Kernel Panic

Connect, Inc.
1701 Quincy Avenue, Suites 5 & 6, Naperville, IL 60540
Ph: (630) 717-7200 Fax: (630) 717-7243
www.connectrf.com

Table of Contents

SCO Kernel Patches for Panics on Pentium Systems

Web Links

http://www.sco.com/cgi-bin/ssl_reference?104530
http://www.sco.com/cgi-bin/ssl_reference?105532

Pentium System Panics with Trap Type 6; no dump is saved to /dev/swap.

Keywords

panic trap kernel invalid opcode pentium querytlb querytbl 386 mpx unix v4 dump swap double type6 no patch /etc/conf/pack.d/kernel/locore.o image locore

Release

SCO UNIX System V/386 Release 3.2 Operating System Version 4.0, 4.1 and 4.2
SCO Open Desktop Release 2.0 and 3.0
SCO Open Server System Release 2.0 and 3.0

Problem

My system panics with a Trap Type 6, but no memory dump gets written.

Cause

There is a flaw in the kernel querytlb() routine that allows the Pentium to execute a 386-specific instruction which causes it to panic with an invalid opcode.

querytlb() is only called when your system is panicking. The fault causes the system to double-panic and it thus fails to write a panic dump to disk. This means that the panic trap type 6 is concealing another type of panic which will show itself after the following patch has been applied.

Solution

You can apply the following patch, as root, to: /etc/conf/pack.d/kernel/locore.o

Use the procedure that follows. Change the /etc/conf/pack.d/kernel directory, make a backup copy of locore.o, patch the locore.o binary, and relink the kernel. Note that the commands issued to _fst are preceded by an *.

```
# cd /etc/conf/pack.d/kernel
# cp locore.o locore.old
# _fst -w locore.o -
* querytlb+5?w 0x9090
querytlb+0x5:     0x375= 0x9090
* querytlb,4?ai
querytlb:    querytlb:     call   near 0x17:0x0
querytlb+0x5:     nop
querytlb+0x6:     nop
querytlb+0x7:     sub    eax, eax
* $q
# cd /etc/conf/cf.d
# ./link_unix
```

Once the relink is complete, reboot your system for the changes to take effect.

http://www.sco.com/cgi-bin/ssl_reference?105532

TA104530 created on 20 April 1994, last updated on 01, September 1998
SSL #: 482366 IT #: os2366

My ODT3 Pentium Pro-based system panics with trap 6 or with k_trap type 0.

Keywords

mpx processor cpu install installation pentium pro trap 6 odt 3.2v4.2 compaq proliant 2500 panic k_trap type 0 v4 unix open server 3.0

Release

SCO UNIX System V/386 Release 3.2 Operating System Version 4.2
SCO Open Server Enterprise System Release 3.0
SCO Open Server Network System Release 3.0
SCO Open Desktop Release 3.0
SCO Open Desktop Lite Release 3.0

Problem

My Pentium Pro-based system intermittently panics either at installation or when booting off the hard disk, with trap 6. I followed the directions in Technical Article 104530, "Pentium System panics with Trap Type 6; no dump is saved to /dev/swap." and I now get a panic k_trap type 0.

Solution

See Technical Article 104530 for information on a workaround for trap 6 problems caused by a querytlb() problem. Follow the procedure in this article and reboot the patched kernel.

Follow the additional instructions below ONLY if you now get a k_trap type 0 panic after following the instructions in Technical Article 104530. To correct a k_trap 0, do the following:

```
# cd /etc/conf/pack.d/pit
# cp Driver.o Driver.orig
# _fst -w Driver.o
* spinwait+2D?w F989 FEE2
* $q
# cd /etc/conf/cf.d
# ./link_unix -y
```

Reboot your system. The above patch corrects a problem with a software delay loop that was optimized out by the compiler and which can cause panics on faster processors.

TA105532 created on 02 May 1997, last updated on 02, June 1997
SSL #: 483282 IT #: os3282

Kernel Panic Debug Methodologies

Affected Packages: All Connect RF Systems

Introduction

This document provides details of troubleshooting methods to be used for UNIX kernel panics.

A UNIX kernel panic is defined as follows:

The system "panics" when it encounters a hardware problem or kernel inconsistency that is so severe the system cannot continue to function. When this happens, the system usually displays a message on the console, and all system activity stops.

The system panic messages begin with the word "PANIC", followed by a diagnostic message. Sometimes this diagnostic explains what caused the panic, but sometimes it only indicates that the kernel was corrupted and the actual cause of the corruption was a few instructions back. The system can also panic without displaying a PANIC message. When this happens, the system simply refuses to process any input from the system console and all other terminals.

This description was taken from the SCO UNIX System Administrators manual on page 486. This is the classical definition of a "kernel panic". It is generally accepted that most panics are caused by hardware problems, such as defective memory, CPU, etc.

There are only three things that can cause the kernel to panic:

1. Hardware
2. Device driver that is linked into the kernel
3. SCO UNIX kernel bug

This list is in order of likelihood. Most panics are caused by either numbers 1 or 2 above. SCO UNIX has been field tested over many years, and it is extremely rare to uncover a bug in the UNIX kernel.

This document is intended to give guidance to support personnel faced with a system experiencing kernel panics.

It must be stated that kernel panics that are not the cause of obvious hardware problems, should be considered an Engineering effort. If your system is experiencing kernel panics and hardware has been ruled out, you should make contact with Engineering for assistance.

This document will not be a detailed guide into kernel debugging, which is beyond the scope of this document and generally beyond the ability of most support personnel, but rather a guide and course to follow. The intent of this procedure is to provide guidance on the proper identification, hardware isolation, and collection of information for Engineering analysis.

Diagnostic Description

The following section is a Tech Note from SCO on UNIX panics. It represents the recommended way of troubleshooting panics. The procedure goes into the details of a simple analysis of the dump image.

It is generally recommended once the hardware has been eliminated, that Connect Engineering be contacted for this type of troubleshooting.

This Note was pulled from the SCO knowledge base. The number for this BBS is listed later in the Reference section.

SCO Tech Note:

Why did my UNIX kernel "panic"?

Release: SCO UNIX System V/386 Release 3.2 Operating System Version 2.x
SCO Open Desktop Release 1.x

Problem: SCO UNIX keeps crashing. The system stops functioning and a message on the console indicates that the SCO UNIX kernel was interrupted by a "panic". SCO UNIX may or may not reboot after this occurrence. How can I determine what happened?

Cause: System panics can be caused by hardware or software problems. The following procedure can be used to diagnose the cause of some system panics.

Solution: When the SCO UNIX kernel panics, it often displays information about the cause of the panic, in the form of a "register dump" and by saving a "system image" in the dump device. The register dump contains the values of the CPU registers at the time of the panic. The system image contains the entire contents of system memory at the time of the panic. This information can be used to examine the state of the kernel when the panic occurred, and can often be used to determine the cause of the panic.

Note: Not all system panics generate a register dump.

There are two steps in diagnosing the cause of a panic. First, you must determine whether the panic is consistent or inconsistent. Second, you must use the crash(ADM) utility to determine what the kernel was executing at the time of the panic.

I. Determining The Consistency Of The Panic

1. Reading the Register Dump

The register dump is displayed on the system console, and is not stored in any file. This information will be lost when the system reboots. In order to preserve this information you need to make sure that the system will not auto-reboot and to make sure that someone writes the panic message down before rebooting. (See section I.3 below for instructions on how to prevent auto-boot.)

If a register dump is present, it will look like the following sample register dump, with 'NNNN' replaced by the actual number of memory pages in your machine.

```
=================================================================
        PANIC:
        cr0 0xFFFFFFEB   cr2  0x00FFFFFF   cr3 0x00002000   tlb  0x00500E80
        ss  0x00000038   uesp 0xD0119554   efl 0x00010282   ipl  0x00000000
        cs  0x00000158   eip  0xD0070488   err 0x00000000   trap 0x0000000E
        eax 0x00FFFFFF   ecx  0x00000000   edx 0x00000305   ebx  0xD00CD780
        esp 0xE0000D40   ebp  0xE0000D64   esi 0xD0119554   edi  0x00000038
        ds  0x00000160   es   0x00000160   fs  0x00000000   gs   0x00000000

        PANIC: Kernel mode trap.  Type 0x0000000E
                Trying to dump NNNN Pages.
        ..................................................................
        ..................................................................
        NNNN Pages dumped

        **   Safe to Power Off   **
              - or -
        ** Press Any Key To Reboot **
```

Note that most of the register dump consists of character strings followed by hexadecimal values. The character strings are the names of the registers in your 80386 or 80486 chip: the values are the contents of those registers at the time of the panic.

Find the characters "cs" in the register dump. This is the chip's Code Segment (CS) register. Find the characters "eip" in the register dump. This is the chip's Instruction Pointer (IP) register. The values in the CS and IP register combine to form the address of the instruction that the kernel was executing at the time of the panic. This value is sometimes called the "PC" value.

Write down these values as a pair of numbers, separated by a colon, and without leading zeros.

Note: Trailing zeros are important. For example, in the register dump above, the value of CS is 0x00000158, and the value of IP is 0xD0070488. Therefore, the PC value you should write down is 158:D0070488.

Important: When your system panics, the person who reboots the system should write down the panic message and (if present) the PC value in your system log.

2. Hardware and Software Panics

Panics can be caused by defective hardware or by a software problem. If the PC value varies widely from panic to panic, you almost certainly have a hardware problem. If you have three or more panics with the same PC value, then it is likely that you have a software problem. Follow the instructions in section II below to try to diagnose your problem.

Note: It is possible for defective RAM to cause multiple panics at the same address. This is a hardware problem, and cannot be fixed via software.

If the PC value varies widely, you can attempt to determine the piece of hardware that is causing the problem. Strip your system down to a minimum configuration: just the hard disk, the video card, and a minimal amount of RAM. Then add hardware piece by piece until the panics begin to occur. If necessary, swap parts of the minimal system (such as the video card) with ones from a system that does not panic. Always keep a notebook of your efforts, and remember that this is likely to be long, frustrating, and difficult search.

Note: One common cause of system panics is a defective power supply or "dirty" power. SCO Support recommends the use of an uninterruptible power supply for critical systems.

3. Preventing SCO UNIX from autobooting.

SCO UNIX can be configured to reboot automatically after a panic.
This is controlled by the value of the "PANICBOOT" variable in the file "/etc/default/boot". To prevent auto-reboots after a panic, edit the file so that this line reads "PANICBOOT=NO". Make sure that the 'P' in "PANICBOOT" is the first character on the line.

II. USING CRASH(ADM)

Important Note: The following procedure is only useful if your panics are due to a software problem. You MUST have at least three panics with identical PC values in order for the results of this procedure to be meaningful.

1. Obtaining a System Dump Image

You will need to save a copy of the system dump image for use with crash(ADM). When you reboot the system, you will see a prompt:

```
There may be a system dump memory image in the swap device.
Do you want to save it? (y/n)>
```

Type 'y' and hit <Return>. Next you will see the prompt:

```
Use Floppy Drive 0 (/dev/rfd0) by default.
Press ENTER to use default device.
Enter valid Floppy Drive number to use if different.
Enter "t" to use tape.>
```

SCO Support strongly recommends that you not use floppy drives to save system dump images. Since the typical SCO UNIX system has many megabytes of memory, you will need to use several floppy disks to save a single image. Problems can arise if you do not have enough floppy disks, or if you insert them in the wrong order.

Type 't' and hit <Return>. Next you will see the prompt:

```
Enter choice of tape drive:

    1 - /dev/rct0
    2 - /dev/rctmini
    n - no, QUIT>
```

Type '1' or '2' as appropriate for your system. Next you will see the prompt:

```
Insert tape cartridge and press return, or enter q to quit.
```

Insert your tape, and press <Return>. You will see a message similar to this one:

```
Wait.
dd if=/dev/swap of=/dev/rct0 bs=120b count=751 skip=0
```

(The actual numbers may vary for your machine.) Finally, you will see the message:

```
Done.  Use /etc/ldsysdump to copy dump from tape or diskettes.
Press return to continue.>
```

At this point, you have saved the system dump image to tape. Press <Return> and continue with the boot process. When the boot process has completed, you will be ready to load the system dump image onto your hard disk. Log in as root, enter the commands:

```
# cd /tmp
# ldsysdump image
```

Note that the pound sign ('#') is a prompt from root's shell: do not type it in. Also note that the argument to ldsysdump(ADM) is the file name in which the system dump image is stored. It is "image" in this example, but can be any file name. You will next see the prompt:

```
Use Floppy Drive 0 (/dev/rfd0) by default.

Press ENTER to use the default.
Enter valid Floppy Drive number to use if different than default.
Enter "t" to use tape drive.
```

Type 't' and press <Return>. Next you will see the prompt:

```
Enter choice of tape drive:
   1 - /dev/rct0
   2 - /dev/rctmini
   n - no, QUIT>
```

Type '1' or '2' as appropriate for your system. Next you will see the prompt:

```
Insert tape cartridge and press return, or enter q to quit. >
```

Insert your tape, and press <Return>. You will see a message similar to this one:

```
Wait.
dd if=/dev/rct0 bs=120b count=751
```

(The actual numbers may vary for your machine.) Finally, you will see the message:

```
System dump copied into image. Use crash(1M) to analyze the dump.
```

At this point, you have a system dump image that you can use with the crash(ADM) utility.

2. Obtaining a Stack Trace

The crash(ADM) utility will allow you to examine the state of your system at the time of the panic. The crash(ADM) manual page describes this command in detail. Knowledge of UNIX kernel internals is necessary to use crash(ADM) to its full potential. However, there are a few simple commands that can give you a good idea about what was going on at the time of the panic.

To invoke crash(ADM), enter the command:

```
# crash -d image
```

You will then see the crash(ADM) prompt, which is a '>' character. Note that the argument to crash(ADM) is the file in which the image is stored. This is "image" in this example, but it can be any file name.

Note: crash(ADM) will only give meaningful results if the file "/unix" is the one that generated the system dump image. If you re-link your kernel, attempting to use the new "/unix" with the old image file will give inaccurate results!

There are three commands that are useful. The 'panic' command shows a limited register dump, plus the partial contents of the kernel stack at the time of the panic. The 'trace' command shows the complete contents of the kernel stack at the time of the panic. By examining the contents of the kernel stack it is often possible to determine the cause of the panic. The 'user' command displays information about the program that was running at the time of the panic. This information can help determine which kernel subsystem was being used when the panic occurred. Type 'quit' to exit from the crash(ADM) command.

3. A sample crash(ADM) session.

Here is a sample crash(ADM) session. The commands entered by the user are in front of the '>' prompt. The rest is output from the crash(ADM) command. In this session, the user entered a 'panic' command, followed by a 'user' command and a 'quit' command.

```
==========================================================
# crash -d image
dumpfile = image, namelist = /unix, outfile = stdout
> panic
System Messages:

WARNING:

Panic String: Kernel mode trap. Type 0x%x
Kernel Trap. Kernel Registers saved at e0000d10
ERR=0, TRAPNO=14
cs:eip=0158:d0070488 Flags=10282
ds = 0160   es = 0160   fs = 0000   gs = 0000
esi= d0119554   edi= 00000038   ebp= e0000d64   esp= e0000d40
eax= 00ffffff   ebx= d00cd780   ecx= 00000000   edx= 000003d5
Kernel Stack before Trap:
STKADDR   FRAMEPTR  FUNCTION  POSSIBLE ARGUMENTS
e0000d40  e0000d64  panopen  (3800,1,2,d0119554)
e0000d6c  e0000d88  s5openi  (d0119554,1,1,d0119554)
e0000d90  e0000dac  copen1   (d0119554,1,e0001148,d00adf20)
e0000db4  e0000de0  copen    (1,403594,e0000e38,7ffffe7c)
e0000de8  e0000df8  open     (403594,0,40358c,d011e274)
e0000e00  e0000e2c  systrap  (e0000e38)
> user
PER PROCESS USER AREA FOR PROCESS 5
USER ID's:      uid: 0, gid: 0, real uid: 0, real gid: 0
PROCESS TIMES:  user: 1, sys: 8, child user: 0, child sys: 0
PROCESS MISC:
        command: sh, psargs: -
        proc slot: 5, cntrl tty:   3,1
        start: Wed May 22 11:31:10 1991
        mem: 18, type: fork
        proc/text lock: none
        inode of current directory: 149
OPEN FILES AND POFILE FLAGS:
        [1]: F#1, 5b           [2]: F#1, 43           [3]: F#1, 0
        [4]: F#2, 0            [5]: F#3, 0
FILE I/O:
        u_base:   402d21, file offset: 3728, bytes: 672,
        segment: sys, cmask: 0077, ulimit: 2097152
        file mode(s): read write
SIGNAL DISPOSITION:
           1: 1bfc      2: 1d50      3: 1d50      4: 1bfc
           5: 1bfc      6: 1bfc       7: 1bfc      8: 1bfc
           9:  default  10: 1bfc     11: 1d50      12: 1bfc
          13: 1bfc      14: 1d50      15: 1d50      16: 1bfc
          17: 1bfc      18:  default  19:  default  20:  default
          21:  default  22:  default  23:  default  24:  default
          25:  default  26:  default  27:  default  28:  default

> quit
```
==

As you can see, the output of crash(ADM) contains a large amount of highly cryptic information. Only a few pieces of it are needed to get an idea of what is going on.

The output of the 'user' command contains information about the process that was executing at the time of the panic. The main thing you should look at is the "PROCESS MISC" section. Under it, you will see two fields that tell you what command was running: "command" is the name of the command, and "psargs" gives the first few arguments to the command. The other fields of interest in the 'user' command output are in the "USER ID's" section, which gives the read and effective uids of the user who ran the process.

In the sample output above, the command being run at the time of the panic was "sh" - the Bourne shell. Since the "psargs" field consists of a "-", this shell was a login shell. Since the values in the "USER ID's" section are all 0, this shell was being run by root.

The most important part of the 'panic' command output is the section marked "Kernel Stack before Trap". The column labeled "FUNCTION" contains the sequence of kernel function calls that led up to the panic. The 'trace' command output is similar, but also shows the sequence of function calls after the event that caused the panic. This stack is displayed as growing upwards, so that functions are called by functions below them in the stack, and call functions that are higher in the stack.

In the sample output above, the kernel panicked while invoking the function panopen(). An examination of the kernel stack shows that this function was called (via copen(), copen1(), and s5openi()) from the function open(). The combination of these two facts suggests that the panic was caused by a bug in the "pan" driver.

Note that the problem may not be in the top-most function in the function stack. For example, if a kernel utility function such as physio() were invoked with improper arguments, physio() would be at the top of the kernel stack, even though the problem was in the function that called it.

4. Diagnosing the Problem

Diagnosing kernel panics requires knowledge of kernel internals and experience in debugging 'C' code.

It is often useful to know in which file the problematic function is located. This can sometimes tell you if the problematic function is part of the operating system or if it is part of a third-party or SCO-supplied driver.

Note: The following shellscript, Script #1, requires that you have the nm(CP) program installed on your system. If you do not have it installed, please use Script #2. Script #2 uses strings(C) in place of nm(CP).

nm(CP) is provided with the SCO UNIX Development System and the Open Desktop Development System.

strings(C) is provided with the standard OS.

The shell script below will print out the names of the kernel component files that reference a particular function. This example searches for the function sioopen(): substitute the appropriate function name for "sioopen". Log in as root and enter the command:

(Script #1)

```
# for i in `find /etc/conf -name '*.o' -print`
> do
> nm $i 2> /dev/null | grep sioopen > /dev/null && echo $i
> done
```

-or-

(Script #2)

```
# for i in `find /etc/conf -name '*.o' -print`
> do
> strings $i 2> /dev/null | grep sioopen > /dev/null && echo $i
> done
```

The '#' and '>' characters at the left are prompts from root's shell: Do not type them in. Be sure to get the "`" and "'" quotes right!

Write down the list of file names printed by the above command. For this example, you will get the output:

```
/etc/conf/pack.d/sio/Driver.o
```

You can then search for the file name in the /etc/perms files. For this example, you would type:

```
egrep /etc/conf/pack.d/sio/Driver.o /etc/perms/*
```

This will give as output:

```
/etc/perms/inst:LINK f644 root/sys 1 ./etc/conf/pack.d/sio/Driver.o N04
```

Since the name after the colon is "LINK", you know that this kernel routine is part of the SCO-supplied link kit. If the file does not appear in the perms files, or if the package name that appears before the colon is part of a third-party device driver, then you know that the problem is with that driver.

Hints: If a routine is mentioned in a file named Driver.o, it is usually part of a device driver, and the driver prefix is the name of the directory in which the Driver.o file is located. If a function name ends in "read", "write", or "ioctl" it is usually part of a character device driver. If a function name ends in "strategy" it is usually part of a block device driver. Kernel utility function names include bcopy(), copyio(), cpass(), passc(), getc(), geteblk(), getablk(), mapphys(), memget(), putc(), fubyte(), subyte(), vtop(), vtop2(), and vtopreg().

5. Contacting SCO Support

If you are still unable to determine the cause of the panic, you may want to contact SCO support. Follow the steps in section II.1 above, and then enter the following commands:

```
# crash -d image -w /tmp/crash.out
dumpfile = image, namelist = /unix, outfile = /tmp/crash.out
> panic
> trace
> user
> quit
```

As before, the '#' and '>' prompts are generated by the crash(ADM) command, so do not type them in. Print out the file "/tmp/crash.out" and send it via fax or email to your support provider.

If you are contacting SCO Support, we ask that you provide the following information along with the crash(ADM) output.

- The exact version of the SCO UNIX System V/386 Operating System you are running.
- The brand and model number of the computer you are using.
- A complete description of the hardware configuration, including the brand and model number of every peripheral card in the machine.
- A listing of all device drivers that you have installed on the machine.
- A listing of all software you have installed on the machine that re-linked the kernel as part of the installation.
- The amount of RAM installed in your computer.

SEE ALSO: autoboot(ADM)
crash(ADM)
messages(M)
nm(CP)

Appendix: What are Traps, Interrupts, and Exceptions?

Keywords: interrupts traps panics exceptions fault nmi abort protection unix xenix kernel error messages what why

Release: SCO XENIX Operating System Release 2.3.4 and earlier
SCO UNIX System V/386 Release 3.2 Operating System Version 4.2 and earlier

Problem: I am receiving a trap, interrupt, or exception message, and don't know what it means.

Solution: Traps, interrupts, and exceptions are basically the same thing and are handled the same by the CPU. Below are some quick definitions:

Interrupts: Used to handle external asynchronous events (i.e., signals that come from the bus are interrupts 33-255 in decimal or 0x21-0xFF in hex). The message "Trap 0xFF", which translates into "Trap 255", indicates bad hardware; Only a piece of hardware on the bus can generate that number.

Exceptions: Used to handle instruction faults.

Trap: Exceptions that are reported immediately after the execution of the instruction that caused the problem. User defined interrupts are examples of traps.

Interrupts 19-32 decimal or 0x13-0x20 in hex are INTEL RESERVED.

Key for the type column below:

NMI = Non-maskable interrupts. These provide a method of servicing very high priority interrupts. They are generated by hardware.

Fault = Exceptions that are detected and serviced before the execution of the faulting instruction (i.e., page faults).

Abort = Exceptions that do not permit the precise location of the instruction causing the exception to be determined. These are used to report severe errors such as a hardware error or illegal values in system tables.

Below is a table that shows a list of interrupts, what type it is, and a description of what could cause that interrupt.

Int	Type	Cause
0x00	Fault	Divide error
0x01	Trap	Not used in XENIX or UNIX
0x02	NMI	NMI interrupt
0x03	Trap	Not used in XENIX or UNIX
0x04	Trap	Interrupt on overflow
0x05	Fault	Array bounds check
0x06	Fault	Invalid op code. Can be caused by any illegal instruction, cacheing, bad hardware, bad instruction from memory or bad memory itself.
0x07	Fault	Device not available (i.e., coprocessor trap for one that does not exist).
0x08	Abort	Double fault. Causes the classic double panic. The kernel panicked while it was panicking. This is caused by any illegal instruction from a third party software package or from bad RAM.

0x09 Abort	Coprocessor segment overrun.
0x0A Fault	Invalid task state, corresponds to how the 386 does 8086 emulation. SCO VP/ix related problems or bad hardware.
0x0B Fault	Segment not present. Addressing problem. Suspect bad memory or 3rd party device driver conflict.
0x0C Fault	Stack fault. Only occurs in kernel mode. Fixed size stack is being overflowed. For example, a bad internal modem could be flooding the stack.
0x0D Fault	General protection trap. Caused by any memory reference or invalid address. Could also be a parity error from bad RAM.
0x0E Fault	Page fault, addressing problem because the kernel tried to reference a page that it couldn't bring in to memory. Could be a memory problem (bad RAM) or a 3rd party device driver bug.
0x0F	Not Used in XENIX or UNIX.
0x10 Fault	Coprocessor Error, hardware problem.
0x11 Fault	Alignment Check. Caused by memory reference to data at unaligned data address. Could be a 3rd party device driver bug.
0x12	Machine Check (Pentium only), model dependent.

References

SCO UNIX System Administrators Guide Num. 3.2v4.2b Dated 1 February 1993
SCO Knowledge Base BBS (408) 426-9495

About This Document

This document is based on the following Technical Documents in our Lotus Notes database that have been made obsolete: T1080 and D1225.

Please let us know about any errors in this document at: http://207.241.78.223/isoxpert/calltrak.nsf/WebTracking?OpenForm.