第 8&9 章家庭作业

——201808010515　计科 1805 黄茂荣

8.18 考虑下面的程序：

code/ecf/forkprob2.c

```
1   #include "csapp.h"
2
3   void end(void)
4   {
5       printf("2");
6   }
7
8   int main()
9   {
10      if (Fork() == 0)
11          atexit(end);
12      if (Fork() == 0)
13          printf("0");
14      else
15          printf("1");
16      exit(0);
17  }
```

code/ecf/forkprob2.c

判断下面哪个输出是可能的。注意：atexit 函数以一个指向函数的指针为输入，并将它添加到函数列表中（初始为空），当 exit 函数被调用时，会调用该列表中的函数。

A. 112002

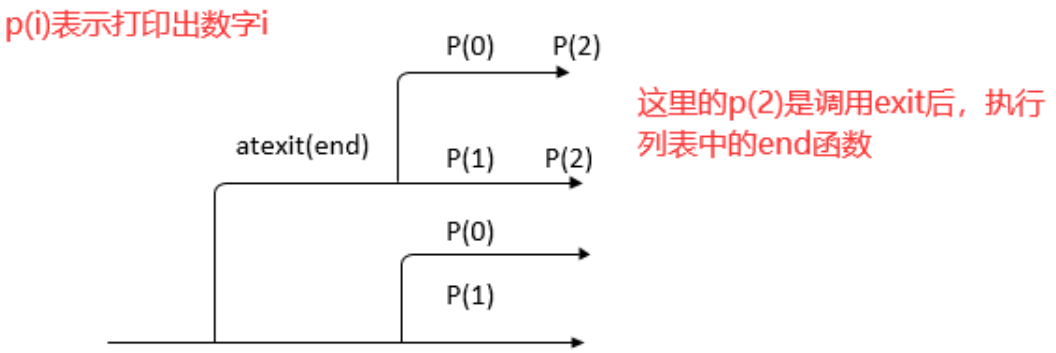
B. 211020

C. 102120

D. 122001

E. 100212

可能的输出有 A,C,E。分析：从题中可以知道 atexit 函数会将一个函数添加到函数列表中去，当 exit 被调用时，会执行列表中的函数。根据 main 函数的内容画出如下的进程图，每个水平的箭头对应于从左到右执行指令的过程，每个垂直的箭头对应于 fork 函数的执行。



由图可以看出输出序列有 6 个数字，其中必须有 12,02，而 B 中没有包含 12，D 中没有包含 02，故选择 A,C,E。

8.23 你的一个同事想要使用信号来让一个父进程对发生在一个子进程中的事件计数。其思想是每次发生一个事件时，通过向父进程发送一个信号来通知它，并且让父进程的信号处理程序对一个全局变量counter加一，在子进程终止之后，父进程就可以检查这个变量。然而，当他在系统上运行图8-41中的测试程序时，发现当父进程调用printf时，counter的值总是2，即使子进程向父进程发送了5个信号。他很困惑，向你寻求帮助。你能解释这个程序有什么错误吗？

```
1   #include "csapp.h"
2
3   int counter = 0;
4
5   void handler(int sig)
6   {
7       counter++;
8       sleep(1); /* Do some work in the handler */
9       return;
10  }
11
12  int main()
13  {
14      int i;
15
16      Signal(SIGUSR2, handler);
17
18      if (Fork() == 0) {  /* Child */
19          for (i = 0; i < 5; i++) {
20              Kill(getppid(), SIGUSR2);
21              printf("sent SIGUSR2 to parent\n");
22          }
23          exit(0);
24      }
25
26      Wait(NULL);
27      printf("counter=%d\n", counter);
28      exit(0);
29  }
```

co

图 8-41　家庭作业 8.23 中引用的计数器程序

在子进程中时，代码中用 for 循环发送 5 次信号给父进程，但实际计数值只有 2。可能的原因是：子进程发送第 1 个信号后，父进程接受并捕获了第 1 个信号，当程序还在处理第 1 个信号时，第 2 个信号发送了并添加到了待处理信号集合里。接着，在程序还在处理第 1 个信号时，第 3 个信号到达了，但因为同一类型的待处理信号不会排队等候，故第 3 个信号被丢弃，同理第 4,5 个信号也被丢弃。一段 时间后，第 1 个信号处理结束，内核注意到有一个待处理的信号，此时父进程就会接收这个信号并进行处理。当处理结束后，由于第 3,4,5 个信号已被丢弃，故没有信号可以进行处理了，counter 计数值为 2。

从题中可以看到不能用信号来对其他进程中发生的事件计数。

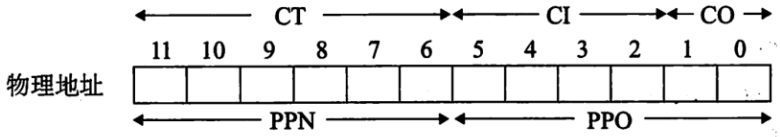
下图为 9.6.4 节示例的相关存储系统

位	标记位	PPN	有效位	标记位	PPN	有效位	标记位	PPN	有效位	标记位	PPN	有效位
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

a) TLB：四组，16 个条目，四路组相联

VPN	PPN	有效位	VPN	PPN	有效位
00	28	1	08	13	1
01	—	0	09	17	1
02	33	1	0A	09	1
03	02	1	0B	–	0
04	—	0	0C	–	0
05	16	1	0D	2D	1
06	—	0	0E	11	1
07	—	0	0F	0D	1

b) 页表：只展示了前 16 个 PTE



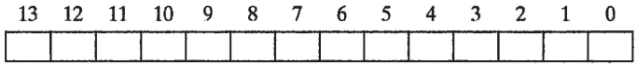
索引	标记位	有效位	块 0	块 1	块 2	块 3
0	19	1	99	11	23	11
1	15	0	—	—	—	—
2	1B	1	00	02	04	08
3	36	0	—	—	—	—
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	—	—	—	—
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	—	—	—	—
A	2D	1	93	15	DA	3B
B	0B	0	—	—	—	—
C	12	0	—	—	—	—
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	—	—	—	—

c) 高速缓存：16 个组，4 字节的块，直接映射

9.11 在下面的一系列问题中，你要展示 9.6.4 节中的示例存储器系统如何将虚拟地址翻译成物理地址，以及如何访问缓存。对于给定的虚拟地址，请指出访问的 TLB 条目、物理地址，以及返回的缓存字节值。请指明是否 TLB 不命中，是否发生了缺页，是否发生了缓存不命中。如果有缓存不命中，对于“返回的缓存字节”用“–”来表示。如果有缺页，对于“PPN”用“–”来表示，而 C 部分和 D 部分就空着。

虚拟地址：0x027c

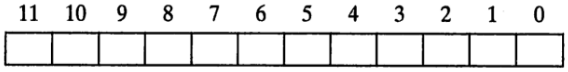
A. 虚拟地址格式



B. 地址翻译

参数	值
VPN	
TLB 索引	
TLB 标记	
TLB 命中？（是/否）	
缺页？（是/否）	
PPN	

C. 物理地址格式



D. 物理地址引用

参数	值
字节偏移	
缓存索引	
缓存标记	
缓存命中？（是/否）	
返回的缓存字节	

根据 9.6.4 节示例的存储系统模型知，每个页面的大小为 2^6=64 字节，故虚拟地址和物理地址中低 6 位分别作为 VPO 和 PPO，虚拟地址的高 8 位作为 VPN，物理地址的高 6 位作为 PPN。TLB 有 4 各组，四路组相连。高速缓存是 16 个组，4 字节的块，直接映射。

A 虚拟地址格式

虚拟地址为 0x027c，化成 2 进制如下：

0	0	0	0	1	0	0	1	1	1	1	1	0	0

B 地址翻译

根据上面的分析，VPN 是虚拟地址高 8 位，即 00001001=0x09,因为 TLB 有 4 个组，所以 VPN 的低 2 位是 TLB 索引，即 0x01,高 6 位为 TLB 标记，即 0x02。根据 TLB 索引和 TLB 标记，在表中查找，没有找到，说明 TLB 为没有命中。接着到主存中寻找 PTE，有效位为 1，找到 PPN 为 0x17，能找到说明没有缺页。

位	标记位	PPN	有效位	标记位	PPN	有效位	标记位	PPN	有效位	标记位	PPN	有效位
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

a) TLB：四组，16 个条目，四路组相联

根据TLBI和TLBT查找

VPN	PPN	有效位
00	28	1
01	—	0
02	33	1
03	02	1
04	—	0
05	16	1
06	—	0
07	—	0

VPN	PPN	有效位
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

发现未命中，从主存中取出相应的PTE，对应PPN的值为0x17

b) 页表：只展示了前 16 个 PTE

参数	值
VPN	0x09
TLB 索引	0x01
TLB 标记	0x02
TLB 命中（是/否）	否
缺页（是/否）	否
PPN	0x17

C 物理地址格式

将 PPN=0x17 和 PPO=VPO=0x3c 和起来得到物理地址 0x5fc

0	1	0	1	1	1	1	1	1	1	0	0

D 物理地址引用

根据上面的分析，每个块是 4 字节，物理地址的低 2 位作为块偏移（CO），即 0x00，有 16 组，故组索引（CI）为 4 位,即 0x0f，最后剩下的高 6 位是标记（CT）,即 0x17。根据缓存索引和缓存标记，在表中查找，发现没有与之对应的内容，故缓存不命中。

索引	标记位	有效位	块 0	块 1	块 2	块 3
0	19	1	99	11	23	11
1	15	0	—	—	—	—
2	1B	1	00	02	04	08
3	36	0	—	—	—	—
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	—	—	—	—
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	—	—	—	—
A	2D	1	93	15	DA	3B
B	0B	0	—	—	—	—
C	12	0	—	—	—	—
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	—	—	—	—

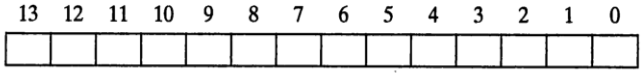
c) 高速缓存：16 个组，4 字节的块，直接映射

参数	值
字节偏移	0x00
缓存索引	0x0f
缓存标记	0x17
缓存命中（是/否）	否
返回的缓存字节	—

9.12 对于下面的地址，重复习题 9.11：

虚拟地址：0x03a9

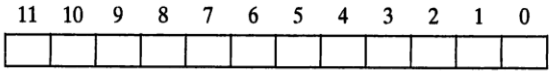
A. 虚拟地址格式



B. 地址翻译

参数	值
VPN	
TLB 索引	
TLB 标记	
TLB 命中？（是/否）	
缺页？（是/否）	
PPN	

C. 物理地址格式



D. 物理地址引用

参数	值
字节偏移	
缓存索引	
缓存标记	
缓存命中？（是/否）	
返回的缓存字节	

根据 9.6.4 节示例的存储系统模型知，每个页面的大小为 2^6=64 字节，故虚拟地址和物理地址中低 6 位分别作为 VPO 和 PPO，虚拟地址的高 8 位作为 VPN，物理地址的高 6 位作为 PPN。TLB 有 4 各组，四路组相连。高速缓存是 16 个组，4 字节的块，直接映射。

A 虚拟地址格式

虚拟地址为 0x03a9，化成 2 进制如下：

0	0	0	0	1	1	1	0	1	0	1	0	0	1

B 地址翻译

根据上面的分析，VPN 是虚拟地址高 8 位，即 00001110=0x0e,因为 TLB 有 4 个组，所以 VPN 的低 2 位是 TLB 索引，即 0x02,高 6 位为 TLB 标记，即 0x03。根据 TLB 索引和 TLB 标记，在表中查找，没有找到，说明 TLB 为没有命中。接着到主存中寻找 PTE，有效位为 1，找到 PPN 为 0x11，能找到说明没有缺页。

位	标记位	PPN	有效位	标记位	PPN	有效位	标记位	PPN	有效位	标记位	PPN	有效位
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	–	1	02	–	0

a) TLB：四组，16 个条目，四路组相联

根据TLBI和TLBT查找

VPN	PPN	有效位
00	28	1
01	—	0
02	33	1
03	02	1
04	—	0
05	16	1
06	—	0
07	—	0

VPN	PPN	有效位
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

没有找到，接着到主存中找PTE，有效位为1，VPN=0xE找到PPN为0x11

b) 页表：只展示了前 16 个 PTE

参数	值
VPN	0x0e
TLB 索引	0x02
TLB 标记	0x03
TLB 命中（是/否）	否
缺页（是/否）	否
PPN	0x11

C 物理地址格式

将 PPN=0x11 和 PPO=VPO=0x29 起来得到物理地址 0x469

0	1	0	0	0	1	1	0	1	0	0	1

D 物理地址引用

根据上面的分析，每个块是 4 字节，物理地址的低 2 位作为块偏移（CO），即 0x01，有 16 组，故组索引（CI）为 4 位,即 0x0a，最后剩下的高 6 位是标记（CT）,即 0x11，根据缓存索引和缓存标记，在表中查找，发现没有与之对应的内容，故缓存不命中。

索引	标记位	有效位	块 0	块 1	块 2	块 3
0	19	1	99	11	23	11
1	15	0	—	—	—	—
2	1B	1	00	02	04	08
3	36	0	—	—	—	—
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	—	—	—	—
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	—	—	—	—
A	2D	1	93	15	DA	3B
B	0B	0	—	—	—	—
C	12	0	—	—	—	—
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	—	—	—	—

c) 高速缓存：16 个组，4 字节的块，直接映射

参数	值
字节偏移	0x01
缓存索引	0x0a

缓存标记	0x11
缓存命中（是/否）	否
返回的缓存字节	—