

Re: oops in :snd_pcm_oss:resample_expand+0x19c/0x1f0

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-09/msg07122.html>

- *From:* Vivek Goyal <vgoyal@xxxxxxxxxx>
 - *Date:* Mon, 25 Sep 2006 11:30:48 -0400
-

On Mon, Sep 25, 2006 at 12:03:59AM +0200, Jean-Marc Saffroy wrote:

On Sun, 24 Sep 2006, Andrew Morton wrote:

I keep a crash dump from kdump,

Whoa. Impressed. Which distro are you using and how did you go about this and how much fuss was it to set up?

Well, it sure didn't work right out of the box... But it's not that hard, really: the main problems are lack of proper end-user docs and packaging, as usual.

Hi Jean,

Thanks for trying out kdump. Things are still in state of flux when it comes to packaging and I am hoping that its a matter of few more months and things will be in a much better shape.

A lot of kdump functionality is in user space when it comes to configuring kdump, saving the core dump, filtering the image and then finally doing the analysis. Complete user space automation is still being implemented by distros and issues are being resolved.

One of the issues is how to bring all the distros on the same platform. SuSE and RedHat are doing quite some amount of automation for using kdump. For example, the small script you wrote for saving the dump, is already part of Fedora and SuSE and will be available in RHEL. I have no idea what's going on in debian when it comes to packaging.

A serial console on another PC very much helped in troubleshooting problems (eg. my kdump kernel's initrd was initially too large), but it's not required (though it's always a nice thing to have

anyway).

The basic instructions for setting up kdump are here:

<http://lse.sourceforge.net/kdump/kdump-test-reports/test-plan.txt>

More docs:

- Documentation/kdump/kdump.txt
- Distros (SuSE, RedHat) are now also packaging a README file in /usr/share/doc.... They are also putting together some man page info for kexec sytem call.

I followed them, and also had to:

- reduce the uncompressed size of my initramfs (was 50+ MB initially, now ca. 22MB), by listing only the required modules (lsmod w/ some hand picking), see /etc/initramfs-tools/initramfs.conf

Hmm.. This looks like another packaging issue. Now we are automating the process, so that a custom initrd is generated for kdump kernel and dump will be saved from the initrd itself. This custom initrd will be only big enough contain the functionality to perform the job of bringing the dump device up and saving the dump. User shall have to just say "service kdump start" and initrd will be generated and kdump kernel will be loaded into the memory.

Developers from NEC are also working on providing a capability to filter the dump so that final core size is small and managable. In fact this filtering also can be done from initrd itself.

- reserve more RAM for the kdump kernel (now I use "crashkernel=96M@16M")

Looks like big initramfs and running the init scripts in second kernel might be taking some toll. We have been able to take dumps successfully by reserving 64MB on x86_64 boxes. Now with dump capturing capability from initrd, it should become even better as we don't have to run user space init scripts reducing memory pressure.

The distro is the AMD64 Debian etch, with two vanilla 2.6.18 kernels: one "regular" SMP kernel with CONFIG_KEXEC=y, which is what I use, and one UP kernel with CONFIG_PROC_VMCORE=y and a different load address, which is activated on a crash; other than that they are the same.

Now we are also looking at a completely relocatable kernel so that the need to build a separate kernel for dumping purposes is gone. Few weeks back Eric posted the patches on LKML for RFC and testing purposes. We have been testing out those patches and fixing few bugs. In a couple of weeks we should be able to re-post the patches for another round of review. So things should become even better once relocatable kernel is in.

The crude script below is run at boot time and configures kdump properly for me. All oopses invoke panic ("kernel.panic_on_oops = 1" in /etc/sysctl.conf), and thus the kdump kernel is activated. This kernel boots on the very same root fs, runs the script below again, this time to save the dump and reboot to the regular kernel. I did not try to have the kdump kernel reset the vga console, so it stays in graphic mode and shows garbage until the final reboot.

Now latest kexec-tools has one patch to re-initialize frame buffer over kdump boot but I am not sure what can we do if we are in X at the time of crash? One suggestion was that at-least just blink the keyboard lights while dump is being saved.

So at this point of time, I am not sure how can we reset the display if user was in graphic mode at the time of crash.

Now from an oops to being back to work, I would say it takes about 2-3 minutes (the longer part is the cold reboot, then the 30" timeout on the Windows loader ;-)). I would certainly not do kernel development on my desktop (backups? what backups?), but oopses do occur from time to time (oh the joys of SMP on a desktop), and it's great to have a dump the first time I see one.

Oh and I wish I could use gdb on a kdump core. :-)

Currently we can use gdb but only for linearly mapped region. You are right its just a matter of re-generating the elf headers and remap the vmalloc areas to enable module debugging in gdb. I can not do it after the crash so probably the best place would be do it in user space. A program can read /proc/vmcore and regenerate the headers for enabling module debugging with gdb.

Would be nice, but this is much better than what we usually get, no?

Re: oops in :snd_pcm_oss:resample_expand+0x19c/0x1f0

I'm used to having LKCD for debugging at work (on IA64), it has raised my expectations. :-)

But using gdb would be so much nicer, and I'm sure we're close to being able to use it, it's probably only a matter of messing a bit with the ELF image provided by kdump (it's really a good old core dump): ie. remapping vmalloc'ed memory pages, so modules can be accessed, and ideally adding processor context for each task (currently there is one per CPU). Crash does all this stuff internally and has other nice commands, but I don't find it quite as comfortable as gdb (with a few good scripts gdb would certainly be on par with crash).

Hmm.. Crash vs gdb is an interesting issue. I have not used gdb very extensively on core dumps, but with my limited experience, I found "crash" to be more friendly. Crash has got so many in-built commands tailored for kernel debugging and gdb lacks all those. Yes, we can write gdb scripts to implement those, but last time Alexander Nyberg wrote few gdb scripts to dump all the threads and it was so slow.

Look at Documentation/kdump/gdbmacros.txt

So what's issue with crash? Is it just a matter of being more familiar with gdb or gdb has got advantages over "crash" when it comes to kernel debugging?

Thanks
Vivek

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>