# 8.11

4行。

# 8.15

5行。

Exit不是子进程，而是return，说明两个子进程都要到回到main函数去打印那里的hello。

# 8.19

2^n行。

# 8.23

可能在第一个信号发给父进程之后，父进程进入handler，并且阻塞了SIGUSR2，第二个信号依然可以发送，然而，之后的3个信号便会被抛弃了。因为是连续发送，所以很可能是没等上下文切换，这5个信号就同时发送了。所以只有2个信号被接收。

# 9.11

A.虚拟地址0x027c

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

B.地址翻译

|  |  |
| --- | --- |
| 参数 | 值 |
| VPN | 0x09 |
| TLB索引 | 0x01 |
| TLB标记 | 0x02 |
| TLB命中 | No |
| 缺页 | No |
| PPN | 0x17 |

C.物理地址格式

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *11* | *10* | *9* | *8* | *7* | *6* | *5* | *4* | *3* | *2* | *1* | *0* |
| *0* | *1* | *0* | *1* | *1* | *1* | *1* | *1* | *1* | *1* | *0* | *0* |

D.物理地址引用

|  |  |
| --- | --- |
| 参数 | 值 |
| 字节偏移 | *0x0* |
| 缓存索引 | *0xF* |
| 缓存标记 | *0x17* |
| 缓存命中 | *No* |
| 返回缓存字节 | *-* |

# 9.15

|  |  |  |
| --- | --- | --- |
| 请求 | 块大小 | 块头部 |
| *malloc(3)* | *8* | *0x9* |
| *malloc(11)* | *16* | *0x11* |
| *malloc(20)* | *24* | *0x19* |
| *malloc(21)* | *32* | *0x21* |

# 9.19

1. a; 对于伙伴系统，如果要申请大小为33的空间，那么需要分配64个空间。如果申请大小为65的空间，那么块大小就需要128，所以最多可能有约50%的空间被浪费。b中，最佳适配要搜索所有空间，所以肯定比首次适配要慢一些。c，边界标记主要功能是释放一个块时，能立即和前后空闲块合并。如果空闲块不按顺序排列的话，其实也能够和前一个或者后一个空闲块进行合并，但如果要和前后一起合并，可能会有些困难，那需要搜索前后块在空闲链表中的位置，并且删除一个再进行合并。可以参考P576，LIFO方法。d，其实任何分配器都可能有外部碎片，只要剩余的空闲块大小和足够但是单个都不够，就会产生外部碎片。
2. d; 块大小递增，那么最佳适配法找到的块和首次适配找到的块是同一个，因为最佳适配总是想找一个刚好大于请求块大小的空闲块。a，块大小递减，首次适配很容易找到，所以分配性能会很高。b，最佳适配方法无论怎样，都要搜索所有的链表（除非维护成块大小递增的链表）。c，是匹配的最小的。
3. c; 保守的意思就是所有可能被引用的堆都会被标记，int像指针，所以可能认为它表示的地址是正在被引用的（实际上它只是个int）。