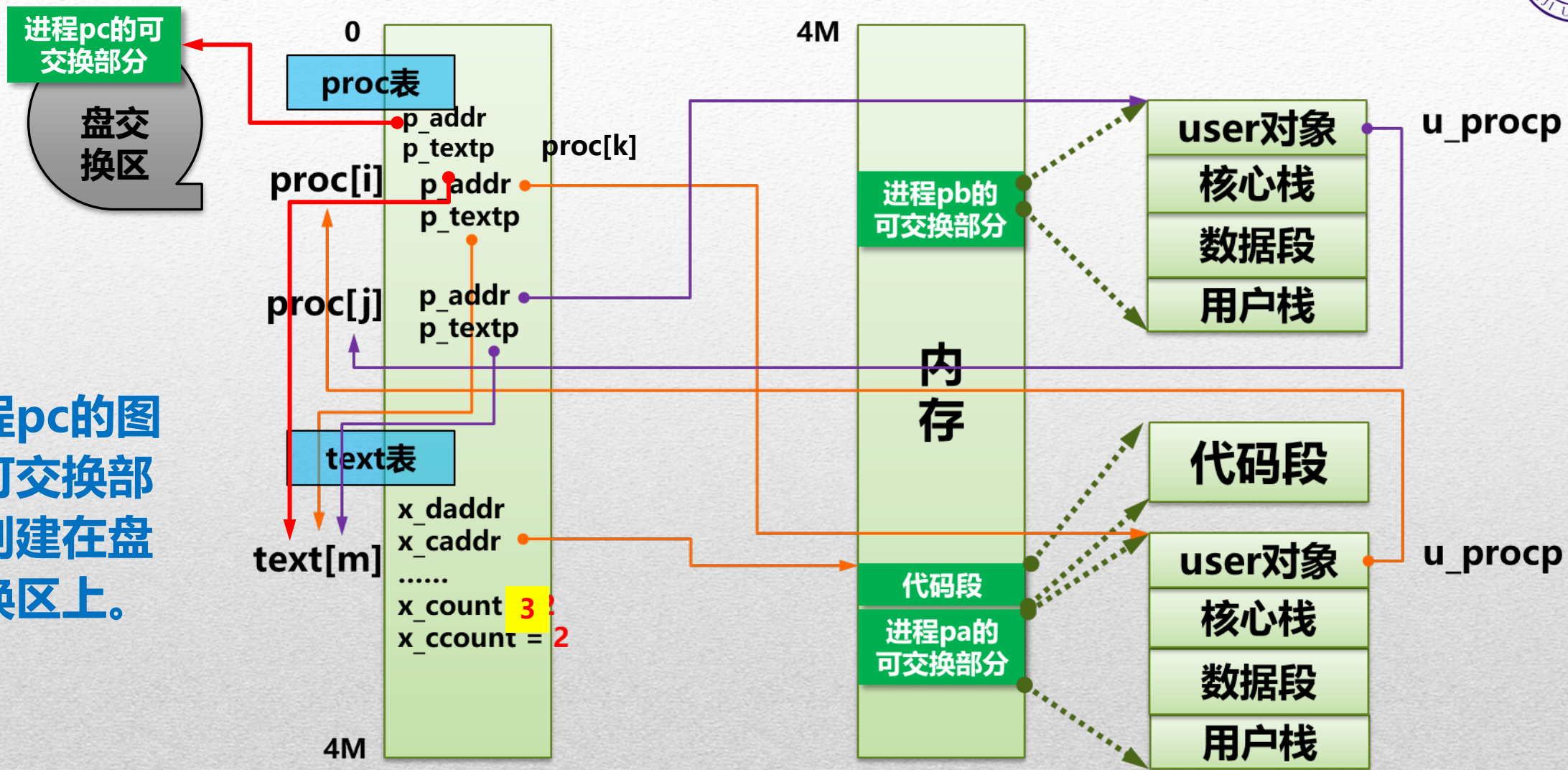
```
#include < fcntl.h>
char buffer[2048];
int version = 1;
main( argc, argv)
int   argc;
char *argv[];
{
    int a, b;
    ……;
    sum(a, b);
    exit(0);
}
int sum( var1, var2)
int var1, var2;
{
    int count;
    count = var1 +var2;
    return(count);
}
```

第一次执行……创建进程pa

第二次执行……创建进程pb

第三次执行……创建进程pc

如果此时内存空间不足。。。



进程pc的图象可交换部分创建在盘交换区上。

程序并发执行带来的问题……

 资源共享  各种程序活动的相互依赖与制约

为了解决程序并发执行带来的问题：

 程序 →  进程

一组数据与指令代码的集合	结构特征 代码段、数据段、堆栈段、进程控制块
静态的 存放在某种介质上	动态性，具有生命周期 “由创建而产生，由调度而执行，由撤销而消亡”

进程是程序的一次运行过程!!!

- 多个进程实体可同时存在于内存中并发执行
- 独立运行、独立分配资源和独立接受调度的基本单位
- 按不可预知（异步）的速度向前推进

理解程序和进程在结构上的差异

本节小结：

1　UNIX进程的进程图象

2　UNIX中与进程管理相关的类结构

请阅读讲义：72页 ~ 82页（2.7.2 ~2.7.3节）

请阅读代码ProcessManager.h，Text.h，User.h，Process.h

P02：UNIX V6++进程的栈帧