选择

1. 下列关于虚存和缓存的说法中，**正确**的是：

A. TLB 是基于物理地址索引的高速缓存

B. 多数系统中，SRAM 高速缓存基于虚拟地址索引

C. 在进行**线程**切换后，TLB条目绝大部分会失效

D. 多数系统中，在进行进程切换后，SRAM 高速缓存中的内容不会失效

答案：D

解析：属于中等题。考察虚拟内存和高速缓存的关系，并且和进程/线程有一定联系。

A 课本原话：A TLB is a small, virtually addressed cache where each line holds a block consisting of a single PTE.

B 课本原话：Although a detailed discussion of the trade-offs is beyond our scope here, most systems opt for physical addressing.

C 线程共享虚拟地址空间，所以 TLB 中条目不会失效

D 课本原话：With physical addressing, it is straightforward for multiple processes to have blocks in the cache at the same time and to share blocks from the same virtual pages.

1. 阅读下列代码并回答选项。（已知文件“input.txt”中的内容为“12”，头文件没有列出）

void \*Mmap(void \*addr, size\_t length, int prot, int flags,

int fd, off\_t offset);

int main(){

int status;

int fd = Open("./input.txt", O\_RDWR);

char\* bufp = Mmap(NULL, 2, PROT\_READ | PROT\_WRITE ,

MAP\_PRIVATE, fd, 0);

*if* (Fork()>0){

*while*(waitpid(-1,&status,0)>0);

\*(bufp+1) = '1';

Write(1, bufp, 2); // 1: stdout

bufp = Mmap(NULL, 2, PROT\_READ, MAP\_PRIVATE, fd, 0);

Write(1, bufp, 2);

}

*else*{

\*bufp = '2';

Write(1, bufp, 2);

}

close(fd);

return 0;

}

在shell中运行该程序，正常运行时的终端输出应为

1. 221112 B. 222121 C. 222112 D. 221111

答案：A

解析：属于中等题。主要考察对私有对象COW机制及mmap函数的理解。

程序打开文件描述符fd后，使用mmap函数创建了一块映射到文本“./input.txt”所在区域的私有虚拟内存区域，其地址存储在指针bufp中。父进程创建完子进程后，首先等待子进程结束。子进程在试图对\*bufp写入‘2’时，触发了一次COW。它在物理内存中创建一个“input.txt”文本所在页面的副本，并在新的页面里完成写入操作，写入完成后子进程bufp所在地址的第一个字节值为新写入的‘2’，第二个字节值为原始值‘2’。子进程结束后，其缓冲区内”21”的值正常输出。父进程在试图对\*(bufp+1)写入‘1’，同样触发了一次COW，bufp所在地址的第一个字节为原始值‘1’，第二个字节为新写入的‘1’，故父进程输出“11”。最后父进程重新创建一块映射到“input.txt”所在页面的虚拟内存并输出。由于上述写入都是在新的物理页面下完成的，“input.txt”所在页面没有发生修改，故最后一步父进程的输出为“12”。综上，最终在终端的输出为“221112”。

大题（15分）

IA32体系采用**小端法**、32位虚拟地址和两级页表。两级页表大小相同，页大小都是4 KB = 212 Byte，结构也相同。TLB 采用**直接映射**，4位组索引。TLB 和页表每一项格式如图所示：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 12 | 11 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address of 4KB page frame | Ignored | G | P  A  T | D | A | P  C  D | P  W  T | U  /  S | R  /  W | P |

部分位的含义如下：

0 (P): 1表示存在，0表示不存在

1 (R/W)：1表示可写，0表示只读

2 (U/S)：1表示内核模式，0表示用户模式

当系统运行到某一时刻时，TLB**有效位为1**的条目如下（未列出部分都是无效的）：

|  |  |  |
| --- | --- | --- |
| 索引（十进制） | TLB 标记 | 内容 |
| 0 | 0x0400 | 0x0ec91313 |
| 3 | 0x02ff | 0x5d2bac01 |
| 5 | 0xd551 | 0x019fa42d |
| 11 | 0x55a6 | 0xfdd3c66b |
| 13 | 0x5515 | 0xb591926b |

一级页表的基地址为0x00e66000，物理内存中的部分内容如下（均为十六进制）：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 地址 | 内容 | 地址 | 内容 | 地址 | 内容 |
| 00615000 | 21 | 00615001 | 2d | 00615002 | ee |
| 00615003 | c0 | 006154d0 | ff | 006154d1 | a0 |
| 00e66001 | a1 | 00e66002 | a4 | 00e66003 | 67 |
| 00e66004 | 21 | 00e66005 | 57 | 00e66006 | 61 |
| 00e66007 | 00 | 2167e000 | 42 | 2167e001 | 67 |
| 2167e002 | 9a | 2167e003 | 7c | c0ee2000 | 6f |
| c0ee2001 | d5 | c0ee2002 | 7e | c0ee24d0 | 48 |
| c0ee24d1 | 83 | c0ee24d2 | ec | c0ee2d21 | 11 |
| c0ee2d22 | 6b | c0ee2d23 | 82 | c0ee2d24 | 8a |

1. 将 cache 清空。访问一个在主存中的虚拟地址，TLB命中，**没有**触发缺页异常，这一过程中，需要访问物理内存（主存）\_\_\_\_\_次（3分）。具体来说，如果该虚拟地址为 y = 0xd5515213，y 地址所具有的**实质权限**是\_\_\_\_\_\_\_\_\_（多选题，选对才得分2分）。

①可写 ②只读 ③用户模式权限 ④内核模式权限

1. 不考虑第一小问，将 cache 清空。访问一个在主存中的虚拟地址，TLB不命中，**没有**触发缺页异常，这一过程中，需要访问物理内存（主存）\_\_\_\_\_次（3分）。具体来说，如果该虚拟地址为x = 0x004004d0，那么x 对应的二级页表起始地址是\_\_\_\_\_\_\_\_\_\_\_\_\_\_（填写16进制，例如0x00123000，2分），x 地址上单字节的**内容**是\_\_\_\_\_\_（填写16进制，例如0x00，1分）。
2. 考虑下面计算矩阵和向量乘法代码：

1 int \*mat\_vec\_mul(int \*\*A, int \*x, int n)

2 {

3 int i, j;

4 int \*y = (int \*)malloc(n \* sizeof(int));

5 *for* (i = 0; i < n; i++)

6 *for* (j = 0; j < n; j++)

7 y[i] += A[i][j] \* x[j];

8 *return* y;

9 }

⑴ 在64位 Linux 机器中运行该代码，输入**同一组合法的参数**后，每次运行返回的向量的值都**不一样**，修复这一错误有一种简单的方法，将第\_\_\_\_行改为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（第二空填写C代码，每空1分，共2分）

可能用到的函数：

void \*memcpy(void \*dest, const void \*src, size\_t n);

void \*memset(void \*s, int c, size\_t n);

void \*calloc(size\_t nelem, size\_t elemsize);

void \*realloc(void \*ptr, size\_t size);

⑵ 在进行前一问的测试**之前**，还在64位 Linux 机器上进行过如下用户代码测试（输入 mat\_vec\_mul的参数都**非零**）：

int \*y = mat\_vec\_mul(A, x, n);

int z = y[0];

结果发生了段错误。通过gdb调试发现mat\_vec\_mul函数内并没有发生段错误，但是在初始化变量z时发生了段错误，后来发现是参数的输入有问题。写出出现这种错误的**充分必要条件**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_（1分）

1. Double fault：Intel处理器中有一种特殊的异常，被称为double fault。此异常发生表明调用某个**故障（fault）**A的处理程序后又触发了另一个故障B。正常情况下，故障B会有相应异常处理程序来处理，因此两个故障B和A可以被顺序解决。但是如果处理器无法正常处理故障B，或是处理了之后依然无法处理故障A，就会产生double fault，并终止（abort）。假设除了缺页异常处理程序外，其他异常处理程序**都不会产生新的故障**。如果在某次**缺页故障**时产生了double fault，其原因可能是\_\_\_\_\_\_\_\_\_\_（不定项选择，都选对才得分，1分）

① 运行缺页故障处理程序时， CPU上的权限位是内核态，但所执行代码段 U/S=0

② 运行缺页故障处理程序时，CPU接收到了键盘发送的Ctrl + C信号

③ 缺页故障处理程序没有加载到主存中

答案：

1. 1 ②④
2. 3 0x00615000 0x48
3. ⑴ 4 int \*y = (int \*)calloc(n, sizeof(int)); 或

int \*y = (int \*)Calloc(n, sizeof(int));

其中 calloc/Calloc 的两个参数只要乘起来等于 n \* sizeof(int) 即算对

⑵ n < 0（或其他等价表达）

1. ③

解析：

1. 考察 TLB 的位划分与表项含义，属于简单题。

题目告知TLB命中且无缺页，因此只需要访问物理页，即访问一次物理内存

根据 y 的值可以得到 TLBT=0xd551，TLBI=5，查表知 y 在TLB条目的内容为0x019fa42d，R/W=0，U/S=1，因此这一页的实质权限为只读、内核模式。

参见课本第9.6和9.7节（P567-582）

1. 考察虚拟内存地址翻译，属于简单+中等题。

题目告知TLB未命中且无缺页，因此需要访问页目录、二级页表和物理页，共访问三次物理内存。

计算得x 的VPN1=1， VPN2=0，PPO =0x4d0。页目录基地址在 0x00e66000，因此相应页目录中所在条目地址为 0x00e66000+1\*4=0x00e66004，从表中按小端法读出0x00615721，因此二级页表存在，首地址为0x00615000，这恰好也是相应二级页表中所在条目的地址，从表中按小端法读出0xc0ee2d21，因此物理页存在，首地址为0xc0ee2000，因此x的物理地址为0xc0ee2000+0x4d0=0xc0ee24d0。从表中读出一个字节，为0x48。

参见课本第9.6和9.7节（P567-582）

1. 本题考查内存安全。

⑴ 考察malloc函数的性质，属于中等题。

malloc函数并不会清零所分配内存区域，因此同样的输入可能会有不同的输出，但mat\_vec\_mul函数别的行都不可能产生这一效果，因此只需要将第4行换为

int \*y = (int \*)Calloc(n, sizeof(int));

Calloc是一个带错误检查的、会将所分配区域置零的函数。

参见课本第9.11.2节正文例子（P610）。

⑵ 内存安全的综合考察，属于难题。

假设malloc函数分配成功，那么y不是空指针。y[0]不可能在外部访问出现段错误，因此这种情况不成立，malloc一定出现了问题，导致y是空指针。但是此时 mat\_vec\_mul 并没有触发段错误，说明函数内部一次对y的访问都没有，这只有可能循环都没有进入，因此必须有n<=0，根据题设，参数是非零的，所以n<0。以上推导出了n<0是必要条件。

下面推导n<0是充分条件，即任何n<0都会导致这样的错误。为此，只需要说明malloc函数一定分配失败。首先，这是Linux 64位系统，因此地址空间为64位，size\_t为64位，用户虚拟地址空间为48位。注意n\*sizeof(int) 会将n的类型转为long，再转为unsigned long，而n只有32位，所以符号扩展之后最高位至少有33个1，乘sizeof(int)，即4之后，最高位还有至少29个1，因此n\*sizeof(int)>263>248，超过了用户虚拟地址空间的大小，因此无论如何，malloc函数都会分配失败。

参见课本第9.8节（P576）、9.9.1节（P588）、2.2.8节（P58-59）。

1. 本题考查对缺页故障、异常处理的综合理解，属于难题。

① 如果CPU权限位是内核态，它自然可以运行用户级代码，不会触发异常。

② 如果缺页故障处理程序运行时，收到外部中断，这不属于故障，因此不会触发。

③ 如果缺页故障处理程序不在主存，那么调用它会触发缺页故障，于是这个故障又会导致调用缺页故障处理程序，但它仍然不在主存，故这一故障无法被解决，于是触发double fault。

相关内容参见课本第8.1节（P502-507）、9.7.2节（P581-582）和Intel 64 and IA-32 Architectures Software Developer’s Manual - Volume 3: System Programming Guide 的第6.15节。

备选题

1. 虚拟内存为内存的使用和管理提供了简化，这样的简化**没有**体现在

A. 编译器将C文件编译为目标文件的过程

B. 链接器生成完全链接的可执行文件的过程

C. 加载器向内存中加载可执行文件和共享对象文件的过程

D. 不同进程共享相同物理页面的过程

答案：A

解析：属于简单题。考察虚拟内存在管理内存方面起到的作用，主要对应中文版课本第566页。

A 程序的编译过程与使用虚拟内存还是使用物理内存的联系较小。

B 课本原话：…Such uniformity greatly simplifies the design and implementation of linkers, allowing them to produce fully linked executables that are independent of the ultimate location of the code and data in physical memory.

C. 课本原话：Virtual memory also makes it easy to load executable and shared object files into memory.

D. 课本原话：… the operating system can arrange for multiple processes to share a single copy of this code by mapping the appropriate virtual pages to in different processes to the same physical pages.

1. 下列选项中**错误**的是
2. 在使用虚拟地址空间的的系统中，程序引用的页面总数**必须不超过**物理内存总大小。
3. **主存中**的每个**有效**字节都有至少一个选自虚拟地址空间的**虚拟地址**和一个选自物理地址空间的**物理地址**。
4. 当在程序中正常调用malloc函数时，操作系统会分配出相应大小（例如k个）的**连续虚拟页面**，并且将它们映射到物理内存中**任意位置**的k个**物理页面**。
5. **不同**进程的多个**虚拟**页面可以映射到**同一个**共享**物理**页面上。

答案：A

解析：属于简单题。主要考察地址空间和虚拟内存相关的基本概念。

A. 如果运行的程序有良好的时间局部性，工作集大小（引用的页面总数）即使超过物理内存总的大小也能有较好的性能。

B. 课本原话：[p561] Each byte of main memory has a virtual address chosen from the virtual address space, and a physical address chosen from the physical address space.

C. 课本原话：[p567] When a program […] requests additional heap space […], the operating system allocates an appropriate number […] contiguous virtual memory pages, and maps them […] arbitrary physical pages located anywhere in physical memory.

D. 课本原话：[p566] Notice that multiple virtual pages can be mapped tothe same shared physical page.

1. 以下关于动态内存分配的说法中，**错误**的是

A. 可以通过调用 sbrk(0) 获取当前进程中堆的顶部地址

B. 如果向一个已经 free 的指针写入数据，**一定会**触发异常

C. 如果使用 malloc(0x10) 获取一个指针，然后写入 0x200 字节大小的数据，**不一定**会触发异常

D. 使用 mmap 也是动态分配内存的方法之一

答案：B

解析：属于中等题。考察对动态内存分配的理解，并与内存保护机制，异常有一定联系。

A 课本原话：If incr is zero, then sbrk returns the current value of brk.

B 已经分配的内存一般不会被 OS 回收，相当于用户空间正常的内存访问

C 只要不超出堆的范围，就不会触发异常。如果超出堆的范围，则触发异常

D 课本原话：While it is certainly possible to use the low-level mmap and munmap functions to create and delete areas of virtual memory, C programmers typically find it more convenient and more portable to use a dynamic memory allocator when they need to acquire additional virtual memory at run time.

1. 以下关于虚拟内存的说法**错误**的是

A. 虚拟内存一般不需要来自应用程序开发者的干涉

B. 虚拟地址空间可以比物理内存更小

C. 连续的虚拟内存**总是**映射到连续的物理内存

D. 目标文件中的.bss段映射到全是二进制零的匿名文件

答案：C

解析：属于中等题。主要考察虚拟内存相关的理解问题。

A. 课本原话：… [virtual memory] works silently and automatically, without any intervention from the application programmer.

B. 课本原话：Interestingly, some early systems […] supports a virtual address space that was smaller than the available physical memory.

C. 课本原话：… the operating system allocates […] contiguous virtual memory pages […] The pages can be scattered randomly in physical memory.

D. 课本原话：The bss area is demand-zero, mapped to an anonymous file […]