第三章

存储管理

主要知识点

1. 一般存储管理
2. UNIX存储管理

主要知识点

1. 一般存储管理

2. UNIX存储管理

一般存储管理

可变分区分配

分配算法

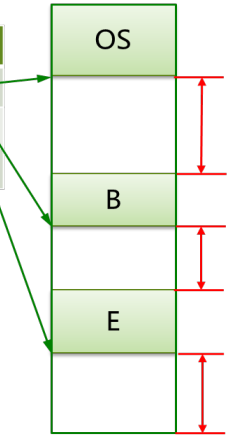
首次适应

- 从第1个空闲区开始顺序找，直至找到一个大小满足要求的空闲分区
- 从该分区中划一块给请求者，余下的仍留空闲分区表中

矛盾和碎片集中在低地址部分

按空闲分区首地址升序排列

索引号	大小	起始地址
1	150	1000
2	70	2000
3	100	3000



循环首次适应

- 从上次找到空闲分区的下一个空闲分区开始查找（采用循环查找方式）
- 从该分区中划一块给请求者，余下的仍留空闲分区表中

缺乏大的空闲分区

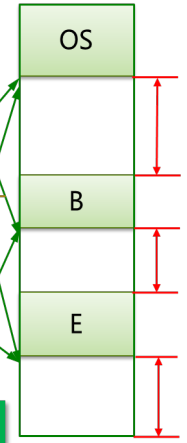
最佳匹配法

- 总把能满足要求，又最小的空闲区分配给进程

避免大材小用，但碎片小

按空闲分区大小升序排列

索引号	大小	起始地址
1	70	2000
2	100	3000
3	150	1000



最坏匹配法

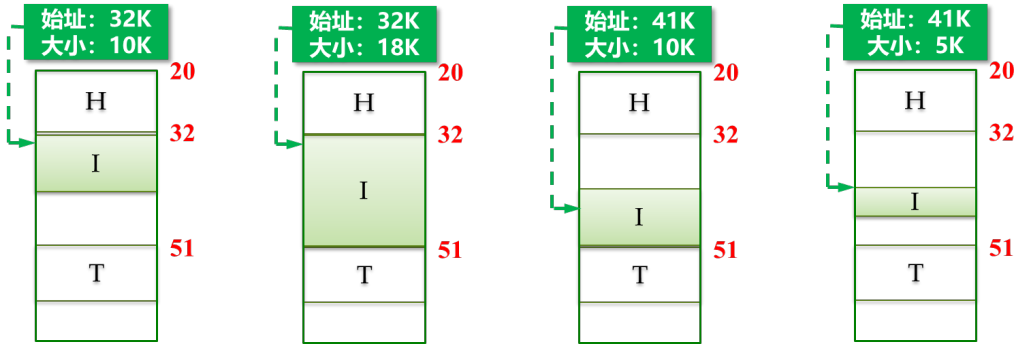
- 总把能满足要求，又最大的空闲区分配给进程，产生碎片的几率最小。

查找效率高，但缺大分区

索引号	大小	起始地址
1	150	1000
2	100	3000
3	70	2000

按空闲分区大小降序排列

内存回收



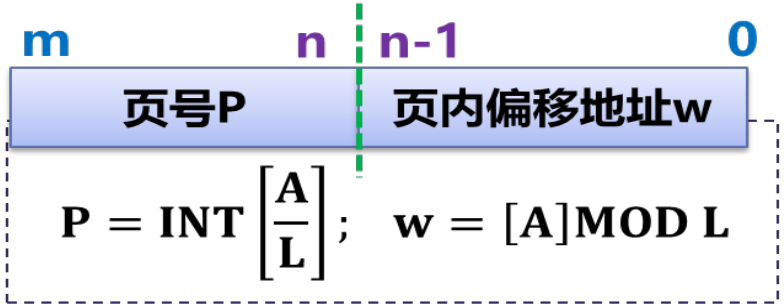
可变分区分配的分配与回收算法：4种分配算法，4种回收情况

一般存储管理

分页存储管理中逻辑地址与物理地址的构成

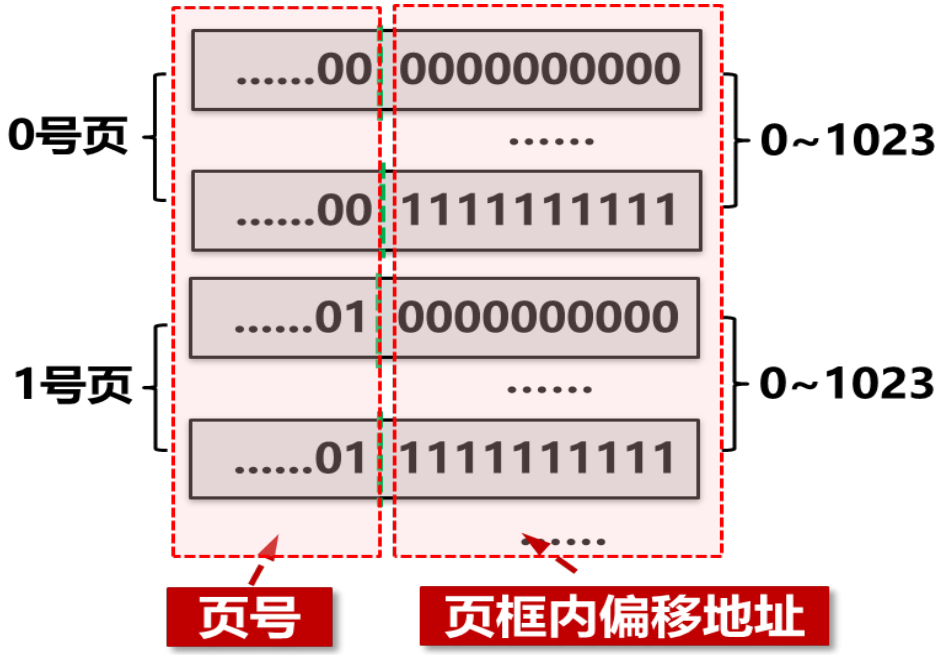
程序逻辑地址空间

地址A，页长：$L=2^n$
程序地址分成两个部分：



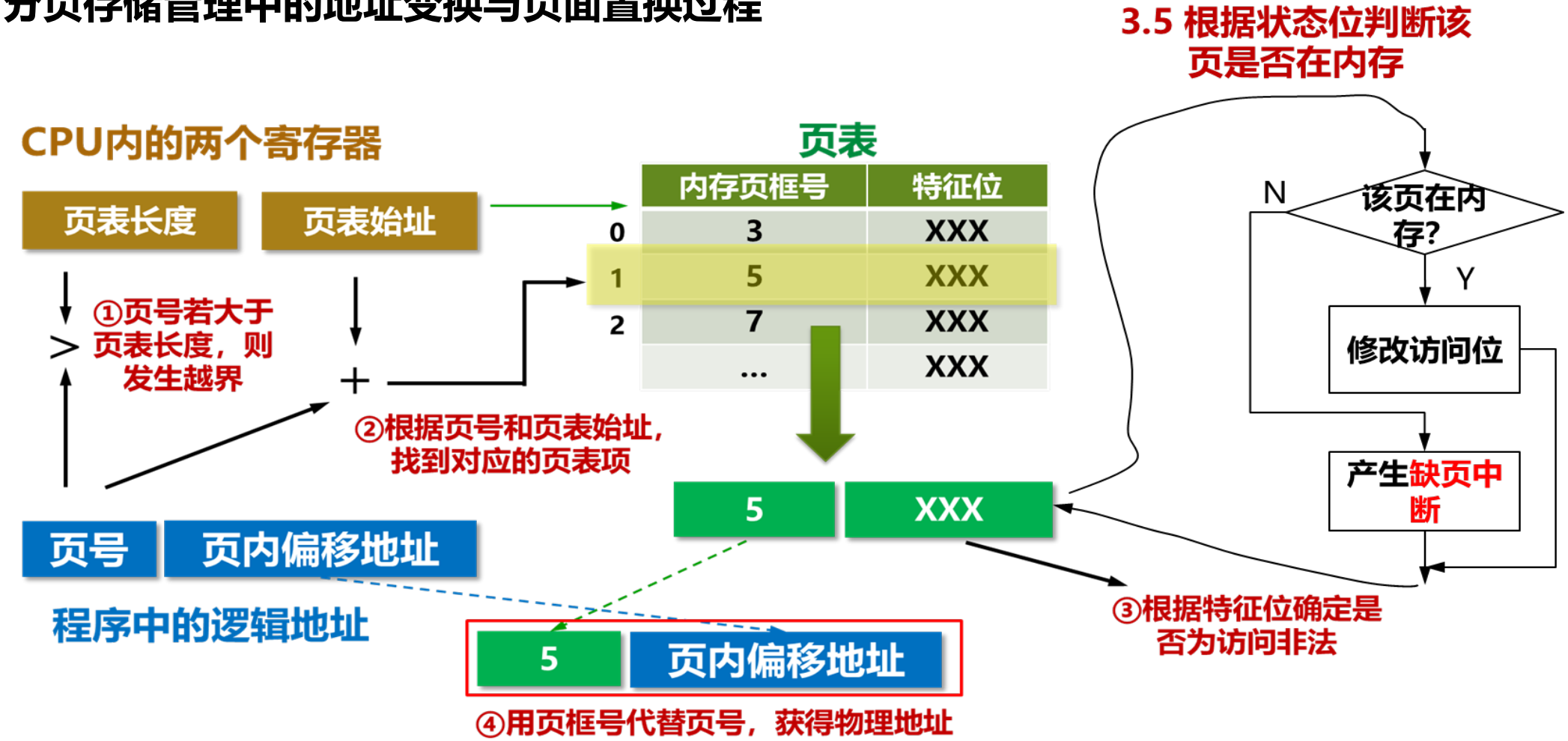
从0开始顺序编址的程序地址，称为线性地址
（一维的，连续的）

例如，某系统的逻辑页面大小1KB



上例中，若A＝2170，则P=2，W=122

分页存储管理中的地址变换与页面置换过程



一般存储管理

1. 在下列动态分区分配算法中，最容易产生内存碎片的是 C

A. 首次适应算法　　B. 最坏适应算法
C. 最佳适应算法　　D. 循环首次适应算法

2. 下列措施中，能加快虚实地址转换的是 C

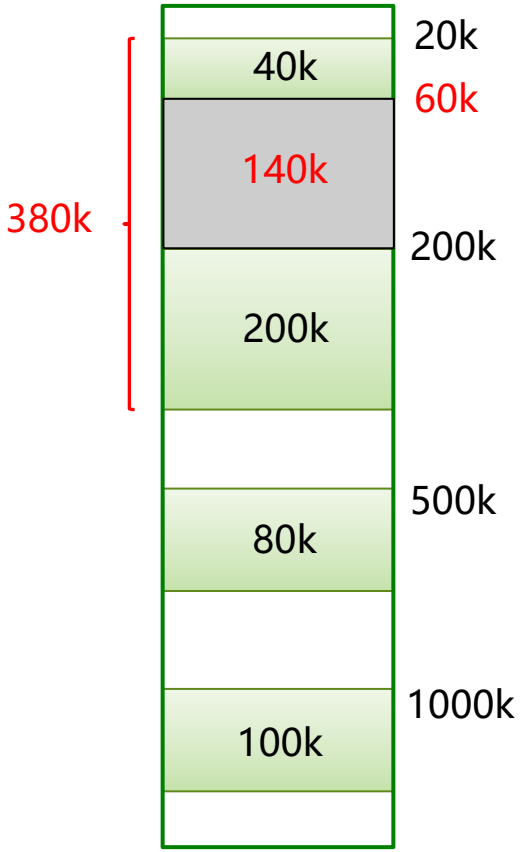
Ⅰ. 增大快表(TLB)容量　　Ⅱ. 让页表常驻内存
Ⅲ. 增大交换区(swap)

A. 仅Ⅰ　　B. 仅Ⅱ　　C. 仅Ⅰ、Ⅱ　　D. 仅Ⅱ、Ⅲ

一般存储管理

3. 某计算机按字节编址，其动态分区内存管理采用最佳适应算法，每次分配和回收内存后都对空闲分区链重新排序。当前空闲分区信息如下表所示。

分区起始地址	20K	500K	1000K	200K
分区大小	40KB	80KB	100KB	200KB



回收起始地址为60K，大小为140KB的分区后，系统中空闲分区的数量、第一个分区的起始地址和大小分别是：

3、500K、80KB 。

一般存储管理

4. 某虚拟存储器用户地址空间有32个页面，每页1KB，主存16KB。假定某时刻为用户的第0，1，2，3号页面分配的物理页号为5，10，4，7，试将虚拟地址0x0A5C和0x0D3C变化成物理地址。

页表
5
10
4
7

1KB页长：　　低10位为页内偏移地址；
32个页面：　　虚地址中至少需要高5位（有效位）为页号；
主存16KB：　　物理地址至少需要高4位（有效位）为页框号。

0A5C = 000 1010 0101 1100　虚页号为2，物理页号为4

000 1010 0101 1100 ⇨ 01 0010 0101 1100 =125C

0D3C = 000 1101 0011 1100　虚页号为3，物理页号为7

000 1101 0011 1100 ⇨ 01 1101 0011 1100 =1D3C

一般存储管理

5. 某计算机系统按字节编址，采用二级页表的分页存储管理方式，虚拟地址格式如下所示：

页目录号(10位)	页表索引(10位)	页内偏移量(12位)

页和页框的大小各为 4KB ，进程的虚拟地址空间大小为 2^{20} 页，假定页目录项和页表项均占4个字节，则进程的页目录和页表最多共占 1025 页。

1024*4B=4K 占用1页
1024*（1024*4B）=1024*4K 占用1024页

一般存储管理

6. 有一128行、128列的整数数组A在系统中按行存放。系统采用页式存储管理，内存一个页框可放128个整数。给数组A赋值分别采用程序段（1）、程序段（2）时，各自产生的缺页中断次数为多少。设在内存中给A分配10个物理页面，并且开始时A的第1个页面已在内存。

程序段（1）：	程序段（2）：
for i:=1 to 128 do for j:=1 to 128 do A[i][j]:=0;	for j:=1 to 128 do for i:=1 to 128 do A[i][j]:=0;

采用代码（1）其访问顺序与数组存放顺序一致，由于第一页已在内存中，所以除了访问第一页时不发生缺页，对其余127页的访问均发生缺页，所以共发生128-1次缺页中断。

采用代码（2）其访问顺序是按列访问，与数组存放顺序不一致，共发生128*128-1次缺页中断。

一般存储管理

7. 在某页式存储管理系统中，一个作业的程序地址空间为3KB，机器内存最大容量为128KB，每个页面的大小为1KB，右表是页表的内容。用户程序中第100号字节单元处有一条指令“MOV ax，(2500)”，该指令的内存地址是_____，该指令操作数的内存地址是_____。

1
2
4

100字节单元的逻辑地址：

00 00 0110 0100 虚页号为0，查表得页框号为1；

物理地址为：

00 00 0110 0100 ⇨ 0 0000 0100 0110 0100 = 1124

另一种计算方法：

2500/1024 可知逻辑地址2500在第2号逻辑页，页内偏移地址为452，所以对应的物理地址为4号，页内偏移地址为452，则物理地址为：1024*4+452=4548

一般存储管理

8. 在一个请求分页系统中，采用LRU页面置换算法，假如一个作业的页面走向为4，3，2，1，4，3，5，4，3，2，1，5，当分配给该作业的物理页框数分别为3和4时，试计算访问过程中所发生的缺页次数和缺页率（假设开始的M个页面已装入主存）。

分配给该作业的物理页框数 M = 3时：

	4	3	2	1	4	3	5	4	3	2	1	5
是否缺页				*	*	*	*			*	*	*
内存中包含页面	4	4	4	1	1	1	5	5	5	2	2	2
	3	3	3	3	4	4	4	4	4	4	1	1
	2	2	2	2	2	3	3	3	3	3	3	5
被淘汰的页				4	3	2	1			5	4	3

缺页7次，缺页率7/12

一般存储管理

8. 在一个请求分页系统中，采用LRU页面置换算法，假如一个作业的页面走向为4，3，2，1，4，3，5，4，3，2，1，5，当分配给该作业的物理页框数分别为3和4时，试计算访问过程中所发生的缺页次数和缺页率（假设开始的M个页面已装入主存）。

分配给该作业的物理页框数 M = 4时：

	4	3	2	1	4	3	5	4	3	2	1	5
是否缺页							*			*	*	*
内存中包含页面	4	4	4	4	4	4	4	4	4	4	4	5
	3	3	3	3	3	3	3	3	3	3	3	3
	2	2	2	2	2	2	5	5	5	5	1	1
	1	1	1	1	1	1	1	1	1	2	2	2
被淘汰的页							2			1	5	4

缺页4次，缺页率4/12

一般存储管理

9. 设某计算机的逻辑地址空间为64KB，页面长4k。在时刻260前，现运行进程PA的页表如下表所示。系统为每个进程固定分配3个内存页框；若缺页，采用FIFO算法置换。

页号	页框号	存在位	装入时间	上次访问时间	修改位	外存起始扇区号
0	0x100	1	130	250	1	0x2000
1	0x102	1	230	240	0	0x2008
2	-	0	-	-	-	0x2016
3	0x106	1	160	170	0	0x2024

当进程执行到时刻260时，要读取逻辑地址为0x27CA的数据，计算该逻辑地址对应的物理地址。

一般存储管理

0x27CA：该逻辑地址对应的页号是2，页内偏移地址是0x7CA

页号	页框号	存在位	装入时间	上次访问时间	修改位	外存起始扇区号
0	0x100	1	130	250	1	0x2000
1	0x102	1	230	240	0	0x2008
2	-	0	-	-	-	0x2016
3	0x106	1	160	170	0	0x2024

由逻辑页号2查表可知，该逻辑页不在内存。会引发缺页中断，缺页中断的处理过程如下：

一般存储管理

① （选换出页）由于采用FIFO算法，可知选择选择0号页换出；

② （回写脏页）由于0号页在内存中曾经被修改过，因此需启动I/O操作将0号页写回磁盘上0x2000开始的8个盘块中；

③ （读入新页）启动I/O将0x2016开始的8个盘块的内容读入0x100号物理页框；

④ （修改页表）修改页表如下：

页号	页框号	存在位	装入时间	上次访问时间	修改位	外存起始扇区号
0	-	0	-	-	-	0x2000
1	0x102	1	230	240	0	0x2008
2	0x100	1	260	260	0	0x2016
3	0x106	1	160	170	0	0x2024

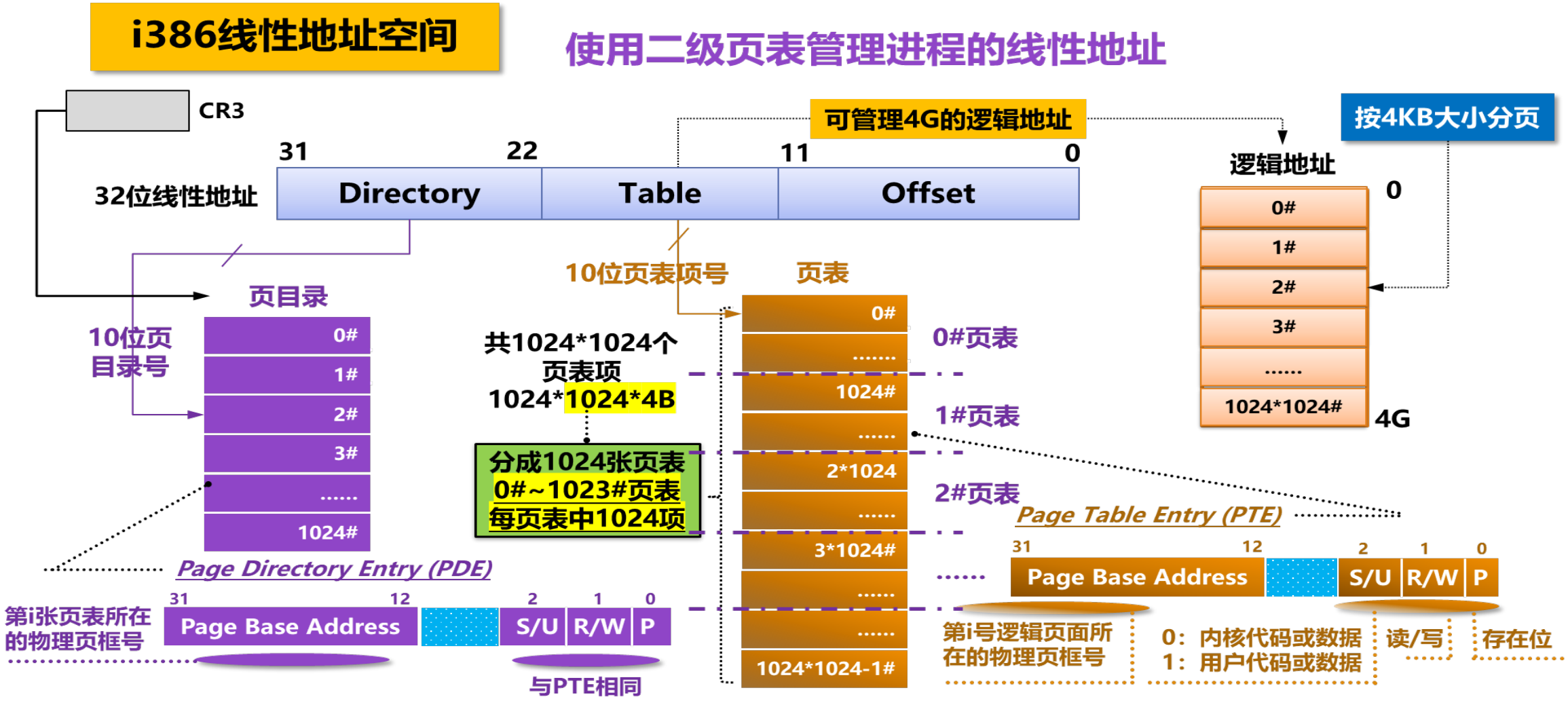
⑤ （形成地址） 0x27CA ⇨ 0x1007CA

主要知识点

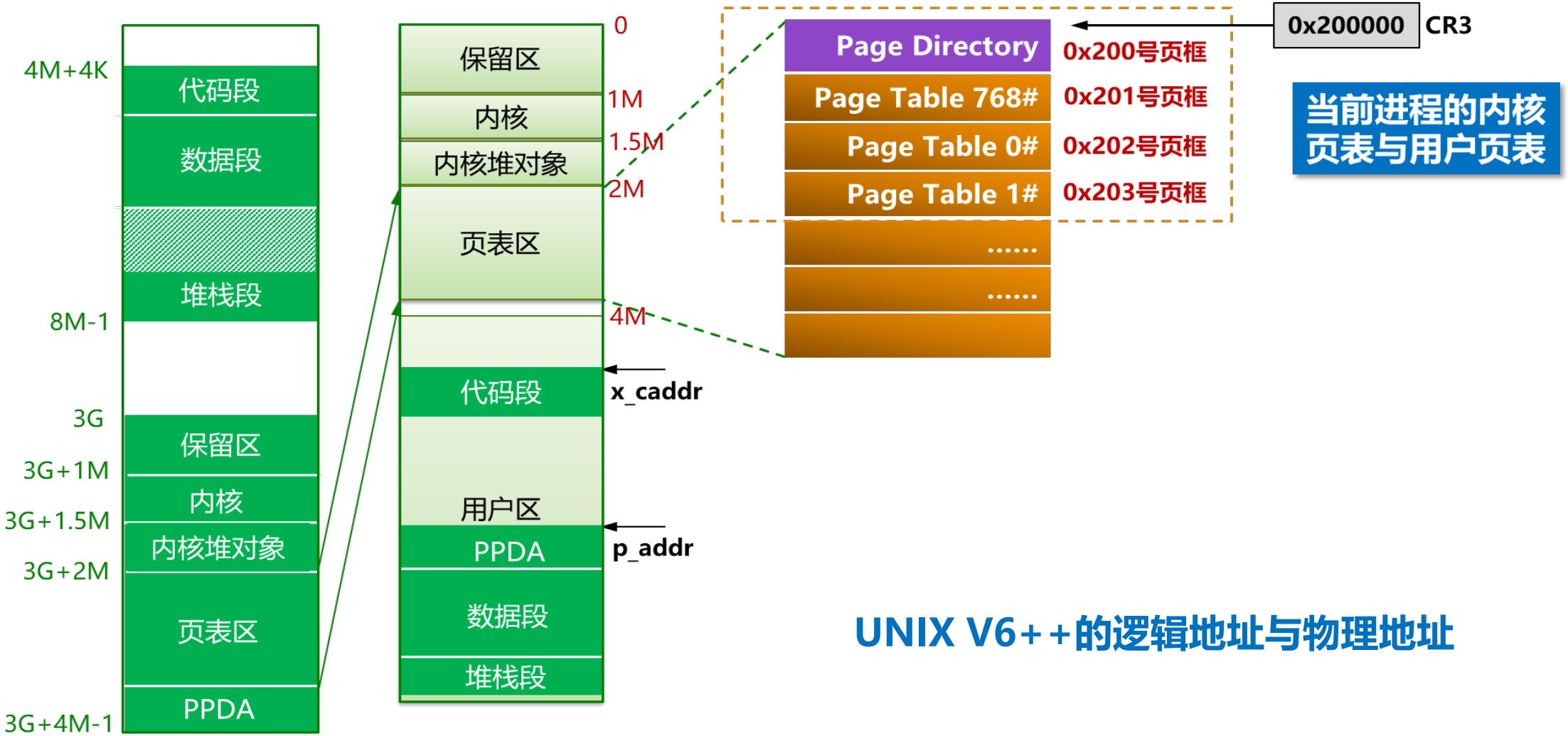
1. 一般存储管理

2. UNIX存储管理

主要知识点

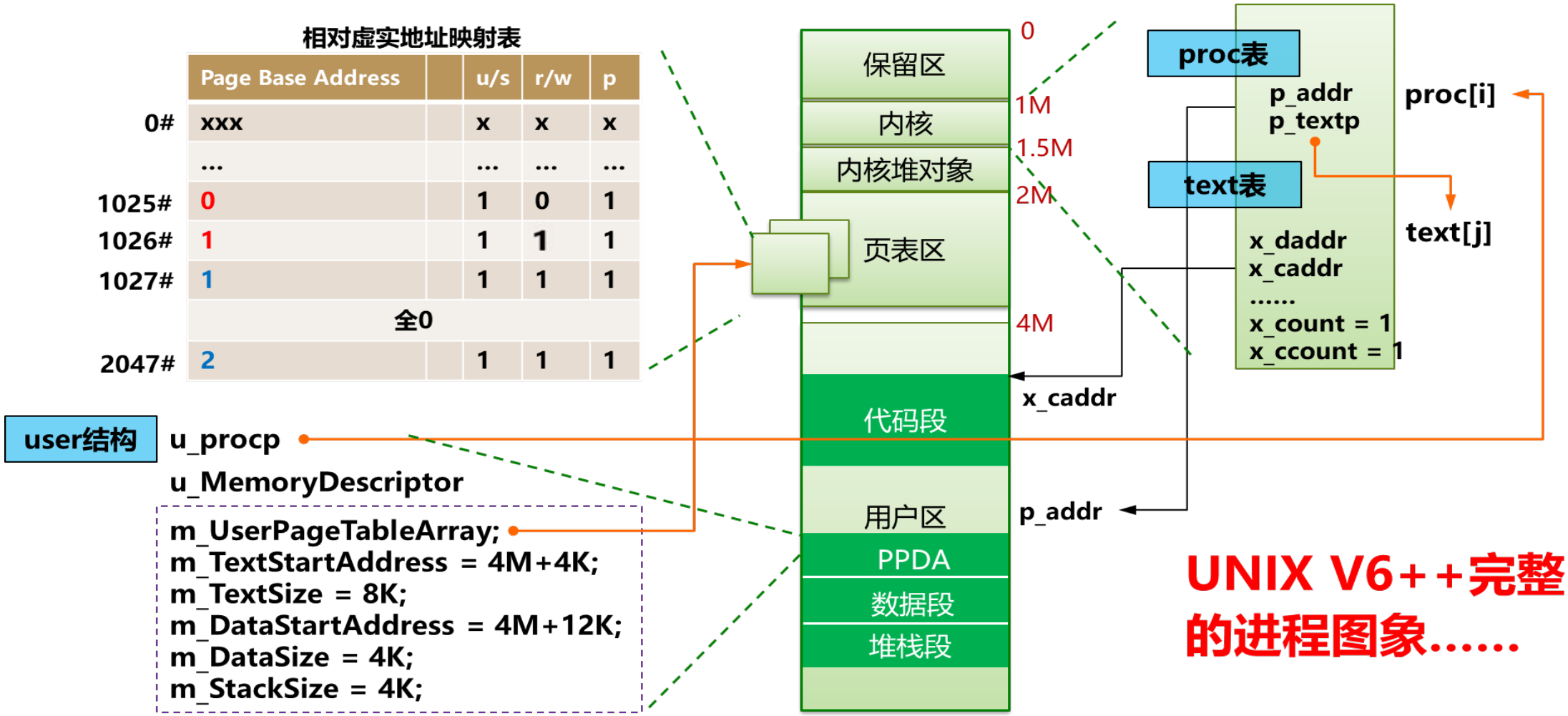

i386线性地址空间
使用二级页表管理进程的线性地址
CR3
可管理4G的逻辑地址
按4KB大小分页
31
22
11
0
32位线性地址
Directory
Table
Offset
逻辑地址
0
0#
1#
2#
3#
.....
1024*1024#
4G
10位页表项号
页表
页目录
10位页
目录号
0#
1#
2#
3#
.....
1024#
共1024*1024个
页表项
1024*1024*4B
分成1024张页表
0#~1023#页表
每页表中1024项
0#页表
1#页表
2#页表
0#
.....
1024#
.....
2*1024
.....
3*1024#
.....
.....
1024*1024-1#
Page Directory Entry (PDE)
第i张页表所在
的物理页框号
31
12
2
1
0
Page Base Address
S/U
R/W
P
与PTE相同
Page Table Entry (PTE)
第i号逻辑页面所
在的物理页框号
0：内核代码或数据
1：用户代码或数据
读/写
存在位

主要知识点



UNIX V6++的逻辑地址与物理地址

主要知识点


相对虚实地址映射表
Page Base Address
u/s
r/w
p
0#
xxx
x
x
x
...
1025#
0
1
0
1
1026#
1
1
1
1
1027#
1
1
1
1
全0
2047#
2
1
1
1
保留区
内核
内核堆对象
页表区
代码段
用户区
PPDA
数据段
堆栈段
0
1M
1.5M
2M
4M
proc表
p_addr
p_textp
proc[i]
text表
text[j]
x_daddr
x_caddr
.....
x_count = 1
x_ccount = 1
x_caddr
p_addr
user结构
u_procp
u_MemoryDescriptor
m_UserPageTableArray;
m_TextStartAddress = 4M+4K;
m_TextSize = 8K;
m_DataStartAddress = 4M+12K;
m_DataSize = 4K;
m_StackSize = 4K;
UNIX V6++完整的进程图象……

复习

进程基本控制块

	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

复习

	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
出错	u_error	ErrorCode	存放错误码，具体数值及其含义请查阅源代码

复习

例题

某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（1）内核在逻辑地址3G+2M+48K处为该进程创建相对虚实地址映射表。画出该进程的相对虚实地址映照表，并填写内存描述符u_MemoryDescriptor中各参数的值。

```
int n;
int main()
{
        n = 2000 ;
        int result = factorial(n) ;
        printf ( "The result is %d", result ) ;
}
int factorial(int n)
{
        if(n>1)
L :         return(n* factorial(n-1));
        else
            return(1);
}
```

```
PageTable*      m_UserPageTableArray = 3G+2M+48K;
unsigned long m_TextStartAddress = 4M+4K;
unsigned long m_TextSize = 4K;
unsigned long m_DataStartAddress = 4M+8K;
unsigned long m_DataSize = 4K;
unsigned long m_StackSize = 4K;
```

	Page Base Address		s/u	r/w	p
0#	xxx		x	x	x
	…		…	…	…
1024#	xxx		x	x	x
1025#	0		1	0	1
1026#	1		1	1	1
	全0				
2047#	2		1	1	1

进程创建时完成相对虚实地址映射表和内存描述符的设置

复习

例题

某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（2）如果该进程为现运行进程，且该进程的p_addr=0x00600000，x_caddr= 0x00500000，请尽量详细地绘制出该进程的图象和4张页表的内容。

	Page Base Address		u/s	r/w	p
0#	xxx		x	x	x
	…		…	…	…
1024#	xxx		x	x	x
1025#	0		1	0	1
1026#	1		1	1	1
	全0				
2047#	2		1	1	1

进程上台时，完成页表的设置

Page Directory（0x200号页框）

	Page Base Address		s/u	r/w	p
0#	0x202		1	1	1
1#	0x203		1	1	1
	……				
768#	0x201		0	1	1
	……				

Page Table 768#（0x200号页框）

	PBA		s/u	r/w	p
0#	0		0	1	1
1#	1		0	1	1
	……				
1023#	0x600		0	1	1

Page Table 0#（0x202号页框）

PBA		s/u	r/w	p
/		/	/	/

Page Table 1#（0x203号页框）

	PBA		s/u	r/w	p
0#	/		/	/	/
1#	0x500		1	0	1
	0x601		1	1	1
	……		…	…	…
1023#	0x602		1	1	1

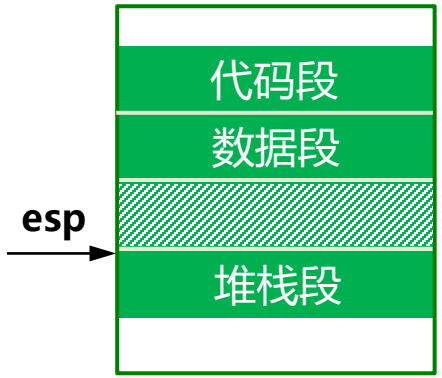
复习

例题

某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（3）T0时刻，ESP=0x007ff000。现运行进程PA进程执行语句L。请写出语句L的寻址过程。

```
int n;
int main()
{
        n = 2000；
        int result = factorial(n)；
        printf（"The result is %d", result）；
}
int factorial(int n)
{
        if(n>1)
L：          return(n* factorial(n-1));
        else
             return(1);
}
```



堆栈需扩展

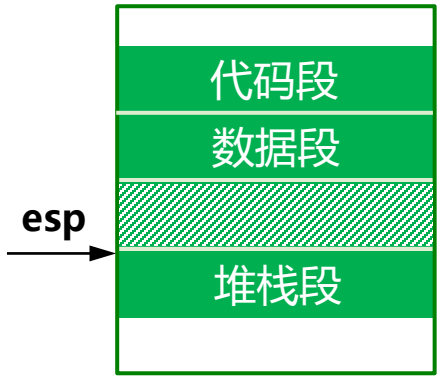
复习

例题

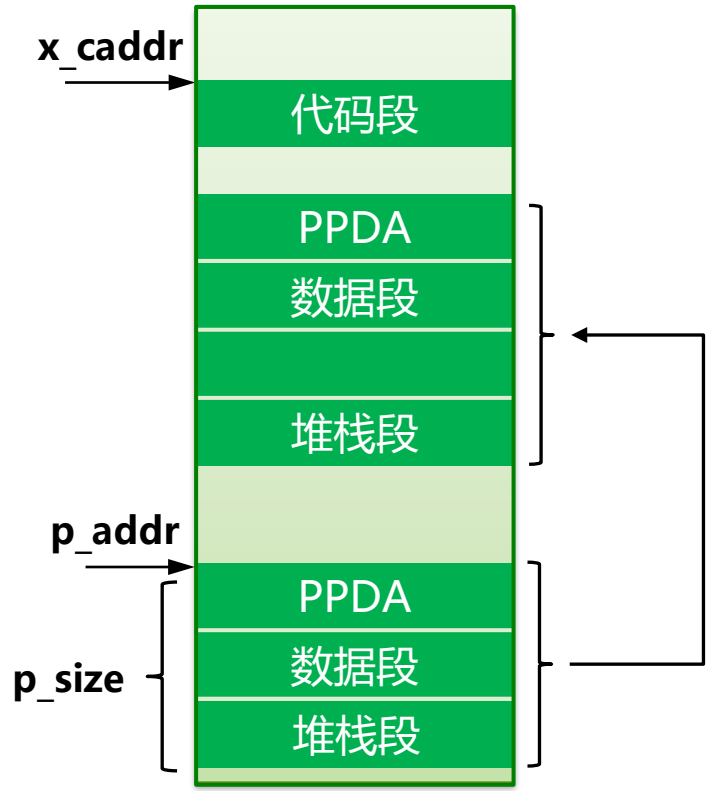
某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（3）T0时刻，ESP=0x007ff000。现运行进程PA进程执行语句L。请写出语句L的寻址过程。

- 申请分配4个连续物理页框，起始页框号假设为new；可交换部分复制到新空间；原有堆栈页向高地址区平移（复制）；



堆栈需扩展



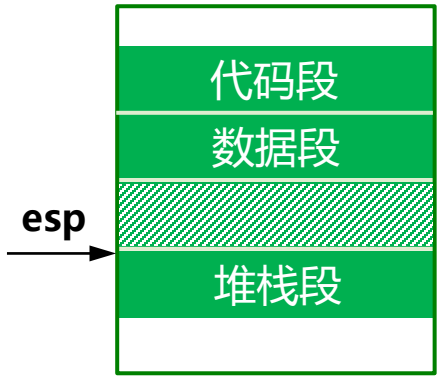
复习

例题

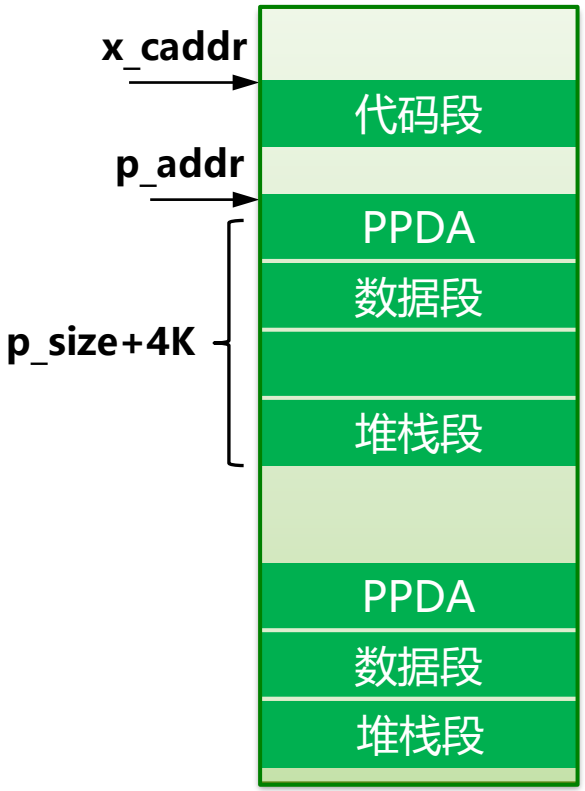
某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（3）T0时刻，ESP=0x007ff000。现运行进程PA进程执行语句L。请写出语句L的寻址过程。

- 申请分配4个连续物理页框，起始页框号假设为new；可交换部分复制到新空间；原有堆栈页向高地址区平移（复制）；
- 修 Process 对 象 ： p_size = p_size+4K， p_addr指向new起始的主存页框起始地址；
- 释放原有的内存空间；



堆栈需扩展



复习

例题

某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（3）T0时刻，ESP=0x007ff000。现运行进程PA进程执行语句L。请写出语句L的寻址过程。

- 申请分配4个连续物理页框，起始页框号假设为new；可交换部分复制到新空间；原有堆栈页向高地址区平移（复制）；
- 修 Process 对 象 ： p_size = p_size+4K， p_addr指向new起始的主存页框起始地址；
- 释放原有的内存空间；
- 重填相对虚实地址映射表和memoryDescriptor；

```
PageTable*      m_UserPageTableArray = 3G+2M+48K;
unsigned long m_TextStartAddress = 4M+4K;
unsigned long m_TextSize = 4K;
unsigned long m_DataStartAddress = 4M+8K;
unsigned long m_DataSize = 4K;
unsigned long m_StackSize = 8K;
```

	PBA		s/u	r/w	p
0#	xxx		x	x	x
…	…		…	…	…
1024#	xxx		x	x	x
1025#	0		1	0	1
	1		1	1	1
	全0				
2046#	2		1	1	1
2047#	3		1	1	1

复习

例题

某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（3）T0时刻，ESP=0x007ff000。现运行进程PA进程执行语句L。请写出语句L的寻址过程。

- 申请分配4个连续物理页框，起始页框号假设为new；可交换部分复制到新空间；原有堆栈页向高地址区平移（复制）；
- 修 Process 对 象 ： p_size = p_size+4K， p_addr指向new起始的主存页框起始地址；
- 释放原有的内存空间；
- 重填相对虚实地址映射表和memoryDescriptor；
- 重填页表。

Page Directory（0x200号页框）

	Page Base Address		s/u	r/w	p
0#	0x202		1	1	1
1#	0x203		1	1	1
	.....				
768#	0x201		0	1	1
	.....				

Page Table 0# （0x202号页框）

PBA		s/u	r/w	p
/		/	/	/

Page Table 768# （0x200号页框）

	PBA		s/u	r/w	p
0#	0		0	1	1
1#	1		0	1	1
	.....				
	new		0	1	1

Page Table 1# （0x203号页框）

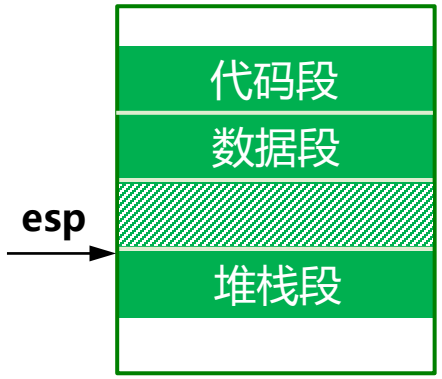
	PBA		s/u	r/w	p
0#	/		/	/	/
1#	0x500		1	0	1
	new+1		1	1	1
	.....		...	...	...
1022#	new+2		1	1	1
1023#	new+3		1	1	1

复习

例题

某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（4）若采用页式（离散）主存管理方式?



堆栈需扩展

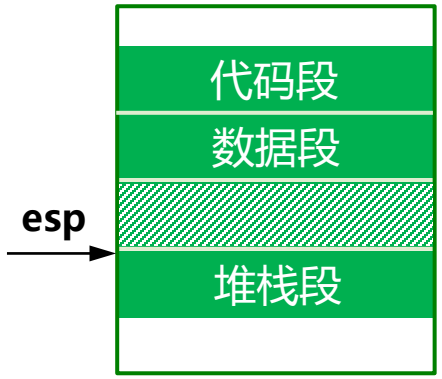
复习

例题

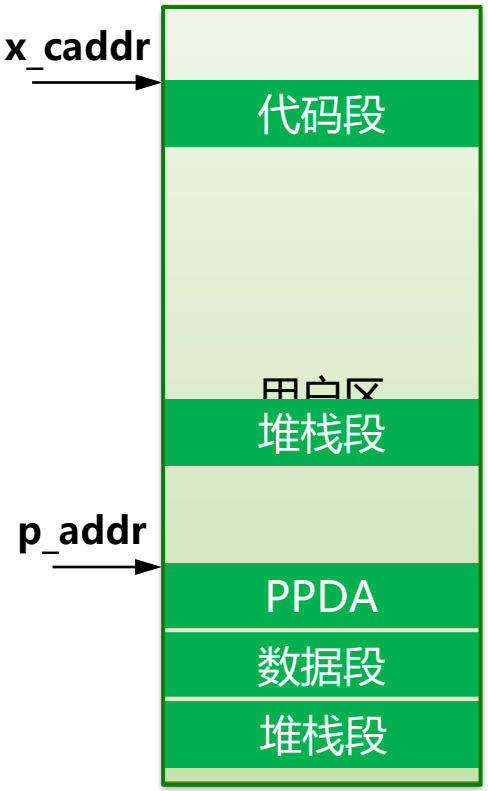
某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

(4) 若采用页式（离散）主存管理方式?

- 申请分配1个~~连续~~物理页框，起始页框号假设为new；~~可交换部分复制到新空间；原有堆栈页向高地址区平移（复制）~~；
- 修 Process 对 象 ： p_size = p_size+4K，~~p_addr指向new起始的主存页框起始地址；~~
- ~~释放原有的内存空间；~~



堆栈需扩展



复习

例题

某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

（4）若采用页式（离散）主存管理方式？

- 申请分配1个~~连续~~物理页框，起始页框号假设为new；~~可交换部分复制到新空间；原有堆栈页向高地址区平移（复制）~~；
- 修 Process 对 象 ： p_size = p_size+4K，~~p_addr指向new起始的主存页框起始地址；~~
- ~~释放原有的内存空间；~~
- 重填相对虚实地址映射表和memoryDescriptor；

```
PageTable*      m_UserPageTableArray = 3G+2M+48K;
unsigned long m_TextStartAddress = 4M+4K;
unsigned long m_TextSize = 4K;
unsigned long m_DataStartAddress = 4M+8K;
unsigned long m_DataSize = 4K;
unsigned long m_StackSize = 8K;
```

	PBA		s/u	r/w	p
0#	xxx		x	x	x
	…		…	…	…
1024#	xxx		x	x	x
1025#	0		1	0	1
	1		1	1	1
	全0				
2046#	XXX		1	1	1
2047#	2		1	1	1

复习

例题

某UNIX V6++系统中，进程PA执行下图所示的应用程序。代码段长1页，数据段长1页，堆栈段长1页。请回答：

(4) 若采用页式（离散）主存管理方式?

- 申请分配1个~~连续~~物理页框，起始页框号假设为new；~~可交换部分复制到新空间；原有堆栈页向高地址区平移（复制）~~；
- 修 Process 对 象 ： p_size = p_size+4K，~~p_addr指向new起始的主存页框起始地址；~~
- ~~释放原有的内存空间；~~
- 重填相对虚实地址映射表和memoryDescriptor；
- ~~重填页表。~~修正页表的2046# PTE。

Page Directory（0x200号页框）

	Page Base Address		s/u	r/w	p
0#	0x202		1	1	1
1#	0x203		1	1	1
	.....				
768#	0x201		0	1	1
	.....				

Page Table 0# （0x202号页框）

PBA		s/u	r/w	p
/		/	/	/

Page Table 768# （0x200号页框）

	PBA		s/u	r/w	p
0#	0		0	1	1
1#	1		0	1	1
	.....				
	0x600		0	1	1

Page Table 1# （0x203号页框）

	PBA		s/u	r/w	p
0#	/		/	/	/
1#	0x500		1	0	1
	0x601		1	1	1
	.....		...	...	...
1022#	new		1	1	1
1023#	0x602		1	1	1