Technical Marketing Competence Center Europe
IBM Deutschland Research & Development GmbH

Linux multipathing: failover mode with preferred paths

Technical White Paper

IBM contact person: Marc Beyerle
System z Specialist
TMCC Europe
marc.beyerle@de.ibm.com

TMCC project number: 8115

Document version: 1.0

Document date: 01/28/2009

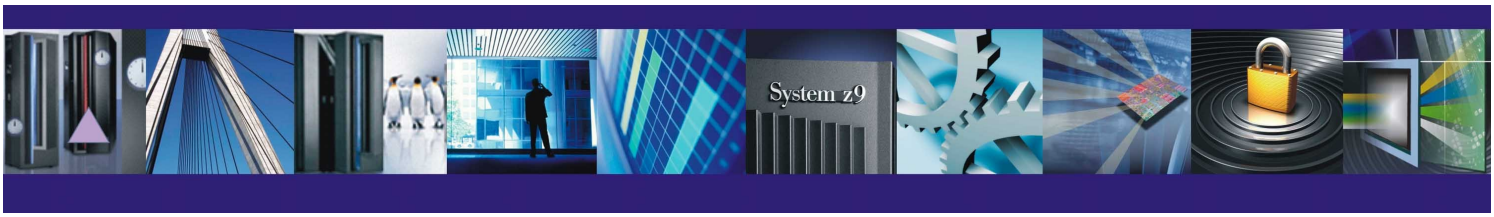

Table of Contents

1 Executive Summary

This Technical White Paper describes the procedure of re-configuring Linux® multipathing for optimized load balancing using the so-called *group-by-prio* mode. Even though the group-by-prio mode may be considered merely a special case of the *failover* mode, the assignment of *active* paths (that is, *primary* paths) and paths in *enabled* state (that is, *secondary* or backup paths) can only be controlled when using the group-by-prio configuration setting. In the case of Linux multipathing, failover means that in the event of a failure of the currently active path, I/O traffic is automatically routed over the next available secondary path.

Motivated by IBM® Mainframe customers asking for this kind of control over the priority assignment procedure, this paper describes the steps necessary to achieve a failover configuration with preferred paths. While most of the contents of this paper holds true for all Linux platforms, the procedures and shell scripts were developed for and tested under Linux on System z®, specifically Novell® SUSE® Linux Enterprise Server 10.

2 Implementation

2.1 Technical background

According to the homepage of the Linux multipath implementation, the term *multipath* simply means that a host can access a *Logical Unit* (LU) by multiple paths. In this terminology, a *Logical Unit Number* (LUN) represents an identifier for an individual SCSI device. For more information about Fibre Channel (FC) Storage Area Networks (SANs), please see the Linux multipath homepage:

- http://christophe.varoqui.free.fr/multipath.html

A *path*, in turn, is defined as a connection from a port in a host to a port in a target device. A port is uniquely identified by a *World Wide Port Name* (WWPN). In the definition of "path", any intermediate routing points in a SAN are ignored. Therefore, the combination of one *Host Bus Adapter* (HBA) to one target is one path, regardless of how it is routed. Hence, a path in zFCP can be represented by the 3-tuple `(ZFCP_ADAPTER, WWPN, LUN)`.

The multipath to the physical storage device is logically divided into so-called *priority groups*. Depending on the configuration, there can be either exactly one or an arbitrary number of priority groups per multipath. Currently, there are five spreading policies for priority groups:

- failover: 1 path per priority group
- multibus: 1 priority group containing all paths to the LUN
- group_by_serial: 1 priority group per serial number
- group_by_prio: 1 priority group per priority value
- group_by_node_name: 1 priority group per target node name

You can visualize priority groups by looking at the output of the `multipath` command line utility. Here is an example for *multibus* mode, which is the default setting for many storage controllers:

```
36005076303ffc54800000000000000111f dm-0 IBM,2107900
[size=40G][features=1 queue_if_no_path][hwhandler=0]
\_ round-robin 0 [prio=2][active]
 \_ 1:0:0:1075789841 sdb   8:16  [active][ready]
 \_ 0:0:0:1075789841 sda   8:0   [active][ready]
```

In the above output, you can see one priority group (marked **red**) containing two SCSI devices (`sda` and `sdb`) for the multipath to LUN `36005076303ffc54800000000000000111f`. If you don't explicitly change the configuration file, the first 1000 I/O operations will be performed over the first logical SCSI device (that is, over the first path to the physical device), and then the next 1000 I/O operations will be performed over the next logical SCSI device (that is, over the next path to the physical device), and so forth, resulting in a round-robin access mode. Linux multipathing cannot access the physical storage device behind a multipath in parallel by using several paths at once. Therefore, multibus mode will always result in the above described round-robin access over the individual paths in the current implementation of Linux multipathing.

One motivation for using failover mode instead of multibus is performance. According to measurements performed by the Linux on System z performance team, *"...failover mode shows better performance, causes lower CPU costs, and ensures better throughput rates than multibus"*. For more details, see the corresponding web page:

- http://www.ibm.com/developerworks/linux/linux390/perf/tuning_rec_dasd_multipath.html

If you are using *failover* mode, then the primary path will be automatically set to the one, which is activated first. See the following output for an example for failover mode:

```
36005076303ffc54800000000000000111f dm-0 IBM,2107900
[size=40G][features=1 queue_if_no_path][hwhandler=0]
\_ round-robin 0 [prio=1][active]
 \_ 1:0:0:1075789841 sdb   8:16  [active][ready]
\_ round-robin 0 [prio=1][enabled]
 \_ 0:0:0:1075789841 sda   8:0   [active][ready]
```

As you can see, failover automatically creates two priority groups (marked **red**), each consisting of one path. By default, both priority groups have the same priority value ("prio=1"), and therefore the sequence of activation of the paths determines the currently active priority group. Incidentally, activating the paths in a specific order is a perfectly legal way to define primary paths, and many customers set up their Linux multipathing using this technique. However, if you do not want to depend on the path activation sequence, the only possibility to define one path as being the primary one, is to use a so-called *priority callout* script.

This script will assign a priority value to each logical SCSI device that it is invoked with. If you want to use this technique, you will have to change your multipath configuration file to use the `group_by_prio` spreading policy. From a configuration file point of view, this is of course different from failover at a first glance. However, the outcome of this setting will behave exactly like a failover configuration, with the sole difference that the priority values generated by the script determine the primary and secondary paths.

2.2 Configuring failover mode with preferred paths

The default spreading policy for priority groups depends on both the storage controller as well as the Linux distribution. Before you start to configure the group-by-prio mode, you have to think carefully about the distribution of paths to the LUNs involved in your setup.

So for example, if your storage unit contains two switches at its entry point, over which the entire I/O traffic is routed into the storage controllers, then you will have to configure the same amount of active paths (that is, primary paths) for each of the two switches. Otherwise, one of the switches will be under heavier load than the other and congestion is likely to occur. Since this paper cannot cover all storage units available on the market, you will have to consult your storage unit's documentation in order to take into account vendor specific implementation details.

For the IBM System Storage™ DS8000™ series of disk storage subsystems, please see the IBM Redbook "IBM System Storage DS8000 – Architecture and Implementation" for a detailed technical description and performance-related setup recommendations:

- http://www.redbooks.ibm.com/abstracts/sg246786.html

Once you have decided on the assignment of the paths, write down the priority setup of your paths in a plain text file called `/root/zfcp_priorities.cfg`. The following is a simple example for one LUN accessed over two paths:

```
0.0.7e00/0x5005076303184548/0x4011401f00000000 1
0.0.7f00/0x5005076303134548/0x4011401f00000000 0
```

The format of the above lines is:

```
<device_bus_id>/<wwpn>/<lun> <priority_value>
```

Note that `0` and `1` are arbitrary priority values. You can use any values you like, but the value for the primary path has to be greater than the value for the secondary path. If you want to use more than two paths (for example: 3) for your failover configuration, you may define a priority value sequence (for example: `0`, `1`, `2`).

The following procedure was tested under Linux on System z installations, where the entire root filesystem resides on *Extended Count Key Data* (ECKD™) devices and only filesystems used by applications are located on SCSI devices. This storage setup is a best practice from a systems programming point of view. A major advantage is that the operating system itself is still bootable, even if there is a misconfiguration in the SCSI setup. Here are the instructions:

- Unmount all file systems, whose underlying devices use SCSI-based multipathing:

  ```
  umount ...
  ```

- Flush all unused multipath device maps:

  ```
  multipath -F
  ```

- Sometimes, a few multipaths remain active. In such a situation, you have to remove them using the device-mapper command line utility:

  ```
  dmsetup remove_all
  ```

- Now set all SCSI devices offline, using the Linux sysfs. You have to repeat the following command for each individual SCSI device:

  ```
  echo 1 > /sys/class/scsi_device/<device_name>/device/delete
  ```

  Alternatively, you can use the script `remove_scsi.sh`, provided in section A.1 on page 11 to perform the above task.

- Double-check that there is no SCSI device left over by issuing:

  ```
  lsscsi
  ```

- Now you have to remove all LUN and WWPN information for each zFCP adapter in the Linux sysfs:

  ```
  echo <lun1> > /sys/bus/ccw/drivers/zfcp/<device_bus_id>/<wwpn>/unit_remove
  echo <lun2> > /sys/bus/ccw/drivers/zfcp/<device_bus_id>/<wwpn>/unit_remove
  ...
  echo <wwpn> > /sys/bus/ccw/drivers/zfcp/<device_bus_id>/port_remove
  ```

Depending on the number of LUNs, the above task may result in a large amount of command line work. Alternatively, you can use the script `clean_zfcp_sysfs.sh`, provided in section A.2 on page 12 to perform the above task.

- Double-check that there is no LUN left over by issuing:

  ```
  lszfcp -D
  ```

- Now configure all zFCP adapters offline:

  ```
  hwdown zfcp-bus-ccw-<device_bus_id>
  ```

  The argument to the `hwdown` script is a so-called *configuration name*. You can find all configuration names for your installation in `/etc/sysconfig/hardware`. However, keep in mind to only set the zFCP devices offline.

- Stop the multipath daemon:

  ```
  service multipathd stop
  ```

- Now modify your multipath configuration file `/etc/multipath.conf` in order to reflect the change to group-by-prio mode. See section A.4 on page 17 for an example. Please note that in our test installation, the `prio_callout` statement had to be placed inside the `device` section and not the `defaults` section. Otherwise, the statement was ignored and the priority callout script was not executed.

- Install your priority callout script (see section A.5 on page 18 for a specific example).

- Install the priority configuration file you created at the beginning of this section. If you created it as `/root/zfcp_priorities.cfg`, you are already done with this step.

- Now configure all zFCP adapters online again:

  ```
  hwup zfcp-bus-ccw-<device_bus_id>
  ```

  Again, you can find all configuration names for your installation in `/etc/sysconfig/hardware`. However, keep in mind to only set the zFCP devices online.

- Now you have to add all WWPN and LUN information for each zFCP adapter in the Linux sysfs:

  ```
  echo <wwpn> > /sys/bus/ccw/drivers/zfcp/<device_bus_id>/port_add
  echo <lun1> > /sys/bus/ccw/drivers/zfcp/<device_bus_id>/<wwpn>/unit_add
  echo <lun2> > /sys/bus/ccw/drivers/zfcp/<device_bus_id>/<wwpn>/unit_add
  ...
  ```

  Depending on the number of LUNs, the above task may result in a large amount of command line work. Alternatively, you can use the script `activate_luns.sh`, provided in section A.3 on page 14 to perform the above task. This script parses all zFCP adapter configuration files found in `/etc/sysconfig/hardware`, so you have to make sure that there are no additional or unused zFCP adapter configuration files in this directory. Furthermore, the script will only recognize zFCP adapter configuration files starting with the default prefix `hwcfg-zfcp-bus-ccw-...`

- Start the multipath daemon:

  ```
  service multipathd start
  ```

- Check the multipath configuration:

  ```
  multipath -ll
  ```

If everything went okay, you will see an output of the multipath command line utility similar to the following:

```
36005076303ffc54800000000000000111f dm-0 IBM,2107900
[size=40G][features=1 queue_if_no_path][hwhandler=0]
\_ round-robin 0 [prio=1][active]
 \_ 0:0:0:1075789841 sda   8:0   [active][ready]
\_ round-robin 0 [prio=0][enabled]
 \_ 1:0:0:1075789841 sdb   8:16  [active][ready]
```

The output looks almost identical to the failover mode output listed earlier in this section. The only noticeable difference is that we now have a dedicated primary path, which is the one with the higher priority ("prio=1").

2.3 Priority callout performance

If your hardware setup contains a large number of LUNs, the execution of the `multipath` command line utility might take a large amount of time, when you are using the priority callout script listed in section A.5 - up to several minutes for some installations. This is due to the fact, that the priority callout script has to map the logical SCSI device that it is called with (for example: `sda`) to the physical zFCP path, over which it is accessed (for example: `0.0.7e00/0x5005076303184548/0x4011401f00000000`). In order to do so, it has to execute the `lszfcp` command line utility for each individual SCSI device.

If you need to improve the performance of the priority callout script, you will have to perform a few changes to the existing setup. The idea behind those changes is to save the current zFCP configuration in a file called `/root/lszfcp.current` and re-use its contents for subsequent executions of the priority callout script. Ideally, you create this file each time your Linux server is rebooted. The standard way of doing this in SUSE Linux is to add a new boot script called `boot.aaa_lszfcp`. You can find the contents of this script in section A.6 on page 20. You activate it by issuing the following command:

- `insserv -v /etc/init.d/boot.aaa_lszfcp`

Now you have to locate the `"lszfcp -D"` command in the priority callout script and replace the corresponding line in the following way:

- Before:

  ```
  ZFCPPATH=$(lszfcp -D | grep "${SCSIDEVICE}\$" ...
  ```

- After:

  ```
  ZFCPPATH=$(cat /root/lszfcp.current | grep "${SCSIDEVICE}\$" ...
  ```

In order to create an initial `/root/lszfcp.current` file, interrupt the re-configuring procedure described in the last section just before starting the multipath daemon and issue the following command:

- `lszfcp -D > /root/lszfcp.current`

This completes the required changes. In our Linux on System z test installation, the `multipath` command line utility was virtually as fast as in a multipath configuration without priority callout script, provided that the above described changes are applied.

2.4 A note on the deprecated statement

If you execute the `multipath` command line utility, you will notice that it produces output containing *"Using deprecated prio_callout..."*. In the original Linux multipathing source code, using a priority callout script is not deprecated at the time of this writing. However, a new mechanism called `libprio` was implemented as part of the ongoing device-mapper development. A discussion about this topic can be found here:

- http://www.linux-archive.org/device-mapper-development/142344-custom-priority-callout.html

This new mechanism ensures that the code, which generates the priority values, is kept in memory in case of a disk loss. The downside of this new approach is that you have to spend quite a lot of development and testing effort compared to writing a priority callout script: You have to write and test C code and include it into the existing `libprio` library.

However, in our setup, the root filesystem resides on ECKD. Therefore, the priority callout script is always available, even in case of a loss of the SCSI devices. Hence, the *"... deprecated"* message can be safely ignored in a mixed ECKD / SCSI environment, where Linux multipathing is only used for the SCSI devices.

2.5 Further information

If you want more information regarding this Technical White Paper or if you need technical assistance in an IBM System z related Proof of Concept project, do not hesitate to contact the author Marc Beyerle (marc.beyerle@de.ibm.com). Alternatively, you may also contact the Technical Marketing Competence Center Europe at tmcc@de.ibm.com

A Shell scripts

All shell scripts in this chapter were written for and tested under *SUSE LINUX Enterprise Server 10* (SLES10) SP2, s390x (64-bit). Nevertheless, please read the following disclaimer for the scripts provided in this appendix:

Disclaimer of warranties

The shell scripts in this document are created by IBM. The shell scripts are provided to you solely for internal usage only. The shell scripts are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the shell scripts, even if they have been advised of the possibility of such damages.

A.1 Set all SCSI devices offline

File name:

`/usr/local/sbin/remove_scsi_devices.sh`

File contents:

```
#!/bin/sh
#
# ------------------------------------------------------------------
# |   REMOVES ALL SCSI DEVICES                                     |
# |                                                                |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 12/12/2008   |
# |   Copyright IBM Corp. 2008                                     |
# ------------------------------------------------------------------
#

# the sysfs directory containing all scsi devices
SYSFS_DIRECTORY="/sys/class/scsi_device"

if [ -e "${SYSFS_DIRECTORY}" ]; then
   # change to the sysfs directory containing all scsi devices
   cd "${SYSFS_DIRECTORY}"

   # find all scsi devices
   DEVICES=$(find . -maxdepth 1 | grep -v "^.$" | cut -c 3-)
else
   # the sysfs directory for the scsi devices does not exist
   DEVICES=""
fi

# print an empty line
echo

if [ -z "${DEVICES}" ]; then
   # print information
   echo "No SCSI devices found."
fi

# iterate over all scsi devices found
for DEVICE in ${DEVICES}; do
   # change to this scsi device's directory
   cd "${SYSFS_DIRECTORY}/${DEVICE}/device"

   # print information
   echo -n "Removing SCSI device ${DEVICE}... "

   # remove this scsi device
   echo 1 > delete

   if [ "${?}" -eq "0" ]; then
      # print success message (otherwise the error is printed above)
      echo "OK"
   fi
done

# print an empty line
echo
```

A.2 Remove all zFCP-related Linux sysfs information

File name:

```
/usr/local/sbin/clean_zfcp_sysfs.sh
```

File contents:

```
#!/bin/sh
#
# ------------------------------------------------------------------
# |   REMOVES ALL ZFCP-RELATED SYSFS ENTRIES                       |
# |                                                                |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 12/12/2008   |
# |   Copyright IBM Corp. 2008                                     |
# ------------------------------------------------------------------
#

# the sysfs directory containing all zfcp adapters
SYSFS_DIRECTORY="/sys/bus/ccw/drivers/zfcp"

if [ -e "${SYSFS_DIRECTORY}" ]; then
   # change to the sysfs directory containing all zfcp adapters
   cd "${SYSFS_DIRECTORY}"

   # find all zfcp adapters
   ADAPTERS=$(find . -maxdepth 1 -name "0.0.*" | cut -c 3-)
else
   # the sysfs directory for the zfcp adapters does not exist
   ADAPTERS=""
fi

if [ -z "${ADAPTERS}" ]; then
   # print information
   echo -e "\nNo zfcp adapters found."
fi

# iterate over all zfcp adapters found
for ADAPTER in ${ADAPTERS}; do
   # change to this zfcp adapter's directory
   cd "${SYSFS_DIRECTORY}/${ADAPTER}"

   # print information
   echo -e "\nNow removing all sysfs entries of zfcp adapter ${ADAPTER}."

   # find all wwpns of this zfcp adapter
   WWPNS=$(find . -maxdepth 1 -name "0x*" | cut -c 3-)

   if [ -z "${WWPNS}" ]; then
      # print information
      echo "No WWPNs found for zfcp adapter ${ADAPTER}."
   fi

   # iterate over all wwpns found
   for WWPN in ${WWPNS}; do
      # change to this wwpn's directory
      cd "${SYSFS_DIRECTORY}/${ADAPTER}/${WWPN}"
```

```
      # find all luns of this wwpn
      LUNS=$(find . -maxdepth 1 -name "0x*" | cut -c 3-)

      if [ -z "${LUNS}" ]; then
         # print information
         echo "No LUNs found for WWPN ${WWPN}."
      fi

      # iterate over all luns found
      for LUN in ${LUNS}; do
         # print information
         echo -n "Removing LUN ${LUN}... "

         # remove this lun
         echo ${LUN} > unit_remove

         if [ "${?}" -eq "0" ]; then
            # print success message (otherwise the error is printed above)
            echo "OK"
         fi
      done

      # change (again) to this zfcp adapter's directory
      cd "${SYSFS_DIRECTORY}/${ADAPTER}"

      # print information
      echo -n "Removing WWPN ${WWPN}... "

      # remove this wwpn
      echo ${WWPN} > port_remove

      if [ "${?}" -eq "0" ]; then
         # print success message (otherwise the error is printed above)
         echo "OK"
      fi
   done
done

# print an empty line
echo
```

A.3 Activate all WWPN and LUN information in sysfs

File name:

```
/usr/local/sbin/activate_luns.sh
```

File contents:

```
#!/bin/sh
#
# ------------------------------------------------------------------------
# |   ACTIVATES ALL WWPNS AND LUNS CONFIGURED IN /etc/sysconfig/hardware   |
# |                                                                        |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 12/12/2008           |
# |   Copyright IBM Corp. 2008                                             |
# ------------------------------------------------------------------------
#

# the /etc directory containing all zfcp adapter configurations
ETC_SYSCONFIG_HARDWARE="/etc/sysconfig/hardware"

# the sysfs directory containing all zfcp adapters
SYSFS_DIRECTORY="/sys/bus/ccw/drivers/zfcp"

if [ -e "${ETC_SYSCONFIG_HARDWARE}" ]; then
   # change to the directory containing all zfcp adapter configurations
   cd "${ETC_SYSCONFIG_HARDWARE}"

   # find all zfcp adapter configurations
   ADAPTERS=$(find -maxdepth 1 -name "hwcfg-zfcp-bus-ccw-*" | cut -c 22-)
else
   # the /etc directory containing all zfcp adapter configurations does not
exist
   ADAPTERS=""
fi

if [ -z "${ADAPTERS}" ]; then
   # print information
   echo -e "\nNo zfcp adapter configuration files found."
fi

# iterate over all zfcp adapter configurations found
for ADAPTER in ${ADAPTERS}; do
   if [ -e "${SYSFS_DIRECTORY}/${ADAPTER}" ]; then
      # change to this zfcp adapter's directory
      cd "${SYSFS_DIRECTORY}/${ADAPTER}"
   else
      # print information
      echo -e "\nNo sysfs entry for zfcp adapter ${ADAPTER} found."

      # skip this zfcp adapter
      continue
   fi

   # print information
   echo -e "\nNow activating all WWPNs and LUNs of zfcp adapter ${ADAPTER}."
```

```
   # source this zfcp adapter's configuration file
   . ${ETC_SYSCONFIG_HARDWARE}/hwcfg-zfcp-bus-ccw-${ADAPTER}

   # find all wwpns of this zfcp adapter
   WWPNS=$(echo ${ZFCP_LUNS} | tr " " "\n" | cut -d ":" -f 1 | sort -i | uniq
-i)

   if [ -z "${WWPNS}" ]; then
      # print information
      echo "No WWPNs found for zfcp adapter ${ADAPTER}."
   fi

   for WWPN in ${WWPNS}; do
      if [ -z "${WWPN}" ]; then
         # skip empty wwpns
         continue
      fi

      # print information
      echo -n "Activating WWPN ${WWPN}... "

      # activate this wwpn
      echo ${WWPN} > port_add

      if [ "${?}" -eq "0" ]; then
         # print success message (otherwise the error is printed above)
         echo "OK"
      fi

      while [ ! -e "${SYSFS_DIRECTORY}/${ADAPTER}/${WWPN}" ]; do
         # print information
         echo "Waiting for WWPN ${WWPN} to come up... ($(date +%r))"

         # give the wwpn some time to come up
         sleep 2
      done

      # change to this wwpn's directory
      cd "${SYSFS_DIRECTORY}/${ADAPTER}/${WWPN}"

      # find all luns of this wwpn
      LUNS=$(echo ${ZFCP_LUNS} | tr " " "\n" | grep ${WWPN} | cut -d ":" -f 2 |
sort -i | uniq -i)

      if [ -z "${LUNS}" ]; then
         # print information
         echo "No LUNs found for WWPN ${WWPN}."
      fi

      # iterate over all luns found
      for LUN in ${LUNS}; do
         if [ -z "${LUN}" ]; then
            # skip empty luns
            continue
         fi

         # print information
```

```
         echo -n "Activating LUN ${LUN}... "

         # activate this lun
         echo ${LUN} > unit_add

         if [ "${?}" -eq "0" ]; then
            # print success message (otherwise the error is printed above)
            echo "OK"
         fi
      done
   done
done

# print an empty line
echo
```

A.4 Multipath configuration file

File name:

```
/etc/multipath.conf
```

File contents:

```
devnode_blacklist {
        devnode "^dasd*"
}

devices {
        device {
                vendor "IBM"
                product "2107900"
                path_grouping_policy "group_by_prio"
                prio_callout "/root/zfcp_prio_config.sh /dev/%n"
                failback "immediate"
                path_checker "tur"
                features "1 queue_if_no_path"
        }
}
```

A.5 Priority callout

File name:

```
/root/zfcp_prio_config.sh
```

File contents:

```
#!/bin/sh
#
# ---------------------------------------------------------------------
# |   CONFIGURES THE PRIORITY OF A SPECIFIC ZFCP PATH                  |
# |                                                                    |
# |   Authors: Holger Smolinski (smolinski@de.ibm.com),                |
# |            Marc Beyerle (marc.beyerle@de.ibm.com), 11/27/2008      |
# |   Copyright IBM Corp. 2008                                         |
# ---------------------------------------------------------------------
#

# the default priority
DEFAULT_PRIORITY=0

# the configuration file for the priorities
ZFCP_CONFIGURATION_FILE="/root/zfcp_priorities.cfg"

# check for an empty device node name
if [ -z "${1}" ]; then
   # write log information
   logger "zfcp_prio_config :: empty device node name specified"

   # output default priority
   echo "${DEFAULT_PRIORITY}"

   # exit successfully
   exit 0
fi

# check for the existence of the configuration file
if [ ! -f "${ZFCP_CONFIGURATION_FILE}" ]; then
   # write log information
   logger "zfcp_prio_config :: configuration file \"${ZFCP_CONFIGURATION_FILE}\"
does not exist"

   # output default priority
   echo "${DEFAULT_PRIORITY}"

   # exit successfully
   exit 0
fi

# write log information
logger "zfcp_prio_config :: called with device node name ${1}"

# determine the scsi device name
SCSIDEVICE=$(lsscsi | grep "${1}\$" | sed "s/^\[\([0-9:]*\)\].*/\1/g")

# check for an invalid device node name
```

```
if [ -z "${SCSIDEVICE}" ]; then
   # write log information
   logger "zfcp_prio_config :: no scsi device found for device node name ${1}"

   # output default priority
   echo "${DEFAULT_PRIORITY}"

   # exit successfully
   exit 0
fi

# write log information
logger "zfcp_prio_config :: determining zfcp path for scsi device ${SCSIDEVICE}"

# determine the zfcp path for this scsi device
ZFCPPATH=$(lszfcp -D | grep "${SCSIDEVICE}\$" | awk '{ print $1; }')

# check for a non-existing zfcp path
if [ -z "${ZFCPPATH}" ]; then
   # write log information
   logger "zfcp_prio_config :: no zfcp path found for scsi device ${SCSIDEVICE}"

   # output default priority
   echo "${DEFAULT_PRIORITY}"

   # exit successfully
   exit 0
fi

# write log information
logger "zfcp_prio_config :: determining priority for zfcp path ${ZFCPPATH}"

# determine this path's priority from the configuration file
PRIORITY=$(grep "${ZFCPPATH}" "${ZFCP_CONFIGURATION_FILE}" | awk '{ print $2; }
')

# check for a non-existing zfcp path in the configuration file
if [ -z "${PRIORITY}" ]; then
   # write log information
   logger "zfcp_prio_config :: no priority value found for zfcp path
${ZFCPPATH}"

   # output default priority
   echo "${DEFAULT_PRIORITY}"

   # exit successfully
   exit 0
fi

# write log information
logger "zfcp_prio_config :: assigning priority = ${PRIORITY} to zfcp path
${ZFCPPATH}"

# output the priority
echo ${PRIORITY}
```

A.6 Boot script

File name:

`/etc/init.d/boot.aaa_lszfcp`

File contents:

```
#!/bin/sh
#
# ---------------------------------------------------------------------
# |   WRITE THE CURRENT ZFCP SETUP TO ROOT'S HOME DIRECTORY           |
# |                                                                   |
# |   Author: Marc Beyerle (marc.beyerle@de.ibm.com), 12/11/2008      |
# |   Copyright IBM Corp. 2008                                        |
# ---------------------------------------------------------------------
#

### BEGIN INIT INFO
# Provides:          boot.aaa_lszfcp
# Required-Start:    boot.loadmodules
# Required-Stop:
# Default-Start:     B
# Default-Stop:
# Description:       Writes the current zfcp configuration to root's home
directory.
### END INIT INFO

lszfcp -D > /root/lszfcp.current
```

IBM Deutschland Research & Development GmbH
Technical Marketing Competence Center Europe
Department 3300
Schoenaicher Str. 220
71032 Boeblingen
Germany