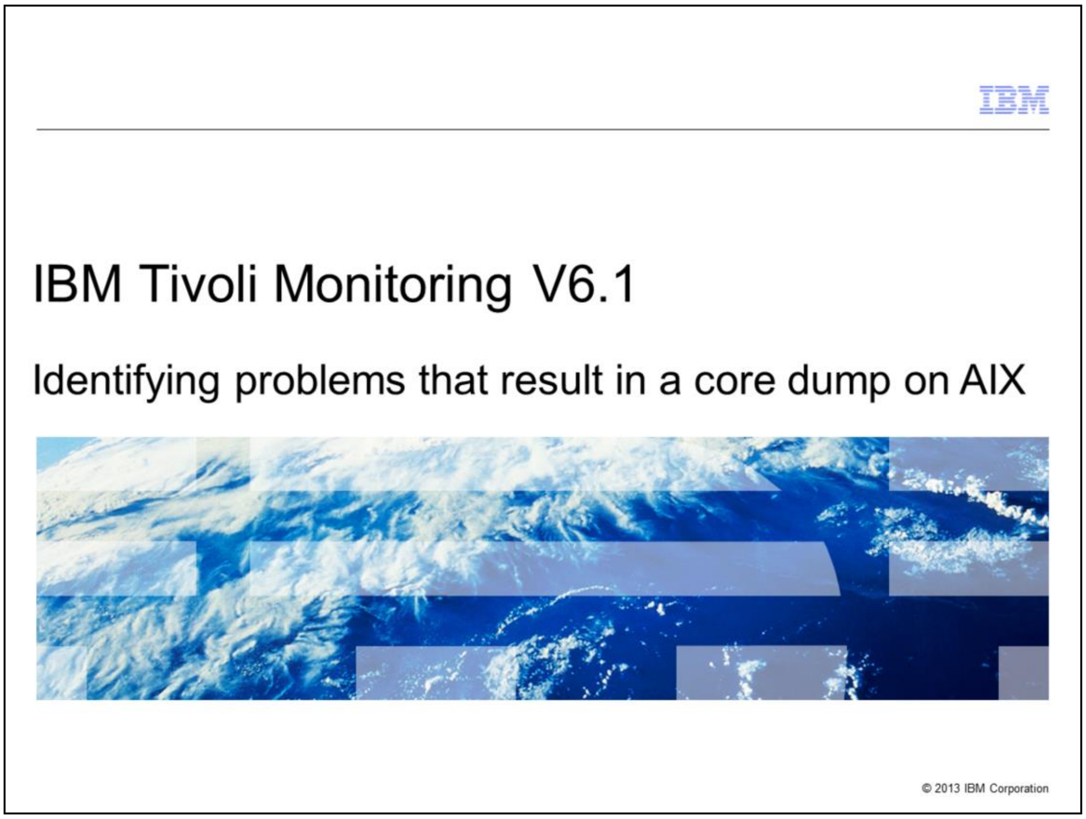
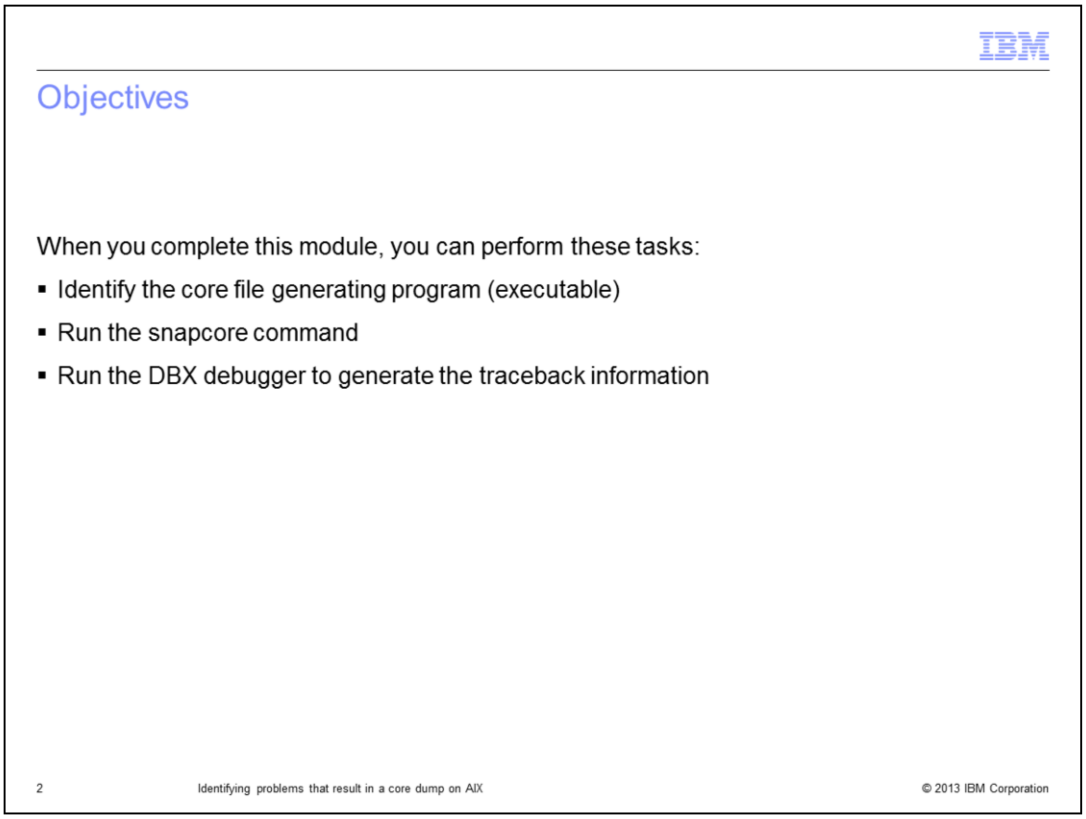


IBM Tivoli Monitoring V6.1

Identifying problems that result in a core dump on AIX

IBM Tivoli® Monitoring V6.1, Identifying problems that result in a core dump on AIX®. In this module, you learn about the steps to identify when a monitoring application is stopping on AIX and how to gather basic documentation to determine if the cause of the stop or crash is a known problem.



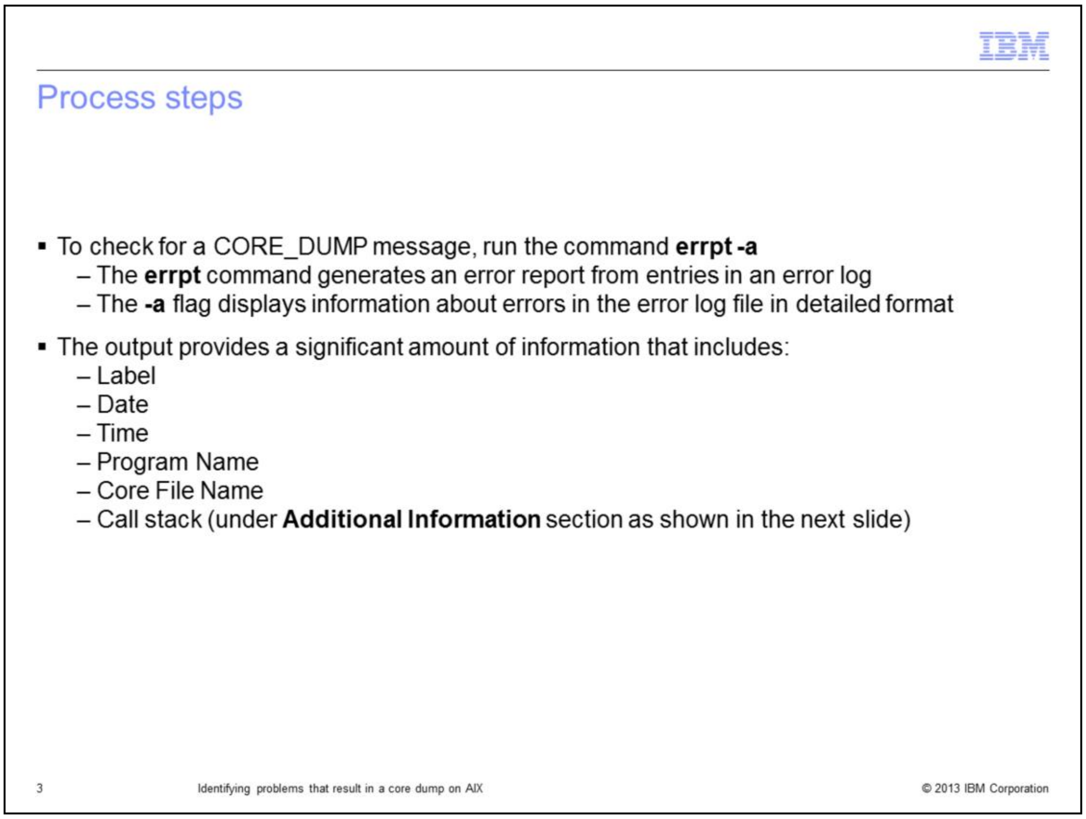
Objectives

When you complete this module, you can perform these tasks:

- Identify the core file generating program (executable)
- Run the snapcore command
- Run the DBX debugger to generate the traceback information

2 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

When you complete this module, you can identify whether a program (executable) produced a core file (core) and gather necessary files for core file analysis on **A**dvanced **I**nteractive e**X**ecutive (AIX) with the **snapcore** utility. You can also review traceback information from the core file to search for known problems. It is important to understand that a core file is a generic symptom. The ability to gather necessary documentation for core analysis and to distinguish between different root causes for cores greatly reduces the time of problem resolution.



Process steps

- To check for a CORE_DUMP message, run the command **errpt -a**
  - The **errpt** command generates an error report from entries in an error log
  - The **-a** flag displays information about errors in the error log file in detailed format
- The output provides a significant amount of information that includes:
  - Label
  - Date
  - Time
  - Program Name
  - Core File Name
  - Call stack (under **Additional Information** section as shown in the next slide)

3 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

The first step is to verify that the process is stopped due to receiving a signal to abort and that a core file exists for analysis.

It is possible for a process to stop with a controlled shutdown which does not result in a core file.

This case requires the analysis of product trace logs to understand why the application shut down, and is not the focus of this presentation.

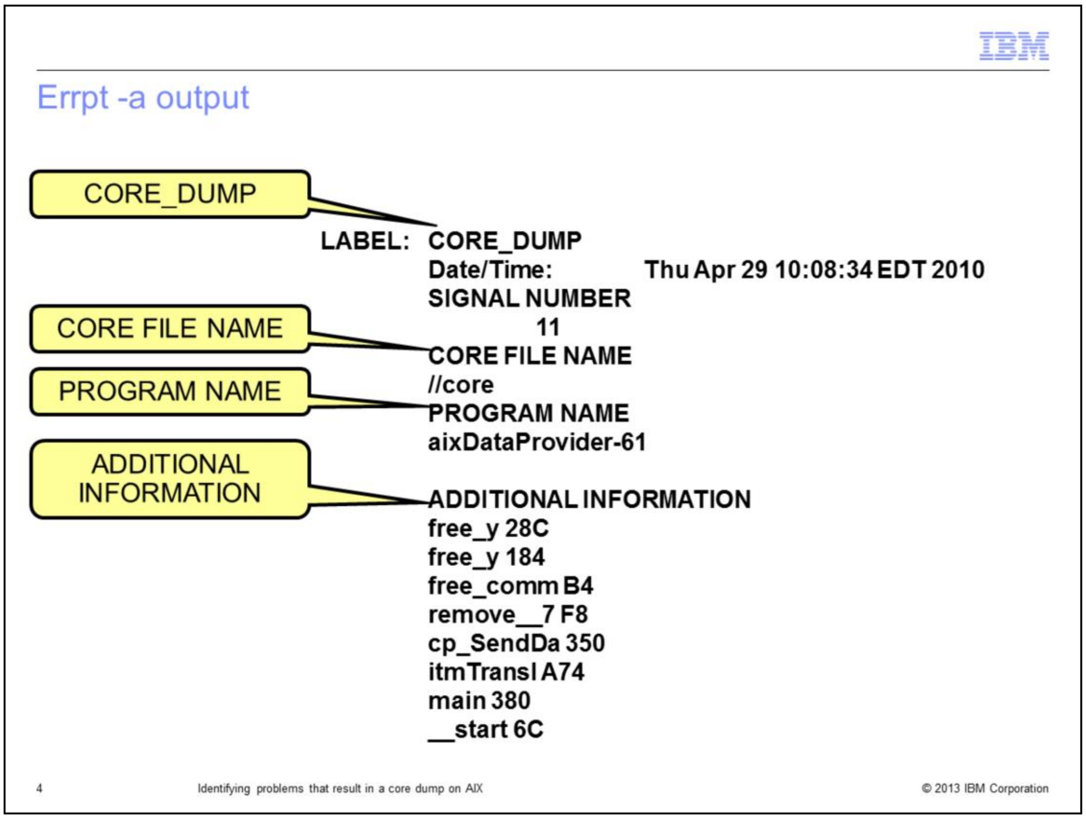
There are several possible reasons for the case where a process stops when it receives a signal from the operating system and no core file is created. Some possible reasons are:

- The application uses its own signal handler

- The **ulimit** settings are not adequate for the core file to be written

- There is no space to write out the core file on the file system …

On an AIX system, you can review the error report by running the command **errpt -a** and look for any message with a label of **CORE_DUMP**.



Errpt -a output

CORE_DUMP
CORE FILE NAME
PROGRAM NAME
ADDITIONAL INFORMATION

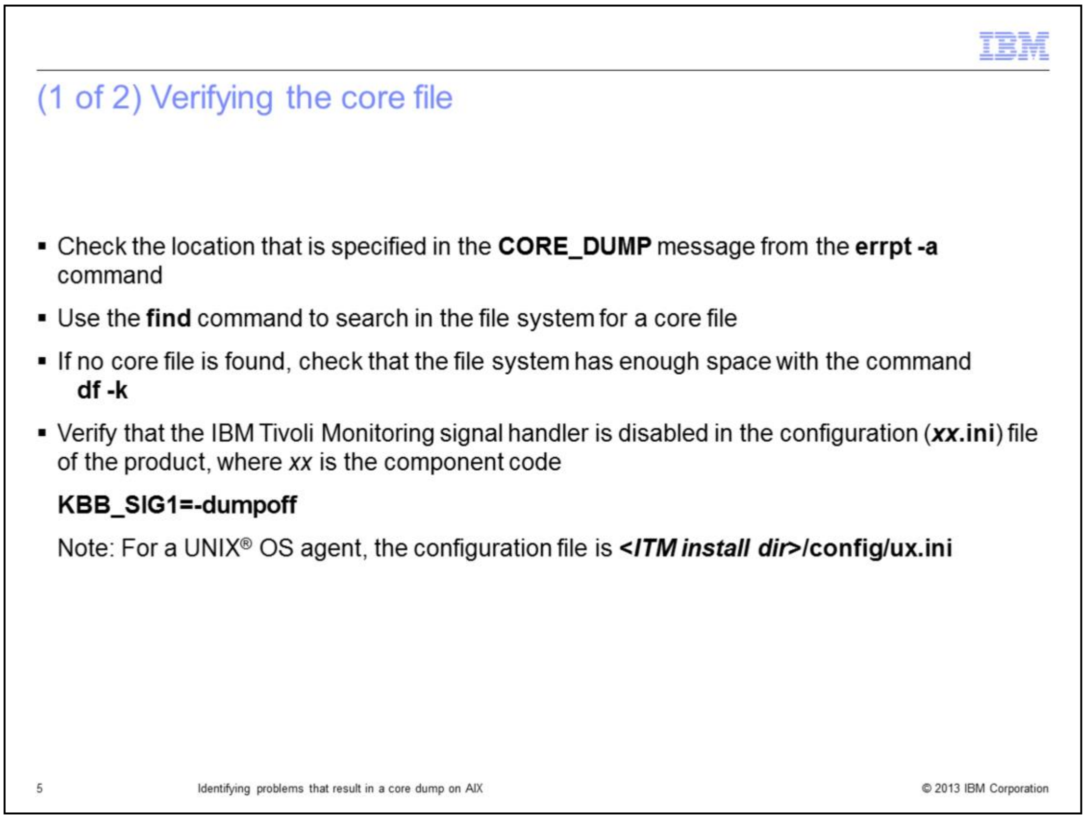
```
LABEL:  CORE_DUMP
        Date/Time:        Thu Apr 29 10:08:34 EDT 2010
        SIGNAL NUMBER
                  11
        CORE FILE NAME
        //core
        PROGRAM NAME
        aixDataProvider-61

        ADDITIONAL INFORMATION
        free_y 28C
        free_y 184
        free_comm B4
        remove__7 F8
        cp_SendDa 350
        itmTransl A74
        main 380
        __start 6C
```

You can use the **CORE_DUMP** message to confirm the program (executable) that produced the core file. The **CORE FILE NAME** section provides the location of the core file. The **PROGRAM NAME** section displays the name of the program that generated the core file.

The **ADDITIONAL INFORMATION** displays the sequence of calls that precedes the failing command, although the function names can be truncated.

It is possible to use just the information from the error report message to find the known issues if it is understood that the full function name might be different. In this example, the fifth call on the stack is **cp_SendDa**. which is the first nine characters of the full function name ***cp_SendData***.

If you cannot find the known issues that are based on the function names as they appear in the error report **CORE_DUMP** message, gather the core file and required associated files for analysis.



(1 of 2) Verifying the core file

- Check the location that is specified in the **CORE_DUMP** message from the **errpt -a** command
- Use the **find** command to search in the file system for a core file
- If no core file is found, check that the file system has enough space with the command
  df -k
- Verify that the IBM Tivoli Monitoring signal handler is disabled in the configuration (***xx*.ini**) file of the product, where *xx* is the component code

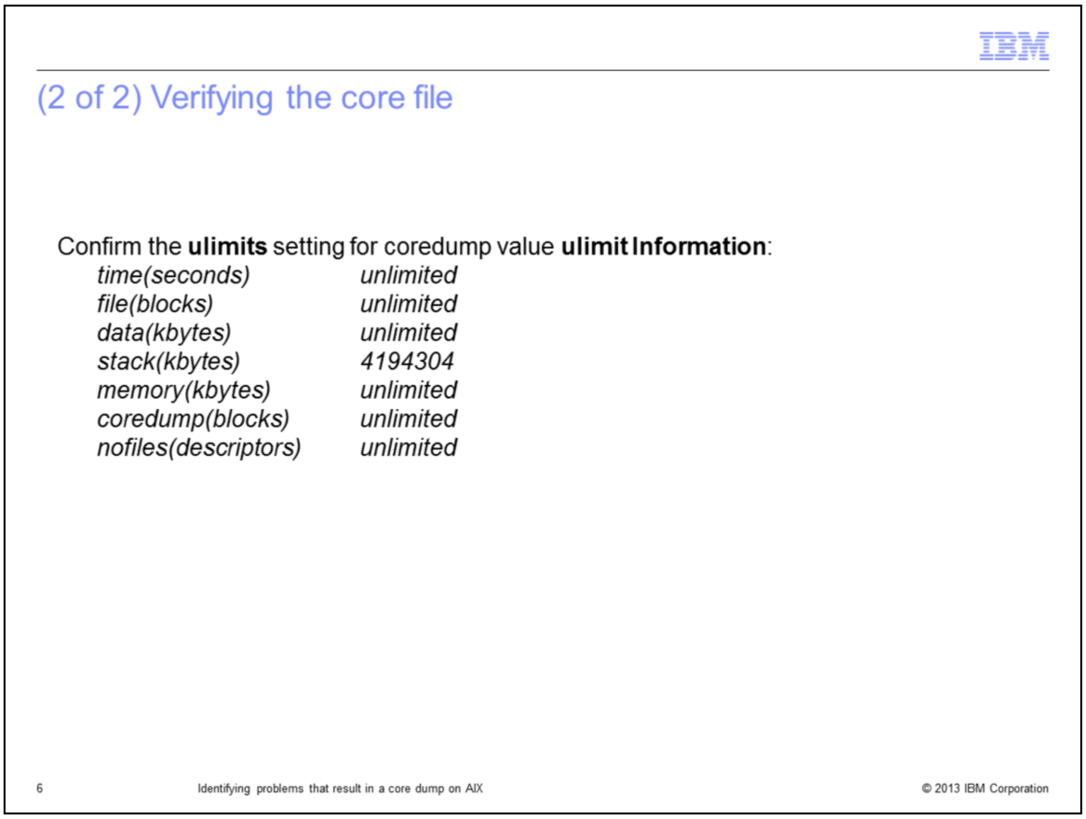
  KBB_SIG1=-dumpoff

  Note: For a UNIX® OS agent, the configuration file is ***<ITM install dir>*/config/ux.ini**

5 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

Verify the file system location from the error report message for the **CORE FILE NAME**. If no core file is present, on a UNIX system, you can run the find command to search for the core file in a different directory.

If you cannot locate the core file, verify that the file system has adequate space to write the core file. Also, check the **.ini** file for the Tivoli Monitoring component to confirm that the Tivoli Monitoring signal handler is disabled. The **KBB_SIG1** signal parameter must be set to **-dumpoff**.



(2 of 2) Verifying the core file

Confirm the **ulimits** setting for coredump value **ulimit Information**:

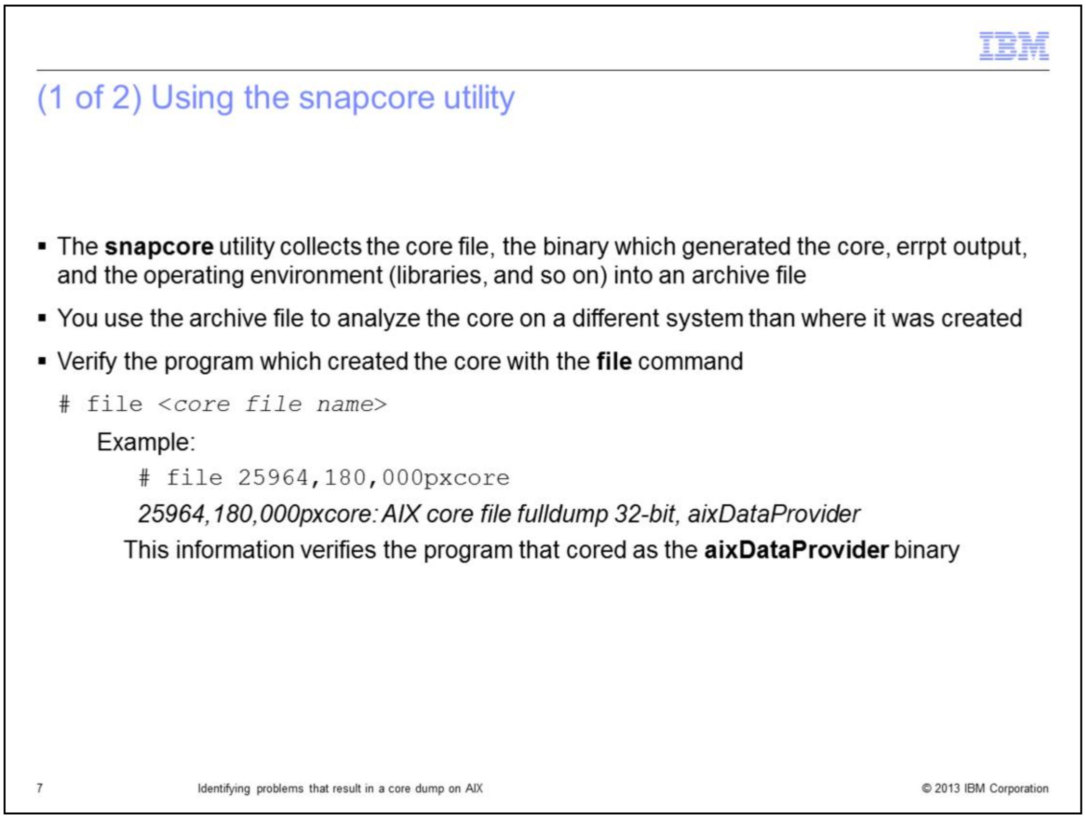
time(seconds)	*unlimited*
file(blocks)	*unlimited*
data(kbytes)	*unlimited*
stack(kbytes)	*4194304*
memory(kbytes)	*unlimited*
coredump(blocks)	*unlimited*
nofiles(descriptors)	*unlimited*

6 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

Check the ulimits setting for core dump.

It might be necessary to reproduce the issue after clearing space on the file system, or increasing coredump ulimit, or disabling the Tivoli Monitoring signal handler.

After the core file is located, use the AIX **snapcore** utility to gather the core and necessary files for core analysis into an archive package to provide to Support for analysis.



(1 of 2) Using the snapcore utility

- The **snapcore** utility collects the core file, the binary which generated the core, errpt output, and the operating environment (libraries, and so on) into an archive file
- You use the archive file to analyze the core on a different system than where it was created
- Verify the program which created the core with the **file** command

```
# file <core file name>
```

Example:

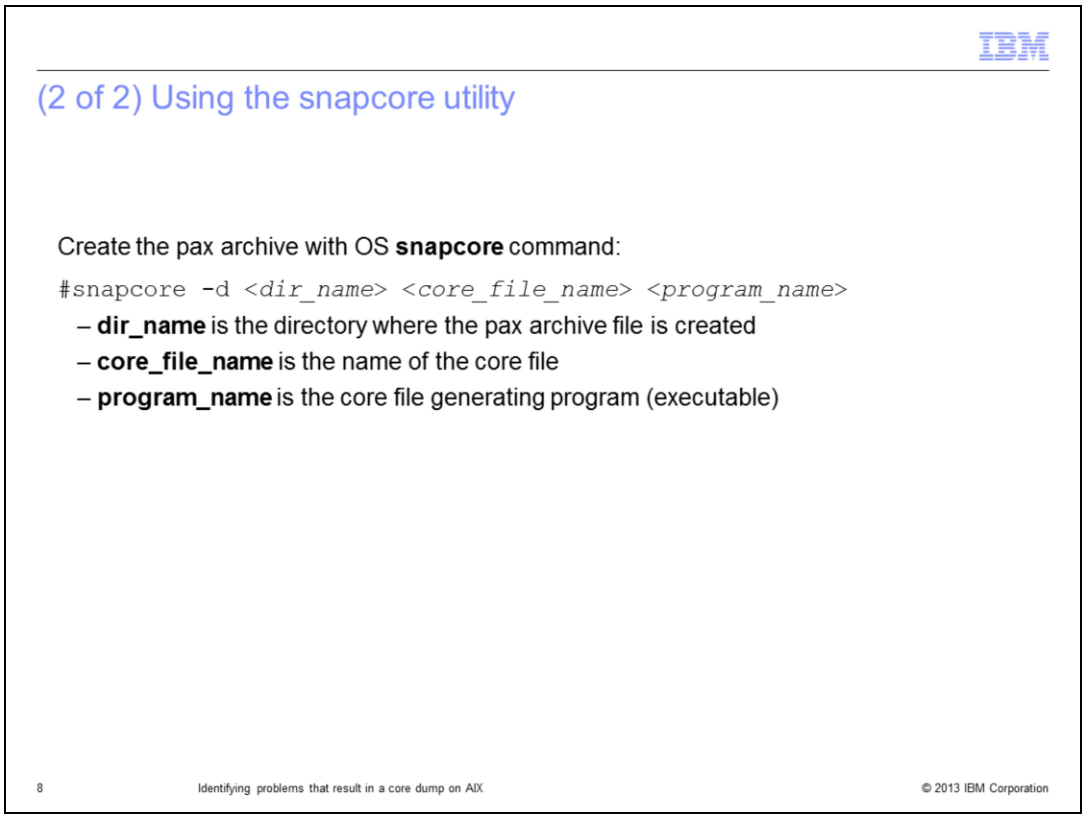
```
# file 25964,180,000pxcore
```

25964,180,000pxcore: AIX core file fulldump 32-bit, aixDataProvider

This information verifies the program that cored as the **aixDataProvider** binary

7 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

Run the file command against the core file to confirm the program that generated the core file. The program must match the program name from the error report message.



(2 of 2) Using the snapcore utility

Create the pax archive with OS **snapcore** command:

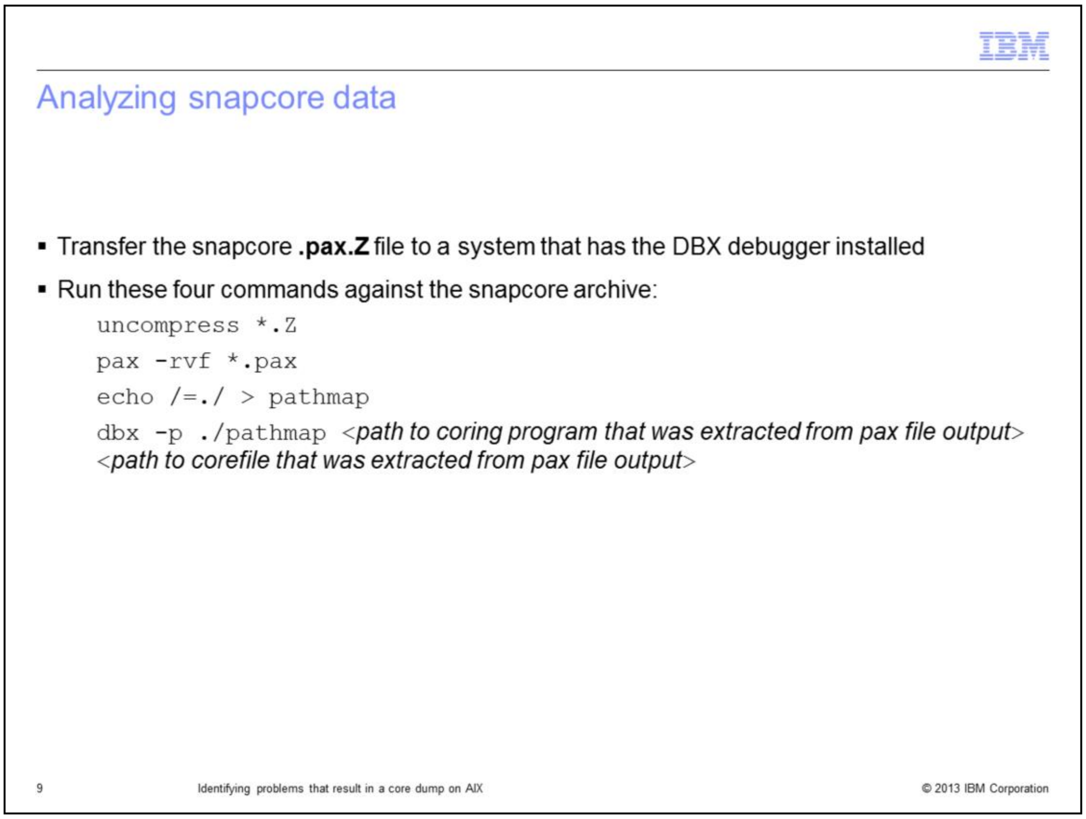
```
#snapcore -d <dir_name> <core_file_name> <program_name>
```

- **dir_name** is the directory where the pax archive file is created
- **core_file_name** is the name of the core file
- **program_name** is the core file generating program (executable)

Run the snapcore command against the core. Specify the output directory, the core file, and the program (executable) that generated the core file.

Use the fully qualified paths and file names for the parameters of the **snapcore** command.

The **snapcore** command creates an archive file with a **.pax** extension in the output directory.



Analyzing snapcore data

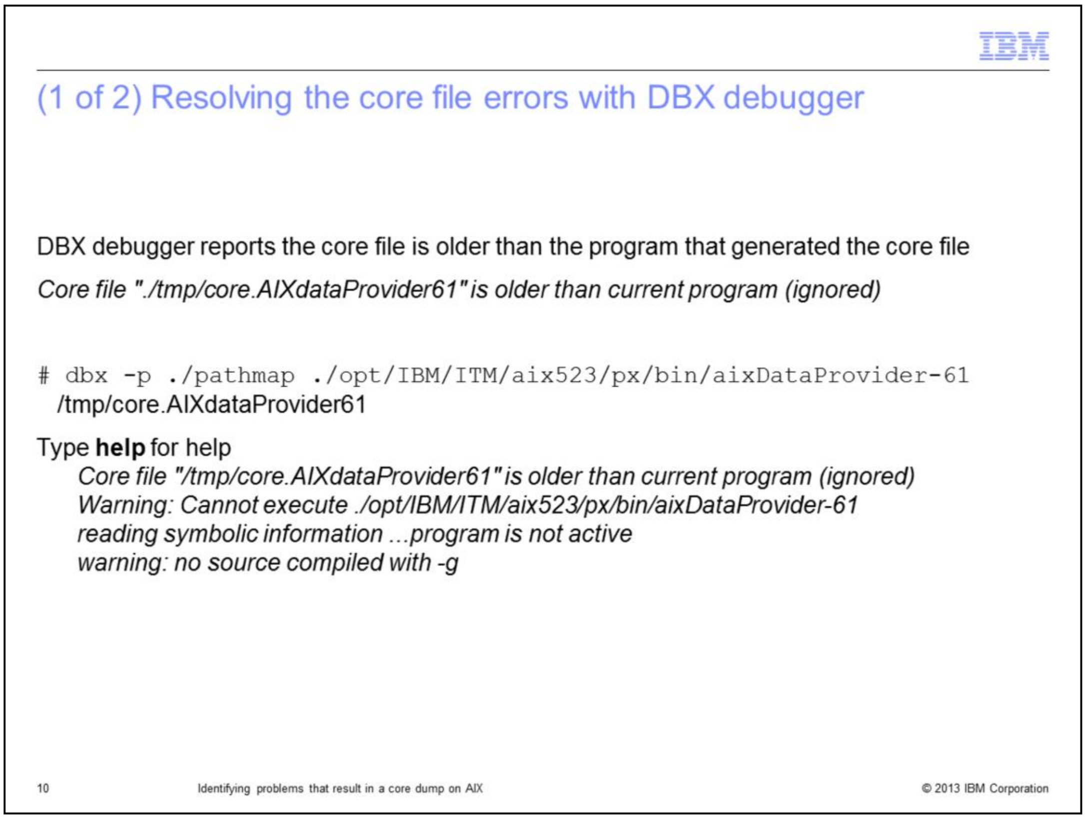
- Transfer the snapcore **.pax.Z** file to a system that has the DBX debugger installed
- Run these four commands against the snapcore archive:

```
uncompress *.Z
pax -rvf *.pax
echo /=./ > pathmap
dbx -p ./pathmap <path to coring program that was extracted from pax file output>
<path to corefile that was extracted from pax file output>
```

9 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

Copy the snapcore archive file to a system where a DBX debugger is installed.

Extract the files from the archive, and launch the DBX debugger against the core file.



(1 of 2) Resolving the core file errors with DBX debugger

DBX debugger reports the core file is older than the program that generated the core file

Core file "./tmp/core.AIXdataProvider61" is older than current program (ignored)

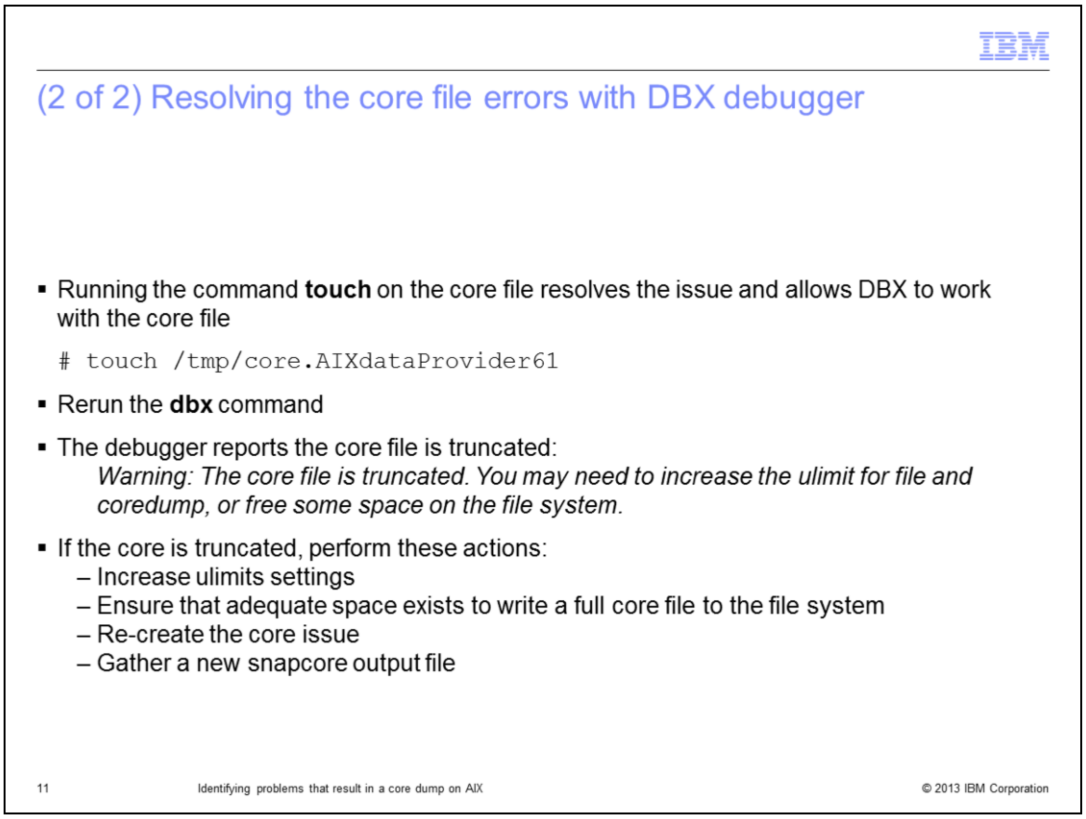
```
# dbx -p ./pathmap ./opt/IBM/ITM/aix523/px/bin/aixDataProvider-61
  /tmp/core.AIXdataProvider61
```

Type **help** for help

```
    Core file "/tmp/core.AIXdataProvider61" is older than current program (ignored)
    Warning: Cannot execute ./opt/IBM/ITM/aix523/px/bin/aixDataProvider-61
    reading symbolic information ...program is not active
    warning: no source compiled with -g
```

10 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

When you run the **dbx** command, it might generate the error message **"Core file "/tmp/core.AIXdataProvider61" is older than current program (ignored)."**



(2 of 2) Resolving the core file errors with DBX debugger

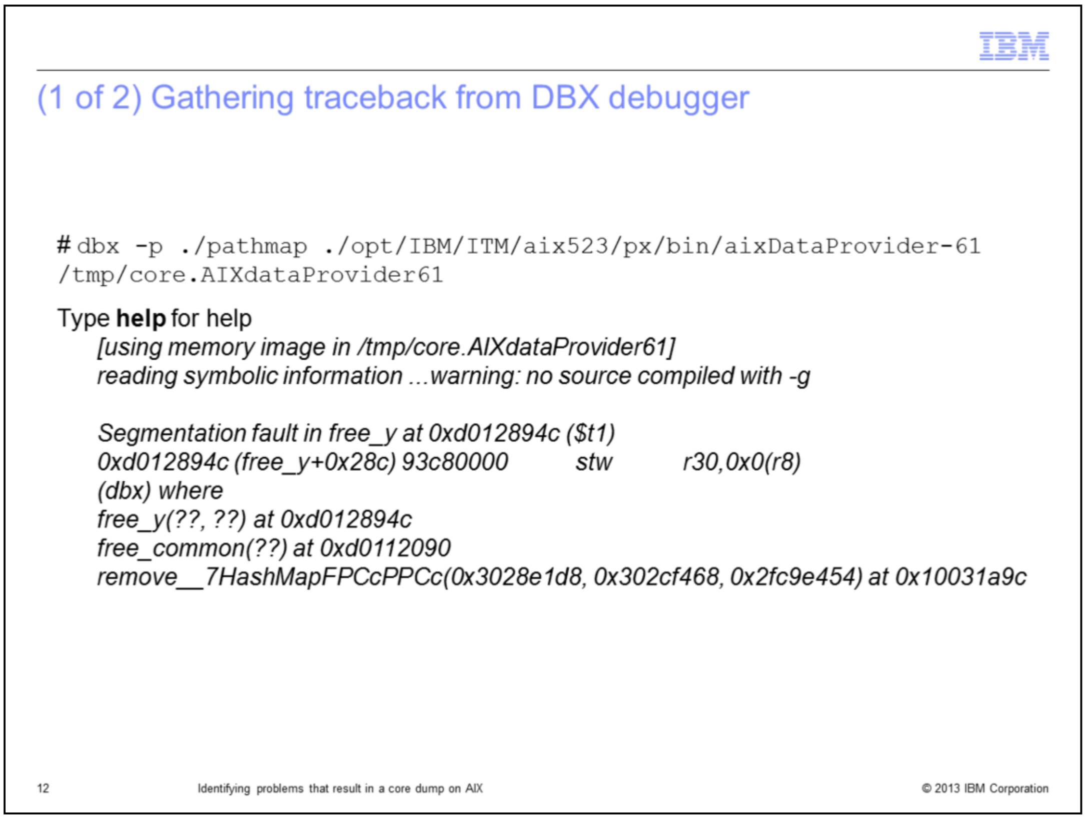
- Running the command **touch** on the core file resolves the issue and allows DBX to work with the core file

```
# touch /tmp/core.AIXdataProvider61
```

- Rerun the **dbx** command
- The debugger reports the core file is truncated:
  Warning: The core file is truncated. You may need to increase the ulimit for file and coredump, or free some space on the file system.
- If the core is truncated, perform these actions:
  – Increase ulimits settings
  – Ensure that adequate space exists to write a full core file to the file system
  – Re-create the core issue
  – Gather a new snapcore output file

11 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

If necessary, run the command **touch** on the core file to update the modification date of the core file. The newer modification date on the core file resolves problems where the debugger reports the core is older than the current program. If the core file is truncated, it might still be possible to gather **where output** to determine the traceback of called functions that lead to the core file. The detailed core analysis requires a full core. If the core is truncated, increase **ulimits** settings, and make sure that adequate space exists to write a full core file to the file system. Re-create the core issue and gather a new snapcore output file.



(1 of 2) Gathering traceback from DBX debugger

```
# dbx -p ./pathmap ./opt/IBM/ITM/aix523/px/bin/aixDataProvider-61
/tmp/core.AIXdataProvider61
```

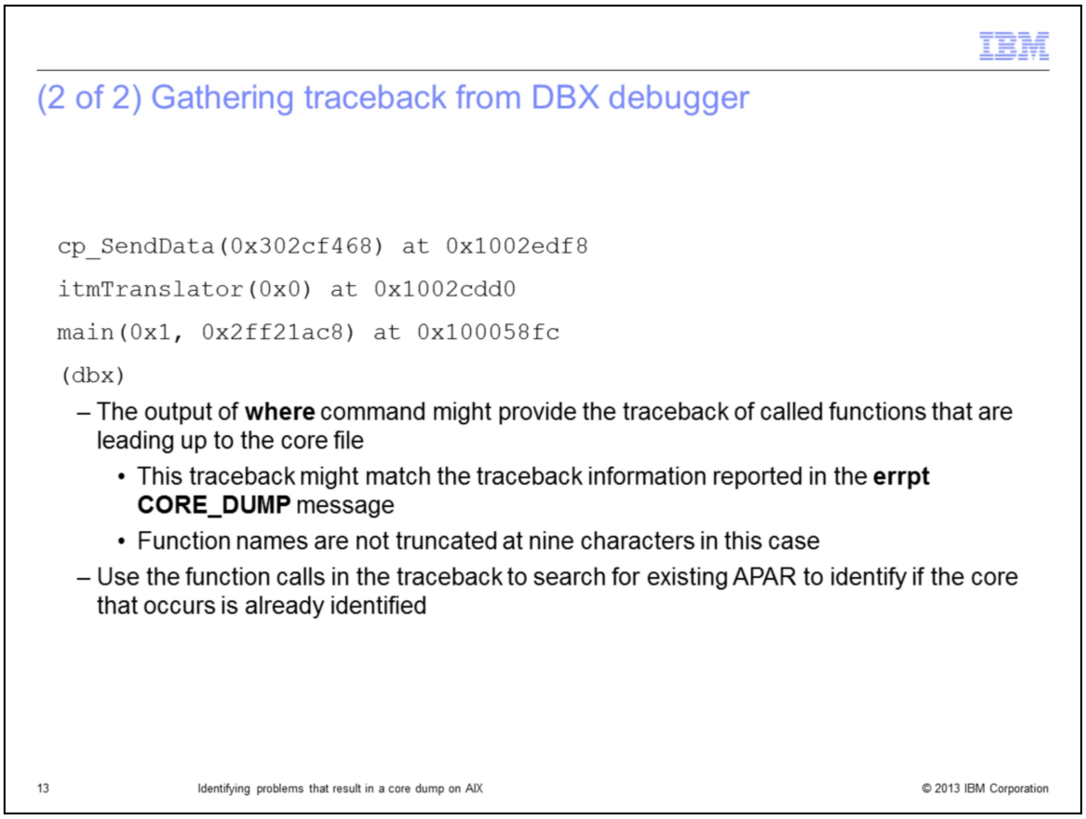
Type **help** for help

```
    [using memory image in /tmp/core.AIXdataProvider61]
    reading symbolic information ...warning: no source compiled with -g

    Segmentation fault in free_y at 0xd012894c ($t1)
    0xd012894c (free_y+0x28c) 93c80000        stw        r30,0x0(r8)
    (dbx) where
    free_y(??, ??) at 0xd012894c
    free_common(??) at 0xd0112090
    remove__7HashMapFPCcPPCc(0x3028e1d8, 0x302cf468, 0x2fc9e454) at 0x10031a9c
```

12 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

If the traceback from DBX debugger does not match the traceback in the error report, the core might be from a different occurrence of the failure.

Now, you might be able to identify the traceback for a core on AIX platform either from the error report for the operating system or with the DBX debugger.



(2 of 2) Gathering traceback from DBX debugger

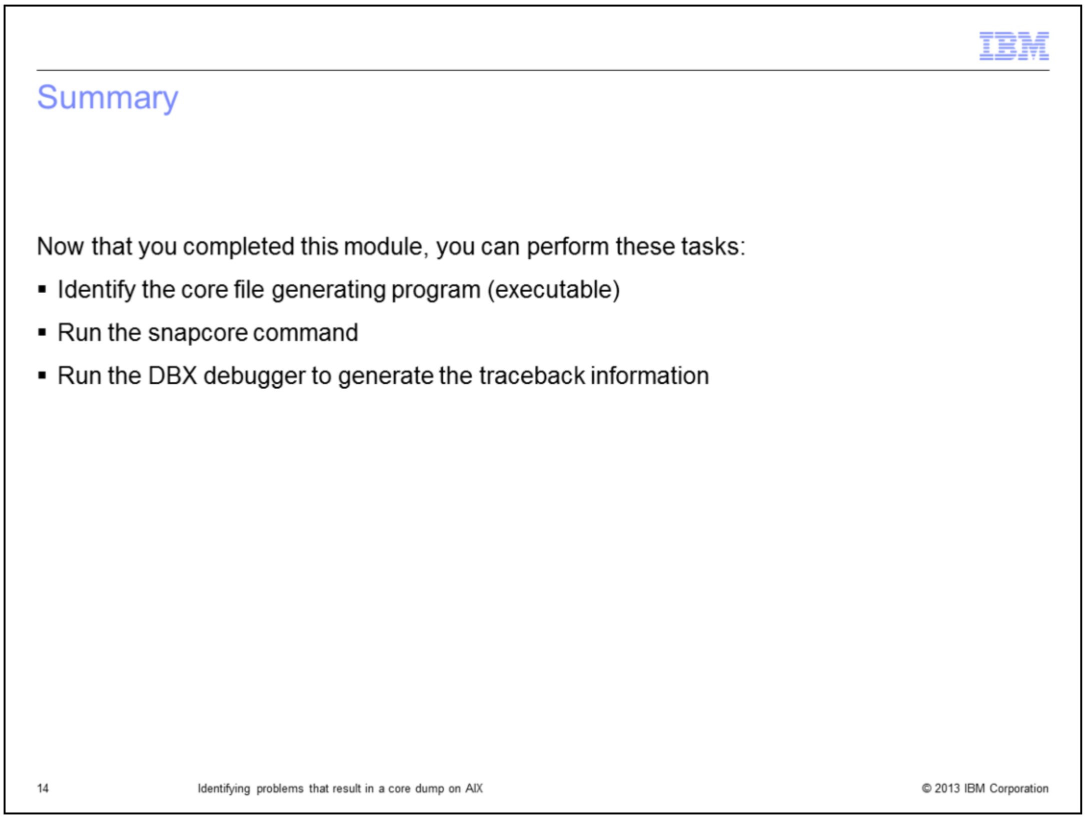
```
cp_SendData(0x302cf468) at 0x1002edf8
itmTranslator(0x0) at 0x1002cdd0
main(0x1, 0x2ff21ac8) at 0x100058fc
(dbx)
```

– The output of **where** command might provide the traceback of called functions that are leading up to the core file
  - This traceback might match the traceback information reported in the **errpt CORE_DUMP** message
  - Function names are not truncated at nine characters in this case

– Use the function calls in the traceback to search for existing APAR to identify if the core that occurs is already identified

13 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

If no matches are found based on the function calls leading up to the core, you can provide the snapcore output file to IBM Support for analysis.



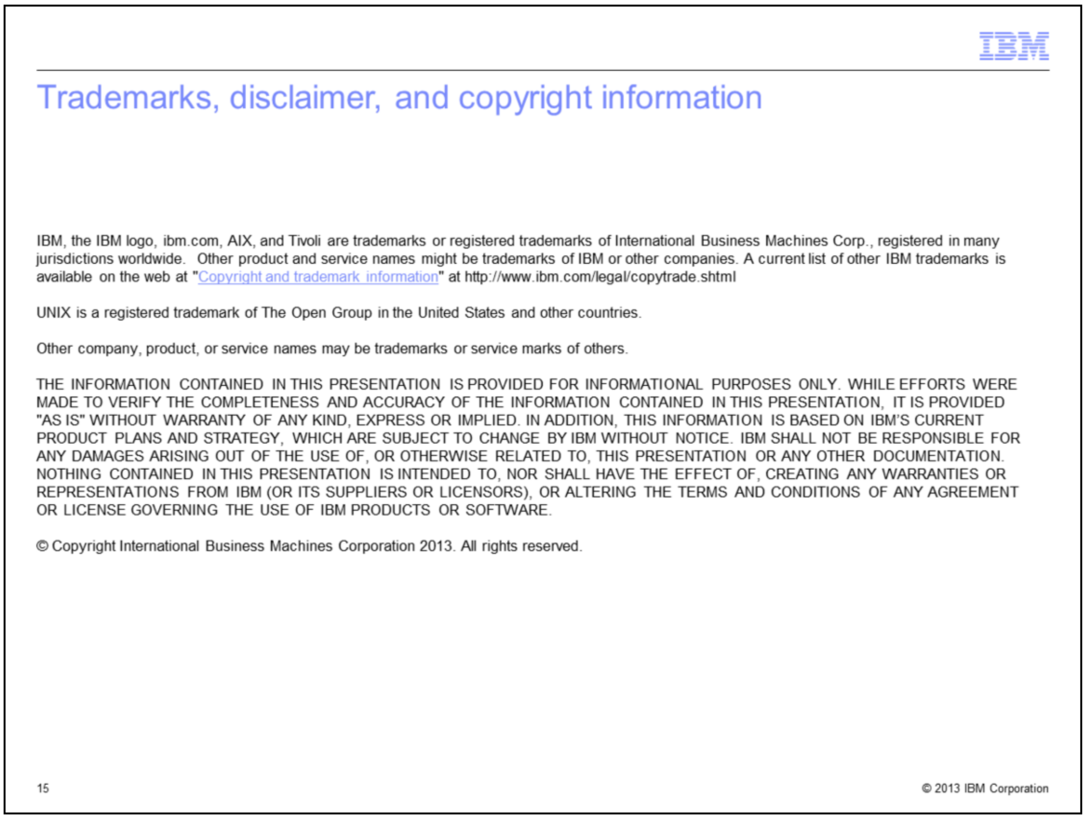
Summary

Now that you completed this module, you can perform these tasks:

- Identify the core file generating program (executable)
- Run the snapcore command
- Run the DBX debugger to generate the traceback information

14 Identifying problems that result in a core dump on AIX © 2013 IBM Corporation

Now that you completed this module, you can identify whether a program (executable) generated a core file. You can also gather necessary files for core analysis on AIX with the snapcore utility and review traceback information from the core to search for known problems.



Trademarks, disclaimer, and copyright information