

qemu QOM(qemu object model)和设备模拟

原创 ayu_ag 最后发布于2016-10-20 14:46:23 阅读数 3770 ☆ 收藏

本文所用qemu为1.5版本的，不是android emulator的。

之前几篇文章介绍的都是android emulator中的设备模拟。一些是android自己加的platform bus上的虚拟设备；一些是qemu自带的虚拟设备，1旧，没有使用到QOM模型。

qemu1.1以及之后的qemu开始有了QOM模型。QOM很大一部分代码是为了实现了C++的继承，公用的东西放到ObjectClass里，一个实例；里，可以有多个实例。

PS：android emulator的代码对应了qemu 0.1x的代码，但是又有一些新版本的qemu的代码porting上去了。

QOM设备模型可以看：

- 1、Qemu中的设备注册：<http://ytliu.info/blog/2015/01/10/qemushe-bei-chu-shi-hua/>
- 2、QEMU 设备模拟：<http://mnstory.net/wp-content/uploads/2014/10/qemu-device-simulation/qemu-device-simulation.pdf>

第二篇pdf讲的非常详细了，但是最后关于PMIO地址和读写函数如何对应起来的，还是有些没清楚的地方。

本文针对这个问题进行一些补充。

初始化内存空间

在memory_map_init(main->cpu_exec_init_all->memory_map_init)中，会设置MemoryRegion改变时的回调函数，memory_map_init是在该的：

```
1 static void memory_map_init(void)
2 {
3     system_memory = g_malloc(sizeof(*system_memory));
4     memory_region_init(system_memory, "system", INT64_MAX);
5     address_space_init(&address_space_memory, system_memory);
6     address_space_memory.name = "memory";
7
8     system_io = g_malloc(sizeof(*system_io));
9     memory_region_init(system_io, "io", 65536);
10    address_space_init(&address_space_io, system_io);
11    address_space_io.name = "I/O";
12
13    memory_listener_register(&core_memory_listener, &address_space_memory);
14    memory_listener_register(&io_memory_listener, &address_space_io);
15    memory_listener_register(&tcg_memory_listener, &address_space_memory);
16
17    dma_context_init(&dma_context_memory, &address_space_memory,
18                    NULL, NULL, NULL);
19 }
```

这个是PMIO的listener，PMIO的MemoryRegion改变后，会调用io_region_add函数，映射PMIO地址和设备读写函数。

普通内存是其他的listener。

```
1 static MemoryListener io_memory_listener = {
2     .region_add = io_region_add,
3     .region_del = io_region_del,
```



1



举报

```
4 | .priority = 0,  
5 | };
```

把注册的listener添加到全局的memory_listeners链表中:

```
1 | void memory_listener_register(MemoryListener *listener, AddressSpace *filter)  
2 | {  
3 |     MemoryListener *other = NULL;  
4 |     AddressSpace *as;  
5 |  
6 |     listener->address_space_filter = filter; // listener是处理那个AddressSpace的  
7 |     if (QTAILQ_EMPTY(&memory_listeners)  
8 |         || listener->priority >= QTAILQ_LAST(&memory_listeners,  
9 |             memory_listeners->priority) {  
10 |         QTAILQ_INSERT_TAIL(&memory_listeners, listener, link);  
11 |     } else {  
12 |         QTAILQ_FOREACH(other, &memory_listeners, link) {  
13 |             if (listener->priority < other->priority) {  
14 |                 break;  
15 |             }  
16 |         }  
17 |         QTAILQ_INSERT_BEFORE(other, listener, link);  
18 |     }  
19 |  
20 |     QTAILQ_FOREACH(as, &address_spaces, address_spaces_link) {  
21 |         listener_add_address_space(listener, as);  
22 |     }  
23 | }
```

对AddressSpace(也就是根MemoryRegion)中的每一个MemoryRegion进行一下listener->region_add

```
1 | static void listener_add_address_space(MemoryListener *listener,  
2 |     AddressSpace *as)  
3 | {  
4 |     FlatRange *fr;  
5 |  
6 |     if (listener->address_space_filter  
7 |         && listener->address_space_filter != as) {  
8 |         return;  
9 |     }  
10 |  
11 |     if (global_dirty_log) {  
12 |         if (listener->log_global_start) {  
13 |             listener->log_global_start(listener);  
14 |         }  
15 |     }  
16 |  
17 |     FOR_EACH_FLAT_RANGE(fr, as->current_map) {  
18 |         MemoryRegionSection section = {  
19 |             .mr = fr->mr,  
20 |             .address_space = as,  
21 |             .offset_within_region = fr->offset_in_region,  
22 |             .size = int128_get64(fr->addr.size),  
23 |             .offset_within_address_space = int128_get64(fr->addr.start),  
24 |             .readonly = fr->readonly,  
25 |         };  
26 |         if (listener->region_add) {  
27 |             listener->region_add(listener, section);  
28 |         }  
29 |     }  
30 | }
```



1



举报

添加PIT设备

MemoryRegion有修改，更新，调用io_region_add函数

添加pit设备时，会调用到memory_region_add_subregion函数，MemoryRegion被修改了，然后会调用到memory_region_transaction_commit函数，调用listener，是在这里映射PMIO地址和设备读写函数的。

注意memory_region_transaction_depth的使用，保证多层调用时，只需要更新一次。

```
1 void memory_region_transaction_commit(void)
2 {
3     AddressSpace *as;
4
5     assert(memory_region_transaction_depth);
6     --memory_region_transaction_depth;
7     if (!memory_region_transaction_depth && memory_region_update_pending) {
8         memory_region_update_pending = false;
9         MEMORY_LISTENER_CALL_GLOBAL(begin, Forward);
10
11         QTAILQ_FOREACH(as, &address_spaces, address_spaces_link) {
12             address_space_update_topology(as);
13         }
14
15         MEMORY_LISTENER_CALL_GLOBAL(commit, Forward);
16     }
17 }
```

真正的更新操作：

```
1 static void address_space_update_topology(AddressSpace *as)
2 {
3     FlatView old_view = *as->current_map;
4     FlatView new_view = generate_memory_topology(as->root);
5
6     address_space_update_topology_pass(as, old_view, new_view, false);
7     address_space_update_topology_pass(as, old_view, new_view, true);
8
9     *as->current_map = new_view;
10    flatview_destroy(&old_view);
11    address_space_update_ioeventfds(as);
12 }
```

```
1 static FlatView generate_memory_topology(MemoryRegion *mr)
2 {
3     FlatView view;
4
5     flatview_init(&view);
6
7     if (mr) {
8         render_memory_region(&view, mr, int128_zero(),
9                             addrrange_make(int128_zero(), int128_2_64()), false);
10    }
11    flatview_simplify(&view);
12
13    return view;
14 }
```

FlatView有点像把链表描述的MemoryRegion搞成了FlatRange数组(FlatRange中记录了MemoryRegion，自然可以获得MemoryRegion中的pit_ioport_ops)：



1



举报

```

1 static void render_memory_region(FlatView *view,
2     MemoryRegion *mr,
3     Int128 base,
4     AddrRange clip,
5     bool readonly)
6 {
7     MemoryRegion *subregion;
8     unsigned i;
9     hwaddr offset_in_region;
10    Int128 remain;
11    Int128 now;
12    FlatRange fr;
13    AddrRange tmp;
14
15    if (!mr->enabled) {
16        return;
17    }
18
19    int128_addto(&base, int128_make64(mr->addr));
20    readonly |= mr->readonly;
21
22    tmp = addrrange_make(base, mr->size);
23
24    if (!addrrange_intersects(tmp, clip)) {
25        return;
26    }
27
28    clip = addrrange_intersection(tmp, clip);
29
30    if (mr->alias) {
31        int128_subfrom(&base, int128_make64(mr->alias->addr));
32        int128_subfrom(&base, int128_make64(mr->alias->offset));
33        render_memory_region(view, mr->alias, base, clip, readonly);
34        return;
35    }
36
37    /* Render subregions in priority order. */
38    QTAILQ_FOREACH(subregion, &mr->subregions, subregions_link) {
39        render_memory_region(view, subregion, base, clip, readonly);
40    }
41
42    if (!mr->terminates) {
43        return;
44    }
45
46    offset_in_region = int128_get64(int128_sub(clip.start, base));
47    base = clip.start;
48    remain = clip.size;
49
50    /* Render the region itself into any gaps left by the current view. */
51    for (i = 0; i < view->nr && int128_nz(remain); ++i) {
52        if (int128_ge(base, addrrange_end(view->ranges[i].addr))) {
53            continue;
54        }
55        if (int128_lt(base, view->ranges[i].addr.start)) {
56            now = int128_min(remain,
57                int128_sub(view->ranges[i].addr.start, base));
58            fr.mr = mr;
59            fr.offset_in_region = offset_in_region;
60            fr.addr = addrrange_make(base, now);
61            fr.dirty_log_mask = mr->dirty_log_mask;
62            fr.readable = mr->readable;
63            fr.readonly = readonly;
64            flatview_insert(view, i, &fr);
65            ++i;
66            int128_addto(&base, now);
67            offset_in_region += int128_get64(now);
68            int128_subfrom(&remain, now);
69        }
70        now = int128_sub(int128_min(int128_add(base, remain),
71            addrrange_end(view->ranges[i].addr)),

```



1



举报

```

72         base);
73         int128_addto(&base, now);
74         offset_in_region += int128_get64(now);
75         int128_subfrom(&remain, now);
76     }
77     if (int128_nz(remain)) {
78         fr.mr = mr;
79         fr.offset_in_region = offset_in_region;
80         fr.addr = addrrange_make(base, remain);
81         fr.dirty_log_mask = mr->dirty_log_mask;
82         fr.readable = mr->readable;
83         fr.readonly = readonly;
84         flatview_insert(view, i, &fr);
85     }
86 }

```



1



MemoryRegion被修改的话，如果有添加，那么会调用到MEMORY_LISTENER_UPDATE_REGION(frnew, as, Forward, region_del)，添加PMI射关系。



```

1  static void address_space_update_topology_pass(AddressSpace *as,
2         FlatView old_view,
3         FlatView new_view,
4         bool adding)
5  {
6      unsigned iold, inew;
7      FlatRange *froid, *frnew;
8
9      /* Generate a symmetric difference of the old and new memory maps.
10     * Kill ranges in the old map, and instantiate ranges in the new map.
11     */
12     iold = inew = 0;
13     while (iold < old_view.nr || inew < new_view.nr) {
14         if (iold < old_view.nr) {
15             frold = &old_view.ranges[iold];
16         } else {
17             frold = NULL;
18         }
19         if (inew < new_view.nr) {
20             frnew = &new_view.ranges[inew];
21         } else {
22             frnew = NULL;
23         }
24
25         if (froid
26             && (!frnew
27                 || int128_lt(froid->addr.start, frnew->addr.start)
28                 || (int128_eq(froid->addr.start, frnew->addr.start)
29                     && !flatrange_equal(froid, frnew)))) {
30             /* In old, but (not in new, or in new but attributes changed). */
31
32             if (!adding) {
33                 MEMORY_LISTENER_UPDATE_REGION(froid, as, Reverse, region_del);
34             }
35
36             ++iold;
37         } else if (froid && frnew && flatrange_equal(froid, frnew)) {
38             /* In both (logging may have changed) */
39
40             if (adding) {
41                 MEMORY_LISTENER_UPDATE_REGION(frnew, as, Forward, region_nop);
42                 if (froid->dirty_log_mask && !frnew->dirty_log_mask) {
43                     MEMORY_LISTENER_UPDATE_REGION(frnew, as, Reverse, log_stop);
44                 } else if (frnew->dirty_log_mask && !froid->dirty_log_mask) {
45                     MEMORY_LISTENER_UPDATE_REGION(frnew, as, Forward, log_start);
46                 }
47             }
48         }

```



举报

```

49 |         ++iold;
      |         50 |         ++inew;
51 |     } else {
52 |         /* In new */
53 |
54 |         if (adding) {
55 |             MEMORY_LISTENER_UPDATE_REGION(frnew, as, Forward, region_add);
56 |         }
57 |
58 |         ++inew;
59 |     }
60 | }
61 | }

```



1



```

1 | #define MEMORY_LISTENER_UPDATE_REGION(fr, as, dir, callback) \
2 |     MEMORY_LISTENER_CALL(callback, dir, (&(MemoryRegionSection) { \
3 |         .mr = (fr)->mr, \
4 |         .address_space = (as), \
5 |         .offset_within_region = (fr)->offset_in_region, \
6 |         .size = int128_get64((fr)->addr.size), \
7 |         .offset_within_address_space = int128_get64((fr)->addr.start), \
8 |         .readonly = (fr)->readonly, \
9 |     }))

```

调用了region_add函数，也就是io_region_add函数：

```

1 | #define MEMORY_LISTENER_CALL(_callback, _direction, _section, _args...) \
2 |     do { \
3 |         MemoryListener *_listener; \
4 | \
5 |         switch (_direction) { \
6 |         case Forward: \
7 |             QTAILQ_FOREACH(_listener, &memory_listeners, link) { \
8 |                 if (_listener->_callback \
9 |                     && memory_listener_match(_listener, _section)) { \
10 |                     _listener->_callback(_listener, _section, ##_args); \
11 |                 } \
12 |             } \
13 |             break; \
14 |         case Reverse: \
15 |             QTAILQ_FOREACH_REVERSE(_listener, &memory_listeners, \
16 |                 memory_listeners, link) { \
17 |                 if (_listener->_callback \
18 |                     && memory_listener_match(_listener, _section)) { \
19 |                     _listener->_callback(_listener, _section, ##_args); \
20 |                 } \
21 |             } \
22 |             break; \
23 |         default: \
24 |             abort(); \
25 |         } \
26 |     } while (0)

```

io_region_add函数处理PMIO地址和设备读写函数的映射关系

```

1 | static void io_region_add(MemoryListener *listener,
2 |     MemoryRegionSection *section)
3 | {
4 |     MemoryRegionIORange *mrio = g_new(MemoryRegionIORange, 1);
5 |

```



举报

```

6 | mrio->mr = section->mr;
7 |     mrio->offset = section->offset_within_region;
8 | iorange_init(&mrio->iorange, &memory_region_iorange_ops,
9 |     section->offset_within_address_space, section->size);
10 | ioport_register(&mrio->iorange);
11 | }

```



1



memory_region_iorange_ops就是IORange->ops，其中读函数为memory_region_iorange_read，真正的设备读写函数保存在：



```

1 | static void memory_region_iorange_read(IORange *iorange,
2 |     uint64_t offset,
3 |     unsigned width,
4 |     uint64_t *data)
5 | {
6 |     MemoryRegionIORange *mrio
7 |     = container_of(iorange, MemoryRegionIORange, iorange);
8 |     MemoryRegion *mr = mrio->mr;
9 |
10 |    offset += mrio->offset;
11 |    if (mr->ops->old_portio) {
12 |        const MemoryRegionPortio *mrp = find_portio(mr, offset - mrio->offset,
13 |            width, false);
14 |
15 |        *data = ((uint64_t)1 << (width * 8)) - 1;
16 |        if (mrp) {
17 |            *data = mrp->read(mr->opaque, offset);
18 |        } else if (width == 2) {
19 |            mrp = find_portio(mr, offset - mrio->offset, 1, false);
20 |            assert(mrp);
21 |            *data = mrp->read(mr->opaque, offset) |
22 |                (mrp->read(mr->opaque, offset + 1) << 8);
23 |        }
24 |        return;
25 |    }
26 |    *data = 0;
27 |    access_with_adjusted_size(offset, data, width,
28 |        mr->ops->impl.min_access_size,
29 |        mr->ops->impl.max_access_size,
30 |        memory_region_read_accessor, mr);
31 | }

```

这里执行真正的读写函数，也就是pit_ioport_ops。

```

1 | static void memory_region_write_accessor(void *opaque,
2 |     hwaddr addr,
3 |     uint64_t *value,
4 |     unsigned size,
5 |     unsigned shift,
6 |     uint64_t mask)
7 | {
8 |     MemoryRegion *mr = opaque;
9 |     uint64_t tmp;
10 |
11 |    if (mr->flush_coalesced_mmio) {
12 |        qemu_flush_coalesced_mmio_buffer();
13 |    }
14 |    tmp = (*value >> shift) & mask;
15 |    mr->ops->write(mr->opaque, addr, tmp, size);
16 | }

```



举报

注册设备的三组读写函数：

```
1 void ioport_register(IORange *ioport)
2 {
3     register_ioport_read(ioport->base, ioport->len, 1,
4         ioport_readb_thunk, ioport);
5     register_ioport_read(ioport->base, ioport->len, 2,
6         ioport_readw_thunk, ioport);
7     register_ioport_read(ioport->base, ioport->len, 4,
8         ioport_readl_thunk, ioport);
9     register_ioport_write(ioport->base, ioport->len, 1,
10        ioport_writeb_thunk, ioport);
11    register_ioport_write(ioport->base, ioport->len, 2,
12        ioport_writew_thunk, ioport);
13    register_ioport_write(ioport->base, ioport->len, 4,
14        ioport_writel_thunk, ioport);
15    ioport_destructor_table[ioport->base] = iorange_destructor_thunk;
16 }
```



在这里注册了PMIO地址对应的读函数是ioport_readb_thunk，这就是一个壳。

```
1 int register_ioport_read(pio_addr_t start, int length, int size,
2     IOPortReadFunc *func, void *opaque)
3 {
4     int i, bsize;
5
6     if (ioport_bsize(size, &bsize)) {
7         hw_error("register_ioport_read: invalid size");
8         return -1;
9     }
10    for(i = start; i < start + length; ++i) {
11        ioport_read_table[bsize][i] = func;
12        if (ioport_opaque[i] != NULL && ioport_opaque[i] != opaque)
13            hw_error("register_ioport_read: invalid opaque for address 0x%x",
14                i);
15        ioport_opaque[i] = opaque;
16    }
17    return 0;
18 }
```

通过opaque可以获取读函数IORange->ops->read(也就是memory_region_iorange_read)。

```
1 static uint32_t ioport_readb_thunk(void *opaque, uint32_t addr)
2 {
3     IORange *ioport = opaque;
4     uint64_t data;
5
6     ioport->ops->read(ioport, addr - ioport->base, 1, &data);
7     return data;
8 }
```

对于读PMIO，KVM_EXIT_IO之后的流程是：

```
kvm_handle_io
->cpu_inb
->ioport_read
```



->ioport_read_table[0][addr](也就是ioport_readb_thunk)
->memory_region_iorange_ops(也就是IORange->ops)
->access_with_adjusted_size(需要mr, 保存了pit_ioport_ops)
->memory_region_read_accessor
->mr-ops

PS:

- 1、Object的parent可能是用来搞总线结构的，比如Object是bus上的设备，parent是bus。
- 2、ObjectProperty里面type为child<的应该就是Object用来记录子Object的，也就是bus记录上面挂的设备的。
- 3、注意QObject和QType(比C语言的type多了ref)，用来折腾ObjectProperty的属性设置的，和之前的Object，ObjectClass不同。
- 4、设置属性都是通过object_property_set_qobject来设置的，会生成visitor，然后调用void object_property_set(Object *obj, QObject *v, const Error **errp)。
- 5、isa_create中创建了Object，调用了pit_class_initfn等初始化函数。
- 6、isa bus的地址_space_io就是系统的system_io:

```
1 static void pc_init_isa(QEMUMachineInitArgs *args)
2 {
3     ram_addr_t ram_size = args->ram_size;
4     const char *cpu_model = args->cpu_model;
5     const char *kernel_filename = args->kernel_filename;
6     const char *kernel_cmdline = args->kernel_cmdline;
7     const char *initrd_filename = args->initrd_filename;
8     const char *boot_device = args->boot_device;
9     has_pvpanic = false;
10    if (cpu_model == NULL)
11        cpu_model = "486";
12    disable_kvm_pv_eoi();
13    enable_compat_apic_id_mode();
14    pc_init1(get_system_memory(),
15            get_system_io(),
16            ram_size, boot_device,
17            kernel_filename, kernel_cmdline,
18            initrd_filename, cpu_model, 0, 1);
19 }
```

pc_init1中:

```
1 if (pci_enabled) {
2     pci_bus = i440fx_init(&i440fx_state, &piix3_devfn, &isa_bus, gsi,
3             system_memory, system_io, ram_size,
4             below_4g_mem_size,
5             0x100000000ULL - below_4g_mem_size,
6             0x100000000ULL + above_4g_mem_size,
7             (sizeof(hwaddr) == 4
8             ? 0
9             : ((uint64_t)1 << 62)),
10            pci_memory, ram_memory);
11 } else {
12     pci_bus = NULL;
13     i440fx_state = NULL;
14     isa_bus = isa_bus_new(NULL, system_io);
15     no_hpet = 1;
16 }
```



1



举报

点赞 1 收藏 分享 ...



ayu_ag

发布了31 原创文章 · 获赞 11 · 访问量 9万+



传统ERP已经过时，2019流行的ERP系统是这一款！

主流erp系统



文章不错哦，我要夸奖作者两句...



1



QEMU中的对象模型——QOM（介绍篇）

阅读数 4186

QEMU提供了一套面向对象编程的模型——QOM，即QEMU Object Module，几乎所有的设备如CPU、内存、总线等...

博文 来自： [YuanruiZJU的博客](#)

QEMU 设备模拟

阅读数 4382

设备模拟目的我们好像不会干一件事而毫无目的，就算不停刷微信朋友圈也是为了打发你无聊的时间。其实最装B的...

博文 来自： [万能的终端和网络](#)

使用 qemu 来模拟 nvme 设备

阅读数 715

使用 qemu 模拟 nvme 设备，本篇可以参考。引用本文请注明出处: https://blog.csdn.net/Hello_NB1/article/details/103444450

博文 来自： [Hello_NB1的博客](#)

QEMU学习笔记——QOM(Qemu Object Model)

阅读数 493

（转载）本文发自<http://www.binss.me/blog/qemu-note-of-qemu-object-model/>，转载请注明出处。QOM(Qem...

博文 来自： [Hugo的博客](#)



软件开发人员外包报价表

软件开发外包

qemu下的USB直通功能介绍

阅读数 48

文章目录1. 查看linux usb设备1.1.通过/sys/kernel/debug/usb/devices文件1.2. 通过lsusb命令2.启动带usb设备的虚...

博文 来自： [daxiatou的专栏](#)

qemu trace使用

阅读数 99

Qemu有自己的Trace框架并支持多个debug/trace后端包括：nop, dtrace, ftrace, log, simple, ust，可以帮助我们分...

博文 来自： [weixin_34144450...](#)

QEMU设备模拟

阅读数 1733

备模拟目的我们好像不会干一件事而毫无目的，就算不停刷微信朋友圈也是为了打发你无聊的时间。其实最装B的回...

博文 来自： [tycoon的专栏](#)

QEMU PCIe设备实现

02-10

PCIe 设备虚拟化QEMU中的实现 包括处理中断的硬件以及Linux如何响应和处理终端。技术分析分享

下载

QEMU 4.0.0 发布，几乎可以模拟任何硬件设备的模拟器

阅读数 945

QEMU4.0.0发布了，此版本更新亮点包括：ARM：实现了一批ARMv8.X的扩展，包括SB、PredInv、HPD、LOR、F...

博文 来自： [weixin_33882452...](#)

QEMU学习笔记——QOM(Qemu Object Model) - Hugo的博客 - CSDN博客

qemu QOM(qemu object model)和设备模拟 - weixin_34004750的博客



举报



做一个小程序大概要多少钱

小程序开发多少钱

qemu 中是怎么模拟的新的设备

kvm_cpu_exec 和 d e m o 中演示的一样转载于:https://www.cnblogs.com/honpey/p/8063875.html...

...i8259 中断控制器虚拟设备_移动开发_ayu_ag的专栏-CSDN博客

QEMU 4.0.0 发布,几乎可以模拟任何硬件设备的模拟器..._CSDN博客

qemu: usb存储设备仿真

qemu既支持仿真虚拟的usb存储设备，也支持连接真实的设备（如U盘）。相关的命令参数为：-usb-device usb-sto...



YuanruiZJU
17篇文章
排名:千里之外



winceos
82篇文章
排名:千里之外



only火车头
70篇文章
排名:千里之外



eHugo
...篇文章
排名:千里之外

使用qemu 来模拟 nvme 设备_Hello_NB1的博客-CSDN博客

QEMU中的对象模型——QOM(介绍篇)_YuanruiZJU的博客-CSDN博客

Windows下编译Qemu模拟器

最近尝试在Windows下编译Qemu模拟器，官网上有相关叙述，我没有看懂是什么意思，贴出来请教大家，谢谢！

QEMU中如何定义所有Device的基类和BUS的基类

本文介绍QEMU如何模拟设备、总线、主板的连接关系。

qemu vl.c源码学习_ayu_ag的专栏-CSDN博客

...qemu-kvm i8254 pit虚拟设备_移动开发_ayu_ag的专栏-CSDN博客

qemu内存模型(3) 内存布局初始化

以i386为例static void memory_map_init(void){ system_memory = g_malloc(sizeof(*system_memory)); m...



传统ERP已经过时，2019流行的ERP系统是这一款！
主流erp系统

qemu使用PCI passthrough设备

qemu使用vfio pci设备

QEMU设备模型

QEMU本身并不是KVM的一部分，其自身就是一个著名的开源虚拟机软件。与KVM不同，QEMU虚拟机是一个纯软件...

QEMU(1) - QOM

Table of ContentsTypeInfo根类型TypeInfoTypeInfo链路初始化TypeInfo ->ModuleEntryDump TypeInfoTypeImpl...

qemu中device和driver的区别 使用9p文件系统

q e m u 配置中经常会出现-driver/-device的选项，可以理解成-driver是后端设备，即一个实际的物理的磁盘；d e ...

imx_6ull开发板（野火、原子）问题解决手册

持续更新中。。。。。。2019.11.03增加问题1，2，311221.无法上网https://blog.csdn.net/weixin_43674943/arti...



软件开发人员外包报价表
软件开发 外包

qemu如何实现面向对象模型QOM（代码讲解）

1. QOM中如何将所有的类储存起来的，并且完整地呈现给使用者的。2. 面向对象编程，有三个重要的特性——封装、...

阅读数 49

博文 来自: weixin_30



1



107



论坛

阅读数 1479

博文 来自: YuanruiZJU的博客

阅读数 132

博文 来自: woi110120130的...

阅读数 7261

博文 来自: RichardYSteven的...

阅读数 345

博文 来自: tengyft的专栏

阅读数 289

博文 来自: lwhuq的博客

阅读数 121

博文 来自: weixin_30741653...

阅读数 408

博文 来自: 5念since



举报

阅读数 1809

博文 来自: YuanruiZJU的博客

qemu USB3.0支持 阅读量 716

分析：主要是在qemu命令行中加入了 -device nec-usb-xhci,id=usb2,bus=pci.0,addr=0x8 这个配置项执行命令：# /... 博文 来自： [li_jiejun的专](#)

qemu代码分析.pdf 1 14

qemu 是使用动态二进制翻译的 cpu 模拟器，它支持两种运行模式：全系统模拟和用户态模拟。在全系统模拟下，qemu ... 博文 来自： [wxx213的专](#)

qemu对象模型——QOM实现分析 81

<http://blog.chinaunix.net/uid-28541347-id-5784376.html> 博文 来自： [wxx213的专](#)

Qemu VirtIO设备模拟分析1-virtio的QOM分析(以VirtIONetPCI为例) 124

基本介绍 本文直接从VirtIO开始分析，在前期需要一些基础只是特别是Qemu的QOM对象模型，有很多其他... 博文 来自： [sungeshil](#)

QEMU调试Linux系统的USB协议栈 355

QEMU调试Linux系统的USB协议栈通过使用QEMU调试Linux系统的USB协议栈来学习USB协议栈。<http://blog.csdn.net/zoomdy> 博文 来自： [zoomdy's b](#)

ubuntu kvm+qemu 加载USB 49

首先参考了：<http://forum.ubuntu.org.cn/viewtopic.php?f=65&t=130210&sid=592708d954990bdd11e...> 博文 来自： [wujay](#)

USB设备仿真框架设计指南——4.DSF中的COM对象 阅读量 260

DSF使用COM自动化对象将DSF服务暴露给设备模拟器。您可以从任何具有COM客户端能力的语言访问这些对象。几... 博文 来自： [SabreWulf](#)

USB设备仿真框架设计指南——5.DSF对象模型 阅读量 144

DSF对所有模拟设备（包括USB设备）使用与设备无关的对象模型。下面的图表显示了DSF对象模型。... 博文 来自： [SabreWulf](#)

qemu2的qom系统分析(-)对象系统 阅读量 85

前边分析machine的注册和选择，发现如果 不了解qom系统是很难分析的。qom系统的说明在include/qom/object... 博文 来自： [woai110120130的](#)

QEMU建模之设备创建总体流程 阅读量 111

(这里的设备创建以中断控制器即openpic为例)1.main函数之前执行type_init1> 在vl.c文件的main函数执行前会先执... 博文 来自： [sinat_38205774的](#)

推荐:国产仿真软件realboard与qemu性能大PK

真刀真枪的和qemu比划，一针见血，振奋人心，本贴转自：<http://lxzhg.download.csdn.net/> 感谢大家关注realboard，... 论坛

virtio的qemu总线与设备模型 阅读量 1万+

（很多内容是网上找的，+上我个人的一点理解，推荐大家去看 <http://mnstory.net/2014/10/qemu-device-simulati...> 博文 来自： [majieyue的专](#)

虚拟化中如何实现设备模拟？ 阅读量 679

在计算机虚拟化领域中，对设备进行模拟是虚拟化实现的基础。设备的模拟主要包括一下三个方面：设备状态的记录... 博文

qemu-kvm 对mmio的模拟 阅读量 609

转：<http://blog.chinaunix.net/uid-28541347-id-5789579.html>MMIO和PIO的区别I/O作为CPU和外设交流的一个渠... 博文 来自： [weixin_42681961](#)

QEMU内在：整体架构和线程模型 阅读量 2334

原文地址：<http://blog.vmsplce.net/2011/03/qemu-internals-overall-architecture-and.html>一篇很不错的文章，... 博文 来自： [CurtisGuo的专](#)

QEMU-System mode emulation分析(2) 阅读量 945

2. System mode下的机器（machine）管理QEMU Sytem mode模拟的整个硬件平台，硬件平台最核心的是我们常... 博文 来自： [lulu901130的专](#)

QEMU通过virtio接收报文处理流程（QEMU2.0.0） 阅读量 1569

1. set_guest_notifiers初始化流程static void virtio_pci_bus_class_init(ObjectClass *klass, void *data){ k-... 博文 来自： [六六哥的博客](#)

virtio gpu 阅读量 1486

virtio gpu 博文 来自： [eva980636的博客](#)

在qemu中增加pci设备并用linux驱动验证 82

声明本文主要针对x86架构进行说明。使用的qemu版本是：qemu-kvm-1.2.0-rc21)PCI结构简介每个PCI设备都有一... 博文 来自： [XscKernel](#)

学Python后到底能干什么？网友：我太难了 阅读量 262

感觉全世界营销文都在推Python，但是找不到工作的话，又有哪个机构会站出来给我推荐工作？笔者冷静分析多方... 博文 来自： [CSDN学院](#)



ayu_ag

[TA的个人主页>](#)

原创 31 粉丝 25 获赞 11 评论 4 访问 9万+

等级: **博客 4**

周排名: **13万+**

积分: **1256**

总排名: **7万+**

勋章:

关注

私信

最新文章

美娜多(manado)潜水游记
ELF函数重定位问题
ld.gold使用指南
android build system中product的继承
(inherit-product)，加载(import-products)
和选择(lunch)
使用libhybris，glibc和bionic共存时的TLS
冲突的问题

分类专栏

编译 6篇
 qemu 14篇
 C和C++ 2篇
 linux和shell 5篇
 汇编 2篇

[展开](#)

归档

2018年5月 1篇
2017年11月 2篇
2017年7月 1篇
2016年12月 1篇



1



举报

2016年10月	8篇
2016年9月	6篇
2016年8月	1篇
2016年6月	2篇
展开	

热门文章

libffi浅析
阅读数 9510

获取urllib2.urlopen失败时的错误页面
阅读数 6210

编译开源软件时，prefix, sysroot, DESTDIR 怎么整
阅读数 6015

ld.gold使用指南
阅读数 4584

qemu参数解析
阅读数 4515

最新评论

使用libhybris，glibc...
ging450：你还，我用你的例子，提示undefined s
ymbol: _Z13android_dl原因PvPKc。可否把代 ...

编译开源软件时，prefix, s...
A694543965：好！！！

ubuntu12.04环境下使用k...
hanf___：哈哈，很有启发，谢谢！

python BaseHTTPSe...
jiangming7：good, and thanks. 我是使用flask作
为服务进程启动一些子进程A,B,C...当flask退出 ...

1 小程序开发	13 分销网站开发
2 免费永久云主机	14 好网站建设公司
3 舆情监测	15 移动隔断墙
4 网站开发公司	16 商城小程序
5 电商系统开发	17 移动环保公厕
6 短信接口	18 专升本
7 免备案的云主机	19 外贸建站 公司
8 高端网站建设	20 icp申请
9 app开发	21 上海市网站建设
10 移动卫生间	22 人力管理系统
11 模拟高尔夫	23 心肺复苏模拟人
12 淘客app	24 软件外包开发

QQ客服

kefu@csdn.net

客服论坛

400-660-0108

工作时间 8:30-22:00

[关于我们](#) [招聘](#) [广告服务](#) [网站地图](#)

京ICP备19004658号 经营性网站备案信息

公安备案号 11010502030143

©1999-2020 北京创新乐知网络技术有限公司

网络110报警服务

北京互联网违法和不良信息举报中心

中国互联网举报中心 家长监护

版权与免责声明 版权申诉

1

举报