[OpenVMS] How to Troubleshoot a Process Stuck in a Compute Loop
Last Technical Review: 13-JUL-1994

OP/SYS: DIGITAL OpenVMS VAX, Versions 5.0 through 6.1

COMPONENT: Memory Management

SOURCE: Compaq Computer Corporation

SUBJECT:

This article describes methods to troubleshoot a process in a compute loop.

DESCRIPTION:

In order to determine why a process is in a compute loop, first make sure the process priority is set to 0 to lessen the effect on other processes when it is no longer suspended. Then, try the following methods to determine where the process may be looping.

1) Issue the 'SHOW PROCESS/CONTINUOUS/IDENTIFICATION =<pid>' command. If it shows the process is running LOGINOUT.EXE, refer to another article in the OPENVMS database that addresses this problem. Use the following keywords to search for the article:

   " OpenVMS Disconnected Process Not Deleted System "

2) If the 'SHOW PROCESS/CONTINUOUS/IDENTIFICATION=<pid>' command shows an image other than LOGINOUT.EXE, it will be necessary to collect PCs (program counters) from that process. A few methods to do this are listed below. All of these techniques assume the process is free-running and not stopped (otherwise all of the PCs will be in the SUSPEND code).

   Following are ways to collect program counters on a process:

   a) Write down PCs while the process is running using the 'SHOW PROCESS/CONTINUOUS/IDENTIFICATION=<pid>' command.

   b) Run a DCL Command Procedure to do this. A sample program named DCL_PC_SAMPLER.COM is given below and is provided AS IS, with no implied warranty or support.

c) Collect PCs from the System Dump Analyzer using the
   following commands:

```
$ SET PROCESS/PRIVILEGE=CMKNRL
$ ANALYZE/SYSTEM
SDA> READ/EXEC                ! Read in OpenVMS symbol table(s)
SDA> SET PROCESS/INDEX=<pid>  ! Go to looping process
SDA> SET LOG HUNG_PROCESS_PC.LOG
SDA> SHOW PROCESS             ! Verify correct process
SDA> SHOW PROCESS/CHANNEL     ! See what files are open
SDA> SHOW PROCESS/IMAGE    ! See where image is located in memory
SDA> EXAMINE/I @PC         ! Repeat as needed
SDA> EXAMINE/I @PC         ! ....... ETC ....
SDA> ..........
SDA> EXIT                  ! Close HUNG_PROCESS_PC.LOG file
```

3) Use the article in the OPENVMS database that can assist in finding where the process PCs are in the program and how large the loop is. The more PCs that are gathered, the better for determining the size of the loop. To find this article, use the following search keywords:

" OpenVMS How To Troubleshoot Hung Process "

4) Sometimes it is useful to get an idea of what subroutine the process is in. One way to do this is to SUSPEND the process and then go into SDA to look at the CALL FRAMES of the process as described in the article mentioned about about troubleshooting hung processes. Then, make a kind of "call map". The process could be resumed and re-suspended to get another "call map". By doing this, it is possible to get an idea of what subroutines keep getting called and which ones are different each time.

For example, the following hypothetical "call map" gives the routine names going DOWN the stack from most recent to least recent call frames (from SHOW CALL down, the SHOW CALL/NEXT results). The MAIN routine that started everything is always on the bottom:

Map 1:	Map 2:	Map 3:	Map 4:	Map 5:
.....	.....	SUB-T	.....	.....
.....	SUB-Y	SUB-R	SUB-Z	.....
SUB-C	SUB-Z	SUB-C	SUB-Q	SUB-M
SUB-B	SUB-B	SUB-B	SUB-B	SUB-B
SUB-A	SUB-A	SUB-A	SUB-A	SUB-A
MAIN	MAIN	MAIN	MAIN	MAIN

In looking at this "call map", the lowest common denominator (common routine) is SUB-B, so this would be a good place to start looking for some kind of program loop. It is often the case that the routine inducing the loop may not have many PC samples, as it could be busy calling other routines.

5) Other things to look at in SDA about looping process are such things as:

   a) Process registers, because sometimes R0 will show an error status. Use the 'SHOW PROCESS/REGISTER' in SDA to see R0. It could have another value, but if the PC is at a location in the program that has an error status, then from SDA do an 'EVALUATE/CONDITION' command on the value in R0 to get some idea of what this error status might be.

   b) The process stack, looking for Signal or Mechanism arrays indicating some kind of error condition. Use the 'SHOW STACK' command to see this.

   c) Depending on how far in rundown the process is, there might be an error status at CTL$GL_FINALSTS (if it is nonzero). Again, the 'EVALUATE/CONDITION' command might decode this status.

6) Set your process index to this process and examine registers and memory locations inside of process space. SDA does NOT know the symbolic names for any of the variables but can look at the Virtual Address of these variables if you know what their Virtual Address is.

EXAMPLE PROGRAM:

Following is the example program DCL_PC_SAMPLE.COM.

```
$! DCL_PC_SAMPLER.COM   <pid>
$!--------------------
$! Uses SHOW PROCESS/CONTINUOUS to get an output file of PCs to look
$! at and extract into a resultant output file.  It expects the a
$! P1 parameter of the PID to take PC samples on.  It will run until
$! an error occurs or the user does a  Control-C to exit.
$!
$!    THIS DCL COMMAND PROCEDURE IS UNSUPPORTED AND PROVIDED AS IS.
$!
$! ************************************************************************
$!                           COPYRIGHT (C) 1994 BY
$!                      DIGITAL EQUIPMENT CORPORATION, MAYNARD
$!                        MASSACHUSETTS.  ALL RIGHTS RESERVED.
$!
$!   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
$!   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
$!   OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES
$!   THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER
$!   PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
$!
$!   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND
$!   SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
$!
$!   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
$!   SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
$!
$!   NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE
$!   ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL EQUIPMENT CORPORATION.
$!
$!   SUPPORT FOR THIS SOFTWARE IS NOT COVERED UNDER ANY DIGITAL SOFTWARE
$!   PRODUCT SUPPORT CONTRACT, BUT MAY BE PROVIDED UNDER THE TERMS OF THE
$!   CONSULTING AGREEMENT UNDER WHICH THIS SOFTWARE WAS DEVELOPED.
$!
$! ************************************************************************
$!
$ IF P1 .NES. "" THEN GOTO LETS_BEGIN
$ WRITE SYS$OUTPUT "Please pass PID as first parameter"
$ EXIT
$ LETS_BEGIN:
$ SET NOON
$ WRITE SYS$OUTPUT "Press CONTROL-C after 1 minute or when ready"
$ DEFINE SYS$OUTPUT SYS$SCRATCH:PC1.TMP
$ SHOW PROC/ID='P1/CONTINUOUS
$ DEASSIGN SYS$OUTPUT
$ WRITE SYS$OUTPUT "Thank You, Please Wait ....."
$ SEARCH SYS$SCRATCH:PC1.TMP/OUT=SYS$SCRATCH:PC2.TMP "Current PC"
$ SORT SYS$SCRATCH:PC2.TMP/NODUP/KEY=(POS=24,SIZE=8,CHARACTER) -
       SYS$SCRATCH:PC3.TMP
$ ! - PROCESS -
$ IF F$TRNLNM("FILEOUT") .nes. "" THEN CLOSE FILEOUT
$ IF F$TRNLNM("FILEIN") .nes. "" THEN CLOSE FILEIN
$ OPEN FILEIN SYS$SCRATCH:PC3.TMP
$ OPEN/WRITE FILEOUT PC.LOG
$ WRITE FILEOUT "Unique PCs"
$ WRITE FILEOUT " "
$ COUNT = 0
$ NEXT_LINE:
```

```
$ READ FILEIN LINEIN/END=NOMORE
$ COUNT = COUNT + 1
$ LINEOUT = F$EXTRACT(24,8,LINEIN)
$ WRITE FILEOUT "| ''LINEOUT' |"
$ GOTO NEXT_LINE
$ NOMORE:
$ WRITE FILEOUT " "
$ WRITE FILEOUT "End of unique PCs, ''count' unique PCs found"
$ IF F$TRNLNM("FILEOUT") .nes. "" THEN CLOSE FILEOUT
$ IF F$TRNLNM("FILEIN") .nes. "" THEN CLOSE FILEIN
$ COUNT1 = 0
$ OPEN FILEIN SYS$SCRATCH:PC2.TMP
$ NEXTLINE1:
$ READ FILEIN LINEIN/END=NOMORE1
$ COUNT1 := COUNT1 + 1
$ GOTO NEXTLINE1
$ NOMORE1:
$ IF F$TRNLNM("FILEIN") .nes. "" THEN CLOSE FILEIN
$ COUNT2 = 0
$ OPEN FILEIN SYS$SCRATCH:PC3.TMP
$ NEXTLINE2:
$ READ FILEIN LINEIN/END=NOMORE2
$ COUNT2 := COUNT2 + 1
$ GOTO NEXTLINE2
$ NOMORE2:
$ IF F$TRNLNM("FILEIN") .nes. "" THEN CLOSE FILEIN
$ RATIO = (COUNT2 / COUNT1) * 100
$! WRITE SYS$OUTPUT "UNIQUENESS VALUE IS ''RATIO'"
$ WRITE SYS$OUTPUT "Unqiue PCs are in PC.LOG"
$ DELETE SYS$SCRATCH:PC%.TMP;*
$ EXIT
```