

Re: How to clear device errors w/o reboot?

Source: <http://unix.derkeiler.com/Newsgroups/comp.os.vms/2005-02/2118.html>

norm.rafael_at_metso.com

Date: 02/23/05

Date: Wed, 23 Feb 2005 11:41:57 -0500

ZDEC has been superseded by CLEAR_ERRORS (which AFAIK is less likely to crash systems).

I don't remember where I got it, but the heads-up came from this venue.

\$ type aaareadme.txt

CLEAR_ERRORS clears the device error count on an Alpha VMS system.

Files in this contribution include:

AAAREADME.TXT – this file

BLD.COM – procedure to build CLEAR_ERRORS exe

CLEAR_ERRORS.C – source

CLEAR_ERRORS.EXE – executable

CLEAR_ERRORS.OBJ – compiled source

SET_ERRORS.C – program to set error count for each device

SET_ERRORS.EXE – executable

SET_ERRORS.OBJ – compiled source

SYSDOC.TXT – system documentation

SET_ERRORS is included because it was useful for testing and debugging CLEAR_ERRORS.

Questions and comments are welcome.

Mark Oakley

Verizon Wireless

5165 Emerald Parkway

Dublin, OH 43017

614/560-8726

mark.oakley@verizonwireless.com

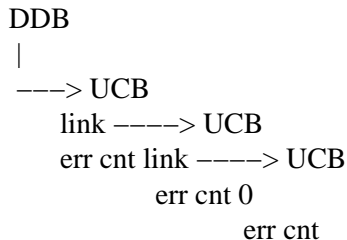
\$ type sysdoc.txt

CLEAR_ERRORS zeroes out the error count for all devices on a VMS Alpha system.

CPU and memory errors are also zeroed out.

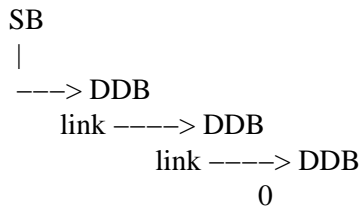
comp.os.vms: Re: How to clear device errors w/o reboot?

The error count for a device is stored in a data structure called a unit control block (UCB). All device UCBs of a particular kind (e.g. DGA1:, DGA2:, etc.) are linked together. The first UCB in the list is pointed to by a device data block (DDB). The last UCB in the list contains a null value (0) as a pointer. So the structure might look like:



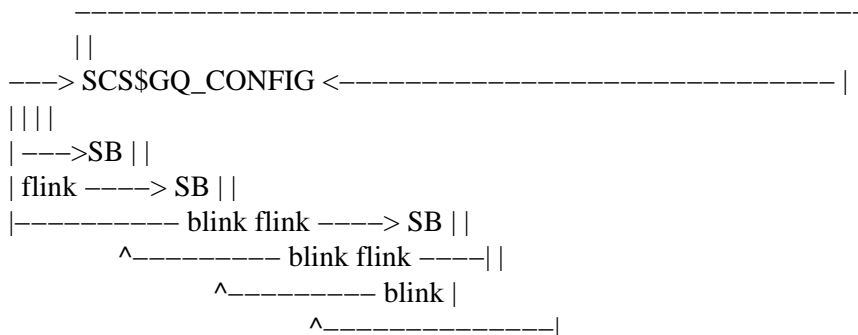
Note that the UCB has over 100 fields. The above diagram just depicts a few fields.

All of the DDBs are linked together. The first DDB in the list is pointed to by an SCS system block (SB). The last DDB contains a null value (0) as a pointer. The structure might look like:



Note that the DDB has over 20 fields. The above diagram just depicts the link field.

All of the SBs are arranged in a doubly linked list. The first SB in the list is pointed to by the global symbol SCSS\$GQ_CONFIG. The structure might look like:



comp.os.vms: Re: How to clear device errors w/o reboot?

Note that the SB has over 20 fields. The above diagram just depicts the link fields.

Each VMS system has at least one SB that points to all the local devices.

If the system is clustered, then there will be one SB for each cluster member.

If there are HSJ, HSC, or HSD controllers connected, there will be one SB for each.

CLEAR_ERRORS starts at SCS\$GQ_CONFIG and works its way through all these linked lists, zeroing the error fields as it goes.

EXE\$GL_MEMERRS and EXE\$CL_MCHKERRS point to memory cells that hold the number of memory and cpu errors, respectively. Both are set to zero.

Zeroing the error fields can only be done from kernel mode and should be appropriately synchronized by acquiring the write mutex on the I/O device database and raising interrupt priority level(IPL). The SYSS\$CMKRNL system service is used to reach kernel mode, and the SCH_STD\$IOLOCKW internal routine

acquires the mutex and raises IPL. An internal routine called SCH_STD\$IOUNLOCK is used to release the mutex and lower IPL.

There is always a risk of a system crash when executing in kernel mode, invoking undocumented routines, and navigating internal data structures. CLEAR_ERRORS performs two steps to minimize the chance of a system crash.

The first is to see if CLEAR_ERRORS is executing on the same release of VMS that it

was linked on. The VMS release id (e.g. V7.3-1) is not stored in the image header. However, the image id of the linker is, and this is compared to the

image id in SYSS\$SYSTEM:LINK.EXE. If they match, then we assume the VMS releases are the same. If they don't match, then we abort. For VMS

V7.1-1H1

the linker image id is A11-39, for V7.2-1 it is A11-50, and for V7.3 and V7.3-1

it is A12-03.

The second step to minimize the chance of a crash is to declare a condition handler as soon as we reach kernel mode. In CLEAR_ERRORS, "nocrash" is the name of the exit handler. If "nocrash" executes it will release the mutex

on the I/O device database if one was acquired and drop IPL. This will avoid a

crash, but the process will likely be deleted if a SYSS\$EXIT is performed.

So

comp.os.vms: Re: How to clear device errors w/o reboot?

"nocrash" calls EXE\$REI_INIT_STACK to change the access mode from kernel to user. EXE\$REI_INIT_STACK turns control over to a routine called "graceful_exit" after the mode change.

CLEAR_ERRORS was compiled, linked, and tested on VMS V7.3 using C V6.5-001. The object was also linked and tested on VMS V7.1-1H1 and V7.2-1.

The error handler ("nocrash") was also tested by causing an access violation in kernel mode at elevated IPL with the I/O database locked, and "nocrash" was able to avoid a system crash.

"JLR" <jl.rayon@gmail.com> wrote on 02/23/2005 11:25:47 AM:

>
> *I have the source of ZDEC.MAR*
>
> ; *Author:*
> ; *Mark Oakley DuPont Experimental Station 12-Nov-1984*
> ; *This program is based upon the VARY program written by*
> ; *Gary Grebus of Battelle Columbus Labs.*
> *Last modification:*
> ; *20-Jul-2001 Paul Gallo Compaq Computer Corporation, CSC*
>
> *If you want to get it send me a mail at*
> *jl.rayon-nospam@gmail.com*
> *(remove "-nospam" from this address).*
>
> *I have tried this ZDEC.MAR in October 2001, and I its works with a*
> *drastic efficiency: the system crashed, and when it restart, the errors*
> *counters have been reset to 0 :)*
>
> *Jean-Luc RAYON*
>