# OpenBMC development obmc-ikmv keyboard and mouse function

原创 Lemon in the Rain Modified on 2025-03-27 10:17:52 Views: 589 Collection 4 Likes 5



Category Column: OpenBMC Article Tags: server C++ linux



> ## Introduction to obmc-ikvm service for OpenBMC development

### 1. Initialization steps

- obmc-ikvm.cpp executes the main() function, creates an ikvm::Manager instance and executes manager.run()
- Create an Input input instance in ikvm_manager.hpp and initialize input in the Manager destructor (args.getKeyboardPath(), args.getPointerPath(), args.getUdcName())
- When creating an Input instance, open the keyboard and mouse UDC device file in the destructor: **hidUdcPath = "/sys/kernel/config/usb_gadget/obmc_hid/UDC"**

C++ AI generated projects 登录复制 run

```cpp
/* @brief Handle of the HID gadget UDC */
std::ofstream hidUdcStream;

Input::Input(const std::string& kbdPath, const std::string& ptrPath,
             const std::string& udc) :
    keyboardFd(-1), pointerFd(-1), keyboardReport{0}, pointerReport{0},
    keyboardPath(kbdPath), pointerPath(ptrPath), udcName(udc)
{
    hidUdcStream.exceptions(std::ofstream::failbit | std::ofstream::badbit);
    hidUdcStream.open(hidUdcPath, std::ios::out | std::ios::app);
}
```



- When there is no KVM session, the keyboard and mouse HID link is disconnected.
- **Only when the first KVM client is opened, that is, when server->numClients = 0, the HID link is established** and the server->input.connect() function is called to configure the USBGadget. At the same time, server->frameCounter = 0.

C++ AI generated projects 登录复制 run

```
enum rfbNewClientAction Server::newClient(rfbClientPtr cl)
{
  Server* server = (Server*)cl->screen->screenData;
  cl->clientData =new ClientData(server->video.getFrameRate(), &server->input);
  cl->clientGoneHook = clientGone;
  cl->clientFramebufferUpdateRequestHook = clientFramebufferUpdateRequest;
  if (!server->numClients++)
  {
      server->input.connect();
      server->pendingResize = false;
      server->frameCounter = 0;
  }
  return RFB_CLIENT_ACCEPT;
}
```

收起 ^

- The main function of the Input::connect() function is to obtain the USB port ID and write it into the hidUdcStream file to configure the USBGadget of the keyboard and mouse.
- If the input parameter udcName is specified, write directly: hidUdcStream << udcName << std::endl;
- If the input parameter is empty, it is automatically obtained from the path /sys/bus/platform/devices/1e6a0000.usb-vhub. The acquisition rule is /sys/bus/platform/devices/1e6a0000.usb-vhub/1e6a0000.usb-vhub:pX/directory contains gadget files, and there is no suspended file

C++ AI generated projects 登录复制 run

```
void Input::connect()
for (const auto& port : fs::directory_iterator(usbVirtualHubPath))
{
    // port=/sys/bus/platform/devices/1e6a0000.usb-vhub/1e6a0000.usb-vhub:pX
    // 确认该路径为目录，而非链接文件
    if (fs::is_directory(port) && !fs::is_symlink(port))
    {
        // 遍历/sys/bus/platform/devices/1e6a0000.usb-vhub/1e6a0000.usb-vhub:pX下的所有文件
        // gadget=/sys/bus/platform/devices/1e6a0000.usb-vhub/1e6a0000.usb-vhub:pX/gadget.Y
        for (const auto& gadget:fs::directory_iterator(port.path()))
        {
            // Kernel 6.0:
            // /sys/.../1e6a0000.usb-vhub:pX/gadget.Y/suspended
            // Kernel 5.15:
            // /sys/.../1e6a0000.usb-vhub:pX/gadget/suspended
            // 确认gadget路径为目录，而非链接文件，且路径中包含gadget，且不存在suspended文件
            if (fs::is_directory(gadget) &&gadget.path().string().find("gadget") !=std::string::npos &&!fs::exists(gadget.path() / "suspended"))
            {
                const std::string portId = port.path().filename();

```

```
twen                hidUdcStream << portId << std::endl;
twen                found = true;
twen                break;
twen            }
 25
 26         }
 27     }
}
```

收起 ^

- After USBGadget is configured, the configuration effect on the device terminal is as follows. UDC control has been successfully assigned, which is equivalent to executing: echo "1e6a0000.usb-vhub:p6">/sys/kernel/config/usb_gadget/obmc_hid/UDC

**sh** AI generated projects 登录复制

```
root@10020220507ABCD:/# cat /sys/kernel/config/usb_gadget/obmc_hid/UDC
1e6a0000.usb-vhub:p6
root@10020220507ABCD:/# ls -l /sys/kernel/config/usb_gadget/obmc_hid/
-rw-r--r--    1 root     root          4096 Nov 17 14:17 UDC
-rw-r--r--    1 root     root          4096 Nov 19 09:05 bDeviceClass
-rw-r--r--    1 root     root          4096 Nov 19 09:05 bDeviceProtocol
-rw-r--r--    1 root     root          4096 Nov 19 09:05 bDeviceSubClass
-rw-r--r--    1 root     root          4096 Nov 19 09:05 bMaxPacketSize0
-rw-r--r--    1 root     root          4096 Nov 17 10:32 bcdDevice
-rw-r--r--    1 root     root          4096 Nov 17 10:32 bcdUSB
drwxr-xr-x    3 root     root             0 Nov 17 10:32 configs
drwxr-xr-x    4 root     root             0 Nov 17 10:32 functions
-rw-r--r--    1 root     root          4096 Nov 17 10:32 idProduct
-rw-r--r--    1 root     root          4096 Nov 17 10:32 idVendor
-rw-r--r--    1 root     root          4096 Nov 19 09:05 max_speed
drwxr-xr-x    2 root     root             0 Nov 17 10:32 os_desc
drwxr-xr-x    3 root     root             0 Nov 17 10:32 strings
```

收起 ^

- After the USBGadget configuration is OK in the void Input::connect() function, open the keyboard device /dev/hidg0 and the mouse device /dev/hidg1 to obtain the corresponding descriptors for keyboard and mouse read and write operations

**C++** AI generated projects 登录复制 run

```
if (!keyboardPath.empty())
{
    keyboardFd =
    open(keyboardPath.c_str(), O_RDWR | O_CLOEXEC | O_NONBLOCK);
    if (keyboardFd < 0)
```

```
    {
        log<level::ERR>("Failed to open input device",entry("PATH=%s", keyboardPath.c_str()),entry("ERROR=%s", strerror(errno)));
        elog<Open>(xyz::openbmc_project::Common::File::Open::ERRNO(errno),xyz::openbmc_project::Common::File::Open::PATH(
        keyboardPath.c_str()));
    }
}

if (!pointerPath.empty())
{
    pointerFd = open(pointerPath.c_str(), O_RDWR | O_CLOEXEC | O_NONBLOCK);
    if (pointerFd < 0)
    {
        log<level::ERR>("Failed to open input device",entry("PATH=%s", pointerPath.c_str()),entry("ERROR=%s", strerror(errno)));
        elog<Open>(xyz::openbmc_project::Common::File::Open::ERRNO(errno),xyz::openbmc_project::Common::File::Open::PATH(
        pointerPath.c_str()));
    }
twen
twen}
```

收起 ^

- The keyboard and mouse write operation API function is as follows

C++ AI generated projects 登录复制 run

```
/* @brief File descriptor for the USB keyboard device */
int keyboardFd;
/* @brief File descriptor for the USB mouse device */
int pointerFd;
/* @brief Data for keyboard report */
uint8_t keyboardReport[KEY_REPORT_LENGTH];
/* @brief Data for pointer report */
uint8_t pointerReport[PTR_REPORT_LENGTH];
```

**2. Uninstallation steps**

- When closing a KVM session, it will first check whether it is the last session. If it is, it will execute the server->input.disconnect() function to disconnect the USB device.

AI generated projects 登录复制

```
void Server::clientGone(rfbClientPtr cl)
{
    Server* server = (Server*)cl->screen->screenData;

    delete (ClientData*)cl->clientData;
    cl->clientData = nullptr;
    //后置操作：numClients返回当前值判断是否==1，然后才递减，当最后一个会话时，执行断开操作
```

```
    if (server->numClients-- == 1)
    {
        server->input.disconnect();
        rfbMarkRectAsModified(server->server, 0, 0, server->video.getWidth(),
                              server->video.getHeight());
    }
}
```

收起 ^

- In the Input::disconnect() function, USBGadget is turned off by clearing the UDC of the keyboard and mouse device, which is equivalent to executing: **echo "">/sys/kernel/config/usb_gadget/obmc_hid/UDC**

C++ AI generated projects 登录复制 run

```
void Input::disconnect()
{
    if (keyboardFd >= 0)
    {
        close(keyboardFd);
        keyboardFd = -1;
    }
    if (pointerFd >= 0)
    {
        close(pointerFd);
        pointerFd = -1;
    }

    try
    {
        hidUdcStream << "" << std::endl; //
    }
    catch (std::ofstream::failure& e)
    {
        log<level::ERR>("Failed to disconnect HID gadget",entry("ERROR=%s", e.what()));
    }
}
```

收起 ^

**3. Data read and write operations**

- The length of keyboard and mouse data is 8Bytes/6Bytes respectively, and the corresponding write operation functions are in ikvm_input.cpp
- The Retry mechanism is introduced in the two write functions, which will try to rewrite 5 times with an interval of 10ms.
- In actual testing, it was found that 10ms is not a suitable value. When continuous transmission fails, the frame rate of the imported KVM video may drop, causing the KVM screen to freeze.

C++ AI generated projects 登录复制 run

```cpp
static constexpr int KEY_REPORT_LENGTH = 8;
static constexpr int PTR_REPORT_LENGTH = 6;
 static constexpr int HID_REPORT_RETRY_MAX = 5;

bool Input::writeKeyboard(const uint8_t* report)
void Input::writePointer(const uint8_t* report)

bool Input::writeKeyboard(const uint8_t* report)
{
    std::unique_lock<std::mutex> lk(keyMutex);
    uint retryCount = HID_REPORT_RETRY_MAX;

    while (retryCount > 0)
    {
        if (write(keyboardFd, report, KEY_REPORT_LENGTH) == KEY_REPORT_LENGTH)
        {
            return true;
        }

        if (errno != EAGAIN)
        {
            if (errno != ESHUTDOWN)
            {
                log<level::ERR>("Failed to write keyboard report",entry("ERROR=%s", strerror(errno)));
            }

            break;
        }

        lk.unlock();
        std::this_thread::sleep_for(std::chrono::milliseconds(10));
        lk.lock();
        retryCount--;
    }

    return false;
}
```

收起 ^

**4. Defect Analysis**

- The current OpenBMC architecture has a keyboard and mouse Write() retransmission timeout of 10ms. This will cause the KVM frame rate to drop and the interface to freeze when the mouse is moved quickly on the Bios interface.

- The current OpenBMC architecture keyboard and mouse Write() operation is violent and direct, especially when Fd is defined as O_NONBLOCK. A large amount of data impact will cause Driver USB transmission abnormality, thus causing the keyboard and mouse to be invalid.
- A better solution is to adjust the retransmission interval and add a Poll mechanism before sending (the driver provides this definition, but obmc-ikvm does not call it)



```
static __poll_t f_hidg_poll(struct file *file, poll_table *wait)
{
    struct f_hidg   *hidg  = file->private_data;
    __poll_t    ret = 0;

    poll_wait(file, &hidg->read_queue, wait);
    poll_wait(file, &hidg->write_queue, wait);

    if (WRITE_COND)
        ret |= EPOLLOUT | EPOLLWRNORM;

    if (hidg->use_out_ep) {
        if (READ_COND_INTOUT)
            ret |= EPOLLIN | EPOLLRDNORM;
    } else {
        if (READ_COND_SSREPORT)
            ret |= EPOLLIN | EPOLLRDNORM;
    }

    return ret;
}

bool Input::writeKeyboard()
{
    struct pollfd fds[1];

    fds[0] = keyboardFd;
    fds[1].events = POLLOUT;

    while(retryCount > 0)
    {
        ret = poll(fds, 1, 200)
        if(ret == 0)
        {
            log<level::ERR>("Write keyboard report timeout!");
        }
        else if(ret < 0)
        }
            log<level::ERR>("Write keyboard report error!");
        }
```

```
        else if(fds[0].revents & POLLOUT)
        {
            if (write(keyboardFd, report, KEY_REPORT_LENGTH) == KEY_REPORT_LENGTH)
            {
                return true;
            }
            ..............
        }
    }
}
```

收起 ^