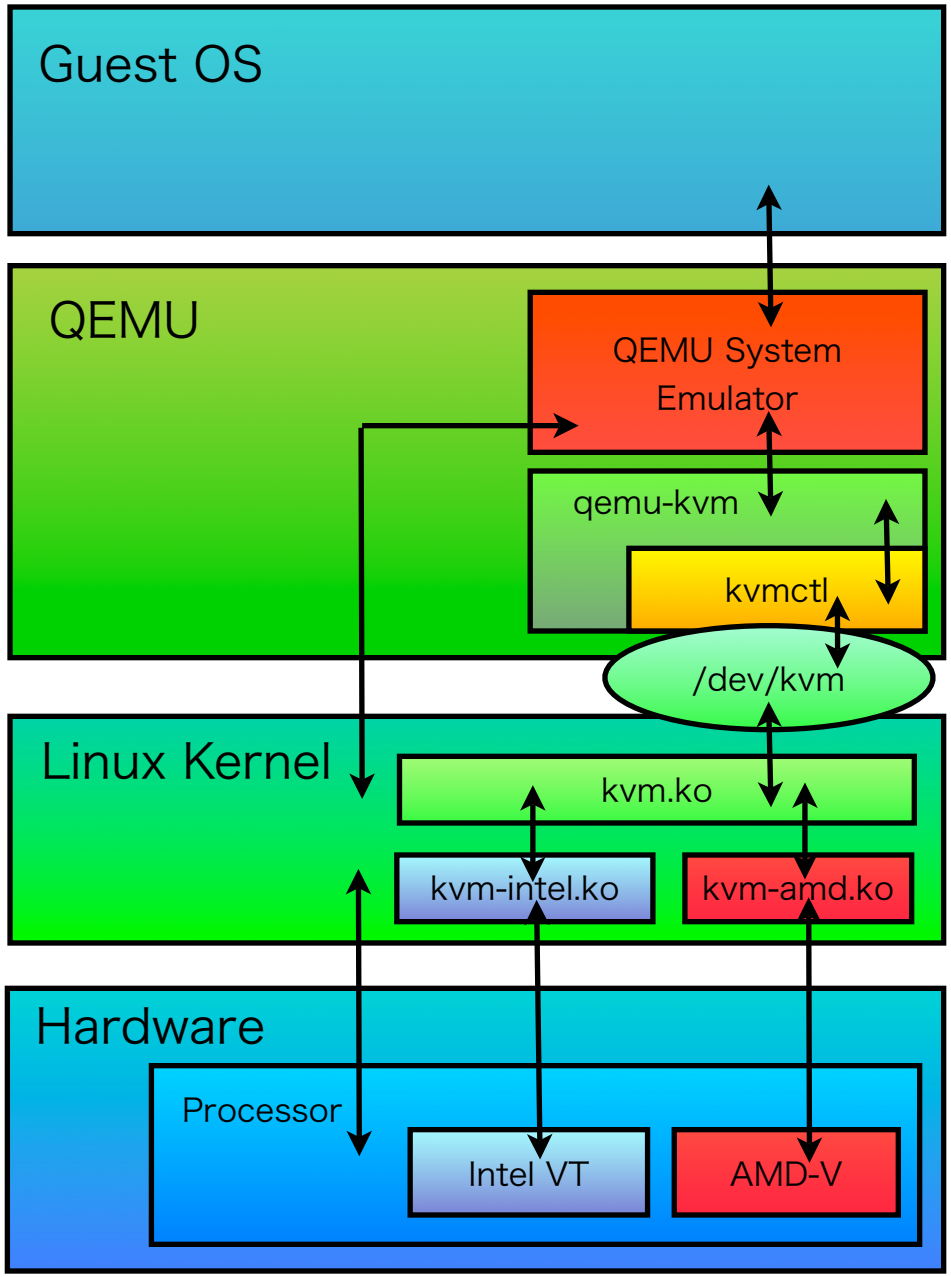
濃いバナ
KVMの仕組み

仮想化友の会

平 初 <htaira@pantora.net>

KVMの仕組み



KVMはQEMUの拡張機能

- KVM拡張機能(qemu-kvm.c)が同時にコンパイル
- カーネル空間で動くコンポーネントとユーザ空間で動くコンポーネントが存在

2種類のカーネルモジュール

- Intel VT(Intel Virtualization Technology)向けのカーネルモジュール → kvm-intel.ko
- AMD-V(AMD Virtualization)向けのカーネルモジュール → kvm-amd.ko

KVMとQEMU間の通信方法

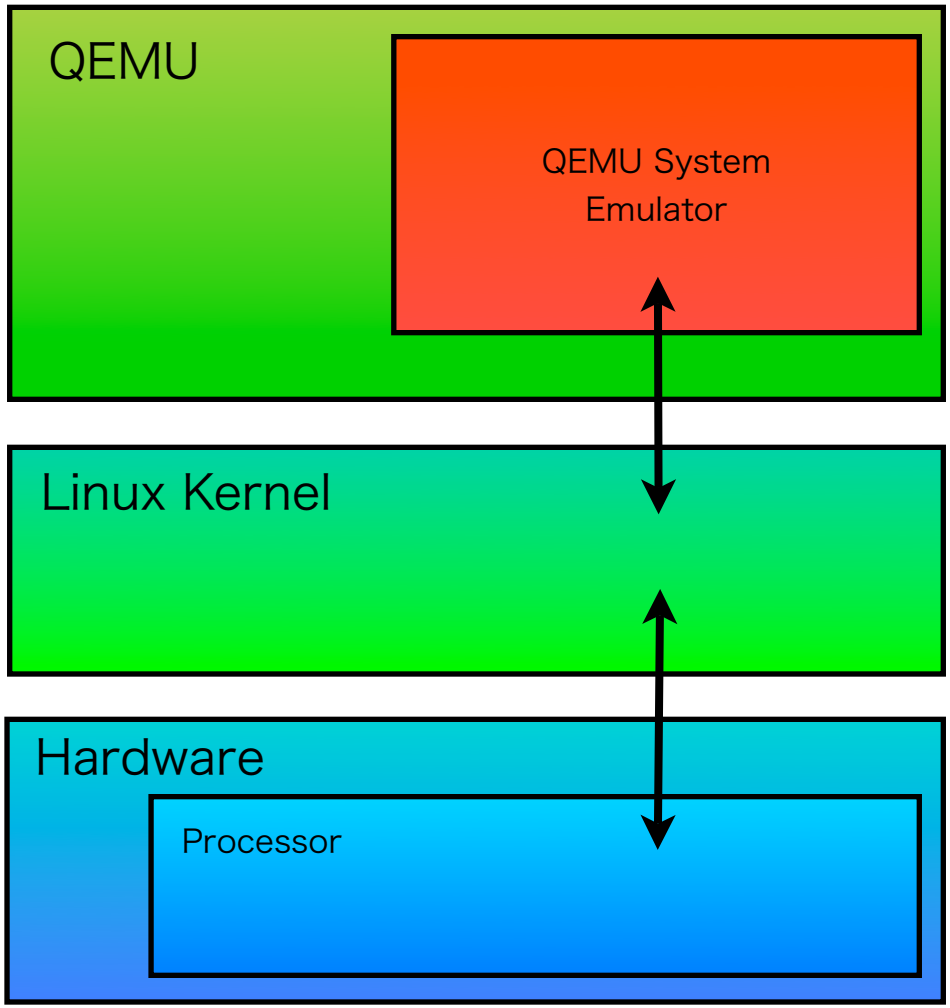
- KVMキャラクタデバイス(/dev/kvm)を経由
- 操作はmmap()、ioctl()などの関数
- 操作はユーザ空間で動くライブラリ(kvmctl.c)を経由し行うこと

KVMが有効な場合

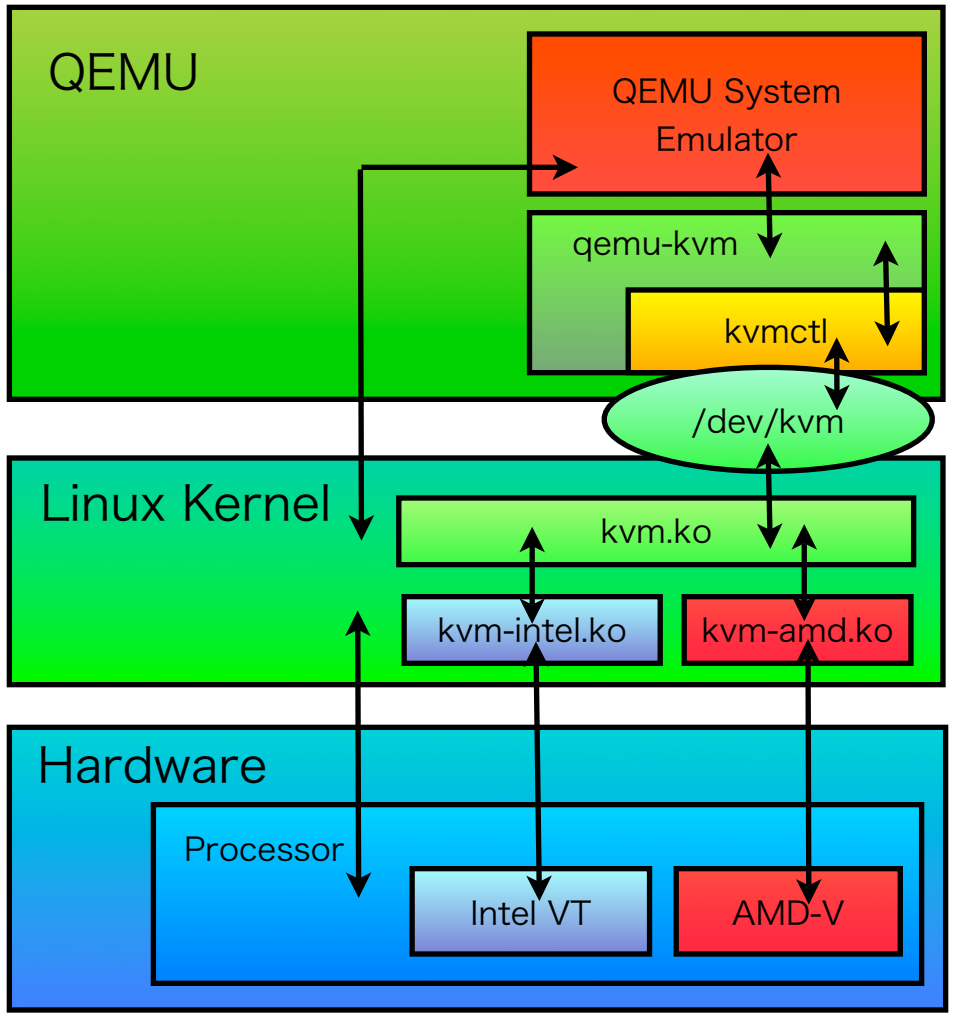
- /dev/kvmが存在する
- kvm_allowed == 1 (QEMUの中の変数)

QEMUとKVM比較

QEMUの仕組み



KVMの仕組み



KVMのソースコード解説

ソースコードのディレクトリ構造 1/2

- kvm-15/
  - kvm ... QEMUのラッパスクリプト(Python)
- kvm-15/qemu/ ... KVMに対応したQEMU
  - vl.c ... QEMU System Emulatorのコアコンポーネント
  - qemu-kvm.c ... QEMUのKVM拡張機能に関する関数が含まれる
  - qemu-kvm.h ... QEMUのKVM拡張機能に関する関数のヘッダーファイル
  - cpu-exec.c ... QEMUのゲストOSのCPU命令を解釈する関数が含まれる
- kvm-15/kernel/ ... KVMカーネルモジュール
  - kvm_main.c ... QEMUのKVM拡張機能に関する関数のヘッダーファイル
  - x86_emulate.c ... x86リアルモードをエミュレーションするコンポーネント
  - kvm_vmx.c ... Intel VT用のカーネルモジュール
  - vmx.c ... Intel VT使用時にIntel VT用命令を処理するコンポーネント
  - mmu.c ... Intel VTの時にメモリ管理を行うコンポーネント
  - kvm_svm.c ... AMD-V用のカーネルモジュール
  - svm.c ... AMD-V使用時にAMD-V用命令を処理するコンポーネント

KVMのソースコード解説

ソースコードのディレクトリ構造 2/2

- kvm-15/user/ ... ユーザ空間で動くコンポーネント
  - kvmctl.c ... KVMキャラクタデバイス(/dev/kvm)と通信するコンポーネント
  - main.c ... KVM機能を試すテストアプリケーション
  - test/ ... ほとんどアセンブラで書かれた小さなOSが入っています。
- kvm-15/scripts/ ... KVMの周辺スクリプト
- kvm-15/drivers/ ... ゲストOS向けパラバーチャルドライバ
  - hypercall.c ... Linux向けパラバーチャルドライバ
    (SCSIドライバやNICドライバが今後このドライバを経由するはず)

KVMのソースコード解説

KVMの起動

```
# kvm --image ./hda.img
```

※現在、Debian GNU/Linuxでは、まだスクリプトは使われていません。

kvmスクリプト(kvm-15/kvm)

```
110 def vendor():
111     for x in file('/proc/cpuinfo').readlines():
112         m = re.match(r'vendor_id[ \t]*: *([a-zA-Z]+),*', x)
113         if m:
114             return m.group(1)
115     return unknown
```

```
117 vendor_module = {
118     'GenuineIntel': 'kvm-intel',
119     'AuthenticAMD': 'kvm-amd',
120     }[vendor()]
121
122 if options.kvm and options.reload:
123     for module in [vendor_module, 'kvm']:
124         remove_module(module)
125     for module in ['kvm', vendor_module]:
126         insert_module(module)
127     for i in range(5):
128         if os.access('/dev/kvm', os.F_OK):
129             break
130         time.sleep(0.1 + 0.2 * i)
131     if not os.access('/dev/kvm', os.F_OK):
132         print '/dev/kvm not present'
```

1-1 Intel用カーネルモジュールとAMD用カーネルモジュールのどちらを読み込むかチェックする。
(/proc/cpuinfoの情報より自動判別)

1-2 最適なカーネルモジュールが読み込まれる。

1-3 KVMキャラクタデバイス(/dev/kvm)があるかチェックする。

1-4 CPUが64bitアーキテクチャがチェックし、最適なQEMUを起動する。
(Pythonのplatformライブラリによりカーネルの種類は自動判別)

KVMのソースコード解説

KVMの起動

kvmスクリプト(kvm-15/kvm)

```
146 import platform
147 arch = platform.machine()
148 # check 32-bit userspace on 64-bit kernel
149 if platform.architecture()[0] == '32bit':
150     arch = 'i386'
151
152 if arch == 'x86_64':
153     cmd = 'qemu-system-' + arch
154 else:
155     cmd = 'qemu'
156
157 local_cmd = 'qemu/' + arch + '-softmmu/' + cmd
158 if os.access(local_cmd, os.F_OK):
159     cmd = local_cmd
160 else:
161     cmd = '/usr/bin/kvm'
162
163 qemu_args = (cmd, '-boot', bootdisk,
164              '-hda', disk, '-m', '384',
165              '-serial', 'file:/tmp/serial.log',
166              '-usbdevice', 'tablet'
167           )
```

1-4 CPUが64bitアーキテクチャがチェックし、
最適なQEMUを起動する。
(Pythonのplatformライブラリによりカーネルの種類は自動判別)

KVMのソースコード解説

KVMの起動

2-1 KVMを使うかどうかチェックされる。
2-2 KVMを使う場合(kvm_allowed == 1)、KVMを初期化する関数kvm_qemu_init()が呼ばれる。

main() in kvm-15/qemu/vl.c

```
7338            signal(SIGTTIN, SIG_IGN);
7339        }
7340    #endif
7341
7342    #if USE_KVM
7343        if (kvm_allowed) {
7344        if (kvm_qemu_init() < 0) {
7345            fprintf(stderr, "Could not initialize KVM, will disable KVM support¥n");
7346            kvm_allowed = 0;
7347        }
7348        }
7349    #endif
```

kvm_qemu_init()
in kvm-15/qemu/qemu-kvm.c

```
653         .pre_kvm_run = pre_kvm_run,
654     };
655
656     int kvm_qemu_init()
657     {
658         /* Try to initialize kvm */
659         kvm_context = kvm_init(&qemu_kvm_ops, saved_env);
660         if (!kvm_context) {
661             return -1;
662         }
663
664         return 0;
665     }
666
667     int kvm_qemu_create_context(void)
668     {
```

KVMのソースコード解説

KVMの起動

kvm_qemu_init()
in kvm-15/qemu/qemu-kvm.c

```
653     .pre_kvm_run = pre_kvm_run,
654 };
655
656 int kvm_qemu_init()
657 {
658     /* Try to initialize kvm */
659     kvm_context = kvm_init(&qemu_kvm_ops, saved_env);
660     if (!kvm_context) {
661         return -1;
662     }
663
664     return 0;
665 }
666
667 int kvm_qemu_create_context(void)
668 {
```

3-1 kvm_init()が呼ばれる。

kvm_init()
in kvm-15/user/kvmctl.c

```
178 kvm_context_t kvm_init(struct kvm_callbacks *callbacks,
                         void *opaque)
180 {
181     int fd;
182     kvm_context_t kvm;
183     int r;
184
185     fd = open("/dev/kvm", O_RDWR);
186     if (fd == -1) {
187         perror("open /dev/kvm");
188         return NULL;
189     }
190     r = ioctl(fd, KVM_GET_API_VERSION, 0);
191     if (r == -1) {
192         fprintf(stderr, "kvm kernel version too old¥n");
193         goto out_close;
194     }
195     if (r < EXPECTED_KVM_API_VERSION) {
196         fprintf(stderr, "kvm kernel version too old¥n");
197         goto out_close;
198     }
199     if (r > EXPECTED_KVM_API_VERSION) {
200         fprintf(stderr, "kvm userspace version too old¥n");
201         goto out_close;
202     }
203     kvm = malloc(sizeof(*kvm));
204     kvm->fd = fd;
205     kvm->vm_fd = -1;
206     kvm->callbacks = callbacks;
207     kvm->opaque = opaque;
208     kvm->dirty_pages_log_all = 0;
209     memset(&kvm->mem_regions, 0, sizeof(kvm->mem_regions));
210
211     return kvm;
212 out_close:
213     close(fd);
214     return NULL;
215 }
```

仮想化友の会

KVMのソースコード解説

KVMの起動

kvm_init()

in kvm-15/user/kvmctl.c

```
178 kvm_context_t kvm_init(struct kvm_callbacks *callbacks,
179                        void *opaque)
180 {
181     int fd;
182     kvm_context_t kvm;
183     int r;
184
185     fd = open("/dev/kvm", O_RDWR);
186     if (fd == -1) {
187         perror("open /dev/kvm");
188         return NULL;
189     }
190     r = ioctl(fd, KVM_GET_API_VERSION, 0);
191     if (r == -1) {
192         fprintf(stderr, "kvm kernel version too old¥n");
193         goto out_close;
194     }
195     if (r < EXPECTED_KVM_API_VERSION) {
196         fprintf(stderr, "kvm kernel version too old¥n");
197         goto out_close;
198     }
199     if (r > EXPECTED_KVM_API_VERSION) {
200         fprintf(stderr, "kvm userspace version too old¥n");
201         goto out_close;
202     }
203     kvm = malloc(sizeof(*kvm));
204     kvm->fd = fd;
205     kvm->vm_fd = -1;
206     kvm->callbacks = callbacks;
207     kvm->opaque = opaque;
208     kvm->dirty_pages_log_all = 0;
209     memset(&kvm->mem_regions, 0, sizeof(kvm->mem_regions));
210
211     return kvm;
212  out_close:
213     close(fd);
214     return NULL;
215 }
```

4-1 KVMキャラクタデバイス(/dev/kvm)がオープンされる。

4-2 KVMのAPIバージョン確認が行われる。

ioctl(fd, KVM_GET_API_VERSION, 0);

kvm_dev_ioctl()

in kvm-15/kernel/kvm_main.c

```
2247 static long kvm_dev_ioctl(struct file *filp,
2248                            unsigned int ioctl, unsigned long arg)
2249 {
2250     void __user *argp = (void __user *)arg;
2251     int r = -EINVAL;
2252
2253     switch (ioctl) {
2254     case KVM_GET_API_VERSION:
2255         r = KVM_API_VERSION;
2256         break;
2257     case KVM_CREATE_VM:
2258         r = kvm_dev_ioctl_create_vm();
2259         break;
2260     case KVM_GET_MSR_INDEX_LIST: {
2261         struct kvm_msr_list __user *user_msr_list = argp;
2287         default:
2288             ;
2289         }
2290 out:
2291         return r;
2292 }
```

/dev/kvm

kvm_context_t 構造体

int fd …

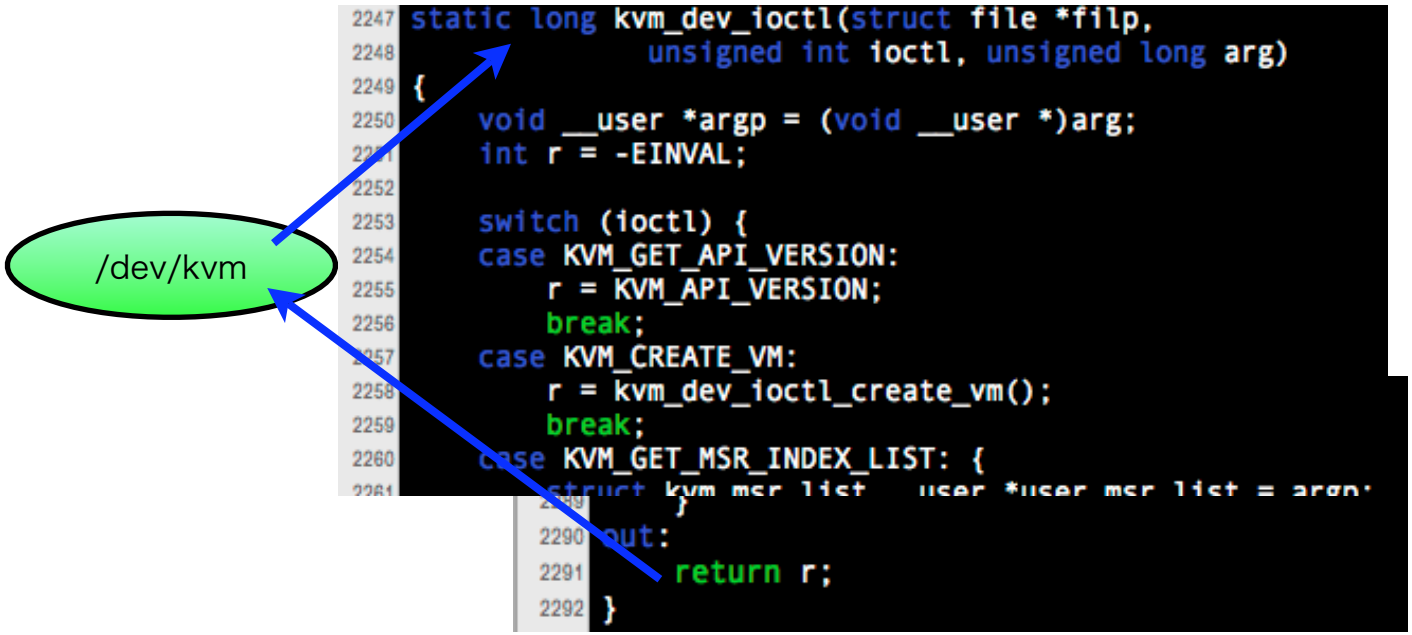
/dev/kvm

KVMのソースコード解説

KVMの起動

5-1 switch文でKVM内の処理が分岐される。
5-2 KVMのAPIバージョン(KVM_API_VERSION)を返す。

kvm_dev_ioctl()
in kvm-15/kernel/kvm_main.c



KVMのソースコード解説

KVMの起動

kvm_init()

in kvm-15/user/kvmctl.c

```
178 kvm_context_t kvm_init(struct kvm_callbacks *callbacks,
179                        void *opaque)
180 {
181     int fd;
182     kvm_context_t kvm;
183     int r;
184
185     fd = open("/dev/kvm", O_RDWR);
186     if (fd == -1) {
187         perror("open /dev/kvm");
188         return NULL;
189     }
190     r = ioctl(fd, KVM_GET_API_VERSION, 0);
191     if (r == -1) {
192         fprintf(stderr, "kvm kernel version too old¥n");
193         goto out_close;
194     }
195     if (r < EXPECTED_KVM_API_VERSION) {
196         fprintf(stderr, "kvm kernel version too old¥n");
197         goto out_close;
198     }
199     if (r > EXPECTED_KVM_API_VERSION) {
200         fprintf(stderr, "kvm userspace version too old¥n");
201         goto out_close;
202     }
203     kvm = malloc(sizeof(*kvm));
204     kvm->fd = fd;
205     kvm->vm_fd = -1;
206     kvm->callbacks = callbacks;
207     kvm->opaque = opaque;
208     kvm->dirty_pages_log_all = 0;
209     memset(&kvm->mem_regions, 0, sizeof(kvm->mem_regions));
210
211     return kvm;
212  out_close:
213     close(fd);
214     return NULL;
215 }
```

4-3 kvm_context_t 構造体が作られ初期化される。

KVMのソースコード解説

KVMの起動

2-3 KVM拡張ページ空間(KVM_EXTRA_PAGES)を考慮し、
必要な物理メモリサイズ(phys_ram_size)を計算する。
2-4 kvm_qemu_create_context()を呼び出す。

main() in kvm-15/qemu/vl.c

```
7431 #if USE_KVM
7432     /* Initialize kvm */
7433     if (kvm_allowed) {
7434         phys_ram_size += KVM_EXTRA_PAGES * 4096;
7435         if (kvm_qemu_create_context() < 0) {
7436             fprintf(stderr, "Could not create KVM context¥n");
7437             exit(1);
7438         }
7439     } else {
```

kvm_qemu_create_context()
in kvm-15/qemu/qemu-kvm.c

```
667 int kvm_qemu_create_context(void)
668 {
669     int i;
670
671     if (kvm_create(kvm_context, phys_ram_size, (void**)
    &phys_ram_base) < 0) {
672     kvm_qemu_destroy();
673     return -1;
674     }
675     kvm_msr_list = kvm_get_msr_list(kvm_context);
676     if (!kvm_msr_list) {
677     kvm_qemu_destroy();
678     return -1;
679     }
680     for (i = 0; i < kvm_msr_list->nmsrs; ++i)
681     if (kvm_msr_list->indices[i] == MSR_STAR)
682         kvm_has_msr_star = 1;
683     return 0;
684 }
```

KVMのソースコード解説

KVMの起動

kvm_qemu_create_context()
in kvm-15/qemu/qemu-kvm.c

```
667 int kvm_qemu_create_context(void)
668 {
669     int i;
670
671     if (kvm_create(kvm_context, phys_ram_size, (void**)
    &phys_ram_base) < 0) {
672     kvm_qemu_destroy();
673     return -1;
674     }
675     kvm_msr_list = kvm_get_msr_list(kvm_context);
676     if (!kvm_msr_list) {
677     kvm_qemu_destroy();
678     return -1;
679     }
680     for (i = 0; i < kvm_msr_list->nmsrs; ++i)
681     if (kvm_msr_list->indices[i] == MSR_STAR)
682         kvm_has_msr_star = 1;
683     return 0;
684 }
```

3-2 kvm_create()を呼び出す。

kvm_create_context()
in kvm-15/user/kvmctl.c

```
227 int kvm_create(kvm_context_t kvm, unsigned long memory, void **vm_mem)
229 {
230     unsigned long dosmem = 0xa0000;
231     unsigned long exmem = 0xc0000;
232     int fd = kvm->fd;
233     int r;
234     struct kvm_memory_region low_memory = {
235         .slot = 3,
236         .memory_size = memory  < dosmem ? memory : dosmem,
237         .guest_phys_addr = 0,
238     };
239     struct kvm_memory_region extended_memory = {
240         .slot = 0,
241         .memory_size = memory < exmem ? 0 : memory - exmem,
242         .guest_phys_addr = exmem,
243     };
244
245     kvm->vcpu_fd[0] = -1;
246
247     fd = ioctl(fd, KVM_CREATE_VM, 0);
248     if (fd == -1) {
249         fprintf(stderr, "kvm_create_vm: %m¥n");
250         return -1;
251     }
277     r = ioctl(fd, KVM_CREATE_VCPU, 0);
278     if (r == -1) {
279         fprintf(stderr, "kvm_create_vcpu: %m¥n");
280         return -1;
281     }
282     kvm->vcpu_fd[0] = r;
283     return 0;
284 }
```

KVMのソースコード解説

KVMの起動

4-4 仮想マシン作成が作成される。
ioctl(fd, KVM_CREATE_VM, 0);

kvm_create_context()
in kvm-15/user/kvmctl.c

```
227 int kvm_create(kvm_context_t kvm, unsigned long memory, void **vm_mem)
228 {
229     unsigned long dosmem = 0xa0000;
230     unsigned long exmem = 0xc0000;
231     int fd = kvm->fd;
232     int r;
233     struct kvm_memory_region low_memory = {
234         .slot = 3,
235         .memory_size = memory  < dosmem ? memory : dosmem,
236         .guest_phys_addr = 0,
237     };
238     struct kvm_memory_region extended_memory = {
239         .slot = 0,
240         .memory_size = memory < exmem ? 0 : memory - exmem,
241         .guest_phys_addr = exmem,
242     };
243
244     kvm->vcpu_fd[0] = -1;
245
246     fd = ioctl(fd, KVM_CREATE_VM, 0);
247     if (fd == -1) {
248         fprintf(stderr, "kvm_create_vm: %m¥n");
249         return -1;
250     }
251     kvm->vm_fd = fd;
278     if (r == -1) {
279         fprintf(stderr, "kvm_create_vcpu: %m¥n");
280         return -1;
281     }
282     kvm->vcpu_fd[0] = r;
283     return 0;
284 }
```

kvm_dev_ioctl()
in kvm-15/kernel/kvm_main.c

```
2247 static long kvm_dev_ioctl(struct file *filp,
2248                           unsigned int ioctl, unsigned long arg)
2249 {
2250     void __user *argp = (void __user *)arg;
2251     int r = -EINVAL;
2252
2253     switch (ioctl) {
2254     case KVM_GET_API_VERSION:
2255         r = KVM_API_VERSION;
2256         break;
2257     case KVM_CREATE_VM:
2258         r = kvm_dev_ioctl_create_vm();
2259         break;
2260     case KVM_GET_MSR_INDEX_LIST: {
2287     default:
2288         ;
2289     }
2290 out:
2291     return r;
```

/dev/kvm

5-3 switch文でKVM内の処理が分岐される。
5-4 kvm_dev_ioctl_create_vm()が呼ばれる。