41

AIX

March 1999

In this issue

update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 550955
From USA: 01144 1635 33823
E-mail: HarryLewis@compuserve.com

North American office

Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions

If you have anything original to say about AIX, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you actively be helping the free exchange of information, which benefits all AIX users, but you will also gain professional recognition for your expertise and that of your colleagues, as well as being paid a publication fee – Xephon pays at the rate of £170 ($250) per 1000 words for original material published in AIX Update.

To find out more about contributing an article, see *Notes for contributors* on Xephon's Web site, where you can download *Notes for contributors* in either text form or as an Adobe Acrobat file.

Editor

Harold Lewis

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, scripts, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £175.00 in the UK; $265.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 1995 issue, are available separately to subscribers for £15.00 ($22.50) each including postage.

AIX Update on-line

Code from *AIX Update* is available from Xephon's Web page at www.xephon.com (you'll need the user-id shown on your address label to access it).

Printed in England.

New features in AIX 4.3.2

IBM announced AIX 4.3.2 at the beginning of October 1998. This is an evolutionary release that contains numerous improvements in various fields of operating system support. The main areas that have been addressed are networking, logical volume management, system management, scalability, and performance. Additional improvements and updates were announced in AIX's graphics, NLS, and document search, plus a number of other miscellaneous features of the operating system. In this article I describe the improvements and enhancements at a level that enables users to determine the features that are most important to them. This should act as a guide to the ones to explore further using the IBM documentation, and may also help users to decide whether to upgrade to this version.

A word of caution, though: newly introduced features are the ones with the most bugs. The reader is, therefore, strongly advised to test features thoroughly in a non-production environment before deploying them in actual production systems.

NETWORKING

IPv6

Support for the IPv6 protocol was incorporated into AIX with release 4.3 (announced in October 1997). That release provides support only for the 'host-only' mode of operation, and no support is provided for routing and/or forwarding of IP packets between different interfaces co-existing on the same host. AIX 4.3.2 adds support for IPv6 routing with some limitations.

This support enables routing of unicast, multicast, and anycast packets as well as multi-homed link local and site local addresses (see *IPv6 – an overview*, *AIX Update* Issue 40, February 1999).

A new daemon called **ndpd-router** has been introduced to support IPv6 routing. Additionally, the commands **ndp-host**, **ndp**, **route**, **autoconf6**, **hostnew**, **ifconfig**, and **no** have been modified.

ifconfig accepts the following new flags:

-*U* Displays interfaces that are up.

-*m* When **-m** is followed by an interface name, it displays all media types supported by the interface.

-*d* Displays interfaces that are down.

-*a* Displays information about all interfaces on the system.

no accepts the following new flags:

- *ip6forwarding*
  If set to '1', this enables forwarding of packets; the default value, '0', prevents the forwarding of packets.
- *multi_homed*
  Specifies the level of multi-homed IPv6 host support.
- *main_if6*
  Specifies the interface to use for link-local addresses. Used by **autoconf6** to set up initial routes.
- *main_site6*
  Specifies the interface to use for site-local addresses. Used only if **multi_homed** is set to '3'.
- *site6_index*
  Specifies maximum interface number for site local routing.

Another important change is support for **gated** version 3.5.9. This daemon enables support for gateway routing for the RIPng, EGP, BGP+ (versions 1, 2, 3, and 4), HELLO, OSPF, and SNMP protocols.

LDAP V2.1

Version 2.1 of Lightweight Directory Access Protocol provides many improvements over Version 1.1.1, which shipped with AIX 4.3.1. The protocol uses a DB2 V5.0 database for data storage (the version of DB2 provided with AIX can be used only by LDAP). By utilizing DB2's advanced features, LDAP is able to offer true multi-threaded operation for multiple concurrent LDAP clients efficiently. This

version of LDAP is based on RFC 1777, 1778, 1779 (LDAP Version 2), plus IETF LDAP Version 3 extensions.

LDAP Version 2 support has been enhanced to support aliases. Support for ACLs has been updated to enable role-based authorization and assignment of multiple user ownership for a single entry, and to remove the requirement that every node's ACL is set in order to permit users access to their own information.

The following Web servers are now supported by LDAP on AIX: Apache, Lotus Domino Go, Netscape FastTrack, and Netscape Enterprise.

LDAP supports clients using either the LDAP or HTTP protocols to access directory information. Both LDAP Version 2 and 3 are supported. The Java Naming and Directory Interface (JNDI) is included with LDAP, allowing the development of LDAP-enabled Java applications. In addition to the development of AIX-based clients, IBM provides the LDAP Client Pack (which can be ordered as PRPQ 5799-GAN) for the development of LDAP clients for Windows NT, Windows 95, Solaris, and HP-UX.

A sample directory named *Directory Sample1* ships with LDAP to allow the testing of LDAP client applications.

LDAP V2.1's use of DB2, and its improvements in replication and performance, enable it to support up to 15,000,000 entries and still offer peak sub-second response time for searches.

BIND DNS (NAMED) V8.1.2

BIND (Berkeley Internet Name Daemon) has been updated to version 8.1.2, which provides the following enhancements:

- Secure dynamic DNS updates, which address security problems stemming from RFC 2136 and allow the secure distribution of DNS data from primary to secondary servers.
- The new incremental zone transfer method enables zone data located on secondary DNS servers to be updated by the transfer of cumulative changes since the last transfer from the primary DNS server. This feature is described in RFC 1995.

- The implementation of RFC 1996 enables users to configure a method by which the primary DNS server can inform secondary servers that zone data has been changed. This feature decreases the period during which secondary DNS servers hold data that's out-of-sync with the primary DNS server.
- A file conversion utility that enables the mapping of dynamic keywords from the *named.boot* file to their functionally equivalent ones in the *named.conf* file.
- Incorporation of **named4** and **named8** functions into single **named** daemon that provides the combined functionality.

RDIST V6.1.3

The **rdist** (remote host file distribution) command has been updated to version 6.13, providing the following new functions:

- The ability to update multiple hosts in parallel. This feature improves update times when a large number of hosts are involved in an update operation. The feature can be disabled by the user or limited to a maximum number of hosts that may be updated in parallel.
- It is now possible to check for the availability of enough free space on the target host before executing a file transfer.
- Improved error handling comes via the ability to skip hosts that don't respond to transfers within a specified interval.
- Improved security comes as a result of the separation of functionality into distinct client and server parts: **rdist** and **rdistd**. Only **rdist** is run as 'set-uid' program.
- To enable compatibility with legacy systems, the old version of the utility is shipped as **/usr/bin/oldrdist**, and is invoked if **rdist** is invoked with **-Server** option.

Network Buffer Cache and send_file system call

In order to improve the performance of network-intensive applications, such as file and Web servers, AIX 4.3.2 implements an in-kernel

Network Buffer Cache (NBC) that enables the temporary storage of recently transferred data in memory. This cache is utilized by the **send_file** system call, which caches files recently sent over the network, re-sending them if the same data is requested again. Buffers for the cache are allocated from the network memory allocator, which has been expanded to 1 GB in AIX 4.3.2. This saves I/O and CPU cycles, and increases the server's network performance.

Below is the synopsis of the use of **send_file**.

```
#include <sys/socket.h>

ssize_t send_file(Socket_p, sf_iobuf, flags)

int *Socket_p;
struct sf_parms *sf_iobuf;
uint_t flags;
```

The size of the NBC and the values of various cache-tuning parameters can be altered using the **no** command:

- *nbc_limit*
  Sets the maximum size of NBC in kilobytes.
- *nbc_max_cache*
  Sets the maximum size of a cache object in NBC in bytes. The default is 131,072 bytes (128 KB).
- *nbc_min_cache*
  Sets the minimum size of a cache object in NBC in bytes. The default is 1 byte.
- *send_file_duration*
  Specifies the time during which cached file objects accessed in NBC using the **send_file** system call remain validated. This attribute is expressed as a number of seconds, the default being 300 seconds or five minutes. A value of '0' means that the cache is validated on every access.

NBC utilization statistics may displayed using **netstat -c** command.

The **send_file** call supports only the TCP/IP protocol (sockets of type SOCK_STREAM in address family AF_INET).

TCP checksum offload on ATM 155 Mbps PCI adapter

The device driver for the ATM 155 PCI Adapter has been modified to offload TCP checksum processing from the AIX TCP/IP protocol stack to the adapter hardware. This reduces the number of CPU cycles needed to process network packets, improving overall performance.

The command **ifconfig** has been updated to support following options to support this function:

- *checksum_offload*
  This enables checksum offload, if the adapter associated with the interface supports it. It's enabled by default for adapters that support the feature.
- *-checksum_offload*
  This disables checksum offload.

This feature doesn't work with IPv6, UDP datagrams, or when IPSEC is enabled.

Improved device driver for IBM 10/100 Mbps Ethernet adapters

The device driver for the IBM 10/100 Mpbs PCI adapter has been improved to support AUI and BNC ports, in addition to the previously available TP connector. Performance has also been improved by reducing the number of data copy operations required. The device driver supports following network speeds:

- 10 (10 Mbps, half-duplex)
- 20 (10 Mbps, full-duplex)
- 100 (100 Mbps, half-duplex)
- 200 (100 Mbps, full-duplex)
- Auto negotiate (the default state).

The device driver is backward-compatible with AIX 4.2 and is available in the following software packages:

- *Devices.pci.23100020.diag*
- *Devices.pci.23100020.rte*.

SDLC/BSC support for the IBM 4-port PCI adapter

This adds support for SDLC and BSC protocols to the 4-port PCI adapter (ARTIC960Hx). This support is necessary as the newly announced PCI-based RS/6000s are unable to use previously available ISA cards that were used with these protocols.

Each of the adapter ports is able to support one of the following physical interfaces: V.24, V.35, V.36, and X.21.

The support is available via the following PTFs:

- *IX81860* for AIX 4.3.2
- *IX81861* for AIX 4.2.1

Miscellaneous networking enhancements

- AIX 4.3.2 reduces the number of network packets that are exchanged between Web servers and clients. This is done by delaying certain ACK messages and piggy-backing them with the next packet that is sent. Two new flags have been introduced for **no** command to support this networking property: **delayack** and **delayackports**.
- The locking of incoming and outgoing packets by the networking code has been optimized in order to shorten the time during which the lock is held.

LOGICAL VOLUME MANAGER

Support for big Volume Groups

AIX 4.3.2 introduces a new volume group format that permits the inclusion of up to 128 physical volumes (PVs) in a single VG. The number of Logical Volumes (LVs) that can be included in VGs that conform with this new format is 512. Previous versions of AIX supported only up to 32 PVs in a single VG, which could comprise up to 256 LVs. Support is provided for up to 1,024 PVs and 1,024 LVs in a single VG. VGs created by previous versions of AIX are fully supported and conversion aids are provided to change LVs from old to new ‘big’ format.

The list below summarizes the changes made to various VG management commands in order to support the new format.

mkvg command:

- **-B** flag
  Directs **mkvg** to create a big VG.
- **-G** flag
  Directs **mkvg** to create a big VG and allocate enough space to allow the inclusion of up to 1,024 PVs and LVs when this is supported in a future release of AIX.

chvg command:

- **-B** flag
  Directs **chvg** to convert a small VG to a big VG.

importvg command:

- **-R** flag
  Directs **importvg** to restore the user ID, group ID, and permissions of imported LV (applies to big VGs only).

mklv and **chlv** commands:

- **-U** flag
  Specifies the user ID for logical volume special file (applies to big VGs only).
- **-G** flag
  Specifies the group ID for logical volume special file (applies to big VGs only).
- **-P** flag
  Specifies permissions for logical volume special file (applies to big VGs only).

All new options described above must be entered from the command line, as no **smit** or **wsm** support for them is available.

The following limitations apply to the use of big VGs:

- *rootvg* cannot be converted to big VG format.

- Concurrent access to big VGs is not supported.
- Big VGs are not supported on previous releases of AIX.

Enhanced support for on-line mirror back up

On-line mirror back up for LVM was introduced in AIX 4.3.1. This feature improves system availability by supporting on-line back up without interrupting access to the original data. Originally, the feature was supported only on a single computer and didn't allow users to access the back up mirror as a filesystem (all access to the back-up copy had to be through special interfaces). The following enhancements were introduced in AIX 4.3.2:

- Concurrent access for multiple computers using the concurrent mode of IBM's HACMP software.
- Support for filesystem access to a back-up mirror by standard commands and APIs.

The command **chlvcopy**, which marks or unmarks a mirror copy of a LV as an on-line back-up, has been updated to support this new feature. The following flags have been added:

-P Directs the system to maintain information about the existence of an on-line back-up copy across reboots and informs additional machines (in a concurrent-mode HACMP environment).

-l This flag is followed by the name of the back-up logical volume. If missing, it's provided by the system.

-w Allows a back-up copy to be writable (the default is read-only).

Below are the limitations of the current implementation of on-line mirror back up:

- LVs cannot be removed when an on-line back-up copy exists.
- No changes to the original LV structure or its attributes are allowed when an on-line back-up exists.
- All partitions of a logical volume must be fresh before you can mark an on-line copy as an on-line back-up.

- Only one copy can be designated as an on-line back-up copy.

Note also that this facility is currently documented only in the manual pages.

SYSTEM MANAGEMENT AND UTILITIES

Deferred paging space allocation

Earlier versions of AIX used, by default, a so-called ‘late allocation’ policy for managing paging space. Essentially this meant that a disk block used for paging was allocated only when the page was actually accessed by the program, and not when the program initially allocated the page. AIX 4.3.2 improves late allocation by deferring allocation of the disk block until a ‘page out’ is necessary. This ‘deferred late allocation’ policy decreases paging space requirements of systems that have large physical memory.

TTY remote reboot

AIX 4.3.2 adds an option that allows users safely to reboot a machine that has stopped responding to the network, but is still able to process device interrupts. This feature allows system administrators to instruct the machine to perform a reboot or a system dump if a certain string is entered on a device connected to the system’s native serial port.

New printer support

AIX 4.3.2 adds support for the following types of printer:

- Lexmark Optra 1200 colour printer
- Lexmark Optra 40 colour printer
- Lexmark Optra 45 colour printer
- Lexmark Optra K 1220 laser printer
- Hewlett-Packard Laserjet 8000
- Hewlett-Packard Colour Laserjet 8500
- IBM InfoPrint 32.

Print job administration

AIX 4.3.2 allows the management of up to 999,999 print jobs per print queue. This ability is relevant only to local queues, as the LPD protocol supports the management of only up to 999 print jobs per print queue.

NIM improvements

AIX 4.3.2 improves the robustness of NIM (Network Installation Manager) by limiting the duration of locks set on various resources to no more than is necessary given the duration of the critical path of an operation. This allows faster completion of various NIM actions.

Additionally, when simultaneous tasks are performed on a number of objects, the system manager has the ability to limit the size of a group on which an operation is to act in parallel and to define a timeout value that, when expired, causes the termination of the NIM operation on the group.

Another enhancement is that support for ATM networks has been added to NIM.

Web-based System Manager security enhancements

Remote administration support using the WSM has been made secure by the incorporation of the Secured Socket Layer (SSL) protocol. This protocol encrypts all data exchanged between the managed machine and the remote client, preventing unauthorized observation of the data.

The software required to implement this feature is included in the AIX Bonus Pack in the *sysmgt.websm.security* package. Users in Canada and the USA may opt for strong encryption via the *sysmgt.websm.security-us* package.

The configuration and use of the SSL protocol requires the generation and distribution of public and private encryption keys for each managed machine. These keys can be obtained from an external Certificate Authority or generated by the WSM server for use in the Intranet. For maximum security, clients should contact a WSM-managed machine using the HTTPS protocol. To enable this option,

the HTTP server that runs on the managed machine (such as the Lotus Go Web server, which is supplied in the Bonus Pack) must be configured to support HTTPS protocol.

Web-based System Manager diagnostic enhancements

A new application has been added to the WSM allowing the system administrator to format and certify certain hard disk types and read-write optical media. It is also possible to perform Error Log Analysis and diagnostic and service aid functions on devices that support these functions using WSM.

Web Based System Manager application registration

This function allows system administrators to register any application that can be run over an intranet or the Internet as a WSM application. The application can be defined for a single remote host or a group of hosts.

System back up and installation enhancements

The **makesysb** and **savevg** commands have been enhanced to store information about the block size of the tape device with the back-up image. This new option has been added for commands that are used to list and restore the back-up image, thus allowing the retrieval of block size information, and also allowing the block size of devices to be set to match the value that is stored with the back-up image. This greatly improves the time it takes to restore or list files on image tapes.

The 'non-prompted' install method used with a custom *bosints.data* file has been enhanced with the addition of the *EXISTING_SYSTEM_OVERWRITE* variable. If this variable is set to ' ', any disk may be used for the system install; if it's set to 'no', only disks containing no VGs can be used; and if it's set to 'yes', only disks in current *rootvg*, or those containing no volume groups, may be used.

New service aids

The System Memory Exerciser can be used to test memory on CHRP machines. This program is provided as a service aid, and can, therefore, be run only in service or maintenance mode.

Diagnostic Error Log analysis now supports the newly announced Advanced I/O Drawer for the S7A.

Alternate disk installation enhancements

The alternate disk installation method, which was introduced in AIX 4.3, is intended to allow the maintenance of multiple bootable AIX releases and/or maintenance levels on-line. A full discussion of the operation of this method is beyond the scope of this article, so I'll just list the new options of the **alt_disk_install** command that are available in AIX 4.3.2:

- **alt_disk_install –q <disk>**
  Used to establish the boot disk of the VG.
- **alt_disk_install –v <new VG name> <disk>**
  Used to rename the alternate disk VG.
- **alt_disk_install –W <disk>**
  Used to wake up the VG.
- **alt_disk_install –S**
  Used to put the VG to sleep.
- **alt_disk_install –X [<VG>]**
  Used to clean up the alternate VG.

SCALABILITY AND PERFORMANCE ENHANCEMENTS

Increased system limits in IPC subsystem

The Inter-Process Communication subsystem has been enhanced to support significantly more usage per system. The number of message queues, semaphores, and shared memory regions have all been increased from 4,096 at AIX 4.3.1 to 131,072 on AIX 4.3.2.

IPC supporting commands, such as **ipcrm** and **ipcs**, have been updated to support the new limits.

Kernel scalability enhancements

AIX 4.3.2 is able to support up to 32 GB of physical memory on 64-

bit RS/6000 S70 and S7A SMP servers. A dedicated memory segment has been added for the kernel heap, doubling the size of the heap from 256 MB to 512 MB. Also, the maximum size of the MBUF pool has increased, and is now 1 GB on CHRP hardware and 256 MB on other types of hardware.

The pipe buffer is now allocated between 16 and 64 MB of memory, significantly increasing the number of pipes that may be opened simultaneously.

NFS server performance has also been significantly enhanced by the implementation of a 'vnode' cache in the JFS component of the kernel. The cache enables NFS server code to translate an NFS file handle to a local vnode structure more efficiently.

Improvement of nice and schedtune commands

The scheduler component of AIX 4.3.2 treats **nice** values as directives and not suggestions when calculating 'niced' process priorities. This causes processes that are run in the background or are invoked with increased **nice** values (either with the **nice** or **renice** command, the **nice( )** library routine, or the **renice( )** system call) to consume significantly less CPU if they run on a heavily CPU-loaded system.

Improvement SRC subsystem

In previous versions of AIX, when the **srcmstr** daemon fails and is re-spawned, the new invocation is unable to manage daemons that were started by its previous invocation. The **srcmstr** daemon in AIX 4.3.2 records all successfully started subsystems in an external file, which may then be read by subsequent invocations of the daemon. This modification enables a newly spawned instance to control daemons invoked by the previous one.

Thread suspend/resume

A new 'pthread' state, the *suspend* state, has been added to the pthread data structure. Programmers are now able to suspend pthreads that are running and resume their execution later using two new pthread library routines (**pthread_suspend_np( )** and **pthread_continue_ np( )**).

The routines **pthread_attr_setsuspendstate_np( )** and **pthread_ attr_getsuspendstate_np( )** are used to set and get the 'suspendstate' attribute of a pthread.

Thread-safe libsrc library

In earlier versions of AIX, *libsrc.a* library subroutines were neither thread-safe nor re-entrant. This is fixed in AIX 4.3.2, so that routines that didn't require interface changes to become thread-safe and re-entrant are now re-entrant, while thread-safe versions of all other routines have been added to the *<previos-name>_r* library.

Additional enhancements

A new *libpthdebug.a* library has been added to allow the debugger application to access the internal structures of the pthreads library.

The binder library *libld.a* has been updated to support the creation of applications to manipulate both 32-bit and 64-bit objects in a transparent manner, without the need to know beforehand the format of the object.

Another enhancement is that DHCP server has been re-implemented as a multithreaded application to enable more scalable performance.

Additional information is included in the core files to enable more efficient debugging of shared memory errors.

MISCELLANEOUS IMPROVEMENTS

OpenGL enhancements

OpenGL version 1.2, released in March 1998, is the second revision since the original Version 1.2. Several extensions have been made to the graphics library, among them:

- *MultiDraw Array Extension*
  This enables users to group together multiple primitives and send them to the adapter with just one call.
- *Texture Mirrored Repeat Extension*
  This enables users to specify texture maps that have no discontinuities at the edges.

- *Colour Blend Extension*
  This enables the user to create blended and/or translucent colours without using an alpha buffer.

OpenGL 1.2 support is available in AIX 4.3.2 only for the GXT3000P PCI Graphics Accelerator. Users of other graphics adapters are limited to functions contained in OpenGL 1.1 and OpenGL 1.1 Extensions. The optional Imaging Extension Subset is not supported by the IBM OpenGL 1.2 implementation at this time.

GraPHIGS enhancements

Performance of polygon rendering and throughput to the graphics adapter has been improved. Support for the euro currency symbol has been added to font support of GraPHIGS.

National language support enhancements

Universal Code Character Set (UCS) versions of locales that were previously available for AIX in other formats are now available. This enables improved support for the requirements of languages worldwide.

Japanese Code Page 943 is supplied to enable interoperability with Microsoft Windows Clients in Japan, and TrueType font support has been added to the Korean locale (*ko_KR* or *IBM-eucKR*).

The standard AIX messaging facility for error logging and reporting has been enhanced to allow the programmer to create applications that are able to log extended error messages (up to 1,000 bytes) in any supported NLS.

The AIX Documentation Search Service is extended to support the searching of documentation stored in Double Byte Character Set (DBCS). The following codesets are supported:

Language	CCSID	Codeset	Locale
Japanese	932	IBM-932	Ja_JP
Korea	970	IBM-eucKR	ko_KR
Simplified Chinese	1383	IBM-eucCN	zh_CN
Traditional Chinese	950	Big5	Zh_CN

To enable the documentation to be displayed properly, the browser must be able to support the codeset being searched. Only a single language may be searched at a time.

Euro currency symbol support

The euro has been adopted by the European Monetary Union (EMU) as common currency to be adopted by all EMU members. Its usage in banking and industry began in January 1999. The EMU plans to replace the existing national currencies completely by the euro by the year 2002. AIX support for the euro currency symbol is based on the new Unicode locales. Both the existing national currency symbols and the euro currency symbol are available. The following Unicode locales have euro currency support:

Language	Locale
Catalan	CA_ES
Dutch(Belgium)	NL_BE
Dutch	NL_NL
Finnish	FI_FI
French(Belgium)	FR_BE
French	FR_FR
German	DE_DE
Italian	IT_IT
Portuguese	PT_PT
Spanish	ES_ES

SUMMARY

My admittedly limited experience of AIX 4.3.2 (given it was released only recently) is that it operates more efficiently than previously available versions. It's a solid base for enterprise-class computing and should continue to evolve to support computing well into the 21st century.

REFERENCES

1 *AIX Version 4.3 Differences Guide*, SG24-2014-01, IBM.

A Polak
System Engineer
APS (Israel)

AIX dump devices

In common with a few other aspects of a well-tuned AIX environment, the dump device is something that's carefully set up in case something you hope will never happen happens. It's fairly rare to have to deal with dumps, which means that knowledge of them belongs in the same category as what to do when other disasters strike your system. In other words, it's not everyday knowledge.

DUMP DEVICES

A dump is your operating system's famous last words as it experiences a system crash and fails. The term 'dump' (which is also used for back-up data) is meant to convey the sense of taking the system's unstructured data and storing it somewhere as quickly as possible. A dump is the operating system's attempt to provide the administrator with as much information as possible about the state of the system at that moment when severe trouble was encountered and other means of recovery and self-healing failed.

What happens is that part of the main memory is copied to non-volatile storage (disk) so that the administrator can use other tools later on to pin down the cause of the operating system crash that initiated the dump. So, if AIX is unable to go on, it will try to give you information on what happened to allow you to identify a misbehaving application or bug in the operating system.

As main memory can contain a large amount of data (today's RS/6000 systems, such as S70s, typically have 16 gigabytes) – most of it not related to the problem that caused the system to crash – a dump normally doesn't include all data, including only the so-called 'working set' (the part of memory that was actually 'in use' the moment the crash occurred) and some general system information, such as file, process, and user tables.

While it may seem obvious to some, it's probably worth pointing out that, if the system throws a dump, this in itself doesn't cure anything and you still have to reboot the system that crashed.

DUMP DEVICES

In order to make the dump available to the administrator, it has to be stored on a disk in what's known as a 'dump device'. In the different versions of AIX, there are different default policies on how to provide a default dump device to the operating system.

In AIX Version 3, the operating system's installation process created a dedicated logical volume in Volume Group *rootvg* called *hd7*. *hd7* was normally located at the edge of disk and had the type 'dump' specified. The only thing the administrator had to worry about was increasing the size of the device to keep it in line with the size of the main memory and the way the system was used.

AIX Version 4 changed that: in order to conserve disk space, especially on newer systems with up to 16 gigabytes RAM (and still more to come!), the default dump device was no longer a dedicated logical volume but the paging space, logical volume *hd6*.

This dual-use idea works fine in most cases; during normal operation, *hd6* is used as paging space. In case of a system crash, AIX uses *hd6* as a dump device, overwriting paging space information that's not needed under those circumstances. *hd6* also has the advantage that, in most cases, paging space is at least the same size as main memory, so there is no doubt that the dump device is large enough to hold the dump information.

On the other hand, using *hd6* as the dump device has one major implication – when you try to reboot after a system crash, AIX knows that there is a dump stored in *hd6*. In order to boot the system, *hd6* must be cleared so it can be used as a paging space again. So, during the boot phase, the system tries to copy the dump to the filesystem in order to free the paging space. After successfully rebooting you'll find the dump image as *vmcore.0* (or a higher number, if more than one dump is stored in your system) in the directory */var/adm/ras*. If there's not enough space in the filesystem, the system halts and offers you a menu where you can copy the dump to other devices – for instance, a tape drive.

To avoid the problem of the system not coming up because the dump has to be saved somewhere, we recommend you create a dedicated

dump device in AIX Version 4 to mirror the one we had back in AIX Version 3. Another option is to tell the system that it should continue booting even if the dump cannot be copied to the filesystem – the disadvantage is that you'll lose information that could be worthwhile for troubleshooting the system.

CREATING A DUMP DEVICE

Let's start by displaying information about the dump device's current settings using the command **sysdumpdev -l**:

```
# sysdumpdev -l
primary              /dev/hd6
secondary            /dev/sysdumpnull
copy directory       /var/adm/ras
forced copy flag     TRUE
always allow dump    FALSE
```

As you can see, the primary dump device is by default set to *hd6*, which is also the paging space. If the primary dump device becomes unavailable, the system tries to dump to */dev/sysdumpnull* which is a kind of */dev/null* (bit bucket) for system dumps. The copy directory specifies where to copy the dump during the reboot – it's set to */var/adm/ras*. The 'forced copy' flag indicates whether copying the dump is mandatory, so that the system prompts you during reboot if the data can't be saved to the specified copy directory. If not, the system continues booting even if the dump was not saved to the filesystem. The 'always allow dump' flag controls whether you can use the reset button to start a dump.

Before creating a dump device, you may wish to determine the likely size of dumps:

```
# sysdumpdev -e
0453-041 Estimated dump size in bytes: 25763840
```

We recommend you run this command with the system in use, as the more work the system has to perform the more information that's stored in memory, and the more space that's needed to hold a dump. Based on the number of bytes displayed by **sysdumpdev**, you can calculate the size of storage (given as the number of physical partitions) you need to allocate to the dump device.

A dump device is created as a logical volume with the **mklv** command:

```
mklv -y hd7 -t sysdump rootvg PP_COUNT
```

Instead of *PP_COUNT*, specify the number of physical partitions based on the value returned by **sysdumpdev -e**. You may also want to change the name of the logical volume. In this example we use *hd7* as a reminder of how AIX Version 3 named the dump device. However it's a good idea to use a descriptive name like *lvsysdump*.

After creating the logical volume, we have to define it as the new primary dump device for the system:

```
# sysdumpdev -p /dev/hd7 -P -K
primary              /dev/hd7
secondary            /dev/sysdumpnull
copy directory       /var/adm/ras
forced copy flag     TRUE
always allow dump    TRUE
```

Using **sysdumpdev**, we set the newly created logical volume as the primary dump device and make the change permanent using the **-P** flag.

CREATING A DUMP

There are several ways to start a dump manually. Note that, in all of them, the system comes to a halt and has to be rebooted after the dump has been written to the dump device. So, starting a dump manually is not the sort of task an administrator does on a regular basis, but may be just one of many ways to troubleshoot a system. One way to initiate a dump is with the command:

```
sysdumpstart
```

This starts a dump to the primary (option **-p**) or secondary (option **-s**) dump device. Following a kernel dump, the following values may be displayed on the three-digit LED display:

0c0 Dump completed successfully

0c1 I/O error during dump

0c2 Dump in progress

0c4 Dump device is too small

0c5 Dump internal error

0c6 Prompt for making secondary dump device ready

0c7 Dump waiting for a remote host

0c8 Dump disabled as no dump device is assigned

0c9 Dump in progress

0cc System switched from primary to secondary dump device.

In addition to **sysdumpdev**, there are other ways to initiate a dump. Depending on the setting of the 'always allow dump' flag (explained below), one may use the reset button in normal or service mode to create a dump. Also, the key sequence Ctrl+Alt+1 may be used to initiate a dump to the primary dump device, and Ctrl+Alt+2 may be used to initiate a dump to the secondary device.

USE A DUMP

To use a dump, start by displaying information about the latest dump available on the dump devices:

```
# sysdumpdev -L
Device name:         /dev/hd7
Major device number: 10
Minor device number: 9
Size:                21557760 bytes
Date/Time:           Thu Aug 28 11:44:45 CDT 1998
Dump status:         0
dump completed successfully
```

sysdumpdev displays all the information about the previous system dump. The output lists not only your dump device but also the size and the date of the dump. Using this information, you can decide whether the dump is complete and therefore worth saving. Also, if you need to copy the dump to another device, you can use the size information to calculate the space you need.

```
# savecore /var/adm/ras
0481-193 Saving /unix in /var/adm/ras/vmunix.0
0481-183 Saving 21558272 bytes of system dump in
/var/adm/ras/vmcore.0
```

As soon as the dump information is saved as a file on disk, you may want to continue by copying the information to an external medium, such as tape:

```
tar -cvf/dev/rmt0 vmcore.0
```

In order to add some information about the system to the dump, use the **snap** command. This first collects data about users, filesystems, etc, and then adds the dump to it, and writes all the information to a directory or (as in the example line below) a tape device.

```
snap -aD -o /dev/rmt0
```

Don't forget to remove the information generated by **snap** or the **vmcore** files when no longer needed – these files take up a large amount of space and may fill up your filesystem.

Once you have a dump, the next thing to do is analyse the information in it. The command to use is **crash** (you have to specify the dump file available):

```
crash vmcore.0
```

crash continues by displaying a '>' prompt, indicating that it's ready to process subcommands, such as **stat**, which displays statistics about the dump. The subcommands '?' or **help** are used to display commands that are available in the environment provided by **crash**.

Analysing dump information is way beyond the scope of this article. Please refer to **crash**'s **man** page to get an idea of what type of information is available. It's worth pointing out that, despite all the subcommands that are available from **crash**, it is still very difficult to analyse dumps, as you need a considerable amount of other information, such as how the process and file tables are structured. There are, however, workshops and books on the market to assist you in building up the necessary knowledge.

MORE SETTINGS

Sometimes it's necessary to start a dump manually. This can be done, as described above, using the **sysdumpstart** command. You may also do this using the reset button on an AIX machine that has a key mode switch. If the 'always allow dump' attribute is set to *FALSE* (you can

display the attribute using **sysdumpdev -l**), you have to switch the key to the 'Service' position and then press the reset button. If the parameter is set to *TRUE* (it can be set to true using the command below), you can start a dump by pressing the reset button when the key is in the 'Normal' position.

```
sysdumpdev -K
```

Beware that, by setting this flag, you lose the ability just to reset the system and carry out no other action when you press the reset button.

It's worthwhile verifying the value of the forced copy flag using **sysdumpdev -l**. If you want your system to come up even if there isn't enough space in */var/adm/ras* to save information, use the command below to set the flag to *FALSE*.

```
sysdumpdev -d /var/adm/ras
```

This enables the system to continue booting after a crash, even if there isn't enough space, though this means that the dump is lost. If you consider the dump very important, then switch the forced copy flag to *TRUE* using the command below.

```
sysdumpdev -D /var/adm/ras
```

After crashing and writing the dump to the specified dump device, the system displays an LED code, such as *0c0* (see above), and waits for the user to restart the system manually. In order to cut down the time before the system comes up again, I recommend that you switch AIX's *autorestart* attribute to *TRUE*.

To display the setting use:

```
lsattr -E -l sys0
```

To change the setting use:

```
chdev -l sys0 -a autostart=true
```

The combination of setting the *autostart* parameter to *TRUE* and creating a dedicated dump device is especially recommended for servers. They ensures that, in the event of a crash, the system comes up rapidly and the information necessary to analyse the crash is available in the dump device.

MORE INFORMATION

For more information on this subject refer to:

- The operating system manuals and the on-line documentation
- *AIX Performance Tuning* by Frank Waters, published by Prentice-Hall.

Michael Abel, IBM Certified AIX Technical Expert (CATE)
Senior Consultant
res nova Unternehmensberatung GmbH (Germany)

The AIX priority mechanism

Processes can be given priorities to assign a degree of importance to the job. A higher priority should ensure that a process is given a 'larger slice of the cake', meaning that it is given more time to run, as AIX works on the principle that the job with the highest priority is the one to run.

Priorities are assigned using the **nice** command. This command changes the default **nice** value of a job. The default **nice** values are 20 for foreground processes and 24 for a background processes. This **nice** value is added to a base priority of 40 to give default priorities of 60 (foreground process) and 64 (background process).

nice -5 proc1 lowers the priority of the process *proc1* by 5. It does this by adding five to the priority. This appears to be incorrect – 'lowers' the priority by 'adding 5'? It is, in fact, correct. We talk about higher priorities running first but the numbers don't appear to agree with this – the reason is that higher priority corresponds to lower numeric priority.

To see this, let us look at a simple example.

```
# sleep 10000 &
[1] 12828
# ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
200001 A 0 12828 16404 0 64 24 2573 64 9f48e58 pts/0 0:00 sleep
240001 A 0 16404 35346 2 61 20 2772 136 pts/0 0:00 ksh
200001 A 0 19490 16404 12 66 20 2213 176 pts/0 0:00 ps
```

The **sleep** command runs in the background with the default priority.

As can be seen, the priority for this **sleep** is 64.

```
# nice -5 sleep 10000 &
[2] 19492
# ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
200001 A 0 12828 16404 0 64 24 2573 64 9f48e58 pts/0 0:00 sleep
240001 A 0 16404 35346 1 60 20 2772 136 pts/0 0:00 ksh
200001 A 0 19492 16404 1 69 29 2370 60 9f48b98 pts/0 0:00 sleep
200001 A 0 40230 16404 13 66 20 2213 236 pts/0 0:00 ps
```

A second **sleep** command is run with a **nice** value of **-5**. This lowers the priority of the process by 5. As you can see above, the numeric value for the second sleep has actually been increased by 5. Yes, it can be confusing, but we talk about lowering a priority with **nice** when the numeric priority value increases. The rule, as stupid as it sounds, is high priority value means low priority, low priority value means high priority.

Continuing with this inverted logic, the **nice** command can be run as follows:

```
nice --5
```

This ‘double negative’ (minus minus) is a positive, so the priority increases by 5. (Remember that the priority value decreases by 5.) The only user that can use the double minus is *root*, so only *root* can increase a priority.

The **nice** command is used to run a job at a certain priority. To modify the priority of a running job, use the **renice** command.

```
renice -5 <pid>
```

The above changes the priority of the process whose process id is <*pid*>. The priority value change (in the above example by 5) does not increase or decrease the old **nice** value – it replaces it with the new one.

For example, process *1789* is running in the foreground with a **nice** value of –5, and thus a priority value of 64 (40 + 20 + 5). Running **renice -5** changes the priority to 55 (40 + 20 – 5). So in the case of **renice**, a minus *reduces* the priority value, whereas a minus increases the priority value with **nice**. Surely IBM could have come up with a more logical approach!

If you are checking the priority of a process, you'll notice that priorities don't appear to be constant. Run **ps -l** a few times and look at the priority of the shell – you'll see that it changes. Priorities change for a simple reason – the kernel changes them. Why does the kernel change them? That's just the way that AIX is implemented – the kernel regularly changes the priority of running processes.

The kernel obeys the following rules:

1 Run the job with the highest priority that is waiting for CPU.

2 The longer a process is active in the CPU, the more its priority should be lowered.

3 Don't allow the priority to be lowered too much.

These rules exist to ensure that higher priority jobs get most access to the CPU, but not at the cost of preventing all other jobs from getting some CPU time.

These are the rules, but how does the kernel enforce them?

A process's priority is determined by adding the base priority, the **nice** value, and a 'penalty value'. The penalty value is determined by how long the process has control of the CPU – the longer it has control, the more it's punished, and the greater its penalty value. We've already seen how to change the **nice** value, so now let's examine how to manage the penalty value.

Every 10 milliseconds the kernel checks which process is in control of the CPU. The process that's in control has its CPU usage counter incremented by one. (The usage value can be seen in the 'C' column of **ps -l**.) This usage value is divided by two (by default) and added to the process's priority (thus effectively lowering priority). Therefore, high priority jobs, which have more access to the CPU, eventually

have their priority lowered to allow other, lower priority jobs to access the CPU. This satisfies rule 2 above. To satisfy rule 3, AIX needs to ensure that the punishment isn't excessive. Firstly, there is a maximum value for the usage counter – 120. This, by default, means a maximum priority penalty of 60 (120 ÷ 2). Based on this, a process that is punished to its maximum may never regain control of the CPU, despite starting with a high priority. AIX ensures that the penalty is not excessive by halving (by default) the punishment every second. This ensures that a high priority process will not hog the CPU, but will not be punished to the extent that its priority is permanently lowered below that of all other processes.

You'll notice that I mentioned 'default' twice. By default, the penalty is equal to half of the CPU usage counter. By default, the punishment is halved every second. These defaults can be changed using the **schedtune** command. The two **schedtune** flags that set the 'punishment system' are **-r** and **-d**.

R determines the ratio of CPU usage to priority penalty. CPU usage is multiplied by *R* to give the priority penalty. The default value is 0.5, which ensures that the penalty is half of the usage. To change *R*, run **schedtune -r *X*** , where *X* is the required ratio.

D determines the reduction in priority penalty that occurs every second. The usage value is multiplied by *D* every second. The default value is also 0.5, which ensures that the penalty is halved every second. To change *D*, run **schedtune -d *X***, where *X* is the required reduction.

By changing the values for *R* and *D*, we can change the way processes' priorities change on the system, and thus the response time for processes. I mentioned above that the default values are 0.5. We change this value using **schedtune -r *X***, but the setting is not straightforward – it wouldn't be, would it? **schedtune -r 8** sets the *R* value to 0.25, and **-r 24** sets it to 0.75.

The values specified as arguments to **schedtune** are divided by 32, thus **-r 16** sets *R* to 0.5.

To test the effect of changing the *R* and *D* values, I created the test listed below.

PRIGET (RUNS FOR ABOUT 15 TO 20 SECONDS)

```
rm progpri?
rm progpri?
clear
nice --0 ./prog1 &
nice -0 ./prog2 &
loop=0
while [ $loop -lt 50 ]
do
tput cup 10 5
echo completed $loop out of 50
echo "$loop :: `ps -lf | grep prog1 | grep -v grep`" >> progpri1
echo "$loop :: `ps -lf | grep prog2 | grep -v grep`" >> progpri2
loop=`expr $loop + 1`
done
./killer
awk '{print $1 "::" $9}' progpri1 > pri1
awk '{print $1 "::" $9}' progpri2 > pri2
echo "prog 1 counted to `tail -1 /tmp/prog1count`"
echo "prog 2 counted to `tail -1 /tmp/prog2count`"
```

PROG1 AND PROG2

```
#! /usr/bin/ksh
rm /tmp/prog1count 2> /dev/null
count=0
while true
do
count=`expr $count + 1`
echo $count >> /tmp/prog1count
done
```

The **pritest** program runs two processes (*prog1* and *prog2*). The priorities of these processes are changed over the test runs by changing their **nice** values (using the fourth and fifth lines of **pritest**). The two processes simply enter an infinite loop that outputs a counter to a file. The effect of priorities can be seen by the difference in the number of write operations carried out by each process. The idea is that the higher priority process will run more often and thus write more to the file (the file will contain a higher counter reading). The effect of changing *R* and *D* can be determined by the change in the number of write operations by each process. The remainder of **pritest** is concerned with recording the changing priorities of *prog1* and *prog2* and reporting the difference in the number of writes performed by *prog1* and *prog2*.

R and *D* were changed in each run by assigning *proc1* and *proc2* the following **nice** values:

```
Proc1          0        --10          --20

Proc2          0         -10           -20
```

This ensures that the effect of the changes can be seen on a range of **nice** values.

Let us examine one run of the test. *R* and *D* are at their defaults of 0.5; the **nice** value of *proc1* is – –20; and that of *proc2* is –20. Therefore *proc1* is at the maximum specified priority, *proc2* is at the minimum.

The priorities of the two processes were recorded by **priget** fifty times, and they're plotted on a graph to show how they change as the processes runs (Figure 1).

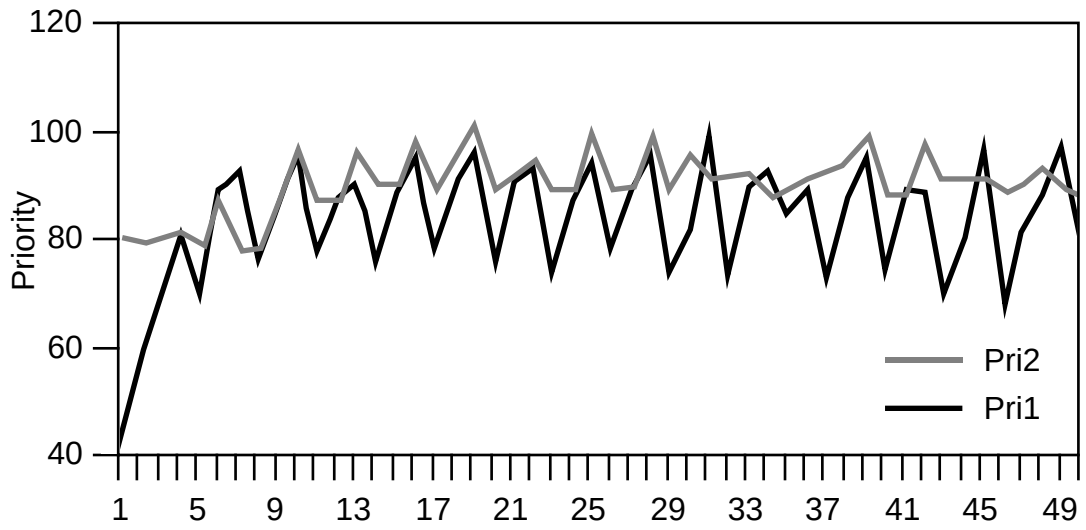

Figure 1: Process priorities recorded by ***priget***

We can see from the graph how the priorities of both processes increase and decrease as a result of the punishment and reduction in punishment determined by the *R* and *D* parameters. Remember that a high priority value is actually a low priority. Therefore, the lowest process on the graph is the one that runs (the highest priority process runs first). The priority of *proc1* is represented by the *pri1* line, and that of *proc2* by *pri2*.

We can see that *proc1* usually has the highest priority (lowest priority value) and should therefore run more than *proc2*. The average priority for *proc1* is 83.7, compared with 90.62 for *proc2*. The 'punishment' feature of the AIX scheduler ensures that, even though *proc2* has a lower priority, it still gets some CPU time.

The average values of the priorities of the two processes do not give a good indication of how often each process runs. All we can tell from the averages is that *proc1* runs more than *proc2*. As *proc1* has the higher priority 40 times out of 50, we can say that *proc1* ran 80% of the time, and *proc2* 20% of the time.

priget also records the number of times that each process writes a counter to disk. The actual figures for the counters are: *proc1* wrote a counter 405 times and *proc2* 128 times. The percentages are: *proc1* 76%, *proc2* 24%. This shows that the number of counters written gives a good indication of the priority of the process. For this reason, I used the difference between the number of counters written by the two processes as a method of testing the changes in the *R* and *D* parameters. The change in the difference for the various parameter values is indicative of the effect on priorities of changing *R* and *D*.

Note that the percentage of times *proc1* has the highest priority (80%) is not exactly equal to the percentage of times *proc1* writes its counter to disk (76%). The difference is a result of the fact that processes can relinquish control of the CPU voluntarily. They usually do so if they are waiting for an event and decide to let another process run, rather than 'hogging' the CPU. One such event is waiting for disk I/O.

CHANGING R

The effect of changing *R* is measured by the change in the difference between the number of writes performed by *proc1* and *proc2*. The graph in Figure 2 shows the difference for four values of *R*: 0, 8, 16, and 32.

For each value of *R*, three priorities are used for *proc1* and *proc2*. They are : {– –20, –20}, {– –10, –10}; and {0, 0}. Figure 2 overleaf shows how the difference in writes changes.

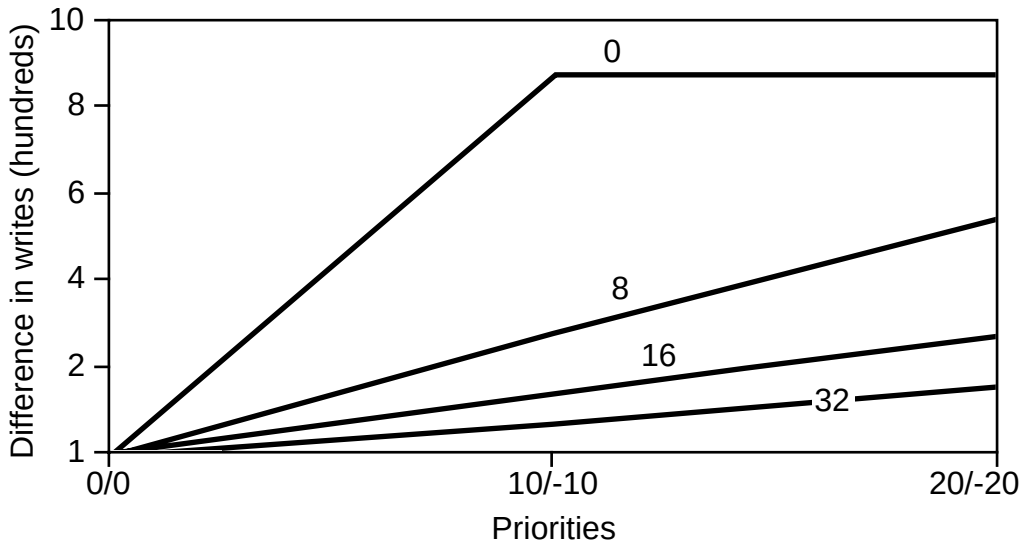

Figure 2: Writes performed by proc1 and proc2

You can see from the graph that the larger the value for *r* given as a parameter of **schedtune**, the less difference there is in the number of writes carried out by high and low priority processes. With **schedtune -r 32**, there is relatively little difference between processes running with **nice** values of – –20 and –20. We are, in effect, reducing the effect of priorities. The CPU is shared more equally among processes.

With **schedtune -r 0**, the opposite happens. Priorities are rigidly enforced with higher priority jobs having much more access to the CPU. In this scenario, there is little sharing of the CPU among processes; higher processes just have control of the CPU. This is useful if processes are of high importance and high priority jobs must be guaranteed access to the CPU at the expense of lower priority jobs. In fact, **schedtune -r 0** disables the 'punishment' of processes. It doesn't matter how long the process has been in control of the CPU, priorities are not reduced. This is known as 'fixed priority scheduling'.

There is no right or wrong value for *R*; it's up to system administrators to decide which value is best suited to their environment. Some systems require fixed priorities, others require that priorities should have no effect. Some systems require a balance between the two. Depending on the system, a value of *R* can be arrived at.

The logic behind all of this is that *R* determines how much 'punishment' is meted out to processes that are frequently in control of the CPU (higher priority processes).

Remember that the kernel keeps track of how many times each process has been in control of the CPU. This usage counter is incremented each time the kernel finds the process active on the CPU. The usage counter is multiplied by *R* to give a 'punishment'. The punishment is added to the process's priority value (lowering its priority). Therefore, the larger the value of *R*, the more processes are punished. The more high priority processes are punished, the more the CPU is shared with lower priority processes. *R* is derived by dividing the parameter given to **schedtune -r** by 32. Thus the larger the value of **-r**, the larger the value of *R*, and the larger the 'punishment' the scheduler metes out. This punishment can be seen by a smaller difference in the writes between the high and low priority processes.

CHANGING D

Again, the difference in the number of writes performed by *proc1* and *proc2* is used to show the effect of changing this parameter. The graph in Figure 3 shows the difference for four values of *D*: 0, 8, 16, and 32.

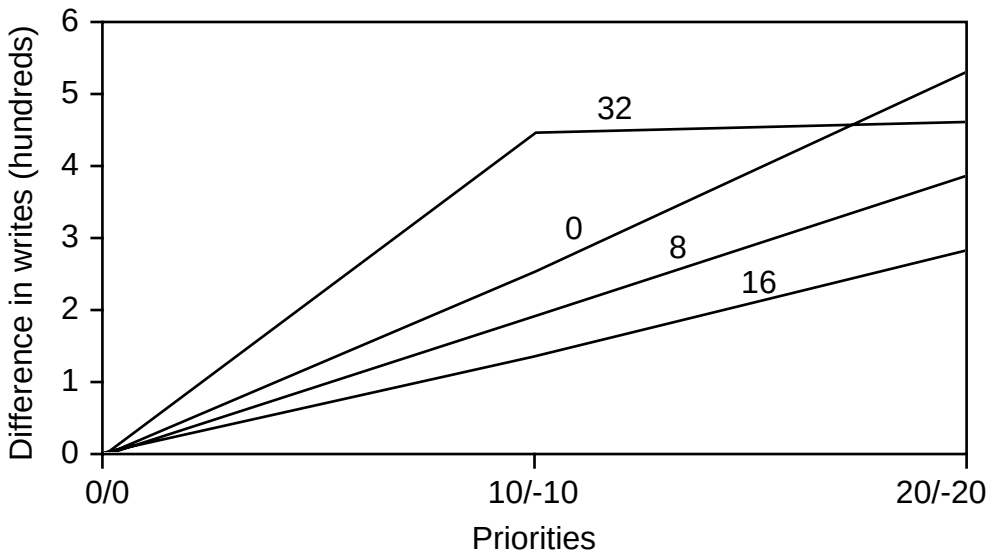

Figure 3: Number of writes for four values of D

For each value of *D*, three priorities are used for {*proc1*, *proc2*}, they are: {––20, –20}; {––10, –10}; and {0, 0}. The graph shows how the difference in the number of writes changes.

As you can see, the graph is not as straightforward as that for changing *R*. It seems strange that the lines appear out of sequence – usually it's either 32 above 16 above 8 above 0 or the other way around. In this case, 32 appears out of place. In fact, it's not. The graph is correct and it shows that the AIX scheduler is working as it should.

While *R* is the 'punishment', *D* is the reduction in punishment. As mentioned earlier, the punishment is reduced every second. The amount by which it is reduced depends on *D*. *D* is derived by dividing the parameter given to **schedtune -d** by 32. The punishment is multiplied by *D* to give the new reduced punishment. Therefore **schedtune -d 16** (the default) would set *D* to 0.5, so the punishment is halved every second.

The reason that the line for **-d 32** seems out of place is that it is a unique value. Setting **-d 32** means that the punishment is never reduced – if **-d 32**, then *D* is 32 ÷ 32 (1).

Every second, the punishment is reduced by multiplying it by *D*. In this case the punishment remains the same. If the punishment is never decreased, it just increases every time the process gets access to CPU. Consequently, you would expect high priority jobs quickly to be punished out of contention for CPU. *proc1* would run first as it has the highest priority. It would then be punished until its priority was below *proc2*. *proc2* would then run and be punished below *proc1*. *proc1* then would then run again and be punished below *proc2*. The two processes would simply take turns on the CPU.

If this were the case, both would run more or less the same number of times and the difference in writes would be close to zero. As you can see from the graph, this is not the case.

This is because AIX imposes a maximum punishment of 60. *proc1* and *proc2* both reach this maximum and then contend based on their assigned priorities. As *proc1* has the higher priority, it has complete control of the CPU. This is why the graph shows the biggest difference in writes when **schedtune** is run with **-d 32**.

So, leaving aside the 'anomaly' of **-d 32**, the other curves appear in order, with the one for *D* = 0 being associated with the greatest difference in the number of writes and the one for *D* = 16 being associated with the smallest difference. The logic behind this is that, if **schedtune -d** is set to 0, *D* is also set to 0 (0 ÷ 32). Therefore, every second the punishment value is removed (multiplied by zero). This means that we have, effectively, fixed priorities. They are 'less fixed' than we get by using **schedtune -r 0**. Setting *R* to zero ensures no punishment, setting *D* to zero means that punishment does occur (depending on the value of *R*) but it is cleared every second.

With *D* set to 16, we have the smallest reduction in punishment (halved every second). Obviously a small reduction in punishment means that punishment has more of a levelling effect on priorities. We can see this in the small difference in writes between the high and low priority processes.

SUMMING UP

This article is not intended to tell you what values to use for *D* and *R*. That decision depends on your system and how you want it to run. Be careful when changing these values – some software, such as Oracle, expects its processes to get a fair share of the CPU. As always, test your system thoroughly after making any changes. Fortunately, the **schedtune** command is not permanent – it only applies till next boot. So, if you do make a complete mess and one process hogs the CPU to the extent that nothing else runs, a reboot will fix it. To make the changes permanent, add the command to your start-up files.

And now a few questions. (The one you're probably asking is simply 'What!?')

- What were the IBM designers thinking when they designed this?
- Why does a high priority value map to a low priority?
- Why use negative and double negative parameters with **nice**?
- Why does **renice** not work in the same way as **nice**?
- Why can't I set *R* and *D* directly, instead of having to set the two

indirectly, using a value that is then divided by 32?

- Why am I asking these questions – will IBM ever change?

Now, I can answer this last one – I doubt it!

John McAvoy (Ireland)

Tuning AIX parameters

tap ('tuning AIX parameters') is a shell script that allows you to tune any tunable parameter related to virtual memory, CPU scheduling, and network-related aspects of AIX 4.1. Among other commands, the shell script uses two release-specific hardware-dependent utilities called **vmtune** and **schedtune**. These programs are shipped with AIX, but are not installed as part of a standard installation.

vmtune

The location of this program is */usr/samples/kernel*. This directory also includes the source file *vmtune.c* and a *README* file for more specific information.

schedtune

The location of this program is */usr/samples/kernel*. This directory also includes the source file *schedtune.c* and a *README* file.

LOGICAL GROUPING OF PARAMETERS

The script displays tuneable parameters under the following menus:

- Virtual memory
- CPU scheduler
- Network.

Executing a particular menu option does not compel the user to change the related parameter. The user can simply read the prologues that are associated with the tuning of the parameter and see the current value without changing it.

TAP

```
#################################################################################
#
#   Name:     tap.sh (tune AIX parameters)
#
#   Overview: This script allows the user to observe and modify
#             tuneable parameters for virtual memory, CPU scheduling,
#             and network functions.
#
#   Notes:    1 The script contains following functions:
#               DefineVariables
#               MoveCursor
#               DisplayMessage
#               IsDigit
#               HandleInterrupt
#               ProcessExit
#               DisplayROOTMenu
#               DisplayVMTUNEMenu
#               TuneVirtualMemory
#               PrepareParameterFileForDisplay
#               LoadVirtualMemoryTuneParameterFile
#               DisplayNetworkTuneMenu
#               TuneNetwork
#               LoadNetworkTuneParameterFile
#               DisplaySchedulerTuneMenu ;;
#               TuneScheduler
#               LoadSchedularTuneParameterFile
#               main
#
#  History
#  Date      Author    Description
#  ---------------------------------------------------------------
#  12/06/98  A Zaman   Initial Build
#
#################################################################################

#################################################################################
#
#  Name     : DefineVariables
#
```

```
#  Overview : The function displays the main menu.
#
#################################################################################
DefineVariables ()
{
# Variables
OPTION=          # Option selected
NEW_VALUE=       # Specified new value for parameter in question
CUR_VALUE=       # Existing value for parameter in question
COMMAND=         # Command being executed to set the new value
ERM=             # System error message
TRUE=0
FALSE=1

# Escape sequences
ESC="\0033["
RVON= [7m                  # Reverse video on
RVOFF= [27m                # Reverse video off
BON= [5m                   # Blinking on
BOFF= [25m                 # Blinking off

# menu titles
#
VMTUNE="${RVON}Virtual Memory Tuning${RVOFF}"
NETWORK="${RVON}Network Specific Tuning${RVOFF}"
SCHEDULER="${RVON}Scheduler Specific Tuning${RVOFF}"

# Temporary files
TEMP_FILE="/tmp/tap_$$.tmp"
PARAM_FILE="${TEMP_FILE}_1"
ERROR_FILE="${TEMP_FILE}_3"

# Messages
INVALID_OPTION="\${OPTION}, is an invalid option${RVOFF}"
COMMAND_FAILED="Failed to execute command \${COMMAND}${RVOFF}"
COMMAND_SUCCEEDED="Successfully executed command \${COMMAND}${RVOFF}"
SYSTEM_ERROR="\${ERM}${RVOFF}"
BLANK="\\\t\\\t\\\t\\\t\\\t\\\t\\\t\\\t\\\t\\\t\\\t   ${RVOFF}"
INVALID_VALUE="\${NEW_VALUE} is an invalid value${RVOFF}"
INTERRUPT="Program interrupted ! Quitting early${RVOFF}"

# Prefixes
ERROR="${RVON}${BON}tap.sh:ERROR:${BOFF}"
INFO="${RVON}${BON}tap.sh:INFO:${BOFF}"

# Exit codes
FEC=1                    #  Failure exit code
SEC=0                    #  Success exit code

SLEEP_DURATION=5         #  Number of seconds for sleep command
```

```
# Define signals
SIGNEXIT=0  ;  export SIGNEXIT   # Normal exit
SIGHUP=1    ;  export SIGHUP     # Session disconnected
SIGINT=2    ;  export SIGINT     # ctrl-c
SIGTERM=15  ;  export SIGTERM    # kill command
}

##############################################################################
#
#   Name        : DisplayMessage
#
#   Description : This function displays a message
#
#   Input       : Message type (E = Error, I = Informative)
#                 Message code
#
##############################################################################
DisplayMessage (  )
{
trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

MESSAGE_TYPE=$1
MESSAGE_TEXT="`eval echo "$2"`"

MoveCursor 24 1

if [ "${MESSAGE_TYPE}" = "E" ]
then
   echo  "`eval echo ${ERROR}`${MESSAGE_TEXT}\c"
   sleep ${SLEEP_DURATION}
elif [ "${MESSAGE_TYPE}" = "SE" ]
then
   # Display system error
   echo  "${RVON}${MESSAGE_TEXT}\c"
   sleep ${SLEEP_DURATION}
elif [ "${MESSAGE_TYPE}" = "B" ]
then
   echo  "${RVON}${MESSAGE_TEXT}\c"
else
   echo  "`eval echo ${INFO}`${MESSAGE_TEXT}\c"
   sleep ${SLEEP_DURATION}
fi
}

##############################################################################
#
#  Name     : ProcessExit
#
#  Overview : This function removes all the temporary files and exits
#             with exit code supplied
```

```
#
#  Input    : Exit code
#
################################################################################
ProcessExit ()
{
EXIT_CODE=$1

# Remove all the temporary files
rm -f ${ERROR_FILE}
rm -f ${PARAM_FILE}
rm -f ${SPECIFIC_PARAM_FILE}

# Remove all split files
rm -f ${TEMP_FILE}.*

exit  ${EXIT_CODE}
}

################################################################################
#
#  Name     : HandleInterrupt
#
#  Overview : This function displays a message and calls ProcessExit
#             with failure exit code.
#
################################################################################
HandleInterrupt ()
{
DisplayMessage I "${INTERRUPT}"

echo "${RVOFF}"
ProcessExit  $FEC
}

################################################################################
#
#   Name        : MoveCursor
#
#   Description : This function moves the cursor to a given point.
#
#   Input       : y-coordinate value
#                 x-coordinate value
#
################################################################################
MoveCursor (  )
{
trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

YCOR=$1
```

```
XCOR=$2

print -n    "${ESC}${YCOR};${XCOR}H"
}

###############################################################################
#
#  Name        : IsDigit
#
#  Description : This function checks whether a string contains
#                only digits.
#
#  Input       : A string
#
#  Returns     : TRUE or FALSE
#
#  Notes       : 1. The function returns FALSE if the string contains
#                   letters
#
#              : 2. The function returns TRUE if the string contains
#                   only digits
#
###############################################################################
IsDigit  ()
{
trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

STRING=$1

# Get the string length
STRLEN=`echo $STRING | wc -c`

# Strip off carriage return
STRLEN=`expr $STRLEN - 1`

STARTPOS=1
ENDPOS=1

while true
do
   if  [  $STARTPOS  -gt   $STRLEN ]
   then
         break
   fi
   DIGIT=`echo $STRING  |  cut -c $STARTPOS-$ENDPOS`
   if [ "$DIGIT" != "0" -a "$DIGIT" != "1" -a "$DIGIT" != "2" -a \
        "$DIGIT" != "3" -a "$DIGIT" != "4" -a "$DIGIT" != "5" -a \
        "$DIGIT" != "6" -a "$DIGIT" != "7" -a "$DIGIT" != "8" -a \
        "$DIGIT" != "9" ]
   then
```

```
        return  $FALSE
   fi

   STARTPOS=`expr  $STARTPOS + 1`
   ENDPOS=${STARTPOS}

done

return $TRUE
}

#############################################################################
#
#  Name     : DisplayROOTMenu
#
#  Overview : This function displays the main menu.
#
#############################################################################
DisplayROOTMenu ()
{
trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

while true
do
  clear
  echo "    ************************************ "
  echo "    *           TUNABLE  AREA          * "
  echo "    *                                  * "
  echo "    *          1. Virtual memory       * "
  echo "    *          2. CPU scheduler        * "
  echo "    *          3. Network              * "
  echo "    *          4. Exit                 * "
  echo "    *                                  * "
  echo "    ************************************ "
  echo "            Enter option--->\c            "

  # Read the option selected
  read  OPTION

  # Process option
  case  ${OPTION}  in
     "") DisplayMessage E "${INVALID_OPTION}" ;;
      1) DisplayVirtualMemoryTuneMenu ;;
      2) DisplaySchedulerTuneMenu ;;
      3) DisplayNetworkTuneMenu ;;
      4) ProcessExit $ESC ;;
      *) DisplayMessage E "${INVALID_OPTION}" ;;
  esac
done
}
```

```
#############################################################################
#
#  Name     : DisplayVirtualMemoryTuneMenu
#
#  Overview : This function displays the $VMTUNE menu.
#
#############################################################################
DisplayVirtualMemoryTuneMenu ()
{
trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

while true
do
  clear
  echo "    ********************************** "
  echo "    *          ${VMTUNE}               * "
  echo "    *                                  * "
  echo "    *          1. maxfree              * "
  echo "    *          2. maxperm              * "
  echo "    *          3. maxpgahead           * "
  echo "    *          4. minpgahead           * "
  echo "    *          5. maxpin               * "
  echo "    *          6. minfree              * "
  echo "    *          7. minperm              * "
  echo "    *          8. npskill              * "
  echo "    *          9. npswarn              * "
  echo "    *         10. numclust             * "
  echo "    *         11. numfsbuf             * "
  echo "    *         12. ROOT menu            * "
  echo "    *         13. Exit                 * "
  echo "    *                                  * "
  echo "    ********************************** "
  echo "            Enter option ---->\c        "

  # Read the option selected
  read  OPTION

  # Process the option selected
  case ${OPTION} in
      1) CUR_VALUE="`vmtune | head -4 | tail -1  | awk {'print $6'}`" ;
         COMMAND="vmtune -F" ;
         TuneVirtualMemory "MAXFREE" ;;

      2) CUR_VALUE="`vmtune | head -4 | tail -1  | awk {'print $2'}`" ;
         COMMAND="vmtune -P" ;
         TuneVirtualMemory "MAXPERM" ;;

      3) CUR_VALUE="`vmtune | head -4 | tail -1  | awk {'print $4'}`" ;
         COMMAND="vmtune -R" ;
         TuneVirtualMemory "MAXPGAHEAD" ;;
```

```
      4) CUR_VALUE="`vmtune | head -4 | tail -1  | awk {'print $3'}`" ;
         COMMAND="vmtune -r" ;
         TuneVirtualMemory "MINPGAHEAD" ;;

      5) CUR_VALUE="`vmtune | head -8 | tail -1  | awk {'print $1'}`" ;
         COMMAND="vmtune -M" ;
         TuneVirtualMemory "MAXPIN" ;;

      6) CUR_VALUE="`vmtune | head -4 | tail -1  | awk {'print $5'}`" ;
         COMMAND="vmtune -f" ;
         TuneVirtualMemory "MINFREE" ;;

      7) CUR_VALUE="`vmtune | head -4 | tail -1  | awk {'print $1'}`" ;
         COMMAND="vmtune -p" ;
         TuneVirtualMemory "MINPERM" ;;

      8) CUR_VALUE="`vmtune | head -8 | tail -1  | awk {'print $3'}`" ;
         COMMAND="vmtune -k" ;
         TuneVirtualMemory "NPSKILL" ;;

      9) CUR_VALUE="`vmtune | head -4 | tail -1  | awk {'print $8'}`" ;
         COMMAND="vmtune -w" ;
         TuneVirtualMemory "NPSWARN" ;;

     10) CUR_VALUE="`vmtune | head -8 | tail -1  | awk {'print $4'}`" ;
         COMMAND="vmtune -c" ;
         TuneVirtualMemory "NUMCLUST" ;;

     11) CUR_VALUE="`vmtune | head -8 | tail -1  | awk {'print $5'}`" ;
         COMMAND="vmtune -b" ;
         TuneVirtualMemory "NUMFSBUF" ;;

     12) return $TRUE ;;

     13) ProcessExit $SEC;;

     "") DisplayMessage E "${INVALID_OPTION}" ;;

      *) DisplayMessage E "${INVALID_OPTION}" ;;
 esac
done
}

#############################################################################
#
#  Name     : DisplaySchedulerTuneMenu
#
#  Overview : This function displays the $SCHEDULER menu.
#
#  Notes    : Each option on this menu corresponds to a parameter
```

```
#             of the schedtune command.
#
###############################################################################
DisplaySchedulerTuneMenu ()
{
trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

while true
do
  clear
  echo "    ************************************************ "
  echo "    *     ${SCHEDULER}                               * "
  echo "    *                                                * "
  echo "    *  1. Maximum fork retry interval                * "
  echo "    *                                                * "
  echo "    *  2. Memory load control parameters             * "
  echo "    *     2a. High memory over commitment threshold  * "
  echo "    *     2b. Memory over commitment threshold       * "
  echo "    *     2c. Minimum level of multi-programming     * "
  echo "    *  3. Process priority calculation               * "
  echo "    *     3a. Parameter r                            * "
  echo "    *     3b. Parameter d                            * "
  echo "    *  4. Time slice expansion amount                * "
  echo "    *  5. Restore default values of all parameters   * "
  echo "    *  6. ROOT menu                                  * "
  echo "    *  7. Exit                                       * "
  echo "    *                                                * "
  echo "    ************************************************ "
  echo "            Enter option ---->\c          "

  # Read option selected
  read  OPTION

  # Process option selected
  case ${OPTION} in
      1) CUR_VALUE="`schedtune | tail -1  | awk {'print $6'}`" ;
         COMMAND="schedtune -f" ;
         TuneScheduler "MAX_FORK_RETRY_INTERVAL" ;;

     2a) CUR_VALUE="`schedtune | tail -1  | awk {'print $1'}`" ;
         COMMAND="schedtune -h" ;
         TuneScheduler   "HIGH-MEMORY-OVERCOMMITMENT-THRESHOLD" ;;

     2b) CUR_VALUE="`schedtune | tail -1  | awk {'print $2'}`" ;
         COMMAND="schedtune -p" ;
         TuneScheduler   "PROCESS-MEMORY-OVERCOMMITMENT-THRESHOLD" ;;

     2c) CUR_VALUE="`schedtune | tail -1  | awk {'print $3'}`" ;
         COMMAND="schedtune -m" ;
         TuneScheduler   "MINIMUM-LEVEL-OF-MULTIPROGRAMMING" ;;
```

```
     3a) CUR_VALUE="`schedtune | tail -1  | awk {'print $8'}`" ;
         COMMAND="schedtune -r" ;
         TuneScheduler  "PROCESS_PRIORITY_CALCULATION" ;;

     3b) CUR_VALUE="`schedtune | tail -1  | awk {'print $7'}`" ;
         COMMAND="schedtune -d" ;
         TuneScheduler  "PROCESS_PRIORITY_CALCULATION" ;;

      4) CUR_VALUE="`schedtune  | tail -1  | awk {'print $9'}`" ;
         COMMAND="schedtune -t" ;
         TuneScheduler   "TIME_SLICE_EXPANSION_AMOUNT" ;;

      5) CUR_VALUE="" ;
         COMMAND="schedtune -D" ;
         TuneScheduler   "RESTORE-DEFAULTS" ;;

      6) return $TRUE ;;

      7) ProcessExit $SEC;;

     "") DisplayMessage E "${INVALID_OPTION}" ;;

      *) DisplayMessage E "${INVALID_OPTION}" ;;
 esac
done
}

###############################################################################
#
#  Name     : DisplayNetworkTuneMenu
#
#  Overview : This function displays the $NETWORK menu.
#
###############################################################################
DisplayNetworkTuneMenu ()
{
trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

while true
do
  clear
  echo "    ********************************************** "
  echo "    *                 ${NETWORK}                 * "
  echo "    *                                            * "
  echo "    *   1. arpt_killc        17. tcp_keepintvl   * "
  echo "    *   2. ipforwarding      18. tcp_mssdflt     * "
  echo "    *   3. ipfragttl         19. tcp_recvspace   * "
  echo "    *   4. ipqmaxlen         20. tcp_sendspace   * "
  echo "    *   5. ipsendredirects   21. tcp_ttl         * "
  echo "    *   6. loop_check_sum    22. thewall         * "
```

```
echo "    *   7. lowclust          23. udp_recvspace   * "
echo "    *   8. lowmbuf           24. udp_sendspace   * "
echo "    *   9. maxttl            25. udp_ttl         * "
echo "    *  10. mb_cl_hiwat       26. biod_count      * "
echo "    *  11. nonlocsrcroute    27. ROOT Menu       * "
echo "    *  12. rfc1122addrchk    28. Exit            * "
echo "    *  13. rfc1323                               * "
echo "    *  14. sb_max                                * "
echo "    *  15. subnetsarelocal                       * "
echo "    *  16. tcp_keepidle                          * "
echo "    *                                            * "
echo "    ********************************************** "
echo "           Enter option ---->\c          "

# Read option selected
read  OPTION

# Process option selected
case ${OPTION} in
    1) CUR_VALUE="`no -o arpt_killc |  awk {'print $3'}`" ;
       COMMAND="no -o arpt_killc=" ;
       TuneNetwork "ARPT_KILLC" ;;

    2) CUR_VALUE="`no -o ipforwarding | awk {'print $3'}`" ;
       COMMAND="no -o ipforwarding=" ;
       TuneNetwork   "IPFORWARDING" ;;

    3) CUR_VALUE="`no -o ipfragttl | awk {'print $3'}`" ;
       COMMAND="no -o ipfragttl=" ;
       TuneNetwork   "IPFRAGTTL" ;;

    4) CUR_VALUE="`no -o ipqmaxlen | tail -1  | awk {'print $3'}`" ;
       COMMAND="no -o ipqmaxlen=" ;
       TuneNetwork   "IPQMAXLEN" ;;

    5) CUR_VALUE="`no -o ipsendredirects | awk {'print $3'}`" ;
       COMMAND="no -o ipsendredirects=" ;
       TuneNetwork   "IPSENDREDIRECTS" ;;

    6) CUR_VALUE="`no -o loop_check_sum  | awk {'print $3'}`" ;
       COMMAND="no -o loop_check_sum=" ;
       TuneNetwork   "LOOP_CHECK_SUM" ;;

    7) CUR_VALUE="`no -o lowclust  |  awk {'print $3'}`" ;
       COMMAND="no -o lowclust=" ;
       TuneNetwork   "LOWCLUST" ;;

    8) CUR_VALUE="`no -o lowmbuf  | awk {'print $3'}`" ;
       COMMAND="no -o lowmbuf=" ;
       TuneNetwork   "LOWMBUF" ;;
```

```
 9) CUR_VALUE="`no -o maxttl  | awk {'print $3'}`" ;
    COMMAND="no -o maxttl=" ;
    TuneNetwork   "MAXTTL" ;;

10) CUR_VALUE="`no -o mb_cl_hiwat  |  awk {'print $3'}`" ;
    COMMAND="no -o mb_cl_hiwat=" ;
    TuneNetwork   "MB_CL_HIWAT" ;;

11) CUR_VALUE="`no -o nonlocsrcroute |  awk {'print $3'}`" ;
    COMMAND="no -o nonlocsrcroute=" ;
    TuneNetwork   "NONLOCSRCROUTE" ;;

12) CUR_VALUE="`no -o rfc1122addrchk | awk {'print $3'}`" ;
    COMMAND="no -o rfc1122addrchk=" ;
    TuneNetwork   "RFC1122ADDRCHK" ;;

13) CUR_VALUE="`no -o rfc1323 | awk {'print $3'}`" ;
    COMMAND="no -o rfc1323=" ;
    TuneNetwork   "RFC1323" ;;

14) CUR_VALUE="`no -o sb_max | awk {'print $3'}`" ;
    COMMAND="no -o sb_max=" ;
    TuneNetwork   "SB_MAX" ;;

15) CUR_VALUE="`no -o subnetsarelocal |  awk {'print $3'}`" ;
    COMMAND="no -o subnetsarelocal=" ;
    TuneNetwork   "SUBNETSARELOCAL" ;;

16) CUR_VALUE="`no -o tcp_keepidle | awk {'print $3'}`" ;
    COMMAND="no -o tcp_keepidle=" ;
    TuneNetwork   "TCP_KEEPIDLE" ;;

17) CUR_VALUE="`no -o tcp_keepintvl  | awk {'print $3'}`" ;
    COMMAND="no -o tcp_keepinvtl=" ;
    TuneNetwork   "TCP_KEEPINTVL" ;;

18) CUR_VALUE="`no -o tcp_mssdflt | awk {'print $3'}`" ;
    COMMAND="no -o tcp_mssdflt=" ;
    TuneNetwork   "TCP_MSSDFLT" ;;

19) CUR_VALUE="`no -o tcp_recvspace | awk {'print $3'}`" ;
    COMMAND="no -o recvspace=" ;
    TuneNetwork   "TCP_RECVSPACE" ;;

20) CUR_VALUE="`no -o tcp_sendspace | awk {'print $3'}`" ;
    COMMAND="no -o tcp_sendspace=" ;
    TuneNetwork   "TCP_SENDSPACE" ;;

21) CUR_VALUE="`no -o tcp_ttl |  awk {'print $3'}`" ;
    COMMAND="no -o tcp_ttl=" ;
```

```
          TuneNetwork   "TCP_TTL" ;;

      22) CUR_VALUE="`no -o thewall | awk {'print $3'}`" ;
          COMMAND="no -o  thewall=" ;
          TuneNetwork   "THEWALL" ;;

      23) CUR_VALUE="`no -o udp_recvspace |  awk {'print $3'}`" ;
          COMMAND="no -o udp_recvspace=" ;
          TuneNetwork   "UDP_RECVSPACE" ;;

      24) CUR_VALUE="`no -o udp_sendspace  | awk {'print $3'}`" ;
          COMMAND="no -o  udp_sendspace=" ;
          TuneNetwork   "UDP_SENDSPACE" ;;

      25) CUR_VALUE="`no -o udp_ttl  | awk {'print $3'}`" ;
          COMMAND="no -o  udp_ttl=" ;
          TuneNetwork   "UDP_TTL" ;;

      26) CUR_VALUE="`ps -eaf | grep biod | grep -v "grep" | awk
      ➤  {'print $10'}`" ;
          COMMAND="chnfs -b " ;
          TuneNetwork   "BIOD_COUNT" ;;

      27) return $TRUE ;;

      28) ProcessExit $SEC;;

      "") DisplayMessage E "${INVALID_OPTION}" ;;

       *) DisplayMessage E "${INVALID_OPTION}" ;;
 esac
done
}

#############################################################################
#
#  Name     : TuneVirtualMemory
#
#  Overview : This function sets the parameter passed.
#
#  Input    : Parameter name
#
#############################################################################
TuneVirtualMemory ()
{
trap "HandleInterrupt " $SIGINT  $SIGTERM $SIGHUP

TUNE_PARAM="$1"

# Prepare file containing parameter description
```

```
PrepareParameterFileForDisplay  "VM"  $TUNE_PARAM

# Append current value to the parameter description file
echo "\nCurrent Value : ${CUR_VALUE} \n"  >> ${SPECIFIC_PARAM_FILE}

# Let user view the file
clear
more  ${SPECIFIC_PARAM_FILE}

# Ask for the new value
while true
do
   echo "Enter new value for $TUNE_PARAM (q to quit):\c"
   read NEW_VALUE
   case  ${NEW_VALUE}  in
     q|Q) return ${TRUE} ;;
      "") : ;;
      * ) # Check for numeric, positive number
          if  IsDigit  "${NEW_VALUE}"  -a ${NEW_VALUE} -ge 0
          then
                break ;
          else
               DisplayMessage E  "${INVALID_VALUE}" ;
               clear
               more  ${SPECIFIC_PARAM_FILE}
          fi ;;
   esac
done
```

This article concludes in next month's issue of *AIX Update* with the remainder of the code for **tap**.

Arif Zaman
DBA/Developer
High-Tech Software Ltd (UK)

Who is to blame for the core file?

A core file is generated in the current directory when one of a number of errors occurs. Errors such as memory address violations, illegal instructions, bus errors, and user-generated quit signals commonly cause a core dump. The core file that's generated contains a memory

image of the terminated process. It usually points to a possible programming error and should be investigated. But how do you locate core files and, what's even more difficult, how do you establish who's responsible for their creation?

ERROR LOG ENTRY

If a core file has been created on your system, you should have an error log entry similar to the following:

```
---------------------------------------------------------------------
LABEL:          CORE_DUMP
IDENTIFIER:     C60BB505

Date/Time:       Wed Aug 26 20:17:36
Sequence Number: 202843
Machine Id:      003509754900
Node Id:         klauser
Class:           S
Type:            PERM
Resource Name:   SYSPROC

Description
SOFTWARE PROGRAM ABNORMALLY TERMINATED

Probable Causes
SOFTWARE PROGRAM

User Causes
USER GENERATED SIGNAL

        Recommended Actions
        CORRECT THEN RETRY

Failure Causes
SOFTWARE PROGRAM

        Recommended Actions
        RERUN THE APPLICATION PROGRAM
        IF PROBLEM PERSISTS THEN DO THE FOLLOWING
        CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
SIGNAL NUMBER
          11
USER'S PROCESS ID:
       18584
FILE SYSTEM SERIAL NUMBER
```

```
           9
INODE NUMBER
         8194
PROGRAM NAME
myProgram
ADDITIONAL INFORMATION
main 14
__start 50
??

Symptom Data
REPORTABLE
1
INTERNAL ERROR
0
SYMPTOM CODE
PCSS/SPI2 FLDS/myProgram SIG/11 FLDS/main VALU/14 FLDS/__start
-------------------------------------------------------------------------
```

In the above entry, you can see that the program responsible for the generation of the core file, *myProgram*, is listed under *PROGRAM NAME*. I wrote a C program named *myProgram.c* in which I made an elementary programming error. The resulting program, **myProgram**, tried a memory access that resulted in a memory address violation error that sparked off a core file dump.

THE 'COREPATH' PACKAGE IN /USR/SAMPLES/FINDCORE

You can configure AIX to detect when core files are created and automatically mail a message to *root* . The instructions for setting this up are in the *README* file in the */usr/samples/findcore* directory. These programs are delivered with the *bos.sysmgt.serv_aid* fileset.

FINDING AND IDENTIFYING THEM YOURSELF

An alternative method is to find and identify core dumps yourself.

1 Log in as *root*.

2 Find the core file(s):

```
find / -name core -ls
```

3 **cd** to the directory indicated in step 2.

4 Determine which application created the core file:

```
lquerypv -h core 6b0 16
```

The program that caused the core dump is listed on the '6B0' and '6C0' lines. For example:

```
000006B0  00000000  00000000  00000000 6D795072  |....myPr|
000006C0  6F677261  6D000000  00000000 00000000  |ogram...|
```

WHAT NOW?

If you recognize the program's name as one of your applications, then you need to notify your application's supplier for further problem determination – it's possible that the supplier will want a copy of the core file. If the program listed is an AIX command, or if you are unsure of its origin, then consider working with AIX support on this problem.

You could run **dbx** on the binary executable that caused the core dump. This displays the offending system call. In the following example, the program that caused the core dump is called *myProgram*. After running **dbx** against it (see below), the offending function was identified as *strcpy()*. Yes, I admit it, it's my fault!

```
dbx myProgram core
Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g

[using memory image in core]

Segmentation fault in strcpy.strcpy [myProgram] at 0x1000031c
0x1000031c (strcpy+0x1c) 7ca01d2a      stswx   r5,r0,r3
(dbx) where
strcpy.strcpy() at 0x1000031c
main() at 0x100002cc
(dbx) quit
```

If the AIX support personnel decide that the core dump needs to be sent to their support centre, then the output of the **snap -g** command should be sent in also. This gathers the output of the **lslpp -hBc** command, which is required to recreate exact operating system environments. Its output is stored in the directory */tmp/ibmsupt*.

Werner Klauser
Klauser Informatik (Switzerland)

Netscape has launched Enterprise Web Server version 3.6, which provides high-availability based on a multi-process architecture – if a Web application crashes, only one of the Web server's processes goes down, the remaining ones providing fail-over to keep hosted sites running. Improvements to the server's engine increase its load handling by a claimed 800%, while a new CGI engine is said to increase performance by up to 1500%. NDS comes bundled with the product, enabling administrators to manage remote servers by treating multiple servers as one cluster. Out now for AIX and a number of other Unixes, prices start at US$1,300 for a 50-publisher, unlimited access licence.

For further information contact:
Netscape Communications, 501 E Middlefield Road, Mountain View, CA 94043, USA
Tel: +1 650 254 1900
Fax: +1 650 528 4138
Web: http://www.netsacape.com

Unipalm-Pipex, 216 Science Park, Milton Road, Cambridge, CB4 4WA, UK
Tel: +44 1223 250120
Fax: +44 1223 250102

* * *

BMC Software has announced SQL-BackTrack, a back-up and recovery tool for databases that are subject to frequent updates or require either point-in-time recovery or the recovery of selected files or tablespaces. The product has facilities for bringing databases back on-line and for extracting specific tables and columns from a full back-up. It integrates with BMC's Patrol Recovery Manager and (to a lesser extent) IBM's ADSM. Versions are available for Oracle, Microsoft SQL Server, Informix-On-line, and Sybase. Product pricing is based on capacity, and all products are available now.

For further information contact:
BMC Software, 2101 CityWest Blvd, Houston, TX 77042, USA
Tel: +1 713 918 8800
Fax: +1 281 242 6523
Web: http://www.bmc.com

BMC Software, Compass House, 207-215 London Road, Camberley, Surrey GU15 3EY, UK
Tel: +44 1276 24622
Fax: +44 1276 61201

* * *

IBM has launched the 3466 Network Storage Manager for AIX 4.3, which includes management functions from ADSM Version 3.1. The product manages other Network Storage Managers and ADSM servers, providing facilities for 'lights-out server automation'. The new Web back-up and archive clients allow remote control of ADSM, and the package also includes configuration and policy management. It's out now, priced at $4,000.

For further information contact your local IBM representative.