Linux on System z
Two stage dumper/kdump framework

Edition: July 8, 2011 for upstream discussion

Author: Michael Holzheu (holzheu at linux.vnet.ibm com)

1 Two stage dumper/kdump framework

1.1 Abstract

With this line item the Linux kdump framework is ported to Linux on System z and is integrated in the already available System z stand-alone dump tools and shutdown actions framework. This leads to the following enhancements for System z kernel dumps:

- Dump time and size can be reduced using page filtering.
- Dump disk space sharing is possible for server farms using network dump.
- Dump setup is made easier using existing kdump setup GUIs of Linux distributions.

The integration with the Linux on System z stand-alone dump tools ensures that the dump reliability with kdump will be as high as with the current solution.

1.2 Description

When an unrecoverable kernel problem occurs on a Linux system, a kernel dump must be created for problem determination. Currently on System z this is done with the architecture-specific stand-alone DASD, tape, or SCSI dump tools. The following problems have been identified with these tools:

- System memory is written to disk without filtering memory pages. Todays disk dump speed is about 100 MiB/s. Thus dumping a system with 100 GiB takes 1000 seconds or about 17 minutes. For some setups this is an unacceptable outage time.
- The resulting dump size can be a problem when transferring the dump to a service organization.
- Automatic dumping requires each system to have a separate dump disk, which results in large amounts of reserved dump disk storage.
- GUI support for the System z stand-alone dump tools setup is almost non-existent. The main reason is that the tools are System z specific and completely new dialogs had to be implemented by the distributors, which has not been done in the past due to resource constraints.

To address these problems, the kdump framework will be implemented for Linux on System z. The following improvements will be made:

- Dump time and size can be reduced using page filtering.
- Dump disk space sharing is possible for server farms using network dump.
- Dump setup is made easier using existing kdump setup GUIs of Linux distributions.

The main disadvantage of kdump and one reason why on System z this dump method was until now not available is that it is not as reliable as the Linux on System z stand-alone dump tools and it cannot be used for early kernel problems. These disadvantages are addressed by combining both methods. It is possible to configure the dump environment in a way that the stand-alone dump tools act as backup dump method in the case of a kdump failure. Therefore compared to other architectures with this line item Linux on System z has the following advantage:

- Improve kdump reliability by combining kdump with the System z stand-alone dump tools.

1.2.1 The kdump concept

The basic idea behind kdump is that in a production Linux system some memory is reserved using the `crashkernel` kernel parameter. In the following this area is called **crashkernel memory**. A backup (kdump) kernel (plus optional initrd) is pre-loaded into the crashkernel memory with the **kexec** tool (see man page [KEX]). The pre-loaded kernel is executed in case of as system crash and uses the crashkernel memory to run. Today, a crashkernel memory size of 64 - 128 MiB is sufficient execute the kdump kernel and userspace.

When the kdump kernel runs, two virtual files `/proc/vmcore` and `/dev/oldmem` are available that represent the dump of the crashed production Linux system. The file `/proc/vmcore` has ELF core format and can be filtered with the **makedumpfile** tool to reduce the dump size. In `/proc/vmcore` the crashkernel memory area is excluded. The `/dev/oldmem` file represents the memory of the crashed system as unstructured linear memory (like `/dev/mem` for the current system). For saving the dump, the files can be copied to an attached disk or sent over network. After the dump is saved, the original system can be restarted by issuing **reboot** in the kdump kernel.

On other architectures like x86 the kdump kernel is relocatable and runs directly in the crashkernel memory. For Linux on System z, the crashkernel memory is swapped with the memory from `[0 - crashkernel size]` before the kdump kernel is then started.

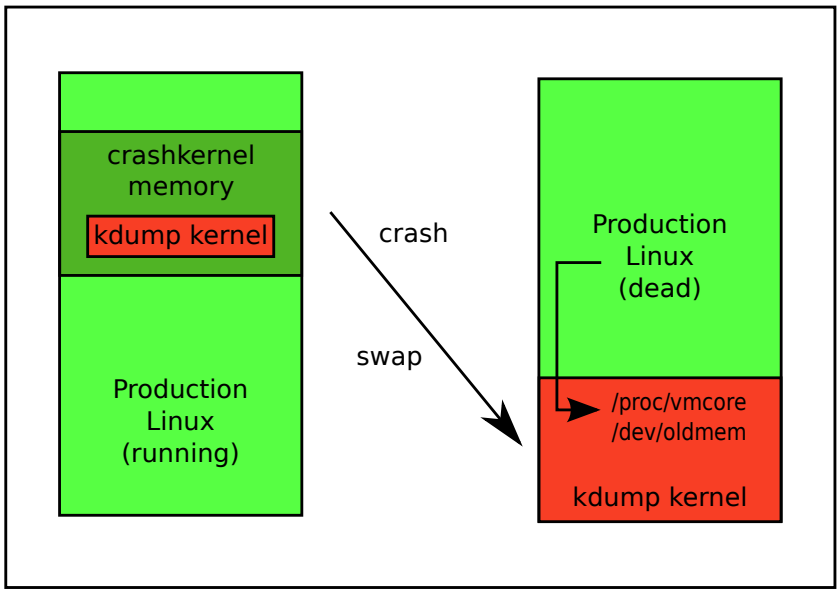

Figure 1: Linux on System z crashkernel memory

Because of the memory swap implementation, for System z memory hotplug, kdump will prevent memory chunks from being set offline for the reserved crashkernel memory **and** the area `[0 - crashkernel size]`.

For Linux on System z running under z/VM the crashkernel memory is only reserved. Until the kdump kernel is started it does not require "real" memory. Linux informs the z/VM hypervisor with diagnose 10 that this memory does not need real backing. Only the pre-loaded kdump kernel and ramdisk consume some megabytes of memory. Over the time z/VM will page out even this memory because it does not change until the kdump kernel is started.

Under LPAR the crashkernel memory is backed with real memory.

For more information on the overall kdump design see the OLS (Ottawa Linux Symposium) papers [GBN05] and [GHO+07].

1.2.2 Debug data: vmcoreinfo / makedumpfile

The **makedumpfile** tool [TO] can be used to reduce the dump size by excluding Linux memory pages. Which page types (e.g. user or cache pages) should excluded can be configured with the `-d` command line option. For doing the filtering, the **makedumpfile** needs datatype and symbol information of the crashed Linux kernel. This data normally is included in a debug file `vmlinux.debug`. Recent Linux kernels contain this information also in memory and for kdump this information is included in the `VMCOREINFO` ELF note section of `/proc/vmcore`. If the **makedumpfile** tool finds this ELF note, no `vmlinux.debug` file is required for page filtering. See also sections 1.5.5 and 1.5.8).

1.2.3 Shutdown action integration

Linux on System z already allows to configure shutdown triggers and actions using a sysfs interface under `/sys/firmware`. Shutdown triggers are events that lead to a shutdown of the Linux system.

The following shutdown triggers are currently available:

- **halt** (shutdown -H)
- **poff** (shutdown -P / halt -p -f)
- **reboot** (shutdown -r / reboot)
- **panic** (kernel panic)

One of the following shutdown actions can be configured for each trigger:

- **stop**: Stop Linux (sigp stop or disabled wait for panic trigger)
- **ipl**: Re-IPL Linux using settings in `/sys/firmware/ipl`
- **reipl**: Re-IPL Linux using settings in `/sys/firmware/reipl`
- **dump**: Dump Linux using settings in `/sys/firmware/dump`
- **dump_reipl**: Dump Linux using settings in `/sys/firmware/dump` and afterwards re-IPL Linux using settings in `/sys/firmware/reipl`
- **vmcmd**: Issue CP command under `/sys/firmware/vmcmd/on_<trigger>` and then stop Linux

With this line item a new trigger `restart` for PSW restart and a new action `kdump` for starting kdump is added to the shutdown action framework.

A human user can trigger `restart` by pressing the "PSW restart" button on the HMC for LPARs or by entering the CP command `system restart` under z/VM.

The default setting for the `restart` trigger is the `reipl` shutdown action. When the `crashkernel` parameter is successfully set (see section 1.5.1), the shutdown action for `panic` and `restart` is automatically changed by the kernel to `kdump`.

See chapters "Shutdown actions", "lsshut", and "chshut" in [IBM11a] and chapter "The dumpconf tool" in [IBM11b] for more information on the current shutdown action implementation.

1.2.4 Stand-alone dump tools integration

The pre-loaded kdump kernel is protected by removing the crashkernel memory from the kernel page tables. But there still is a small likelihood that the pre-loaded kdump kernel is overwritten by DMA caused by device driver bugs. Another drawback is that kdump is not functional until the kdump kernel has been loaded with **kexec**. This line item enables the stand-alone dump tools to detect these two situations. If an overwritten kdump kernel is detected by use of checksums or if no kdump kernel is loaded, a stand-alone

dump is created as backup dump method. If a healthy kdump is detected, the stand-alone dump tool starts the pre-loaded kdump kernel.

For the System z IPL command a loadparm with 8 characters can be specified. With this item the creation of a stand-alone dump can be forced, when specifying the loadparm `sadump`, when the dump tool is IPLed. Then independent from a potentially loaded kdump kernel always a stand-alone dump is created. This loadparm can be used, if there are unexpected problems with kdump.

If a stand-alone dump tool is configured for automatic dump on panic, a kernel panic or PSW restart will trigger kdump indirectly with an IPL of the stand-alone dump tools. In this case, according to `/etc/sysconfig/dumpconf`, the **dumpconf** service script sets the action for the `panic` and the `restart` triggers to `dump` or `dump_reipl`.

Using the the stand-alone dump tools to trigger kdump is a tradeoff between reliability (kdump itself is not 100% reliable) and resource expenses (you have to reserve disk dump space for the stand-alone dump tools).

1.2.5 Comparison kdump on System z with other architectures

The System z implementation of kdump has some advantages compared to other architectures:

- On System z diagnose 308 or IPL perform CPU and I/O subsystem reset. So kdump on s390 is safe against ongoing I/O.
- On z/VM diagnose 10 is used to release the reserved memory. Real/backed memory is required only for the kdump image and ramdisk (currently about 10 MiB). After some time z/VM will page out this memory. Then no real memory will be wasted.
- When kdump segments (kernel or ramdisk) are corrupt, the stand-alone dump tools can be used automatically as backup dump method. The kdump consistency verified by the use of checksums.
- Using the System z stand-alone dump tools, it is still possible to dump early crashes, when kdump is not yet initialized.
- On System z you can start kdump via a stand-alone dump if the system is in a state where it neither panics nor reacts to other inputs (NMIs etc). For example, when the NMI function has been overwritten.
- Because on System z the ELF header for `/proc/vmcore` is built dynamically when the kdump kernel is initialized, memory layout changes caused by memory hotplug are always handled correctly. On other architectures **kexec** has to be called each time a memory layout change is done in order to provide a correct ELF core header.
- On System z crashkernel memory is removed from the kernel page tables. Therefore the likelihood of memory corruption is reduced. Only wrong device DMA can overwrite the loaded kdump kernel. Another advantage is that no page table memory is required.

1.2.6 Usage scenarios

The standard usage scenario for kdump as described in section 1.2.1 is that a large production system preloads the kdump kernel into a rather small crashkernel memory. When kdump is started, it is automatically restricted to the memory area `[0 - crashkernel size]`. With the kdump kernel the dump is written and afterwards the production system is rebooted.

There is another usage scenario where the kdump system itself already starts as production system and offers the service of the system while in the background the dump is written. After the dump has been written, a new kdump kernel can be loaded into crashkernel memory. We call this scenario HA (high availability) scenario because the system service will be available faster than with the standard kdump scenario. Both production system and kdump must have the same memory layout and must be started with `crashkernel=<memory size/2>@<memory size/2>` (see figure 3).

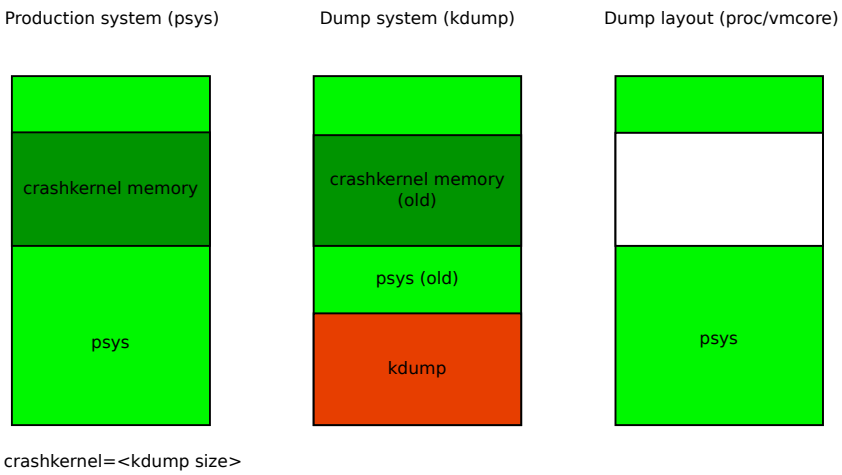

Figure 2: Standard kdump usage

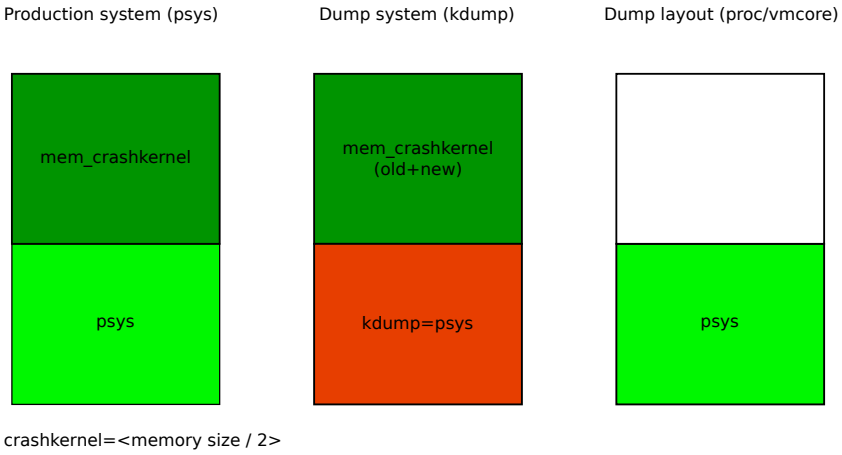

Figure 3: 1:1 HA kdump usage

1.2.7 Overview of kdump patches

With this line item patches for the following packages are provided:

- **kernel**: System z kdump backend support (s390 and common code changes)
  - Add `crashkernel` parameter to reserve crashkernel memory
  - Add `/proc/vmcore` and `/dev/oldmem` backend support
  - Add shutdown action `kdump`
  - Add shutdown trigger `restart`
  - Ensure that kdump re-IPL will reboot the production system
  - Export vmcoreinfo data
- **kexec tools**: Implement the `--load-panic` option for loading kdump kernel. With this option the kdump backup kernel can be pre-loaded into the crashkernel memory using the `kexec_load()` system call.
- **s390-tools**: kdump integration
  - **zipl dump tools**: Trigger kdump using the zipl stand-alone dump tools. Provide `sadump` loadparm support to force stand-alone dump.

  - **zfcpdump**: Trigger kdump using the zfcpdump SCSI stand-alone dump tool. Provide `sadump` loadparm support to force stand-alone dump.
  - **zgetdump**: Add vmcoreinfo ELF note to ELF output dumps. This can be used by tools like **makedumpfile** to reduce the dump size.
  - **lsshut/chshut**: Add support for new shutdown trigger `restart` and new shutdown action `kdump`.
  - **dumpconf**: Add support for `restart` trigger.
- **makedumpfile**: Make tool output usable for s390 line-mode terminals. Currently the tool uses control characters to position the cursor (e.g. for progress messages) which is not possible on line-mode terminals.
- **crash**: Implement restart stack support. The new PSW restart trigger for kdump introduces a new kernel stack that has to be supported by the crash tool.

1.3 Configuring, building and installing

A kernel with kdump support has to be built and installed. The following kernel config options have to be used:

- CONFIG_KEXEC=y
- CONFIG_CRASH_DUMP=y
- CONFIG_PROC_VMCORE=y

Updated packages as listed in section 1.2.7 have to be installed.

1.4 Setting up the environment

To use kdump, configure the memory for the guest virtual machine or LPAR large enough for the the `crashkernel` parameter memory reservation. Therefore if a Linux system without kdump is configured with `n` MiB memory, with kdump `n + crashkernel memory` MiB must be configured for that system.

Note, that under z/VM the crashkernel memory is only reserved and first requires almost no "real" memory (see section 1.2.1).

1.5 Using this feature

1.5.1 Setup of kdump - The crashkernel parameter

The production system has to be started with the `crashkernel` parameter to reserve the required memory for kdump. The amount of the crashkernel memory must be chosen large enough that the kdump kernel and ramdisk can be executed within this memory.

The standard syntax of the parameter is as follows:

```
crashkernel=size[KMG][@offset[KMG]]
```

This parameter reserves the physical crashkernel memory region `[offset, offset + size]` for the kdump kernel image and ramdisk. If `@offset` is omitted, then a suitable offset is selected automatically.

Examples:

- **crashkernel=128M**: Reserve 128 MiB at a suitable memory offset.

- **crashkernel=128M@256M**: Reserve 128 MiB at memory offset 256 MiB.

The extended crashkernel syntax is as follows:

```
crashkernel=range1:size1[,range2:size2,...][@offset]
```

Memory reservation depends on the memory size of the running system. The syntax of `range` is `start-[end]` where `start` and `end` specify the memory size of a system (amount[KMG]). The memory size of a system is the last valid physical address plus one. If `end` is omitted, infinite memory is assumed. If no range is found for a system, no reservation will be made.

The motivation for the extended crashkernel syntax comes from distributors that configure their crashkernel command line automatically with some configuration tool. The tool knows the value of System RAM, but if the user removes RAM, then the system becomes unbootable or at least unusable and error handling is very difficult.

Examples:

- **crashkernel=128M-255M:64M,256M-1G:128M**: Reserve 64 MiB for systems that have 128 to 255 MiB memory and 128 MiB for systems that have 256 MiB to one GiB memory and automatically search a suitable memory offset.
- **crashkernel=128M-255M:64M,256M-:128M**: Reserve 64 MiB for systems that have 128 to 255 MiB memory and 128 MiB for systems that have 256 MiB or more memory and automatically search a suitable memory offset.
- **crashkernel=128M-255M:64M,256M-1G:128M@512M**: Reserve 64 MiB for systems that have 128 to 255 MiB memory and search a suitable memory offset for it. Reserve 128 MiB at memory offset 512 MiB for systems that have 256 MiB to one GiB memory.

There are some restrictions for choosing size and offset for the crashkernel parameter.

- The defined crashkernel memory must be available
- The `offset` must be larger than `size`.
- The memory area `[0, size]` must be available (no memory holes).

If the reservation was successful, in `/proc/iomem` the memory range is shown in the line "**Crash kernel**":

```
# cat /proc/iomem
00000000-3fffffff : System RAM
00000000-00562823 : Kernel code
00562824-0084f91f : Kernel data
00a52000-0138bd8f : Kernel bss
10000000-1fffffff : Crash kernel

# cat /proc/cmdline
dasd=eb90 root=/dev/dasda1 crashkernel=256M@256M
```

When the `crashkernel` parameter is successfully set, the shutdown action for `panic` and `restart` is automatically set by the kernel to `kdump`.

The because of the swap implementation on System z, the memory area from `[0 - crashkernel memory]` cannot be set offline via memory hotplug after the crashkernel memory reservation.

Under z/VM it is guaranteed that the crashkernel memory is not backed with real memory (see section **??**).

Until a kdump kernel is loaded with **kexec**, the size of the crashkernel memory can be reduced at runtime using the `/sys/kernel/kexec_crash_size` file. For example to reduce the size to 64 MiB (67108864 bytes) the following command can be used:

```
# echo 67108864 > /sys/kernel/kexec_crash_size
```

1.5.2 Setup kdump - Loading kdump kernel and initrd

When the crashkernel memory reservation was successful, the kdump kernel and ramdisk (optional) have to be pre-loaded with the **kexec** tool. This is done with the `--load-panic`/`-p` option. With the `--command-line` option the correct kernel parameter line for the kdump kernel has to be specified. A kdump initrd can be specified with the `--initrd` option.

Examples:

```
# kexec --load-panic /boot/image \
    --command-line="dasd=eb90 root=/dev/dasda1 maxcpus=1"
# kexec -p /boot/kdump.image --initrd /boot/kdump.initrd \
    --command-line="dasd=eb90 root=/dev/dasda1 maxcpus=1"
```

With the `--unload` / `-u` option a pre-loaded kdump kernel can be unloaded.

```
# kexec -p -u
```

The file `/sys/kernel/kexec_crash_loaded` shows if currently a kdump kernel is loaded (`1`) or not (`0`).

```
# cat /sys/kernel/kexec_crash_loaded
1
```

1.5.3 Setup kdump - Configure stand-alone dump tools with dumpconf

When the kdump kernel is loaded, the system is prepared for a crash and in case of a kernel panic or PSW restart the kdump kernel will be started directly by the crashing production system.

As described in section 1.2.4, there are situations where kdump may fail and the traditional System z stand-alone dump tools can be used to increase the dump reliability. If this is **not** desired, no further setup is required.

Otherwise a stand-alone dump tool has to be configured using `/etc/sysconfig/dumpconf` and the **dumpconf** service script must be started to activate the new setting. This will set the shutdown action for `panic` and `restart` to `dump` or `dump_reipl`. In this case, a kernel panic or PSW restart automatically will IPL the configured stand-alone dump tool. The dump tool then triggers kdump if a healthy setup is available. Otherwise, as backup method, it writes a stand-alone dump and if `dump_reipl` has been specified, the original production system is booted again.

The following shows a `/etc/sysconfig/dumpconf` sample configuration for a DASD stand-alone dump tool:

```
ON_PANIC=dump_reipl
DUMP_TYPE=ccw
DEVICE=0.0.4e13
```

The configuration is enabled with:

```
# service dumpconf restart
```

See chapter "The dumpconf tool" in [IBM11b] for more information.

1.5.4 Triggering kdump

There are four standard ways to trigger kdump on Linux on System z:

- Kernel panic (`panic` shutdown trigger)

- Magic sysrq 'c'rash (`panic` shutdown trigger)
- PSW restart (`restart` shutdown trigger)
- IPL of stand-alone dump tool

A kernel panic is triggered automatically, when the Linux kernel itself detects an unrecoverable error. For the other triggers external intervention is required. External intervention can be triggered either by a human user or by the z/VM watchdog.

As described in section 1.5.3, if a stand-alone dump tool has been configured with **dumpconf**, the shutdown actions for `panic` and `restart` are set to `dump` or `dump_reipl` and kdump is triggered indirectly by IPLing the configured stand-alone dump tool. Otherwise kdump is triggered directly using the shutdown action `kdump`.

When a kernel panic occurs, the shutdown action configured for the shutdown trigger `panic` is started automatically.

For triggering the `crash` magic sysrq you have to enter `^-c` on the 3270 or HMC operator console. This creates a forced Linux kernel panic and the shutdown action defined for the `panic` trigger is executed.

PSW restart is triggered under z/VM by entering `#cp system restart` on the 3270 console or under LPAR by pressing the `PSW Restart` button on the HMC (LPAR->Recovery->PSW Restart).

To IPL a stand-alone dump tool you have to specify the DASD or tape dump device number or for zfcpdump, the WWPN, LUN and adapter device number. See [IBM11b] for more information on how to start a System z stand-alone dump.

Under z/VM you can also use the watchdog driver for triggering kdump via `system restart`. You can configure the z/VM watchdog as follows:

```
# echo -en "system restart" > /sys/module/vmwatchdog/parameters/cmd
```

For more information on using the watchdog, see chapter "Setting up the watchdog device driver" in [IBM11a].

1.5.5 Accessing the dump

After kdump has been triggered, the pre-loaded kdump kernel is started in the swapped crashkernel memory. Two virtual files `/proc/vmcore` and `/dev/oldmem` are available that represent the dump of the crashed system.

The `/dev/oldmem` file represents the memory of the crashed system as unstructured linear memory.

The file `/proc/vmcore` has ELF core format and contains memory, CPU register and vmcoreinfo information.

The content of the ELF file can be displayed with the **readelf** or **zgetdump** command:

```
# readelf -n /proc/vmcore

Notes at offset 0x000000b0 with length 0x00000e38:
  Owner         Data size       Description
  CORE          0x00000088      NT_PRPSINFO (prpsinfo structure)
  CORE          0x00000150      NT_PRSTATUS (prstatus structure)
  CORE          0x00000088      NT_FPREGSET (floating point registers)
  LINUX         0x00000008      NT_S390_TIMER (s390 timer register)
  LINUX         0x00000008      NT_S390_TODCMP (s390 TOD comparator)
  LINUX         0x00000004      NT_S390_TODPREG (s390 TOD programmable)
  LINUX         0x00000080      NT_S390_CTRS (s390 control registers)
  LINUX         0x00000004      NT_S390_PREFIX (s390 prefix register)
  LINUX         0x00000080      NT_S390_CTRS (s390 control registers)
```

```
  LINUX         0x00000004      NT_S390_PREFIX (s390 prefix register)
  VMCOREINFO    0x0000048d      Unknown note type: (0x00000000)

# zgetdump -i /proc/vmcore
General dump info:
  Dump format........: elf
  Version............: 1
  System arch........: s390x (64 bit)
  CPU count (online).: 1
  Dump memory range..: 1024 MB

Memory map:
  0000000000000000 - 000000001fffffff (512 MB)
  0000000030000000 - 000000003fffffff (256 MB)
```

The dump analysis tool **crash** can directly process the `/proc/vmcore` file in the running kdump kernel:

```
# crash /boot/vmlinux /boot/vmlinux.debug /proc/vmcore
```

To save the dump the `/proc/vmcore` file can be copied to a local disk or copied over network using tools like scp or ftp.

```
# cp /proc/vmcore /dumps/dump.elf
# scp /proc/vmcore user@host:/dumps/dump.elf
```

For reducing the dump size, the **makedumpfile** tool can be used. Because `/proc/vmcore` contains the `VMCOREINFO` ELF note (see 1.2.2), it is not required to specify a `vmlinux.debug` file that contains the debug data for filtering the dump. The following command can be used to copy a compressed and filtered dump to a local disk:

```
# makedumpfile -c -d 31 /proc/vmcore /dumps/dump.kdump
```

With the following command the (flattened) compressed dump can be sent via network:

```
# makedumpfile -F -c -d 31 /proc/vmcore | \
               ssh user@host "cat > /dumps/dump.kdump"
```

For more information on how to use **makedumpfile**, read the corresponding man page [TO].

1.5.6 Reboot production system

The System z kdump implementation ensures that a reboot of the kdump kernel will reboot the device that was specified unter `/sys/firmware/reipl` on the previously crashed production system.

1.5.7 kdump initrd

A kdump initrd is optional. On Linux distributions normally the kdump initrd will automatically do the steps described in 1.5.5. Scripts in the initrd will copy the dump to a configured location and then reboot the system.

Instead of using an initrd theoretically also a monolithic kernel with the root file system on disk could be used.

1.5.8 zgetdump and vmcoreinfo

With this item vmcoreinfo support for the **zgetdump** ELF target format is added. For dumps of kernels that have the new kdump support, **zgetdump** will extract the vmcoreinfo data from the dump memory and adds the `VMCOREINFO` ELF note section to the target ELF dump (see also 1.2.2). Therefore it is no longer necessary to use the debug `vmlinux.debug` file with **makedumpfile** on stand-alone dumps.

Example:

Mount stand-alone DASD dump on /dev/dasdb1 to ELF core file and do page filtering with **makedumpfile** without using a `vmlinux.debug` file:

```
# zgetdump -m /dev/dasdb1 -f elf /mnt
# readelf -n /mnt/dump.elf | grep VMCOREINFO
#    VMCOREINFO      0x0000048d Unknown note type: (0x00000000)
# makedumpfile -c -d 31 /mnt/dump.elf dump.kdump
```

1.5.9 lsshut/chshut

With this line item support for the new shutdown trigger `restart` and the new shutdown action `kdump` is added to the tools **lsshut** and **chshut**.

The **lsshut** tool displays the current setting for the `restart` trigger. In case of the `kdump` action it also shows the size of the crashkernel memory and it shows, if the kdump kernel is currently loaded.

```
# lsshut
Trigger          Action
=======================
Halt             stop
Power off        stop
Reboot           reipl
Restart          kdump (loaded=1, size=256MiB)
Panic            kdump (loaded=1, size=256MiB)
```

With the **chshut** tool it is possible to change the shutdown action for the `restart` trigger.

Examples:

```
# lsshut | grep Restart
Restart          reipl

# chshut restart stop

# lsshut | grep Restart
Restart          stop

# chshut restart vmcmd "msg OPERATOR bye bye" vmcmd "logoff"
Restart          vmcmd ("msg OPERATOR bye bye","logoff")
```

1.5.10 Forcing stand-alone dump with loadparm "sadump"

When a stand-alone dump tool is IPLed and the loadparm `sadump` is specified, the dump tool will create a stand-alone dump independently from a potentially pre-loaded kdump kernel. The loadparm can be specified under z/VM with the IPL CP command and under LPAR on the HMC load panel.

Example: (z/VM)

```
# cp IPL 4711 loadparm sadump
```

1.6 Using online documentation

There are man pages available for the following tools:

- lsshut (updated)
- chshut (updated)
- dumpconf (updated)
- kexec
- makedumpfile

1.7 Intra-kernel interfaces

With this line item a new kernel to kernel/external program communication mechanism is established using memory. A meminfo block containing the following elements is accessible using an ABI defined (absolute zero) lowcore pointer to that structure:

- crashkernel memory base and size
- kdump segments (kernel and ramdisk)
- vmcoreinfo
- re-IPL block (ipib)

The kdump segments and re-IPL block contain checksums.

The production Linux system sets up the meminfo and any external program (e.g. stand-alone dump tools or the kdump kernel) can access this information. E.g. the stand-alone dump tools use the checksums to verify that the pre-loaded kdump is valid and use the crashkernel memory base to find the kdump kernel to execute, if the checksums are valid.

References

[GBN05] Vivek Goyal, Eric W. Biederman, and Hariprasad Nellitheertha. *Kdump, A Kexec-based Kernel Crash Dumping Mechanism*. OLS, 2005. http://lse.sourceforge.net/kdump/documentation/ols2oo5-kdump-paper.pdf.

[GHO+07] Vivek Goyal, Neil Horman, Ken'ichi Ohmichi, Maneesh Soni, and Ankita Garg. *Kdump: Smarter, Easier, Trustier*. OLS, 2007. http://www.kernel.org/doc/ols/2007/ols2007v1-pages-167-178.pdf.

[IBM11a] IBM. *SC33-8411-09: Device Drivers, Features, and Commands*. IBM Corp., 2011. http://www.ibm.com/developerworks/linux/linux390/documentation_dev.html.

[IBM11b] IBM. *SC33-8412-07: Using the Dump Tools*. IBM Corp., 2011. http://www.ibm.com/developerworks/linux/linux390/documentation_dev.html.

[KEX] *kexec man page*. http://www.linuxcertif.com/man/8/kexec/.

[TO] Masaki Tachibana and Ken'ichi Ohmichi. *makedumpfile man page*. http://www.linuxcertif.com/man/8/makedumpfile/.

end of document