63

AIX update

January 2001

In this issue

AIX Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon/QNA
Post Office Box 350100, Westminster CO 80035-0100, USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

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

Trevor Eddolls

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £180.00 in the UK; $275.00 in the USA and Canada; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 1995 issue, are available separately to subscribers for £16.00 ($23.00) each including postage.

AIX Update on-line

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

An SP/2 environment toolbox

Here is an assortment of commands, procedures, and miscellaneous notes that may come in handy in your RS/6000 SP/2 environment. I've collected them during the course of administering our SP/2 system over the past five years. Our SP/2 comprises two frames, a high-speed switch, one Model 604 high node, two MCA wide nodes, and 11 'Silver' wide nodes. Many of the tips in this article apply to any RS/6000 trusted host environment. I believe that some of them will be new even to seasoned SP administrators, and not just to newcomers. This is basically a sampling of many notes and some issues I think are interesting and useful. This article can easily become 'Part 1 of many', because there are many subjects (Oracle, Service Director, NTP, SSA, RVSDs, etc) that I have not touched on but could. I am attempting to keep this one to 2,000 words or so to make it manageable for *AIX Update*, and then – based on feedback – will supply more.

REMOTE AND PARALLEL COMMANDS

1 The *WCOLL* environment variable

Many of these commands rely on the *WCOLL* variable being set to a file name that contains a list of nodes in your collective. In many cases, this will be a list of all nodes. If *WCOLL* is not set, you'll need to use **-w node_list** (or a similar flag) with each command. I keep a file called */nodes* on all of my nodes that has the same list of nodes. The examples below assume a four-node environment with a */nodes* file that contains the following lines:

```
/node1
/node2
/node3
/node4
```

2 **rcp** – remote copy

This is a very useful command for copying files quickly from one node to another. For example:

```
rcp /usr/local/bin/monitor -p hps21:/usr/local/bin/
```

will copy the **monitor** program from one node to another. The **-p** flag keeps the source date on the target file. Use **man rcp** to find out about other options that are available with this command. Also, make sure that you use the switch's IP address/alias for these copies, whenever possible. The speed difference, believe me, is remarkable.

3 **hostlist**

hostlist is not a remote command, as such. However, it is useful for determining the host names of all active nodes (according to the SDR) and sending them to *stdout*, where they can then be piped to a remote or parallel command (see the **pcp** command examples for another example of **hostlist**).

For example:

```
hostlist -av
```

A sample output in a four-node system with *node3* down is:

```
node1
node2
node4
```

The **-a** flag uses the first host name found in the SDR. The **-v** flag causes **hostlist** to return only nodes that are responding, as indicated by the SDR. See **man hostlist** for more information.

4 **pcp** – parallel copy

This utility copies from one source to many targets. For example:

```
pcp -w node1, node3 /tmp/notes /tmp/notes
```

copies */tmp/notes* from your current location to */tmp/notes* on hosts *node1* and *node3*. Hence:

```
hostlist -av | pcp -w - /tmp/notes /tmp/notes
```

copies */tmp/notes* to all active nodes in your collective. Also see the previous section on *WCOLL* and check **man pcp** for more information on **pcp**.

5 **dsh**

dsh distributes a command to a list of nodes, either obtaining a list of nodes from *WCOLL* or using the **-w** flag. For example:

```
dsh oslevel
```

results in the following output for four nodes:

```
node1: 4.3.3.Ø
node2: 4.3.2.Ø
node3: 4.3.3.Ø
node4: 4.2.1.Ø
```

A very handy use of this command is to test Kerberos authentication and host name resolution between all nodes with one command:

```
dsh dsh -wnode1, node2, node3, node4 date
```

This causes each node to try to issue the **date** command remotely on all the other nodes.

6 Tuning node-to-node TCP/IP activity

Tuning node-to-node TCP/IP activity can greatly improve network performance, whether you use an SP switch or not. For example, I've seen the performance of programs like **SYSBACK**, which backs up files to tape on a remote server, improved by 10 to 1.

The values below are derived from a lot of experimentation and some research. They work great for me, but should be considered no more than a starting point for your own experimentation.

Using the 'network options' (**no**) command, set the following six values:

```
thewall = 65536
sb_max = 2Ø97152
tcp_sendspace = 262144
tcp_recvspace = 262144
udp_sendspace = 65536
udp_recvspace = 262144
```

If you have a switch, set *spoolsize* and *rpoolsize* to at least 2,097,152. If these parameters were left at their default values, the current value is probably 524,288. Use **lsattr** to check them first, then use:

```
chdev -l cssØ -a rpoolsize=2Ø97152
```

and:

```
chdev -l css0 -a spoolsize=2097152
```

to set them.

Note that a reboot is necessary for changes to the switch to take effect.

MISCELLANEOUS DEBUGGING AND PROBLEM DIAGNOSES

1 **node_number**

This command simply gets the node number from the ODM, and displays it on *stdout*. If you don't get a value back (or get 'null'), this can explain a lot of 'strange' problems, especially PSSP install/migrate problems.

2 Cloning a node

Here is a handy little procedure to create a 'clean' **mksysb** image for use as the install image on another node.

Carry out the following procedure at the donor node:

- Run the following commands:

```
rm /etc/niminfo*
odmget -q name=sp CuAt > /tmp/odmsave
odmdelete -q name=sp -o CuAt
mkdir /tmp/ssp; mv /etc/ssp/*name* /tmp/ssp
```

- Make sure that the */etc/hosts* file has reliable host names for all the nodes.

- Make the **mksysb** image using the following commands:

```
cp /tmp/ssp/*name* /etc/ssp
odmadd /tmp/odmsave
```

3 Updating a spot without a rebuild

This is handy to update the spot without having to perform a rebuild from scratch. Note that just adding your updated software to *lppsource* may not always work correctly – this procedure will ensure that it does.

In the *lppsource* directory:

```
inutoc .
nim-o check -F lppsource_name
smit nim_res_op
```

select ‘SPOT’, then **smitty update_all**. To finish:

```
setup_server
```

4 Checking MAC address discrepancies

Many NIM problems are caused by MAC address discrepancies, especially after hardware changes. Use the following command to display the MAC addresses of each node:

```
dsh lscfg -vl entØ | grep Network
```

and use the following command to display the MAC address in the SDR, then compare it to the **lscfg** output:

```
SDRGetObjects Node hdw_enet_addr
```

5 Updating the *host_responds* or *switch_responds* state in SDR

Occasionally, usually when new nodes are introduced or PSSP is updated, you get into a situation where *host_responds* and/or *switch_responds* gets ‘stuck’. Use the procedure below to alter the state of the SDR.

– First display the state using the command:

```
SDRGetObjects host_responds
```

– Then issue the command:

```
SDRChangeAttrValues host_responds node_number==xx
```

where host_responds is either ‘0’ (for ‘off’/‘no’) or ‘1’ (for ‘on’/’yes’), and *xx* is the node number.

– Use the same commands for *switch_responds*, substituting *switch_responds* for *host_responds*.

SOME GOOD SP LINKS

• Scripts:

```
http://www.parallix.demon.co.uk/
```

- Performance tuning:

```
http://www.rs6000.ibm.com/support/sp/perf/index.html
```

- RS/6000 SP Resource Centre:

```
http://www.rs6000.ibm.com/support/sp/resctr/index.html
```

- SP User Group:

```
http://spud-web.tc.cornell.edu/HyperNews/get/SPUserGroup.html
```

David Miller
Database Manager
Baystate Health Systems (USA)

The Apache Web server – part 1

INTRODUCTION

The Apache Web server is a public domain software program that can be acquired by anyone. This article describes how you can:

- Acquire and extract the source files
- Compile the source code
- Install the application
- Configure Apache
- Modify Apache.

The article also includes a shell script, aadm.sh, that can be used to administer aspects of Apache.

ACQUIRING APACHE SOFTWARE

Apache's source files can be downloaded from the following site:

```
http://www.apache.org
```

EXTRACTING TAR FILE

The download comprises a **tar** file that uses relative paths for archiving. Use the following procedure to extract the software:

1 Make an application directory (eg */appl/apache*).

2 Place the **tar** file in the application directory and extract the archive. This will install the software in following directories:

```
/appl/apache/apache_<version>
/appl/apache/apache_<version>/cgi-bin
/appl/apache/apache_<version>/conf
/appl/apache/apache_<version>/logs
/appl/apache/apache_<version>/htdocs
/appl/apache/apache_<version>/src
/appl/apache/apache_<version>/icons
```

CREATING THE HTTPD EXECUTABLE (COMPILING)

To make the explanation clear, let’s call the main directory under which the Apache files were copied $APACHE.

Unless you managed to get hold of a binary distribution of Apache, you must now compile it for your specific platform (presumably AIX, though most of the information in this article also applies to other versions of Unix). In order to compile the source files, you need to set compile-time options by editing a configuration file. In particular, you need to specify your system type. You then run a script that generates a makefile and a small piece of C code (modules.c) that’s used to compile Apache.

COMPILATION

Building the Apache Web server requires an ANSI C-compliant compiler. There is no way around this requirement. If your compiler does not meet this requirement, the build won’t work. Also, while the software may build correctly with a C++ compiler, making it ‘compileable’ with C++ is not a goal at this point, so I recommend you use an ANSI C compiler if this doesn’t work.

Apache supports the notion of ‘optional modules’. However, the server has to know which modules are to be part of the compilation process in order for those modules to work. This requires the generation of a short bit of code (*modules.c*) that simply contains a list of modules.

It is also necessary to choose the correct options for your platform:

1 Copy *$APACHE/src/Configuration.tmpl* as *Configuration* and then edit it. This file contains a list of options, settings for various 'rules', a section that lists the modules that are to be compiled in, and the names of files containing the modules. You may also need to:

   – Edit both *EXTRA_CFLAGS|LIBS|LDFLAGS|INCLUDES* and the 'rules'.

   – Uncomment lines corresponding to optional modules that you wish to include (these are in 'Module' at the bottom of the file) and (if necessary) add new lines corresponding to custom modules you have written. (See *API.html* for information on how this is done.)

   Note that 'DBM auth' has to be configured explicitly – if you want it, just uncomment the corresponding line.

2 Run the *$APACHE/src/Configure* script. This generates new versions of the makefiles and *modules.c*. If you want to maintain multiple configurations, you can specify a different configuration file, for example:

```
Configure -file Configuration.ai
```

3 Now compile the program:

```
$ make
```

The modules that are included in the Apache distribution are the ones that have been tested by various members of the Apache development group. Additional modules contributed by others or third-parties with specific needs or functions are available at:

```
http://www.apache.org/dist/contrib/modules/
```

There are instructions on this page for linking the modules into the core Apache code.

If you get a warning about a missing *regex.h* during compilation, set *WANTHSREGEX=yes* in *Configuration* and let the Apache Group know that you needed to do this for your OS either by filling out a problem report form at *http://bugs.apache.org/* or by sending an e-mail message to *apache-bugs@apache.org*. Include the output of the command **uname -a** in your report.

INSTALLING AND CONFIGURING RUN-TIME ENVIRONMENT

After compilation, you'll have a binary called **httpd** in your *$APACHE/src/* directory. If you received a binary distribution of Apache, you should have this file already.

The next step is to edit the configuration files for the server. In the top-level subdirectory, called *$APACHE/conf*, you should find distribution versions of the three configuration files: *srm.conf-dist*, *access.conf-dist*, and *httpd.conf-dist*. Copy them to *srm.conf*, *access.conf*, and *httpd.conf* respectively.

First edit *httpd.conf*. This sets up the server's general attributes, such as the port number, the user account under which it runs, etc. Next edit the file *srm.conf*. This sets up the root of the document tree, special functions, like server-parsed HTML or internal image map parsing, etc. Finally, edit the file *access.conf* – the least you should do is set the base access. Documentation for all these files is located at *http://www.apache.org/docs/*.

Finally, call **httpd** using the option **-f** and the full path of the *httpd.conf* file as follows:

```
$APACHE/src/httpd -f $APACHE/conf/httpd.conf
```

By default, the files *srm.conf* and *access.conf* are located by name, so, if you want to call them by other names, use the *AccessConfig* and *ResourceConfig* directives in *httpd.conf*.

MODIFYING THE SOFTWARE

Use the shell script **aadm.sh** to modify either the Apache or HTML source files – the location of the files is set by the following variables in the shell script:

- APACHE_SOURCE_DIR
- HTML_SOURCE_DIR.

ESTABLISHING DIRECTORY STRUCTURE

Broadly speaking, the directory structure needs to cater for the following types of document:

- Apache source files
- HTML source files.

Use the *DocumentRoot* directive to establish the main file-serving directory. Use the *Alias* directive to set a different location for static files, if required, and use the *ScriptAlias* directive to set a different location for executable files – again, if required.

EXAMPLE

```
    DocumentRoot     /usr/www/htdocs
```

This will serve files from the directory */usr/www/htdocs* and its subdirectories.

```
    DocumentRoot          /usr/www/htdocs
    Alias /user_logon     /usr/www/logon
```

This will serve files from the directory */usr/www/logon*, if the requested URL is:

```
    http://www.xxx.com/user_logon
```

PORT ASSIGNMENT

By default, the **httpd** (the http daemon) will listen to port 80, though this can be changed. Port numbers below 123 are restricted to *root*, so the **httpd** must be run using the *root* account if the default port number is used.

AADM.SH

```
#! /bin/ksh
###########################################################################
```

```
# Name     : aadm.sh (Apache admin)
#
# Overview : The script allows users to administer a subset of
#            Apache's settings.
#
# Notes    : 1. The script contains the following functions:
#               - main
#               - InitializeVariables
#               - DisplayMessage
#               - ProcessExit
#               - FormatUnderscores
#               - DisplayListOfValues
#               - ViewFile
#               - RootUser
#               - GetFileName
#               - PrintFile
#               - HandleInterrupt
#               - MoveCursor
#               - ProcessROOTMenu
#               - ProcessEDITMenu
#               - ProcessVIEWMenu
#               - ProcessRTOMenu
#               - StartHttpListener
#               - PerformSanityCheck
#               - StopHttpListener
#               - ShowHttpdProcessDetails
#               - RestartHttpdImmediately
#               - RestartHttpdAfterCurrentTask
#               - DisplayHttpdRuntimeOptions
#               - EditHttpdConfigurationFile
#               - EditHttpMakedConfigurationFile
#               - ShowCompiledInModuleNames
#               - ShowAvailableConfigurationDirectives
#               - CheckConfigurationFileForSyntax
#               - ProcessMainDirectory
#               - ProcessSubDirectory
#
#            2. The following directories must be set according
#               to requirements :
#               - HTTPD_DIR where httpd executable resides.
#               - APACHE_RUNTIME_DIR where sub-directories 'conf',
#                 'htdocs', 'logs', etc are found.
#               - APACHE_SOURCE_DIR where sub-directories 'src' etc
#                 are found.
#               - HTML_SOURCE_DIR
######################################################################
######################################################################
# Name     : InitializeVariables
#
# Overview : The function initializes all required variables.
```

```
##################################################################################
InitializeVariables ()
{
# locations - Apache root directory
APACHE_ROOT_DIR = "/home/ecatmgr/appl/apache_1.3.4"
# Apache source directory
APACHE_SOURCE_DIR = "${APACHE_ROOT_DIR}/src"
# html source directory
HTML_SOURCE_DIR = "${APACHE_ROOT_DIR}/htdocs"
# httpd daemon directory
HTTPD_DIR = "/usr/local/bin"
# directory containing sub-directories conf, logs, htdocs, etc
# httpd will be started using this location as follows:
# httpd -d "${APACHE_RUNTIME_DIR}"
APACHE_RUNTIME_DIR = "${APACHE_ROOT_DIR}"
# default httpd configuration file
HTTPD_CONFIG_FILE = "${APACHE_RUNTIME_DIR}/conf/httpd.conf"
HTTPD_CONFIG_TEMPLATE = "${APACHE_ROOT_DIR}/conf/httpd.conf-dist"
HTTPD_MAKEFILE_TEMPLATE = "${APACHE_ROOT_DIR}/Makefile.tmpl"
HTTPD_MAKE_CONFIG_FILE = "${APACHE_SOURCE_DIR}/Makefile.config"
DEFAULT_HTTPD_ERROR_LOG_FILE = "${APACHE_RUNTIME_DIR}/logs/error_log"
HTTPD_ERROR_LOG_FILE = ""
# temporary files
ERROR_FILE = "/tmp/aadm_$$.err"
LOV_FILE_1 = "/tmp/aadm_$$_1.lov"
LOV_FILE_2 = "/tmp/aadm_$$_2.lov"
APACHE_SOURCE_LIST = "/tmp/aadm_$$_apache.lst"
HTML_SOURCE_LIST = "/tmp/aadm_$$_html.lst"
TEMP_FILE_1 = "/tmp/aadm_$$_1.tmp"
TEMP_FILE_2 = "/tmp/aadm_$$_2.tmp"
TEMP_FILE_3 = "/tmp/aadm_$$_3.tmp"
# escape sequences
ESC = "\0033["
RVON = "\0033[7m"                  # reverse video on
RVOFF = "\0033[27m"                # reverse video off
BOLDON = "\0033[1m"                # bold on
BOLDOFF = "\0033[22m"              # bold off
BON = "\0033[5m"                   # blinking on
BOFF = "\0033[25m"                 # blinking off
# menu titles
ROOT_MENU = "${RVON}Apache Admin Main Menu${RVOFF}"
EDIT_MENU = "${RVON}Apache Admin Edit Menu${RVOFF}"
VIEW_MENU = "${RVON}Apache Admin View Menu${RVOFF}"
RUNTIME_MENU = "${RVON}Apache Admin Runtime Option Menu${RVOFF}"
# exit codes
SEC = 0
FEC = 1
# return codes
TRUE = 0
FALSE = 1
```

```
# message prefixes
SLEEP_DURATION = 4         #  seconds for sleep command
ERROR = "${RVON}${BON}aadm.sh:ERROR:${BOFF}"
INFO = "${RVON}aadm.sh:INFO: "
# messages
WORKING = "Working"
INTERRUPT = "Program interrupted - quitting"
NOT_TEXT_FILE = "\${FILE_TO_VIEW} is not a text file"
INVALID_OPTION = "Invalid entry"
PRINT_OK = "Successfully submitted the print job"
PRINT_NOT_OK = "Failed to submit the print job"
DIR_NOT_EXIST = "Directory \${DIR_NAME} does not exist"
OSERROR = "\${ERR_MSG}"
ROOT_USER = "The script must be executed from the root account"
OPTION_NOT_IMPLEMENTED = "Option not yet implemented"
HTTPD_NOT_FOUND = "Executable httpd not found in ${HTTPD_DIR}"
HTTPD_NOT_EXECUTABLE = "${HTTPD_DIR}/httpd is not executable"
LISTENER_NOT_STARTED = "Failed to start http listener"
LISTENER_STARTED = "Successfully started http listener"
ADVISE_CHILD = "Advising child processes to end current task and exit"
LISTENER_RESTARTED = "Successfully re-started http listener"
LISTENER_RUNNING = "Instances of http listener are already running"
LISTENER_NOT_RUNNING = "Instances of http listener are not running"
LISTENER_KILLED = "Stopped all instances of http listener"
LISTENER_NOT_KILLED = "Failed to stop all instances of http listener"
CONF_FILE_NOT_FOUND = "File \${FILE_TO_VIEW} does not exist"
FILE_NOT_FOUND = "Configuration file \${FILE} does not exist"
NO_FILE_SELECTED = "No file is selected for viewing"
QUIT_ACCESS_LOG_VIEWING = "Use ctrl-c to quit"
# define signals
SIGHUP = 1; export SIGHUP       # when session disconnected
SIGINT = 2; export SIGINT       # ctrl-c
SIGTERM = 15; export SIGTERM    # kill command
SIGTSTP = 18; export SIGTERM    # interactive stop (ctrl-z)
}
#######################################################################
# Name     : ProcessExit
# Overview : The function removes any temporary files and makes a
#            graceful exit.
# Input    : Exit code
#######################################################################
ProcessExit ()
{
# assign parameter
EXIT_CODE = "$1"
clear
# remove temporary files
rm -f ${LOV_FILE_1}
rm -f ${LOV_FILE_2}
rm -f ${APACHE_SOURCE_LIST}
rm -f ${HTML_SOURCE_LIST}
rm -f ${TEMP_FILE_1}
rm -f ${TEMP_FILE_2}
rm -f ${TEMP_FILE_3}
rm -f ${ERROR_FILE}
# exit gracefully
exit ${EXIT_CODE}
}
######################################################################
# Name     : HandleInterrupt
# Overview : The function calls ProcessExit().
# Notes    : 1. The function first evaluates the variable
#               $FUNCTION_NAME to establish in which function the
#               interrupt occurred. In certain functions, the
#               interrupt simply returns $TRUE and is ignored.
######################################################################
HandleInterrupt ()
{
if ["${FUNCTION_NAME}" = "ViewFile"]
then
    # reset the variable
    FUNCTION_NAME = ""
    return $TRUE
fi
DisplayMessage I "${INTERRUPT}" N
ProcessExit $FEC
}
######################################################################
# Name     : MoveCursor
# Input    : Y and X coordinates
# Overview : It moves the cursor to the required location (Y, X).
######################################################################
MoveCursor ()
{
YCOR = $1
XCOR = $2
echo "${ESC}${YCOR};${XCOR}H"
}
######################################################################
# Name     : DisplayMessage
# Overview : The function displays the incoming message.
# Input    : 1. Message type (E = Error, I = Information)
#            2. Error code
#            3. Message to be acknowledged flag (Y = yes, N = no)
# Notes    : 1. If the message is to be acknowledged, the function
#               displays the message and waits for an input as
#               acknowledgement. If the message is not to be
#               acknowledged, the function appends three dots
#               ('...') at the end of the message to be displayed
#               to indicate that it's not to be acknowledged.
```

```
###############################################################################
DisplayMessage ()
{
MESSAGE_TYPE = $1
MESSAGE_TEXT = "`eval echo $2`"
ACKNOWLEDGE_FLAG = "$3"
# set the message acknowledge flag
if ["${ACKNOWLEDGE_FLAG}" = ""]
then
    ACKNOWLEDGE_FLAG = "Y"
fi
MoveCursor 24 1
if ["${MESSAGE_TYPE}" = "E"]
then
    if ["${ACKNOWLEDGE_FLAG}" = "N"]
    then
        echo "`eval echo ${RVON}${ERROR}`${MESSAGE_TEXT}...${RVOFF}\c"
    else
        echo "`eval echo ${RVON}${ERROR}`${MESSAGE_TEXT}${RVOFF}\c"
    fi
else
    if ["${ACKNOWLEDGE_FLAG}" = "N"]
    then
        echo "`eval echo ${RVON}${INFO}`${MESSAGE_TEXT}...${RVOFF}\c"
    else
        echo "`eval echo ${RVON}${INFO}`${MESSAGE_TEXT}${RVOFF}\c"
    fi
fi
# examine message acknowledge flag
if ["${ACKNOWLEDGE_FLAG}" = "Y"]
then
    read DUMMY
else
    sleep ${SLEEP_DURATION}
fi
return ${TRUE}
}
###############################################################################
# Name     : DisplayHotKeys
# Overview : The function displays hotkeys.
###############################################################################
DisplayHotKeys ()
{
clear
echo "
               Hot Keys Details
               ================
            root = ProcessROOTMenu
            view = ProcessVIEWMenu
            edit = ProcessEDITMenu
            rto  = ProcessRTOMenu (runtime options)
"
read dummy
}
########################################################################
# Name     : FormatUnderscores
# Overview : Assigns an appropriate number of underscores ('=') to
#            the variable UNDERSCORE to be used in conjunction with
#            a header.
# Input    : Line containing the header
########################################################################
FormatUnderscores ()
{
# assign parameter
LINE = "$1"
# initialize UNDERSCORE
UNDERSCORE =
# initialize index
IND = 1
# get no of characters in $LINE
NO_CHARS = `echo "$LINE" | wc -c`
# subtract the carriage return
NO_CHARS = `expr $NO_CHARS - 1`
while ["$IND" -le "$NO_CHARS"]
do
    UNDERSCORE = "${UNDERSCORE} = "
    IND = `expr $IND + 1`
done
}
########################################################################
# Name        : PrintFile
# Description : Prints the named file.
# Input       : The name of the file to be printed.
########################################################################
PrintFile ()
{
FILE_TO_BE_PRINTED = $1
# print file
while true
do
    clear
    echo "Do you wish to print the output file (Y/N)?:\c"
    read REPLY
    REPLY = `echo $REPLY | tr "a-z" "A-Z"`
    case $REPLY in
        N) return $TRUE;;
        Y) break;;
        *) DisplayMessage E "${INVALID_OPTION}" N;;
    esac
done
```

```
# get printer name
while true
do
    clear
    echo "Enter printer name for lp command (q to quit):\c"
    read PRINTER
    case $PRINTER in
         "") : ;;
        q|Q) break;;
          *) lp -d$PRINTER ${FILE_TO_BE_PRINTED} > ${ERROR_FILE} 2>&1;
             if [$? -eq 0]
             then
                 DisplayMessage I "${PRINT_OK}" N;
                 break;
             else
                 DisplayMessage E "${PRINT_NOT_OK}" N;
                 ERR_MSG = `cat ${ERROR_FILE}`;
                 DisplayMessage E "${OSERROR}" Y;
             fi;;
    esac
done
}
#######################################################################
# Name     : RootUser
# Overview : Checks whether the user is root.
# Returns  : TRUE if the user is root
#            FALSE otherwise.
#######################################################################
RootUser ()
{
USER = `id | cut -d'(' -f2 | cut -d')' -f1`
if ["${USER}" = "root"]
then
    return $TRUE
else
    return $FALSE
fi
}
#######################################################################
# Name     : DisplayApacheOptions
# Overview : Displays Apache command line options.
# Input    : Apache options (h v V l L S)
# Returns  : TRUE or FALSE
# Notes    : 1. The function calls the following functions:
#               - DisplayMessage
#               - PrintFile
#               - FormatUnderscores
#######################################################################
DisplayApacheOptions ()
```

```
{
# assign parameter
P_OPTION = "$1"
# check for executable httpd
if [! -f "${HTTPD_DIR}/httpd"]
then
    DisplayMessage E "${HTTPD_NOT_FOUND}"
    return $FALSE
fi
# check for execute permission
if [! -x "${HTTPD_DIR}/httpd"]
then
    DisplayMessage E "${HTTPD_NOT_EXECUTABLE}"
    return $FALSE
fi
# initialize temporary file
DATETIME = `date "+%d/%m/%Y at %H:%M:%S"`
# process option
if ["${P_OPTION}" = "h"]
then
    HEADER = "Runtime options for httpd on ${DATETIME}"
    FormatUnderscores "${HEADER}"
    echo "  ${HEADER}     " > ${TEMP_FILE_1}
    echo "  ${UNDERSCORE}\n" >> ${TEMP_FILE_1}
    ${HTTPD_DIR}/httpd -h > ${TEMP_FILE_2} 2>&1
elif ["${P_OPTION}" = "v"]
then
    HEADER = "Installed Apache version number on ${DATETIME}"
    FormatUnderscores "${HEADER}"
    echo "  ${HEADER}     " > ${TEMP_FILE_1}
    echo "  ${UNDERSCORE}\n" >> ${TEMP_FILE_1}
    ${HTTPD_DIR}/httpd -v > ${TEMP_FILE_2} 2>&1
elif ["${P_OPTION}" = "V"]
then
    HEADER = "Httpd compile settings on ${DATETIME}"
    FormatUnderscores "${HEADER}"
    echo "  ${HEADER}     " > ${TEMP_FILE_1}
    echo "  ${UNDERSCORE}\n" >> ${TEMP_FILE_1}
    ${HTTPD_DIR}/httpd -V > ${TEMP_FILE_2} 2>&1
elif ["${P_OPTION}" = "l"]
then
    HEADER = "List of compiled-in modules on ${DATETIME}"
    FormatUnderscores "${HEADER}"
    echo "  ${HEADER}     " > ${TEMP_FILE_1}
    echo "  ${UNDERSCORE}\n" >> ${TEMP_FILE_1}
    ${HTTPD_DIR}/httpd -l > ${TEMP_FILE_2} 2>&1
elif ["${P_OPTION}" = "L"]
then
    HEADER = "List of configuration directives on ${DATETIME}"
    FormatUnderscores "${HEADER}"
    echo "  ${HEADER}     " > ${TEMP_FILE_1}
    echo "  ${UNDERSCORE}\n" >> ${TEMP_FILE_1}
    ${HTTPD_DIR}/httpd -L > ${TEMP_FILE_2} 2>&1
elif ["${P_OPTION}" = "S"]
then
    HEADER = "httpd Parsed Settings on ${DATETIME}"
    FormatUnderscores "${HEADER}"
    echo "  ${HEADER}     " > ${TEMP_FILE_1}
    echo "  ${UNDERSCORE}\n" >> ${TEMP_FILE_1}
    ${HTTPD_DIR}/httpd -S -f ${HTTPD_CONFIG_FILE} > ${TEMP_FILE_2} 2>&1
fi
# append details to header file
cat ${TEMP_FILE_2} >> ${TEMP_FILE_1}
# view the file
view ${TEMP_FILE_1}
PrintFile "${TEMP_FILE_1}"
return $TRUE
}
#######################################################################
# Name     : StartHttpListener
# Overview : The function starts the http listener.
# Notes    :1. The function calls the following function:
#              - DisplayMessage
#######################################################################
StartHttpListener ()
{
# are there any instances running?
if ps -eaf | grep "httpd" | grep -v "grep" > /dev/null 2>&1
then
    # http listener already running
    DisplayMessage E "${LISTENER_RUNNING}" N
    return $FALSE
fi
# does the executable exist?
if [! -f "${HTTPD_DIR}/httpd"]
then
    DisplayMessage E "${HTTPD_NOT_FOUND}" N
    return $FALSE
fi
# is it executable?
if [! -x "${HTTPD_DIR}/httpd"]
then
    DisplayMessage E "${HTTPD_NOT_EXECUTABLE}" N
    return $FALSE
fi
# start the listener
${HTTPD_DIR}/httpd -d "${APACHE_RUNTIME_DIR}" 2> ${TEMP_FILE_1}
if [$? -ne 0]
```

```
then
    ERR_MSG = `cat ${TEMP_FILE_1} | head -1`
    DisplayMessage E "${LISTENER_NOT_STARTED}" N
    DisplayMessage E "${OSERROR}" Y
    return $FALSE
else
    DisplayMessage I "${LISTENER_STARTED}" N
fi
return $TRUE
}
######################################################################
# Name     : StopHttpListener
# Overview : The function stops the http listener.
# Notes    : 1. The function calls the following function:
#               - DisplayMessage
######################################################################
StopHttpListener ()
{
# are there any instances running?
if ps -eaf | grep "httpd" | grep -v "grep" > /dev/null 2>&1
then
    # http listener running
    :
else
    DisplayMessage E "${LISTENER_NOT_RUNNING}" N
    return $FALSE
fi
# kill all listener processes
kill -TERM `cat ${APACHE_RUNTIME_DIR}/logs/httpd.pid` 2> ${TEMP_FILE_1}
sleep 2
if ps -eaf | grep "httpd" | grep -v "grep" > ${TEMP_FILE_1} 2>&1
then
    DisplayMessage E "${LISTENER_NOT_KILLED}" N
    ERR_MSG = `cat ${TEMP_FILE_1} | head -1`
    DisplayMessage E "${OSERROR}" Y
    return $FALSE
else
    DisplayMessage I "${LISTENER_KILLED}" N
    return $TRUE
fi
}
```

This article concludes in the next issue of AIX Update *with the remainder of this script.*

Arif Zaman
DBA/System Administrator
High-Tech Software (UK)

An information system for storage administrators

Good administration requires a historical database of how your storage has evolved. Such information is required for both capacity planning and problem determination. While the latter is easily overlooked, you should remember that it's not unusual for a filesystem or database tablespace to reach 100% utilization when a program malfunctions – without historical information, the storage administrator might easily choose to increase the size of filesystem or tablespace, which is the wrong course of action. A simple query to the storage database would allow the administrator to establish whether the volume of data had increased abnormally.

We have coded an application that simplifies the storage administrator's life. The application gathers daily storage information from a group of servers (clients). The data is stored in the clients and asynchronously transmitted and stored in a centralized SQL database (server). A Web server connected to the database allows the information to be accessed using an Internet browser. A graphic data representation simplifies data analyses. Reports, based on a SQL query, can be automatically generated by a script program and sent by e-mail to a distribution list.

The applications runs on AIX and gathers storage information from AIX and NT systems. The database was initially MySQL but I have recently changed to IBM DB2 Version 5.2. The application was written in Perl, a very flexible and powerful script language. Perl and IBM DB2 are supported in almost all operating systems, allowing the application to cover a wide range of operating systems. The Web Server runs in AIX and is Apache.

We have tested the application in AIX 4.2.1 to AIX 4.3.3, NT4 and NT5, Perl versions from 5.005_03 to 5.6, and DB2 Version 5.2.

THE AIX CLIENT

A Perl agent program (/stgmon/utils/get_aix_fs.pl) is scheduled by cron to run every hour of every day. It gathers filesystem size and free space. The data is then written to the application statistics directory (/stgmon/stats).

A daemon written in Perl (/stgmon/utils/load_aix_stats.pl), scheduled to run 10 minutes after the get_aix_fs.pl script, reads the statistics directory and inserts the data in the DB2 database. After a successful read and database insert, the data is deleted from the statistics directory.

THE NT CLIENT

A Perl agent program (c:\stgmon\get_nt_fs.pl) is scheduled by the NT scheduler to run every eight hours. It gathers drive sizes and free space information. The data is then written to the application statistics directory (c:\stgmon\stats).

A daemon written in Perl (c:\stgmon\utils\load_nt_stats.pl), running as a never-ending task, periodically reads the statistics directory and inserts the data in the DB2 database. After a successful read and database insert, the data is deleted from the disk.

THE AIX SERVER

The AIX server runs the DB2 database and the Apache Server. The database contains one table, stginfo.fs, which contains the filesystem/drive statistics. The Apache Web server contains a CGI script, /cgi-bin/fs.pl, which connects to the DB2 database and allows a Web user to consult the database.

The communication protocol between the clients (AIX and NT) and the server is TCP/IP.

AIX CLIENT CONFIGURATION

Install the DB2 AIX client. Create a DB2 instance user with the username ‘stadmusr’.

As root create the DB2 client instance:

```
    /usr/lpp/db2_Ø5_ØØ/instance/db2icrt -s client -u nobody stadmusr
```

Edit the /etc/services file and include:

```
db2i1          50000/tcp
db2i1a         50001/tcp
```

Logon as stadmusr and catalogue the DB2 database (note the use of the continuation character, '➤', below to indicate a formatting line break):

```
db2 catalog tcpip node stginfo remote <DatabaseHostName.domain>
➤  server db2i1
db2 catalog database stginfo at node stginfo
```

Test the connection using:

```
db2 connect to stginfo user stadmusr using dummy
```

Copy the application sources to the script directories:

- Copy *get_aix_fs.pl* to */stgadm/scripts/get_aix_fs.pl*
- Copy *load_aix_stats.pl* to */stgadm/scripts/get_aix_stats.pl*.

As root insert the following lines in the root's crontab:

```
0  * * * * /stgadm/scripts/get_aix_fs.pl      1>/dev/null
➤  2>/dev/null
10 * * * * /stgadm/scripts/load_aix_stats.pl  1>/dev/null
➤  2>/dev/null
```

THE NT CLIENT CONFIGURATION

Edit the C:\winnt\system32\drivers\etc\services file and include:

```
db2i1          50000/tcp
db2i1a         50001/tcp
```

Use the DB2 client configuration wizard to catalog the stginfo database. Or if you prefer, use the DB2 command window:

```
db2cmd> db2 catalog tcpip node stginfo remote
➤  <DatabaseHostName.domain> server db2i1
db2cmd> db2 catalog database stginfo at node stginfo
```

Test the connection using:

```
db2cmd> db2 connect to stginfo user stadmusr using dummy
```

Copy the application sources to the script directories:

- Copy *get_nt_drives.bat* to *c:\stgadm\scripts\get_nt_drives.bat*

- Copy *get_nt_drives.pl* to *c:\stgadm\scripts\get_nt_drives.pl*
- Copy *load_nt_stats.bat* to *c:\stgadm\scripts\load_nt_stats.bat*
- Copy *load_nt_stats.pl* to *c:\stgadm\scripts\load_nt_stats.pl*.

Set up the scheduler to run the filesystem data collector (get_nt_drives.bat) and the statistics loader (load_nt_stats.bat) every eight hours.

```
cmd> at 8:00 /every:M,T,W,Th,F,S,Su
     ➤  c:\stgadm\scripts\get_nt_drives.bat
cmd> at 16:00 /every:M,T,W,Th,F,S,Su
     ➤  c:\stgadm\scripts\get_nt_drives.bat
cmd> at 0:00 /every:M,T,W,Th,F,S,Su
     ➤  c:\stgadm\scripts\get_nt_drives.bat
cmd> at 8:10 /every:M,T,W,Th,F,S,Su
     ➤  c:\stgadm\scripts\load_nt_stats.bat
cmd> at 16:10 /every:M,T,W,Th,F,S,Su
     ➤  c:\stgadm\scripts\load_nt_stats.bat
cmd> at 0:10 /every:M,T,W,Th,F,S,Su
     ➤  c:\stgadm\scripts\load_nt_stats.bat
```

DATABASE CONFIGURATION

Install the DB2 UDB Version 5.2 with the development kit. Obtain the DBD::DB2 Perl module from CPAN and install it.

Create two AIX user accounts, one for the DB2 instance with username 'stginfo' and another for the DB2 client connection with username 'stadmuser'. Set 'dummy' as the user 'stadmuser' password.

As root create the DB2 instance:

```
aix> /usr/lpp/db2_05_00/instance/db2icrt -u nobody stginfo
```

In order for this change to take effect logoff and logon again.

Edit the /etc/services file and include:

```
db2i1          50000/tcp
db2i1a         50001/tcp
```

Logon with the db2 instance owner 'stginfo' and create the database with the command:

```
aix> db2start
aix> db2 -tf create_all.ddl
aix> db2set DB2COMM=tcpip
aix> db2 connect reset
aix> db2 force applications all
aix> db2stop
aix> db2start
```

WEB SERVER CONFIGURATION

Download and install the Apache Web server. You can download an installable AIX image from the BULL site:

```
http://www-frec.bull.com:80/cgi-bin/list_dir.cgi/download/out/
```

Download and install in the Apache Web server root directory the Michael Bostock 'JavaScript Graph Builder' from:

```
http://developer.netscape.com/docs/technote/javascript/graph/
```

I have introduced some changes in the **graph.js** javascript program. After install, replace this program by the program listed in this article.

The Apache default Web server configuration is sufficient for this application:

```
Resource                Directory
<root directory>        /usr/local/share/apache
CGI directory           <root directory>/cgi-bin/fs.pl
Document directory      <root directory>/htdocs
```

Copy the application sources to the Web server directories:

- Copy 'JavaScript Graph Builder' images to <root directory> /htdocs/images
- Copy fs.pl to <root directory>/cgi-bin/fs.pl
- Copy graph.js to <root directory>/htdocs.

As root set the AIX file permissions:

```
aix> chmod ugo=rx /usr/local/share/apache/htdocs
aix> chmod ugo=rx /usr/local/share/apache/htdocs/images
aix> chmod ugo=rx /usr/local/share/apache/cgi-bin
aix> chmod ugo=r /usr/local/share/apache/htdocs/images/*
aix> chmod ugo=rx /usr/local/share/cgi-bin/*
```

and start the Web server:

```
aix> /usr/local/bin/apachectl start
```

The application can be accessed in the URL:

```
http://<hostname.domain>/cgi-bin/fs.pl
```

SOURCE FILES

Create_all.ddl

```
create db stginfo
connect to stginfo
create table stginfo.fs
  (
   hostname char(8)     not null,
   sampled  date        not null,
   name     varchar(80) not null,
   fsmax    bigint      not null,
   size     bigint      not null,
   percent  int         not null
  );
alter table stginfo.fs add primary key (hostname,sampled,name);
grant select, insert, update, delete on  stginfo.fs to user stadmusr;
update dbm cfg using SVCENAME db2i1;
```

load_nt_stats.bat

```
c:\<perl path>\perl c:\stgadm\scripts\load_nt_stats.pl
```

load_nt_stats.pl

```
use Getopt::Std;
use vars qw( $opt_d );
use FileHandle;
use Fcntl ':flock'; # import LOCK_* constants
use DBI;
use DBI::DBD; # simple test to make sure it's okay
use POSIX;
use strict;
main:
{
 getopts('d');
 open(TRANSLOCK,"> c:\\tmp\\.load_nt_stats.lock");
 unless (flock(TRANSLOCK,LOCK_EX | LOCK_NB))
  { die "Unable to get lock file\n" }
 print "Lock file granted...\n" if $opt_d;
 load_db2_tbs_dir();
 flock(TRANSLOCK,LOCK_UN);
 close(TRANSLOCK);
}
```

```
sub load_db2_tbs_dir
{
 my $file;
 my $hostname=`hostname`;
 my ($dbh,$stmt,$sql,$vars,$erro,$errstr,$status,$sql_header);
 my (@parms);
 my ($sql_dup,$sql_post);
 chomp($hostname);
 $hostname=~ s/eth/tr/g;
 # Processa directorio de Alarmes
 print "Sending data...\n" if $opt_d;
 DBI->trace(Ø);
 print "Connecting to Database...\n" if $opt_d;
 $dbh = DBI->connect('DBI:DB2:stginfo','stadmusr','dummy');
 if ( $dbh eq "" )
  {
   print "Database connect failure...\n" if $opt_d;
   return;
  };
 $dbh->{PrintError}=Ø;
 opendir(DIR,"c:\\stgadm\\stats\\");
 while(defined($file=readdir(DIR)))
  {
   if ($file =~ /^\./)      {next;}
   open(FILE,"c:\\stgadm\\stats\\$file");
   $status="Fail";
   print "Processing file $file........\n" if $opt_d;
   $_=<FILE>;
   if (m%\»(.+)%)
     {
      print "\nAbout SQL:$1\n\n" if $opt_d;
      $stmt= $dbh->prepare($1);
      $stmt->execute;
      $erro=$dbh->err;
      print "\nDone: $erro\n\n" if $opt_d;
      if (($erro != Ø) and ($erro != -8Ø3) and (! $dbh->errstr =~ /
SQLØ8Ø3N/i))
        {
         $errstr=$dbh->errstr;
         print "ERRNO: $erro Desc: $errstr\n";
         $status="Fail";
         $stmt->finish;
         $dbh->disconnect;
         close(DIR);
         print "End of processing....\n" if $opt_d;
         return;
        }
      $stmt->finish;
      $_=<FILE>;
     }
   if (m%\#(.+)%)
    {
     $sql_header=$1;
     $status="ok";
     while($vars = <FILE>)
      {
       if (m%\@(.+)%)
         { $sql_dup=$1;}
        else
         {
          if (m%\«(.+)%)
           { $sql_post=$_;}
          else
           {
            $sql="$sql_header $vars";
            print "\nAbout SQL:$sql\n\n" if $opt_d;
            $stmt= $dbh->prepare($sql);
            $stmt->execute;
            $erro=$dbh->err;
            print "\nDone: $erro\n\n" if $opt_d;
            if (($erro != 0) and ($erro != -803) and (! $dbh->errstr =~
/SQL0803N/i))
             {
              $errstr=$dbh->errstr;
              print "ERRNO: $erro Desc: $errstr\n";
              $status="Fail";
             }
            $stmt->finish;
           }
        }
      }
    }
   close(FILE);
   if ($status eq "ok")
     {
      print "File $file processed with success........\n" if $opt_d;
      system("del c:\\stgadm\\stats\\$file");
     }
    else
     {print "File $file processed with error........\n" if $opt_d;}
  }
 close(DIR);
 print "Disconnecting from Database...\n" if $opt_d;
 $dbh->disconnect;
 print "End of processing....\n" if $opt_d;
}
```

get_nt_drives.bat

```
c:\<perl path>\perl c:\stgadm\scripts\get_nt_drives.pl
```

get_nt_drives.pl

```
use strict;
use Win32::DriveInfo;
use POSIX;
main:
{
 my($hostname,@mytime);
 my($hostname,$time) = @_;
 my $drive;
 my $SectorsPerCluster;
 my $BytesPerSector;
 my $NumberOfFreeClusters;
 my $TotalNumberOfClusters;
 my $FreeBytesAvailableToCaller;
 my $TotalNumberOfBytes;
 my $TotalNumberOfFreeBytes;
 my $TotalNumberOfFreeBytes;
 my $TotalNumberOfBytes;
 my @drives;
 my $VolumeName;
 my $VolumeSerialNumber;
 my $MaximumComponentLength;
 my $FileSystemName;
 my @attr;
 my $type;
 my $color;
 my $percent;
 my $workdir;
 $workdir="c:\\stgadm\\stats\\";
 $hostname=`hostname`;
 chomp($hostname);
 @mytime=gmtime(time);
 $mytime[4]+=1;
 $mytime[5]+=1900;
 $time="$mytime[5]-$mytime[4]-$mytime[3]";
 open(OUTPUT,"> $workdir.fs_stats") or die "\n\nError, unable to open
file: $workdir.fs_stats\n\n";
 print OUTPUT "#insert into stginfo.fs
(hostname,sampled,name,fsmax,size,percent) values\n";
 @drives = Win32::DriveInfo::DrivesInUse();
 foreach $drive (@drives)
  {
    $type = Win32::DriveInfo::DriveType($drive);
    next unless ($type eq 3);
   ($SectorsPerCluster,
    $BytesPerSector,
    $NumberOfFreeClusters,
    $TotalNumberOfClusters,
    $FreeBytesAvailableToCaller,
    $TotalNumberOfBytes,
    $TotalNumberOfFreeBytes) = Win32::DriveInfo::DriveSpace($drive);
   ($VolumeName,
    $VolumeSerialNumber,
    $MaximumComponentLength,
    $FileSystemName, @attr) = Win32::DriveInfo::VolumeInfo($drive);
   if ($TotalNumberOfBytes >0)
     { $percent=sprintf("%3d",100-100*$TotalNumberOfFreeBytes/
$TotalNumberOfBytes)}
    else
     { $percent='-1';}
   $TotalNumberOfBytes=-1 if $TotalNumberOfBytes eq '';
   $TotalNumberOfFreeBytes=-1 if $TotalNumberOfFreeBytes eq '';
   $TotalNumberOfBytes=floor($TotalNumberOfBytes/1024);
   $TotalNumberOfFreeBytes=floor($TotalNumberOfFreeBytes/1024);
   print OUTPUT
"('$hostname','$time','$drive',$TotalNumberOfBytes,$TotalNumberOfFreeBytes,
$percent)\n";
 }
 close(OUTPUT);
 if (-f "$workdir.fs_stats")
  {
   system("mv $workdir.fs_stats $workdir"."fs_stats.$time");
  }
 if (-f "$workdir"."fs_stats.$time" and
     -f "$workdir.fs_stats")
  {system("delete $workdir.fs_stats");}
}
```

graph.js

```
function Graph(width, height, stacked) {
this.stacked = stacked;
this.width = width || 400;
this.height = height || 200;
this.rows = new Array();
this.addRow = _addRowGraph;
this.addRowHref = _addRowHrefGraph;
this.addRowLabel = _addRowLabelGraph;
this.setXScale = _setXScaleGraph;
this.setXScaleValues = _setXScaleValuesGraph;
this.setTime = _setStartTimeGraph;
this.setDate = _setStartDateGraph;
this.build = _buildGraph;
this.setLegend = _setLegendGraph;
this.writeLegend = _writeLegendGraph;
this.offset = 0;
return this;
}
function _setLegendGraph() {
```

```
this.legends = arguments;
}
function _addRowLabelGraph() {
this.labelrows = new Array();
this.labelrows[this.labelrows.length] = new Array();
var labelrows = this.labelrows[this.labelrows.length-1];
for(var i = 0; i < arguments.length; i++)
labelrows[labelrows.length] = arguments[i];
}
function _addRowHrefGraph() {
this.hrefrows = new Array();
this.hrefrows[this.hrefrows.length] = new Array();
var hrefrow = this.hrefrows[this.hrefrows.length-1];
for(var i = 0; i < arguments.length; i++)
hrefrow[hrefrow.length] = arguments[i];
}
function _addRowGraph() {
this.rows[this.rows.length] = new Array();
var row = this.rows[this.rows.length-1];
for(var i = 0; i < arguments.length; i++)
row[row.length] = arguments[i];
}
function _rescaleGraph(g,posMax) {
g.posMax = posMax | 0, g.negMax = 0, g.c = 0;
for(var i = 0; i < g.rows.length; i++) {
for(var j = 0; j < g.rows[i].length; j++) {
g.c++;
if(g.rows[i][j] > g.posMax) g.posMax = g.rows[i][j];
if(g.rows[i][j] < g.negMax) g.negMax = g.rows[i][j];
}
}
g.vscale = g.height/(g.posMax-g.negMax);
g.hscale = Math.floor(g.width/g.c-1/g.rows[0].length);
}
function _stackRescaleGraph(g,posMax) {
var m, n, c = 0;
g.posMax = posMax | 100, g.negMax = 0;
for(var i = 0; i < g.rows[0].length; i++) {
m = 0; n = 0;
c++;
for(var j = 0; j < g.rows.length; j++) {
if(g.rows[j][i] > 0) m += g.rows[j][i];
else n += g.rows[j][i];
}
if(m > g.posMax) g.posMax = m;
if(n < g.negMax) g.negMax = n;
}
g.vscale = g.height/(g.posMax-g.negMax);
g.hscale = Math.floor(g.width/c)-1;
}
```

```
function _relRescaleGraph(g) {
var m, c = 0;
g.vscale = g.height/100;
for(var i = 0; i < g.rows[0].length; i++) {
m = 0;
c++;
for(var j = 0; j < g.rows.length; j++) {
if(g.rows[j][i] < 0) g.rows[j][i] = 0;
m += g.rows[j][i];
}
var s = 100/m; var k = 0;
for(var j = 1; j < g.rows.length; j++) {
g.rows[j][i] *= s;
g.rows[j][i] = Math.round(10*g.rows[j][i])/10;
if(j != 0) k += g.rows[j][i]*g.vscale;
}
g.rows[0][i] = Math.round(10*(g.height-k)/g.vscale)/10;
}
g.hscale = Math.floor(g.width/c)-1;
g.posMax = 100; g.negMax = 0;
}
function _writeLegendGraph() {
var st = "";
st += "<TABLE BORDER=1 CELLSPACING=0 CELLPADDING=4><TR><TD><FONT
FACE='Arial,Helvetica' SIZE=-1>";
for(var i = 0; i < this.legends.length; i++) {
if(!this.legends[i]) continue;
if(i >= this.rows.length) break;
st += "<IMG SRC=/images/"+i+".gif BORDER=1
HSPACE=3>"+this.legends[i]+"<BR>\n";
}
st += "</FONT></TD></TR></TABLE>";
return st;
}
function _buildRegGraph(g, doc) {
var str = "";
str += "<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0>\n";
if(g.title) {
str += "<TR>\n";
if(g.scale) str += "<TD COLSPAN=3></TD>\n";
if(g.yLabel) str += "<TD></TD>\n";
str += "<TH VALIGN=TOP HEIGHT=30 COLSPAN="+(g.c)+">\n";
str += "<FONT FACE='Arial,Helvetica' SIZE=-1>";
str += g.title;
str += "</FONT></TH></TR>\n";
}
if(g.yLabel) {
g.yLabel = g.yLabel.split("");
g.yLabel = g.yLabel.join("<BR>\n");
str += "<TR>\n";
```

```
var r = 2; if(g.negMax && g.posMax) r++;
str += "<TH ROWSPAN="+r+" ALIGN=LEFT WIDTH=20 NOWRAP>\n";
str += "<FONT FACE='Arial,Helvetica' SIZE=-1>"+g.yLabel+"</FONT></
TD>\n";
}
if(g.posMax > 0) {
if(!g.yLabel) str += "<TR>\n";
if(g.scale) str += _writeScaleGraph(g, 0, g.posMax);
str += "<TD VALIGN=BOTTOM";
if(g.bgColor) str += " BGCOLOR=\""+g.bgColor+"\"";
str += ">";
for(var j = 0; j < g.rows[0].length; j++) {
for(var i = 0; i < g.rows.length; i++) {
if(parseInt(g.vscale*g.rows[i][j]) > 0) {
if (g.hrefrows) str += "<A HREF="+g.hrefrows[i][j]+">";
str += "<IMG BORDER=0 SRC=\"/images/"+i+".gif\" ";
if (g.labelrows)
  {
   str += "ALT=\"";
   str += g.labelrows[i][j];
   str += "\" ";
  }
 else
  {
   str += "ALT=\"";
   if(g.legends && g.legends[i]) str += g.legends[i]+": ";
   str += (g.rows[i][j]+g.offset);
   if(g.dates) str += ", "+g.dates[j];
   str += "\" ";
  }

str += "WIDTH="+parseInt(g.hscale)+" ";
str += "HEIGHT="+parseInt(g.vscale*g.rows[i][j])+" ";
str += ">";
if (g.hrefrows) str += "</a>";
} else
str += "<IMG SRC=\"/images/clear.gif\" WIDTH="+parseInt(g.hscale)+"
HEIGHT=5>";
str += "</TD>\n<TD VALIGN=BOTTOM";
if(g.bgColor) str += " BGCOLOR=\""+g.bgColor+"\"";
str += ">";
}
str += "<IMG SRC=\"/images/clear.gif\" WIDTH=1 HEIGHT=5>";
}
str += "</TD>\n";
}
if(g.legends && g.posMax != 0) {
str += "<TD WIDTH=5 NOWRAP ROWSPAN=3></TD>\n";
str += "<TD ROWSPAN=3>";
str += g.writeLegend();
str += "</TD>\n";
}
if(g.scale || g.xScale) {
if(g.posMax) str += "</TR><TR>\n";
else str += "<TR><TD COLSPAN=2></TD>\n";
str += "<TD BGCOLOR=#000000 COLSPAN="+(g.c+1)+">";
str += "<IMG SRC=\"/images/black.gif\" HEIGHT=1 WIDTH=";
str += parseInt(g.rows[0].length*g.hscale)+" ></TD></TR>\n";
}
if(g.xScale && !g.negMax)
str += _writeXScaleGraph(g);
if(g.negMax < 0) {
if(g.posMax != 0 && !g.scale) str += "</TR>";
str += "<TR>\n";
if(g.scale) str += _writeNegScaleGraph(g, g.negMax, 0);
str += "<TD VALIGN=TOP";
if(g.bgColor) str += " BGCOLOR=\""+g.bgColor+"\"";
str += ">";
for(var j = 0; j < g.rows[0].length; j++) {
for(var i = 0; i < g.rows.length; i++) {
if(parseInt(g.vscale*g.rows[i][j]) < 0) {
str += "<IMG VSPACE=0 HSPACE=0 BORDER=0 ALIGN=TOP SRC=/images/"+i+".gif
WIDTH="+
parseInt(g.hscale)+" HEIGHT="+
parseInt(-1*g.vscale*g.rows[i][j]);
str += " ALT=\"";
if(g.legends && g.legends[i]) str += g.legends[i]+": ";
str += (g.rows[i][j]+g.offset);
if(g.dates) str += ", "+g.dates[j];
str += "\" >";
} else
str += "<IMG SRC=\"/images/clear.gif\" ALIGN=TOP BORDER=0
WIDTH="+parseInt(g.hscale)+
" HEIGHT=5>";
str += "</TD>\n<TD VALIGN=TOP";
if(g.bgColor) str += " BGCOLOR=\""+g.bgColor+"\"";
str += ">";
}
str += "<IMG SRC=\"/images/clear.gif\" ALIGN=TOP WIDTH=1 BORDER=0
HEIGHT=5>";
}
str += "</TD>\n";
if(g.legends && g.posMax == 0) {
str += "<TD WIDTH=5 NOWRAP ROWSPAN=2></TD><TD>";
str += g.writeLegend();
str += "</TD>\n";
}
}
str += "</TD></TR>\n";
if(g.xLabel) {
```

```
str += "<TR>\n";
if(g.scale) str += "<TD COLSPAN=3></TD>\n";
if(g.yLabel) str += "<TD></TD>\n";
str += "<TH COLSPAN="+g.c+" HEIGHT=25 VALIGN=BOTTOM><FONT
FACE='Arial,Helvetica' SIZE=-1>";
str += g.xLabel;
str += "</FONT></TH></TR>\n";
}
str += "</TABLE>\n";
doc.write(str);
}
function _setXScaleGraph(s, skip, inc) {
this.xScale = true;
this.s = s || 0;
this.skip = skip || 1;
this.inc = inc || 1;
}
function _setXScaleValuesGraph() {
this.xScale = true;
this.s = 0;
this.skip = 1;
this.inc = 1;
this.dates = new Array();
for(var i = 0; i < arguments.length; i++)
this.dates[this.dates.length] = arguments[i];
}
function _setStartTimeGraph(hour, min, skip, inc) {
this.xScale = true;
this.sTime = new Date(0, 0, 0, hour, min);
this.skip = skip || 1;
this.inc = inc || 1;
}
function _setStartDateGraph(month, day, year, skip, inc) {
this.xScale = true;
this.sDate = new Date(year, month-1, day);
this.skip = skip || 1;
this.inc = inc || skip || 1;
this.showDate = true;
}
function _setDatesArrayGraph(g) {
if(g.dates) return;
g.dates = new Array();
for(var i = 0; i < g.rows[0].length; i++) {
var dateStr = "";
if(g.sDate) {
if(g.showDay) {
eval('switch(g.sDate.getDay()) {'+
'case 0: dateStr += "Sun"; break;'+
'case 1: dateStr += "Mon"; break;'+
'case 2: dateStr += "Tue"; break;'+
'case 3: dateStr += "Wed"; break;'+
'case 4: dateStr += "Thu"; break;'+
'case 5: dateStr += "Fri"; break;'+
'case 6: dateStr += "Sat"; break;'+
'}');
dateStr += " ";
}
if(g.longDate && g.showDate) {
dateStr += g.sDate.getDate()+"-";
eval('switch(g.sDate.getMonth()) {'+
'case 0: dateStr += "Jan"; break;'+
'case 1: dateStr += "Feb"; break;'+
'case 2: dateStr += "Mar"; break;'+
'case 3: dateStr += "Apr"; break;'+
'case 4: dateStr += "May"; break;'+
'case 5: dateStr += "Jun"; break;'+
'case 6: dateStr += "Jul"; break;'+
'case 7: dateStr += "Aug"; break;'+
'case 8: dateStr += "Sep"; break;'+
'case 9: dateStr += "Oct"; break;'+
'case 10: dateStr += "Nov"; break;'+
'case 11: dateStr += "Dec"; break;'+
'}');
} else if(g.showDate) dateStr += (g.sDate.getMonth()+1)+"/
"+g.sDate.getDate();
if(g.showYear && g.showDate) {
if(g.longDate) dateStr += "-";
else dateStr += "/";
}
if(g.showYear) {
if(g.longYear) dateStr += g.sDate.getFullYear();
else dateStr += (g.sDate.getFullYear()%100);
}
g.sDate.setDate(g.sDate.getDate()+ g.inc);
} else if(g.sTime) {
var hrs = g.sTime.getHours();
if(!g.armyTime) {
var pm = false;
if(hrs == 0) { hrs = 12; }
else if(hrs >= 12) { if(hrs > 12) hrs -= 12; pm = true; }
} else
if(hrs < 10) hrs = "0" + hrs;
dateStr = hrs + ":";
var min = g.sTime.getMinutes();
if(min < 10) min = "0" + min;
dateStr += min;
if(!g.armyTime) { !pm ? dateStr += "am" : dateStr += "pm" ; }
g.sTime.setMinutes(g.sTime.getMinutes()+ g.inc);
} else dateStr = g.s+i*g.inc;
g.dates[i] = dateStr;
```

```
}
}
function _writeXScaleGraph(g) {
var st = "";
if(!g.c) g.c = g.rows[0].length*2-1;
st += "<TR>\n";
if(g.scale) st += "<TD COLSPAN=2></TD>\n";
if(g.yLabel) st += "<TD></TD>\n";
st += "<TD VALIGN=TOP COLSPAN="+(g.c+1)+">";
st += "<IMG SRC=/images/black.gif HEIGHT=10 WIDTH=1>";
st += "<IMG SRC=/images/clear.gif HEIGHT=1 WIDTH=1>";
var n = g.rows[0].length;
var mult = g.rows.length;
if(g.stacked || g.relative) mult = 1;
for(var i = 0; i < n; i++) {
st += "<IMG SRC=/images/clear.gif HEIGHT=1 WIDTH="+(g.hscale*mult)+">";
if((i+1) % g.skip)
st += "<IMG SRC=/images/clear.gif HEIGHT=10 WIDTH=1>";
else
st += "<IMG SRC=/images/black.gif HEIGHT=10 WIDTH=1>";
}
st += "</TD></TR><TR>\n";
if(g.scale) st += "<TD COLSPAN=3></TD>\n";
if(g.yLabel) st += "<TD></TD>\n";
var cspan = g.rows.length;
if(g.stacked || g.relative) cspan = 2;
cspan *= g.skip;
if(g.sDate || g.sTime) _setDatesArrayGraph(g);
var t = 0;
for(var i = 0; i < Math.floor(g.rows[0].length/g.skip+1); i++) {
st += "<TD VALIGN=TOP";
st += " COLSPAN="+cspan; t += cspan;
st += "><FONT FACE='Arial,Helvetica' SIZE=-4><I>";
st += g.dates[i*g.skip] || "";
st += "</I></FONT></TD>\n";
}
var len = g.rows[0].length;
if(!g.stacked) len *= g.rows.length;
if(i < Math.ceil(g.rows[0].length/g.skip)) {
st += "<TD VALIGN=TOP";
st += " COLSPAN="+(len-t);
st += "><FONT FACE='Arial,Helvetica' SIZE=-3><I>";
st += g.dates[i*g.skip];
st += "</I></FONT></TD>\n";
}
st += "</TR>\n";
return st;
}
function _writeNegScaleGraph(g, min, max) {
var h = Math.ceil(g.height/(g.posMax-g.negMax)*g.scale);
var p = -1*g.negMax/(g.posMax-g.negMax);
var n = Math.floor(g.height*p/h);
var st = "";
if(h < 15) {
if(!g.posMax)
alert("Warning! Scale is too small! Please make\nthe scale larger or
make the graph taller.");
st += "<TD></TD><TD></TD><TD></TD>\n"
return st;
}
st += "<TD VALIGN=TOP ALIGN=RIGHT>";
var H = h - 3;
for(var i = Ø; i < n; i++) {
st += "<FONT FACE=Arial,Helvetica SIZE=-3><I>"+(g.scale*-
1*(i+1)+g.offset)+"</I></FONT>";
st += "<IMG SRC=/images/clear.gif WIDTH=1 HEIGHT="+H+"><BR>\n";
}
st += "</TD>\n";
st += "<TD VALIGN=TOP>";
for(var i = Ø; i < n; i++) {
st += "<IMG SRC=/images/clear.gif WIDTH=1 HEIGHT="+(h-1)+"><BR>\n";
st += "<IMG SRC=/images/black.gif WIDTH=6 HEIGHT=1><BR>\n";
}
st += "</TD>\n";
st += "<TD VALIGN=TOP>";
st += "<IMG SRC=/images/black.gif WIDTH=1 HEIGHT="+(g.height*p)+">";
st += "<IMG SRC=/images/clear.gif WIDTH=1 HEIGHT="+(g.height*p)+">";
st += "</TD>\n"
return st;
}
function _writeScaleGraph(g, min, max) {
var h;
var p = g.posMax/(g.posMax-g.negMax);
h = Math.ceil(g.height/(g.posMax-g.negMax)*g.scale);
var n = Math.floor(g.height*p/h);
var st = "";
if(h < 15) {
alert("Warning! Scale is too small! Please make\nthe scale larger or
make the graph taller.");
st += "<TD ROWSPAN=2></TD><TD ROWSPAN=2></TD><TD></TD>\n"
return st;
}
st += "<TD VALIGN=BOTTOM ROWSPAN=2 ALIGN=RIGHT>";
var H = h - 3;
for(var i = -1; i < n; i++) {
st += "<FONT FACE=Arial,Helvetica SIZE=-3><I>"+(g.scale*(n-1)-
g.scale*i+g.offset);
if(g.relative) st += "%";
st += "</I></FONT>";
st += "<IMG SRC=/images/clear.gif WIDTH=1 HEIGHT="+H+"><BR>\n";
```

```
}
st += "</TD>\n";
st += "<TD VALIGN=BOTTOM ROWSPAN=2>";
for(var i = 0; i < n; i++) {
st += "<IMG SRC=/images/black.gif WIDTH=6 HEIGHT=1><BR>\n";
st += "<IMG SRC=/images/clear.gif WIDTH=1 HEIGHT="+(h-1)+"><BR>\n";
}
st += "<IMG SRC=/images/black.gif WIDTH=6 HEIGHT=1><BR>\n";
st += "</TD>\n";
st += "<TD VALIGN=BOTTOM>";
st += "<IMG SRC=/images/black.gif WIDTH=1 HEIGHT="+(g.height*p)+">";
st += "<IMG SRC=/images/clear.gif WIDTH=1 HEIGHT="+(g.height*p)+">";
st += "</TD>\n"
return st;
}
function _buildStackGraph(g) {
if(!g.c) g.c = g.rows[0].length*2-1;
var str = "";
str += "<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0>\n";
if(g.title) {
str += "<TR>\n";
if(g.scale) str += "<TD COLSPAN=3></TD>\n";
if(g.yLabel) str += "<TD></TD>\n";
str += "<TH VALIGN=TOP HEIGHT=30 COLSPAN="+(g.c)+">";
str += "<FONT FACE='Arial,Helvetica' SIZE=-1>";
str += g.title;
str += "</FONT></TH></TR>\n";
}
if(g.yLabel) {
g.yLabel = g.yLabel.split("");
g.yLabel = g.yLabel.join("<BR>\n");
str += "<TR>\n";
var rspan = 2; if(g.negMax && g.posMax) rspan++;
str += "<TH ROWSPAN="+rspan+" ALIGN=LEFT NOWRAP WIDTH=20>";
str += "<FONT FACE='Arial,Helvetica' SIZE=-1>"+g.yLabel+"</FONT></
TH>\n";
}
if(g.posMax > 0) {
if(!g.yLabel) str += "<TR>\n";
if(g.scale) str += _writeScaleGraph(g, 0, g.posMax);
for(var j = 0; j < g.rows[0].length; j++) {
str += "<TD VALIGN=BOTTOM";
if(g.bgColor) str += " BGCOLOR=\""+g.bgColor+"\"";
str += ">";
var k = 0, y = 0, drawn = false;
for(var i = 1; i < g.rows.length; i++)
if(parseInt(g.vscale*g.rows[i][j]) > 0)
k += parseInt(g.vscale*g.rows[i][j]);
if(g.rows.length > 0 && g.relative && (g.height > k)) {
str += "<IMG SRC=/images/0.gif WIDTH="+parseInt(g.hscale)+"
HEIGHT="+(g.height-k)+" ";
str += "ALT=\"";
if(g.legends && g.legends[0]) str += g.legends[0]+": ";
str += Math.round((g.height-k)/g.vscale*10)/10+"%";
if(g.dates) str += ", "+g.dates[j];
str += "\" ><BR>\n";
y++;
drawn = true;
}
for(var i = y; i < g.rows.length; i++) {
if(parseInt(g.vscale*g.rows[i][j]) > 0) {
str += "<IMG SRC=/images/"+i+".gif WIDTH="+parseInt(g.hscale)+"
HEIGHT="+
parseInt(g.vscale*g.rows[i][j])+" ";
str += "ALT=\"";
if(g.legends && g.legends[i]) str += g.legends[i]+": ";
str += g.rows[i][j];
if(g.relative) str += "%";
if(g.dates) str += ", "+g.dates[j];
str += "\" ><BR>\n";
drawn = true;
}
}
if(!drawn) str += "<IMG SRC=/images/clear.gif
WIDTH="+parseInt(g.hscale)+" HEIGHT=1>";
str += "</TD>\n";
str += "<TD";
if(g.bgColor) str += " BGCOLOR=\""+g.bgColor+"\"";
str += "><IMG SRC=/images/clear.gif WIDTH=1 HEIGHT=5></TD>\n";
}
}
if(g.legends && g.posMax != 0) {
str += "<TD WIDTH=5 NOWRAP ROWSPAN=3></TD>\n<TD ROWSPAN=3>";
str += g.writeLegend();
str += "</TD>\n";
}
if(g.scale || g.xScale) {
str += "</TR><TR>\n<TD BGCOLOR=#000000 COLSPAN="+(g.rows[0].length*2);
str += "><IMG SRC=/images/black.gif HEIGHT=1 WIDTH=";
str += parseInt(g.rows[0].length*g.hscale)+" ></TD></TR>\n";
}
if(g.xScale && !g.negMax) {
str += _writeXScaleGraph(g);
}
if(g.negMax < 0) {
if(g.posMax != 0) str += "</TR>\n";
str += "<TR>\n";
if(g.scale) str += _writeNegScaleGraph(g, g.negMax, 0);
for(var j = 0; j < g.rows[0].length; j++) {
str += "<TD VALIGN=TOP";
```

```
if(g.bgColor) str += " BGCOLOR=\""+g.bgColor+"\"";
str += ">";
var drawn = false;
for(var i = 0; i < g.rows.length; i++) {
if(parseInt(g.vscale*g.rows[i][j]) < 0) {
str += "<IMG VSPACE=0 HSPACE=0 BORDER=0 ALIGN=TOP SRC=/images/"+i+".gif
WIDTH="+
parseInt(g.hscale)+" HEIGHT="+
parseInt(-1*g.vscale*g.rows[i][j])+" ";
str += "ALT=\"";
if(g.legends && g.legends[i]) str += g.legends[i]+": ";
str += g.rows[i][j];
if(g.relative) str += "%";
if(g.dates) str += ", "+g.dates[j];
str += "\" ><BR>\n";
drawn = true;
}
}
if(!drawn) str += "<IMG SRC=/images/clear.gif
WIDTH="+parseInt(g.hscale)+" HEIGHT=1>";
str += "</TD>\n";
str += "<TD";
if(g.bgColor) str += " BGCOLOR=\""+g.bgColor+"\"";
str += "><IMG SRC=/images/clear.gif WIDTH=1 HEIGHT=5></TD>";
}
if(g.legends && g.posMax == 0) {
str += "<TD WIDTH=5 NOWRAP ROWSPAN=2></TD>\n<TD>";
str += g.writeLegend();
str += "</TD>\n";
}
}
str += "</TD></TR>\n";
if(g.xLabel) {
str += "<TR>\n";
if(g.scale) str += "<TD COLSPAN=3></TD>\n";
if(g.yLabel) str += "<TD></TD>\n";
str += "<TH COLSPAN="+g.c+" HEIGHT=25 VALIGN=BOTTOM><FONT
FACE='Arial,Helvetica' SIZE=-1>";
str += g.xLabel;
str += "</FONT></TH></TR>\n";
}
str += "</TABLE>\n";
doc.write(str);
}
function _adjustOffsetGraph(g) {
if(g.relative) return;
for(var i = 0; i < g.rows.length; i++)
for(var j = 0; j < g.rows[i].length; j++)
g.rows[i][j] -= g.offset;
}
```

```
function _buildGraph(d) {
doc = d || document;
if(!this.rows) return;
if(this.rows.length == 0) {
doc.write("<TABLE><TR><TD><TT>[empty graph]</TT></TD></TR></TABLE>\n");
return;
}
_adjustOffsetGraph(this);
if(this.xScale) _setDatesArrayGraph(this);
if(this.relative) {
_relRescaleGraph(this);
_buildStackGraph(this, doc);
return;
}
if(this.stacked) {
_stackRescaleGraph(this,this.Max);
_buildStackGraph(this, doc);
return;
}
_rescaleGraph(this,this.Max);
_buildRegGraph(this, doc);
}
```

load_aix_stats.pl

```
#!/usr/local/bin/perl
use Getopt::Std;
use vars qw( $opt_d );
use FileHandle;
use Fcntl ':flock'; # import LOCK_* constants
use DBI;
use DBI::DBD;      # simple test to make sure it's okay

use POSIX;
use strict;
main:
{
 getopts('d');
 $|=1;
 open(TRANSLOCK,"> /tmp/.load_aix_stats.lock");
 unless (flock(TRANSLOCK,LOCK_EX | LOCK_NB))
  { die "Unable to get lock file\n" };
 print "Lock file granted...\n" if $opt_d;
 load_stats_dir();
 flock(TRANSLOCK,LOCK_UN);
 close(TRANSLOCK);
}
sub load_stats_dir
{
 my $file;
 my $hostname=`hostname`;
 my ($dbh,$stmt,$sql,$vars,$erro,$errstr,$status,$sql_header);
 my (@parms);
 my ($sql_dup,$sql_post);
 chomp($hostname);
 #
 print "Sending data...\n" if $opt_d;
 DBI->trace(0);
 print "Connecting to Database...\n" if $opt_d;
 $ENV{DB2INSTANCE}='stadmusr';
 $dbh = DBI->connect('DBI:DB2:stginfo', 'stadmusr', 'dummy');
 if ( $dbh eq "" )
  {
   print "Database connect failure...\n";
   return;
  };
 $dbh->{PrintError}=0;
 opendir(DIR,"/stgadm/stats");
 while(defined($file=readdir(DIR)))
  {
   if ($file =~ /^\./)      {next;}
   open(FILE,"/stgadm/stats/$file");
   $status="Fail";
   print "Processing file $file........\n" if $opt_d;
   $_=<FILE>;
   if (m%\»(.+)%)
     {
      $stmt= $dbh->prepare($1);
      $stmt->execute;
      $erro=$dbh->err;
      if (($erro != 0) and ($erro != -803) and (! $dbh->errstr =~ /
SQL0803N/i))
        {
         $errstr=$dbh->errstr;
         print "ERRNO: $erro Desc: $errstr\n";
         $status="Fail";
         $stmt->finish;
         $dbh->disconnect;
         close(DIR);
         print "End of processing....\n" if $opt_d;
         return;
        }
      $stmt->finish;
      $_=<FILE>;
     }
   if (m%\#(.+)%)
    {
     $sql_header=$1;
     $status="ok";
     while($vars = <FILE>)
```

```
      {
       if (m%\@(.+)%)
         { $sql_dup=$1;}
        else
         {
          if (m%\«(.+)%)
           { $sql_post=$_;}
          else
           {
            $sql="$sql_header $vars";
            print "\nAbout SQL:$sql\n\n" if $opt_d;
            $stmt= $dbh->prepare($sql);
            $stmt->execute;
            $erro=$dbh->err;
            print "\nDone: $erro\n\n" if $opt_d;

            if (($erro != 0) and ($erro != -803) and (! $dbh->errstr =~
/SQL0803N/i))
             {
              $errstr=$dbh->errstr;
              print "ERRNO: $erro Desc: $errstr\n";
              $status="Fail";
             }
            $stmt->finish;
           }
        }
      }
    }
   close(FILE);
   if ($status eq "ok")
     {
      print "File $file processed with success........\n" if $opt_d;
      system("rm /stgadm/stats/$file");
     }
    else
     {print "File $file processed with error........\n" if $opt_d;}
  }
 close(DIR);
 print "Disconnecting from DB2...\n" if $opt_d;
 $dbh->disconnect;
 print "End of processing....\n" if $opt_d;
}
```

get_aix_fs.pl

```
#!/usr/local/bin/perl
use strict;
main:
{
 my(@fs,$res,@mytime);
 my $hostname;
 my $workdir="/stgadm/stats";
 $hostname=`hostname`;
 chomp($hostname);
 @mytime=gmtime(time);
 $mytime[4]+=1;
 $mytime[5]+=1900;
 GetFs(\@fs);
 open(OUTPUT,"> $workdir/.fs_stats") or die "\n\Error, unable to open
file: $workdir/.fs_stats\n\n";
 print OUTPUT "#insert into stginfo.fs
(hostname,sampled,name,fsmax,size,percent) values\n";
 foreach (@fs)
   {
    $res=&GetFsData($_,$hostname,"$mytime[5]-$mytime[4]-$mytime[3]");
   }
 close(OUTPUT);
 if (-f "$workdir/.fs_stats")
  {
   system("chown webadmin.web $workdir/.fs_stats");
   system("mv $workdir/.fs_stats $workdir/fs_stats.$mytime[5]-
$mytime[4]-$mytime[3]");
  }
 if (-f "$workdir/fs_stats.$mytime[5]-$mytime[4]-$mytime[3]" and
     -f "$workdir/.fs_stats")
  {system("rm $workdir/.fs_stats");}
}
sub GetFs
{
 my($fs) = shift;
 my($i);
 $i=0;
 open(SYSTEM,"lsvg -l `lsvg` | grep jfs |");
 while (<SYSTEM>)
  {
   if ( $_ =~ m%(\S+)\s+jfs\s+\S+\s+\S+\s+\S+\s+\S+\s+(\S+)%)
    { if (!( $2 eq "N/A" ))
        {
         $fs->[$i][0]=$2;
         $fs->[$i][1]="/dev/".$1;
         $i++;
        }
    }
  }
 close (SYSTEM);
}
sub GetFsData
{
 my($fs,$hostname,$time) = @_;
 my($i);
 $i=0;
 open(SYSTEM,"df -k $fs->[1] |");
 while (<SYSTEM>)
  {
   chomp();
   if ( $_ =~ m%(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\%\s+(\S+)\s+(\S+)\s+$fs-
>[0]%)
    {
     close (SYSTEM);
     print OUTPUT "('$hostname','$time','$fs->[0]',$2,$3,$4)\n";
    }
   if ( $_ =~ m%(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\%\s+(\S+)\s+(\S+)\s+$fs-
>[1]%)
    {
     close (SYSTEM);
     print OUTPUT "('$hostname','$time','$fs->[0]',$2,$3,$4)\n";
    }
  }
 close (SYSTEM);
}
```

Fernando Carvalho
System Engineer (Portugal)

Debugging a core file

When an application tries to do something that it is not allowed to, such as memory-address violations, illegal instructions, bus errors, and user-generated quit signals, the AIX operating system may protect itself and/or its users by sending the offending application a signal to ‘kill’ it. When this happens, most applications will generate a core file appropriately named ‘core’. A core file is an image of the application from memory and some additional information to help determine the application’s fault.

Core files are often considered a nuisance because they can be large and consume a large amount of disk space. Indeed, AIX has the **/etc/skulker** command that can be activated by root’s cron to watch for these files and periodically delete them. However, core files are actually useful at times in debugging problems with the application or with the AIX operating system. The core file was produced because of an abnormality and it is often very interesting to find out why this abnormality occurred and what can be done to prevent it from occurring again.

The simple procedure described in this article should allow you to gather some basic information to inform you, the third-party software developer, or possibly IBM support why the application is dumping its core.

SOFTWARE REQUIREMENTS

The procedure described uses the **dbx** debugger found in the *bos.adt.debug* software fileset. For core debugging to work reliably you must have access to the original application binary file and any shared libraries that the application was using. If you do not have access to the binary or the libraries, you will get error messages when you try debugging. You may still be able to debug the core file, but debugging is substantially more reliable if you have the original application binary file.

BASIC PROCEDURES

In general, the basic debug procedure for a core file is to determine the application that created the core file and then compare the core file with the original application binary file to determine which function in the application caused the problem.

Unlike other Unix operating systems, AIX's **file** command tells us only what file type the core file is, not which application created it:

```
# file core
core: data or International Language test
```

Use methods described in the March 1999 issue of *AIX Update* article titled *Who is to blame for the core file?* to determine the guilty application. Use **which** or **find** to find its exact directory location. After finding the binary file, invoke dbx with the original binary application file and core as arguments.

For our example we will use our program titled **myProgram** written in C which has an elementary memory access error. Not only once, it has the same error twice!

MYPROGRAM.C

```
#include    <strings.h>
void foo(char *);
main(void) {
    char    *p;
    foo(p);
    (void) strcpy(p, "data");
    return(Ø);
}
void) foo(char *p) {
    (void) strcpy(p, "data");
}
```

We compile it using the **xlc** compiler and the **–g** flag (more on the **–g** flag later).

```
# xlc -g -o myProgram myProgram.c
```

Its error is that we copy five bytes (‘data’ followed by a ‘\0’) to the memory location **p* without first having pointed *p* to a valid memory location. As surmised, executing the program produces a core file:

```
# myProgram
Segmentation fault(coredump)

# ls -l
-rw-r--r--   1 werner   staff      2677Ø Jul 13 19:58 core
-rwxr-xr-x   1 werner   staff       5729 Jul 13 19:58 myProgram
-rw -r—r--   1 werner   staff        166 Jul 13 19:3Ø myProgram.c
```

We call up the **dbx** debugger to find out what when wrong:

```
# dbx myProgram core
reading symbolic information ...
[using memory image in core]
Segmentation fault in strcpy.strcpy [myProgram] at Øx1ØØØØ39c
Øx1ØØØØ39c (strcpy+Øx1c) 7caØ1d2a      stswx   r5,rØ,r3
(dbx)
```

This informs us that the *strcpy()* function is at fault. Actually, we knew that the *strcpy()* was the fault, but we want to know exactly which of the two *strcpy()* derailed first. For this we use **dbx**’s *where* command:

```
(dbx) where
strcpy.strcpy() at Øx1ØØØØ39c
foo(p = (nil)), line 16 in "myProgram.c"
main(Øx1), line 8 in "myProgram.c"
(dbx) quit
```

Now we not only know that it was the first *strcpy()* in the subroutine *foo(),* we're also given the hint that *p* was a 'nil' (null) pointer.

STRIPPED APPLICATIONS

Core files of applications from which the symbol table information was removed (stripped) my be less useful. A symbol table is a table of symbolic names used in a program and their memory locations. The symbol table is part of the executable object generated by the compiler and used by debuggers to analyse the program. In stripped core files, **dbx** will not be able to generate a stack trace as shown in the following new example:

```
# xlc -o myProgram myProgram.c
# myProgram
Segmentation fault(coredump)
# dbx myProgram core
reading symbolic info ...warning: no source compiled with -g
[using memory image in core ]
Segmentation fault in strcpy.strcpy [myProgram] at 0x1000039c
0x1000039c (strcpy+0x1c) 7ca01d2a      stswx   r5,r0,r3
(dbx) where
strcpy.strcpy() at 0x1000039c
foo(0x0) at 0x10000334
main(0x1) at 0x100002cc
(dbx) quit
```

If you can recompile the application and use the **–g** compiler option, future core files from the application can be debugged more effectively. The **–g** option instructs the compiler to retain (that is, do not strip) the debug symbol table information from the application binary file. Because stripped binary files are approximately 15% smaller than unstripped files, most AIX operating system binary files are stripped of their debug information to conserve space. For more information see information on the strip command.

FOR MORE INFORMATION

For more information on the **dbx** command use an Internet search engine and search for '+AIX +dbx +core'.

Werner Klauser
Klauser Informatik (Switzerland)

AIX news

IBM has released Version 3.6.1 of its Tivoli SecureWay Privacy Manager, an access control tool, which now supports AIX and NT.

Designed to help implement privacy policies that protect customer's personal information, it builds on Policy Director Version 3.6's set of user-modifiable privacy roles and categories of data.

Privacy Manager also supports dynamic roles, to access decisions to be based on the relationship between the requestor and the subject of the data.

Applications running on AIX or NT can now implement calls to Privacy Manager for AIX and NT rather than requiring Solaris.

For further information contact your local IBM representative.
URL: http://www.tivoli.com/products.

* * *

IBM has announced Version 4.1 of its WebSphere Commerce Suite, MarketPlace Edition Version (WCS, MPE), providing the means to establish electronic marketplaces. It's designed, says the vendor, so that the technical infrastructure can be implemented and operational in weeks. It includes browser-based tools, preconfigured options, and a range of built-in capabilities.

Built on WCS Version 4.1, it can be adapted and extended to support new features, services, and business models and it supports notification of events such as RFQs, bids, and other requests.

It standards-based component architecture uses Java, XML, PKI, LDAP, and Trading Partner Agreements and can run on a single AIX server or scale over several.

The software provides an integrated set of flexible business functions on a common base platform for rapid application development, allowing for internal application integration with back-end systems and external business integration.

Key components include dynamic trading models (exchange, RFP/RFQ, and auctions), aggregated catalogue management framework, functions for secure membership registration and access control management, business intelligence, and reporting functions.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/webservers.

* * *

IBM has announced Version 2.2 of its WebSphere Payment Manager for commerce merchant activities, in particular for integrating payments into business processes and applications such as accounting, inventory management, and shipping and for CRM and customer call centres, where orders are placed and payments initiated.

It runs on AIX , NT, Solaris, and Windows 2000 and works with databases, which include DB2 UDB, Microsoft SQL Server, and Oracle.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/webservers.