# Приложения

## Приложение А (Листинг DMA для современных устройств)

1. #include <linux/module.h>                      //Динамическая загрузка модулей в ядро.
2. #include <linux/pci.h>                         //определение PCI и прототипы функций
3. #include <linux/slab.h>                        //Распределение памяти
4. #include <linux/dma-mapping.h>               //Список возможных атрибутов, связанных с отображением DMA.
5. #include <linux/dmapool.h>
6. #include "out.c"
7. MODULE\_LICENSE( "GPL" );
8. #define pool\_size 1024
9. #define pool\_align 8
11. /\* обмен данными будет осуществляться в обоих направлениях \*/
12. **static** **int** direction = PCI\_DMA\_BIDIRECTIONAL;
13. // int direction = PCI\_DMA\_TODEVICE ;       //Обмен данными в направлении устройства
14. // int direction = PCI\_DMA\_FROMDEVICE ;     //Обмен данными в направлении от устройства
15. //int direction = PCI\_DMA\_NONE;             //Блокировка обмена данными
17. **static** **int** \_\_init my\_init( **void** )
18. {
19. **char** \*kbuf;                             //буфер DMA
21. dma\_addr\_t handle;                      /\* Dma\_addr\_t может содержать любой действительный адрес DMA или шины для платформы. Оно может
22. \* передаваться устройству для использования в качестве источника или цели DMA. Это характерно для
23. \* данное устройство и может быть передано между физическим адресом ЦП и адресным пространством шины
25. dma-mapping.h :#define DMA\_MAPPING\_ERROR        (~(dma\_addr\_t)0)
26. \*/
28. size\_t size = ( 10 \* PAGE\_SIZE );       //Размер страницы
29. **struct** dma\_pool \*mypool;                //devic.h : dma\_pools:  Dma pools (if dma'ble device).
31. /\* использование метода dma\_alloc\_coherent \*/
32. kbuf = dma\_alloc\_coherent( NULL, size, &handle, GFP\_KERNEL );
33. output( kbuf, handle, size, "This is the dma\_alloc\_coherent() string" );
34. dma\_free\_coherent( NULL, size, kbuf, handle );
36. /\* использование метода dma\_map/unmap\_single \*/
37. kbuf = kmalloc( size, GFP\_KERNEL );                                 // GFP\_KERNEL (\_\_GFP\_\_WAIT | \_\_GFP\_\_IO | \_\_GFP\_\_FS) - выделение производится от имени процесса,
38. //который выполняет системный запрос в пространстве ядра — такой запрос может быть временно переводиться
39. //в пассивное состояние (блокирован).
40. //slab.h
41. handle = dma\_map\_single( NULL, kbuf, size, direction );
42. output( kbuf, handle, size, "This is the dma\_map\_single() string" );
43. dma\_unmap\_single( NULL, handle, size, direction );
44. kfree( kbuf );
46. /\* использование метода dma\_pool \*/
47. mypool = dma\_pool\_create( "mypool", NULL, pool\_size, pool\_align, 0 );
48. kbuf = dma\_pool\_alloc( mypool, GFP\_KERNEL, &handle );
49. output( kbuf, handle, size, "This is the dma\_pool\_alloc() string" );
50. dma\_pool\_free( mypool, kbuf, handle );
51. dma\_pool\_destroy( mypool );
52. **return** -1;
53. }

## Приложение Б (Листинг DMA для устаревших устройств)

1. #include <linux/module.h>
2. #include <linux/pci.h>
3. #include <linux/slab.h>
4. MODULE\_LICENSE( "GPL" );
6. #include "out.c"
8. /\* обмен данными будет осуществляться в обоих направлениях \*/
9. **static** **int** direction = PCI\_DMA\_BIDIRECTIONAL;
10. // int direction = PCI\_DMA\_TODEVICE ;
11. // int direction = PCI\_DMA\_FROMDEVICE ;
12. //int direction = PCI\_DMA\_NONE;

15. **static** **int** \_\_init my\_init( **void** ) {
16. **char** \*kbuf;
17. dma\_addr\_t handle;
18. size\_t size = ( 10 \* PAGE\_SIZE );
19. /\* использование метода pci\_alloc\_consistent \*/
20. kbuf = pci\_alloc\_consistent( NULL, size, &handle );
21. output( kbuf, handle, size, "This is the pci\_alloc\_consistent() string" );
22. pci\_free\_consistent( NULL, size, kbuf, handle );
23. /\* использование метода pci\_map/unmap\_single \*/
24. kbuf = kmalloc( size, GFP\_KERNEL );
25. handle = pci\_map\_single( NULL, kbuf, size, direction );
26. output( kbuf, handle, size, "This is the pci\_map\_single() string" );
27. pci\_unmap\_single( NULL, handle, size, direction );
28. kfree( kbuf );
29. **return** -1;
30. }

## Приложение В (Листинг: Общая часть тестов)

1. **static** **int** \_\_init my\_init( **void** );
2. module\_init( my\_init );
4. MODULE\_AUTHOR( "Sergey Mirzoyan" );
6. #define MARK "=> "
8. **static** **void** output( **char** \*kbuf, dma\_addr\_t handle, size\_t size, **char** \***string** ) {
9. unsigned **long** diff;
10. diff = (unsigned **long**)kbuf - handle;
11. printk( KERN\_INFO MARK "kbuf=%12p, handle=%12p, size = %d\n",
12. kbuf, (**void**\*)(unsigned **long**)handle, (**int**)size );
13. printk( KERN\_INFO MARK "(kbuf-handle)= %12p, %12lu, PAGE\_OFFSET=%12lu, compare=%lu\n",
14. (**void**\*)diff, diff, PAGE\_OFFSET, diff - PAGE\_OFFSET );
15. strcpy( kbuf, **string** );
16. printk( KERN\_INFO MARK "string written was, %s\n", kbuf );
17. }