HVCS IBM "Hypervisor Virtual Console Server" Installation Guide

for Linux Kernel 2.6.4+

Author(s): Ryan S. Arnold <rsa@us.ibm.com>

Date Created: March, 02, 2004 Last Changed: August, 24, 2004

1. Driver Introduction:

This is the device driver for the IBM Hypervisor Virtual Console Server, "hvcs". The IBM hvcs provides a tty driver interface to allow Linux user space applications access to the system consoles of logically partitioned operating systems (Linux and AIX) running on the same partitioned Power5 ppc64 system. Physical hardware consoles per partition are not practical on this hardware so system consoles are accessed by this driver using firmware interfaces to virtual terminal devices.

2. System Requirements:

This device driver was written using 2.6.4 Linux kernel APIs and will only build and run on kernels of this version or later.

This driver was written to operate solely on IBM Power5 ppc64 hardware though some care was taken to abstract the architecture dependent firmware calls from the driver code.

Sysfs must be mounted on the system so that the user can determine which major and minor numbers are associated with each vty-server. Directions for sysfs mounting are outside the scope of this document.

3. Build Options:

The hvcs driver registers itself as a tty driver. The tty layer dynamically allocates a block of major and minor numbers in a quantity requested by the registering driver. The hvcs driver asks the tty layer for 64 of these major/minor numbers by default to use for hvcs device node entries.

If the default number of device entries is adequate then this driver can be built into the kernel. If not, the default can be over-ridden by inserting the driver as a module with insmod parameters.

3.1 Built-in:

The following menuconfig example demonstrates selecting to build this driver into the kernel:

```
Device Drivers  --->
        Character devices  --->
                <*> IBM Hypervisor Virtual Console Server Support
```

Begin the kernel make process.

3.2 Module:

The following menuconfig example demonstrates selecting to build this driver as a kernel module:

```
Device Drivers  --->
        Character devices  --->
                <M> IBM Hypervisor Virtual Console Server Support
```

The make process will build the following kernel modules:

- hvcs.ko
- hvcserver.ko

To insert the module with the default allocation execute the following commands in the order they appear:

```
insmod hvcserver.ko
insmod hvcs.ko
```

The hvcserver module contains architecture specific firmware calls and must be inserted first, otherwise the hvcs module will not find some of the symbols it expects.

To override the default use an insmod parameter as follows (requesting 4 tty devices as an example):

```
insmod hvcs.ko hvcs_parm_num_devs=4
```

There is a maximum number of dev entries that can be specified on insmod. We think that 1024 is currently a decent maximum number of server adapters to allow. This can always be changed by modifying the constant in the source file before building.

NOTE: The length of time it takes to insmod the driver seems to be related to the number of tty interfaces the registering driver requests.

In order to remove the driver module execute the following command:

```
rmmod hvcs.ko
```

The recommended method for installing hvcs as a module is to use depmod to build a current modules.dep file in /lib/modules/*uname -r* and then execute:

```
modprobe hvcs hvcs_parm_num_devs=4
```

The modules.dep file indicates that hvcserver.ko needs to be inserted before hvcs.ko and modprobe uses this file to smartly insert the modules in the proper order.

The following modprobe command is used to remove hvcs and hvcserver in the proper order:

```
modprobe -r hvcs
```

4. Installation:

The tty layer creates sysfs entries which contain the major and minor numbers allocated for the hvcs driver. The following snippet of "tree" output of the sysfs directory shows where these numbers are presented:

```
sys/
|-- *other sysfs base dirs*
|
|-- class
|   |-- *other classes of devices*
|   |
|   `-- tty
|       |-- *other tty devices*
|       |
|       |-- hvcs0
|       |   `-- dev
|       |-- hvcs1
|       |   `-- dev
|       |-- hvcs2
|       |   `-- dev
|       |-- hvcs3
|       |   `-- dev
|       |
|       |-- *other tty devices*
|
|-- *other sysfs base dirs*
```

For the above examples the following output is a result of cat'ing the "dev" entry in the hvcs directory:

```
Pow5:/sys/class/tty/hvcs0/ # cat dev
254:0

Pow5:/sys/class/tty/hvcs1/ # cat dev
254:1

Pow5:/sys/class/tty/hvcs2/ # cat dev
254:2

Pow5:/sys/class/tty/hvcs3/ # cat dev
254:3
```

The output from reading the "dev" attribute is the char device major and minor numbers that the tty layer has allocated for this driver's use. Most systems running hvcs will already have the device entries created or udev will do it automatically.

Given the example output above, to manually create a /dev/hvcs* node entry mknod can be used as follows:

```
mknod /dev/hvcs0 c 254 0
mknod /dev/hvcs1 c 254 1
mknod /dev/hvcs2 c 254 2
mknod /dev/hvcs3 c 254 3
```

Using mknod to manually create the device entries makes these device nodes persistent. Once created they will exist prior to the driver insmod.

Attempting to connect an application to /dev/hvcs* prior to insertion of the hvcs module will result in an error message similar to the following:

```
"/dev/hvcs*: No such device".
```

NOTE: Just because there is a device node present doesn't mean that there is a vty-server device configured for that node.

5. Connection

Since this driver controls devices that provide a tty interface a user can interact with the device node entries using any standard tty-interactive method (e.g. "cat", "dd", "echo"). The intent of this driver however, is to provide real time console interaction with a Linux partition's console, which requires the use of applications that provide bi-directional, interactive I/O with a tty device.

Applications (e.g. "minicom" and "screen") that act as terminal emulators or perform terminal type control sequence conversion on the data being passed through them are NOT acceptable for providing interactive console I/O. These programs often emulate antiquated terminal types (vt100 and ANSI) and expect inbound data to take the form of one of these supported terminal types but they either do not convert, or do not _adequately_ convert, outbound data into the terminal type of the terminal which invoked them (though screen makes an attempt and can apparently be configured with much termcap wrestling.)

For this reason kermit and cu are two of the recommended applications for interacting with a Linux console via an hvcs device. These programs simply act as a conduit for data transfer to and from the tty device. They do not require inbound data to take the form of a particular terminal type, nor do they cook outbound data to a particular terminal type.

In order to ensure proper functioning of console applications one must make sure that once connected to a /dev/hvcs console that the console's $TERM env variable is set to the exact terminal type of the terminal emulator used to launch the interactive I/O application. If one is using xterm and kermit to connect to /dev/hvcs0 when the console prompt becomes available one should "export TERM=xterm" on the console. This tells ncurses applications that are invoked from the console that they should output control sequences that xterm can understand.

As a precautionary measure an hvcs user should always "exit" from their session before disconnecting an application such as kermit from the device node. If this is not done, the next user to connect to the console will continue using the previous user's logged in session which includes using the $TERM variable that the previous user supplied.

Hotplug add and remove of vty-server adapters affects which /dev/hvcs* node is used to connect to each vty-server adapter. In order to determine which vty-server adapter is associated with which /dev/hvcs* node a special sysfs attribute has been added to each vty-server sysfs entry. This entry is called "index" and showing it reveals an integer that refers to the /dev/hvcs* entry to use to connect to that device. For instance cating the index attribute of vty-server adapter 30000004 shows the following:

```
Pow5:/sys/bus/vio/drivers/hvcs/30000004 # cat index
2
```

This index of '2' means that in order to connect to vty-server adapter 30000004 the user should interact with /dev/hvcs2.

It should be noted that due to the system hotplug I/O capabilities of a system the /dev/hvcs* entry that interacts with a particular vty-server adapter is not guaranteed to remain the same across system reboots. Look in the Q & A section for more on this issue.

6. Disconnection

As a security feature to prevent the delivery of stale data to an unintended target the Power5 system firmware disables the fetching of data and discards that data when a connection between a vty-server and a vty has been severed. As an example, when a vty-server is immediately disconnected from a vty following output of data to the vty the vty adapter may not have enough time between when it received the data interrupt and when the connection was severed to fetch the data from firmware before the fetch is disabled by firmware.

When hvcs is being used to serve consoles this behavior is not a huge issue because the adapter stays connected for large amounts of time following almost all data writes. When hvcs is being used as a tty conduit to tunnel data between two partitions [see Q & A below] this is a huge problem because the standard Linux behavior when cat'ing or dd'ing data to a device is to open the tty, send the data, and then close the tty. If this driver manually terminated vty-server connections on tty close this would close the vty-server and vty connection before the target vty has had a chance to fetch the data.

Additionally, disconnecting a vty-server and vty only on module removal or adapter removal is impractical because other vty-servers in other partitions may require the usage of the target vty at any time.

Due to this behavioral restriction disconnection of vty-servers from the connected vty is a manual procedure using a write to a sysfs attribute outlined below, on the other hand the initial vty-server connection to a vty is established automatically by this driver. Manual vty-server connection is never required.

In order to terminate the connection between a vty-server and vty the "vterm_state" sysfs attribute within each vty-server's sysfs entry is used. Reading this attribute reveals the current connection state of the vty-server adapter. A zero means that the vty-server is not connected to a vty. A one indicates that a connection is active.

Writing a '0' (zero) to the vterm_state attribute will disconnect the VTERM connection between the vty-server and target vty ONLY if the vterm_state previously read '1'. The write directive is ignored if the vterm_state read '0' or if any value other than '0' was written to the vterm_state attribute. The following example will show the method used for verifying the vty-server connection status and disconnecting a vty-server connection:

```
Pow5:/sys/bus/vio/drivers/hvcs/30000004 # cat vterm_state
1
```

```
Pow5:/sys/bus/vio/drivers/hvcs/30000004 # echo 0 > vterm_state

Pow5:/sys/bus/vio/drivers/hvcs/30000004 # cat vterm_state
0
```

All vty-server connections are automatically terminated when the device is hotplug removed and when the module is removed.

7. Configuration

Each vty-server has a sysfs entry in the /sys/devices/vio directory, which is symlinked in several other sysfs tree directories, notably under the hvcs driver entry, which looks like the following example:

```
Pow5:/sys/bus/vio/drivers/hvcs # ls
.  ..  30000003  30000004  rescan
```

By design, firmware notifies the hvcs driver of vty-server lifetimes and partner vty removals but not the addition of partner vtys. Since an HMC Super Admin can add partner info dynamically we have provided the hvcs driver sysfs directory with the "rescan" update attribute which will query firmware and update the partner info for all the vty-servers that this driver manages. Writing a '1' to the attribute triggers the update. An explicit example follows:

```
Pow5:/sys/bus/vio/drivers/hvcs # echo 1 > rescan
```

Reading the attribute will indicate a state of '1' or '0'. A one indicates that an update is in process. A zero indicates that an update has completed or was never executed.

Vty-server entries in this directory are a 32 bit partition unique unit address that is created by firmware. An example vty-server sysfs entry looks like the following:

```
Pow5:/sys/bus/vio/drivers/hvcs/30000004 # ls
.   current_vty   devspec       name          partner_vtys
..  index         partner_clcs  vterm_state
```

Each entry is provided, by default with a "name" attribute. Reading the "name" attribute will reveal the device type as shown in the following example:

```
Pow5:/sys/bus/vio/drivers/hvcs/30000003 # cat name
vty-server
```

Each entry is also provided, by default, with a "devspec" attribute which reveals the full device specification when read, as shown in the following example:

```
Pow5:/sys/bus/vio/drivers/hvcs/30000004 # cat devspec
/vdevice/vty-server@30000004
```

Each vty-server sysfs dir is provided with two read-only attributes that provide lists of easily parsed partner vty data: "partner_vtys" and "partner_clcs":

```
Pow5:/sys/bus/vio/drivers/hvcs/30000004 # cat partner_vtys
30000000
30000001
30000002
30000000
30000000

Pow5:/sys/bus/vio/drivers/hvcs/30000004 # cat partner_clcs
U5112.428.103048A-V3-C0
U5112.428.103048A-V3-C2
U5112.428.103048A-V3-C3
U5112.428.103048A-V4-C0
U5112.428.103048A-V5-C0
```

Reading partner_vtys returns a list of partner vtys. Vty unit address numbering is only per-partition-unique so entries will frequently repeat.

Reading partner_clcs returns a list of "converged location codes" which are composed of a system serial number followed by "-V*", where the '*' is the target partition number, and "-C*", where the '*' is the slot of the adapter. The first vty partner corresponds to the first clc item, the second vty partner to the second clc item, etc.

A vty-server can only be connected to a single vty at a time. The entry, "current_vty" prints the clc of the currently selected partner vty when read.

The current_vty can be changed by writing a valid partner clc to the entry as in the following example:

```
Pow5:/sys/bus/vio/drivers/hvcs/30000004 # echo U5112.428.10304
8A-V4-C0 > current_vty
```

Changing the current_vty when a vty-server is already connected to a vty does not affect the current connection. The change takes

effect when the currently open connection is freed.

Information on the "vterm_state" attribute was covered earlier on the chapter entitled "disconnection".

8. Questions & Answers:

Q: What are the security concerns involving hvcs?

A: There are three main security concerns:

1. The creator of the /dev/hvcs* nodes has the ability to restrict the access of the device entries to certain users or groups. It may be best to create a special hvcs group privilege for providing access to system consoles.

2. To provide network security when grabbing the console it is suggested that the user connect to the console hosting partition using a secure method, such as SSH or sit at a hardware console.

3. Make sure to exit the user session when done with a console or the next vty-server connection (which may be from another partition) will experience the previously logged in session.

Q: How do I multiplex a console that I grab through hvcs so that other people can see it:

A: You can use "screen" to directly connect to the /dev/hvcs* device and setup a session on your machine with the console group privileges. As pointed out earlier by default screen doesn't provide the termcap settings for most terminal emulators to provide adequate character conversion from term type "screen" to others. This means that curses based programs may not display properly in screen sessions.

Q: Why are the colors all messed up? Q: Why are the control characters acting strange or not working? Q: Why is the console output all strange and unintelligible?

A: Please see the preceding section on "Connection" for a discussion of how applications can affect the display of character control sequences. Additionally, just because you logged into the console using and xterm doesn't mean someone else didn't log into the console with the HMC console (vt320) before you and leave the session logged in. The best thing to do is to export TERM to the terminal type of your terminal emulator when you get the console. Additionally make sure to "exit" the console before you disconnect from the console. This will ensure that the next user gets their own TERM type set when they login.

Q: When I try to CONNECT kermit to an hvcs device I get: "Sorry, can't open connection: /dev/hvcs*"What is happening?

A: Some other Power5 console mechanism has a connection to the vty and isn't giving it up. You can try to force disconnect the consoles from the HMC by right clicking on the partition and then selecting "close terminal". Otherwise you have to hunt down the people who have console authority. It is possible that you already have the console open using another kermit session and just forgot about it. Please review the console options for Power5 systems to determine the many ways a system console can be held.

OR

A: Another user may not have a connectivity method currently attached to a /dev/hvcs device but the vterm_state may reveal that they still have the vty-server connection established. They need to free this using the method outlined in the section on "Disconnection" in order for others to connect to the target vty.

OR

A: The user profile you are using to execute kermit probably doesn't have permissions to use the /dev/hvcs* device.

OR

A: You probably haven't inserted the hvcs.ko module yet but the /dev/hvcs* entry still exists (on systems without udev).

OR

A: There is not a corresponding vty-server device that maps to an existing /dev/hvcs* entry.

Q: When I try to CONNECT kermit to an hvcs device I get: "Sorry, write access to UUCP lockfile directory denied."

A: The /dev/hvcs* entry you have specified doesn't exist where you said it does? Maybe you haven't inserted the module (on systems with udev).

Q: If I already have one Linux partition installed can I use hvcs on said partition to provide the console for the install of a second Linux partition?

A: Yes granted that your are connected to the /dev/hvcs* device using kermit or cu or some other program that doesn't provide terminal emulation.

Q: Can I connect to more than one partition's console at a time using this driver?

A: Yes. Of course this means that there must be more than one vty-server configured for this partition and each must point to a disconnected vty.

Q: Does the hvcs driver support dynamic (hotplug) addition of devices?

A: Yes, if you have dlpar and hotplug enabled for your system and it has been built into the kernel the hvcs drivers is configured to dynamically handle additions of new devices and removals of unused devices.

Q: For some reason /dev/hvcs* doesn't map to the same vty-server adapter after a reboot. What happened?

A: Assignment of vty-server adapters to /dev/hvcs* entries is always done in the order that the adapters are exposed. Due to hotplug capabilities of this driver assignment of hotplug added vty-servers may be in a different order than how they would be exposed on module load. Rebooting or reloading the module after dynamic addition may result in the /dev/hvcs* and vty-server coupling changing if a vty-server adapter was added in a slot between two other vty-server adapters. Refer to the section above on how to determine which vty-server goes with which /dev/hvcs* node. Hint; look at the sysfs "index" attribute for the vty-server.

Q: Can I use /dev/hvcs* as a conduit to another partition and use a tty device on that partition as the other end of the pipe?

A: Yes, on Power5 platforms the hvc_console driver provides a tty interface for extra /dev/hvc* devices (where /dev/hvc0 is most likely the console). In order to get a tty conduit working between the two partitions the HMC Super Admin must create an additional "serial server" for the target partition with the HMC gui which will show up as /dev/hvc* when the target partition is rebooted.

The HMC Super Admin then creates an additional "serial client" for the current partition and points this at the target partition's newly created "serial server" adapter (remember the slot). This shows up as an additional /dev/hvcs* device.

Now a program on the target system can be configured to read or write to /dev/hvc* and another program on the current partition can be configured to read or write to /dev/hvcs*. Now you have a tty conduit between two partitions.

---

9. Reporting Bugs:

The proper channel for reporting bugs is either through the Linux OS distribution company that provided your OS or by posting issues to the PowerPC development mailing list at:

linuxppc-dev@lists.ozlabs.org

This request is to provide a documented and searchable public exchange of the problems and solutions surrounding this driver for the benefit of all users.