119

AIX update

September 2005

In this issue

A itc PUBLICATION

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs $275.00 in the USA and Canada; £180.00 in the UK; £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 2000 issue, are available separately to subscribers for $24.00 (£16.00) each including postage.

AIX Update on-line

Code from *AIX Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/aix; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

Using smbclient to copy files from Unix to Windows servers

INTRODUCTION

There are several ways to copy files from Unix servers to Windows servers and *vice versa*. In the past I have used FTP, but recently I had to transfer files from an AIX server to a Windows server that did not have an FTP service running. After some research, I found a very useful utility from the Samba software suite called smbclient.

Smbclient is a client that can talk to an SMB/CIFS server. It offers an interface similar to that of the FTP program. Operations permitted include things like getting files from the local machine, putting files from the local machine to the server, retrieving directory information from the server, and so on.

Because smbclient has a command line interface, it is very easy to integrate into scripts.

In this article, I will describe how to install and use smbclient on an AIX server, and I will then discuss the flags I find most useful. Lastly, I will give a sample script that can be used to copy a file from an AIX server to a Windows server.

DOWNLOADING SAMBA

The Samba suite can be found and downloaded from IBM's AIX Toolbox Download page at http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html.

All software found on IBM's AIX Toolbox Download page is in RPM (Redhat Package Manager) format. So you will need to have the rpm.rte LPP installed before you can install the RPMs. The rpm.rte AIX installp format can be found at ftp://

ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/INSTALLP/ppc/rpm.rte.

Minimally, you will need the following RPMs to use smbclient:

1 samba-common-2.2.7-4.aix4.3.ppc.rpm.

2 readline-4.3-2.aix5.1.ppc.rpm

3 samba-client-2.2.7-4.aix4.3.ppc.rpm.

Optionally, you can install the Samba server – samba-2.2.7-4.aix4.3.ppc.rpm.

If you already have a version of Samba installed on your server but it is older than Version 2.2, I strongly suggest upgrading because the smbclient utility has a new **-A** flag that allows you to pass the name of a file for authentication on the remote server (discussed later on).

INSTALLING SAMBA

To install the RPMs you have just downloaded, simply type these commands:

```
# rpm -i samba-common-2.2.7-4.aix4.3.ppc.rpm
# rpm -i readline-4.3-2.aix5.1.ppc.rpm
# rpm -i samba-client-2.2.7-4.aix4.3.ppc.rpm
```

Optionally, you may also want to install the Samba server portion:

```
# rpm -i samba-2.2.7-4.aix4.3.ppc.rpm
```

If you wish to configure the Samba server or any other component not discussed in this article, I suggest you visit http://www.samba.org. There is also plenty of good documentation available on the Internet to help you.

USING SMBCLIENT

Using smbclient is quite simple. Here is the syntax of the **smbclient** command:

```
smbclient {servicename} [password] [-b <buffer size>] [-d debuglevel] [-
```

```
D Directory] [-U username] [-W workgroup] [-M <netbios name>] [-m
maxprotocol] [-A authfile] [-N] [-l logdir] [-I destinationIP] [-E] [-c
<command string>] [-i scope] [-O <socket options>] [-p port] [-R <name
resolve order>] [-s <smb config file>] [-T<c|x>IXFqgbNan] [-k]
```

You can find the details of all these options on the smbclient man page. I will describe the ones I find most useful, which will be used below in my script:

- {servicename} – *servicename* is the name of the service you want to use on the server. A service name takes the form *//server/service* where *server* is the NetBIOS name of the SMB/CIFS server offering the desired service, and *service* is the name of the service offered.
- -I IP-address – IP address is the address of the server to connect to. It should be specified in standard 'a.b.c.d' notation. Using this parameter will force the client to assume that the server is on the machine with the specified IP address and the NetBIOS name component of the resource being connected to will be ignored.
- -A authfile '' allows you to specify a file from which to read the username and password used in the connection. The format of the file is:

  ```
  username = <value>
  password = <value>
  domain = <value>
  ```

  Note: make certain that the permissions on the file restrict access from unwanted users.
- -c command string – *command string* is a semicolon-separated list of commands to be executed instead of prompting from stdin.

SAMPLE SCRIPTS USING SMBCLIENT

Here is a sample Korn shell script that I use in a cron to copy a file from an AIX server over to a Windows server. If the file copy to the remote Windows server fails, I send an e-mail alert to the administrator.

```
#!/bin/ksh
#
# Script to copy files from Unix servers to
#  Windows servers using smbclient

# Change the values of these variables to match your environment
WIN_SERVER_IP="172.26.240.8"   # IP address of the remote Windows server
WIN_SERVER="pt1826"            # Hostname of the remote Windows server
SHARE="C$"                     # Share name on the remote Windows server
LOGFILE="smbclient.log"        # Full path to logfile
TRANSFER_FILE="testfile.txt"
                    # Name of file to transfer to remote Windows server
AUTHFILE="smbauthfile"         # Full path to authentication file
BASENAME='basename $0'         # Name of this script
MAIL="/usr/bin/mail"           # Full path to mail command
MAILTO=admin@domain.com        # Who we send mail to in case of error

# DO NOT CHANGE ANYTHING AFTER THIS
#                                 UNLESS YOU KNOW WHAT YOU ARE DOING !!!

/usr/local/bin/smbclient //$WIN_SERVER/$SHARE -l $WIN_SERVER_IP -A
$AUTHFILE -c "put $TRANSFER_FILE $TRANSFER_FILE" > $LOGFILE

if [ $? -ne 0 ]
then
      echo "smbclient Error transferring file $TRANSFER_FILE to
$WIN_SERVER" >> $LOGFILE
       $MAIL -s "Script $BASENAME failed" $MAILTO < $LOGFILE
fi

# Clean up
/usr/bin/rm -f $LOGFILE
```

Note: you will have to change the values of the variables at the top of the script to match your environment.

Here is my sample authentication file called *smbauthfile*:

```
username = winuser
password = badpass
```

GOTCHAS

Here are some things to check for if you can't get smbclient to work correctly:

1 Verify that the server name and IP address match correctly.

2 Make sure the Windows share name is the actual name and not a subdirectory of the share. To verify the shares available on the Windows server, type:

```
# /usr/local/bin/smbclient -L windows_server_name
```

This command will return all the shares on the server named *windows_server_name*.

3 Ensure that your username has write permissions in the Windows share.

4 Ensure that your username and password are valid on the Windows server.

If you are still experiencing problems, I suggest you try the **smbclient** command interactively to make sure everything works as you expect.

```
# /usr/local/bin/smbclient -U winuser
```

Once the client is running, the user is presented with the prompt:

```
smb:\>
```

Now, run all your **smbclient** subcommands as you would in the script.

Still having problems with the script after you successfully run the commands interactively? Add the debug flag to the **smbclient** command:

```
-d|--debug=debuglevel
```

debuglevel is an integer from 0 to 10. If this parameter is not specified, the default value is zero.

The higher the value, the more detail will be logged to the log files about the activities of the server. At level 0, only critical errors and serious warnings will be logged. Level 1 is a reasonable level for day-to-day running – it generates a small amount of information about operations carried out.

Levels above 1 will generate considerable amounts of log

data, and should be used only when investigating a problem. Levels above 3 are designed for use only by developers and generate huge amounts of log data, most of which is extremely cryptic.

Elvio Prattico
Consultant
PRATTICO Consulting (Canada)

Considerations for implementation of point-in-time copy back-ups for online data

Enterprise Storage System arrays provide a way to perform a point-in-time copy of one set of storage volumes to a different set. The resultant copy is then used to serve either as a back-up of the original data or is utilized for data mining or reporting without affecting the performance of volumes containing the original data. Examples of such technologies are IBM ESS, TotalStorage FlashCopy, and EMC BCV.

In order to ensure the integrity of the filesystems created on the target volumes during point-in-time copy operations on AIX, it is recommended that the filesystems on the source volumes be unmounted prior to taking the point-in-time copy. When online (hot back-up) copies are required, the filesystems cannot be unmounted and alternative measures are required to ensure the integrity of the target volumes.

DETAILED PROBLEM DESCRIPTION

If any writes are occurring to the filesystems on the source volumes when the point-in-time copy is established, unexpected errors may occur, compromising the integrity of the filesystems on the target volumes, and making the target copy unusable.

The AIX filesystem logging mechanism has been designed to recover from a power failure or a system crash. In those situations, I/O to all volumes containing the user data, metadata, and filesystem logs stops at the same time. This situation creates a point-in-time image of the filesystem and JFS log, guaranteeing that when the filesystem is mounted and the log is replayed, all the filesystem metadata will be fixed (if required) to stay in a consistent state. In situations where writing operations continue to be applied to either the log or the filesystem itself, while writes to the other have been stopped, this guarantee cannot be made.

The result of replaying a log in the situation where all I/O does not stop at the same point-in-time is a filesystem that may contain metadata corruption. There will be no indication of the corruption at the time the copy is mounted; the mount operation will trigger the replay of the log, and the filesystem will be mounted. However, at some later time, when the inconsistent metadata is accessed, there will be problems. If the corruption is recognized as such, then the system may crash. If it is not recognized when it is used, then loss or corruption of user data may result.

The following methods should be used when performing a point-in-time copy of a mounted filesystem on AIX.

JFS2 FREEZE/THAW

The latest levels of AIX 5.2 and 5.3 provide a freeze/thaw function for the JFS2 filesystem. FlashCopy of a mounted filesystem is supported when the JFS2 freeze/thaw function is used. The freeze/thaw function is currently available for AIX 5.2 with APAR IY66043 or (IY59928 and IY59770). The 5.3 version is available in APAR IY59929 or maintenance level 1.

The following is a list of steps to be performed when this option is chosen:

1 Set the application to online back-up mode, if possible:

- For Oracle databases:
  - put the database (or selected table spaces) into online back-up mode (ALTER DATABASE BEGIN BACKUP or ALTER TABLSPACE … BEGIN BACKUP)
  - suspend Oracle I/O activity (ALTER SYSTEM SUSPEND).
- For DB2 UDB databases, issue the following command on all DB2 partitions residing on the filesystem to be frozen:

```
DB2 SET WRITE SUSPEND FOR DATABASE
```

2 Issue a **sync** command.

This step is not mandatory, but is recommended when filesystem cacheing is used in order to minimize the time required to flush dirty pages from the filesystem cache. If **sync** is not performed, dirty pages will be flushed by the **chfs freeze** operation in the step below. However, **sync** is a multi-threaded operation and can flush dirty pages faster than **chfs**, which is a single-threaded process.

3 Issue the file system **freeze** command:

```
chfs -a freeze=<timeout in seconds> /filesystemname
```

4 Issue the point-in-time copy command to the required LUNs of the underlying storage subsystem (those that contain filesystem data or filesystem log files). Wait for the point-in-time copies to complete.

5 Issue the filesystem thaw command:

```
chfs -a freeze=off  /filesystemname
```

6 Set the application back to normal mode:

- For Oracle databases:
  - resume Oracle I/O activity (ALTER SYSTEM RESUME)

– take the database (or selected table spaces) out of online back-up mode (ALTER DATABASE END BACKUP or ALTER TABLSPACE … END BACKUP).

- For DB2 UDB databases, issue the following command on all DB2 partitions residing on the filesystem to be thawed:

```
DB2 SET WRITE RESUME FOR DATABASE
```

7 Verify the integrity of the target volumes and filesystem (see section below).

STORAGE-BASED MULTI-VOLUME CONSISTENCY GROUPS

Some storage subsystems support multi-volume consistency groups that allow a single consistent point-in-time copy across multiple LUNs. For example, consistency groups are available as part of the Copy Services Version 2 features available with IBM storage arrays.

Consistency groups provide a point-in-time target copy that is comparable to what the source copy would look like following a power failure or system crash.

The following is a list of steps to be performed when this option is chosen:

1 Set the application to online back-up mode:

- For Oracle databases, put the database (or selected table spaces) into online back-up mode (ALTER DATABASE BEGIN BACKUP or ALTER TABLSPACE … BEGIN BACKUP).
- For DB2 UDB databases, issue the following command on all DB2 partitions residing on the filesystem to be frozen:

```
DB2 SET WRITE SUSPEND FOR DATABASE
```

2 Issue a point-in-time copy command for the consistency

group associated with a particular filesystem or set of raw devices. Wait for point-in-time copy operations to complete.

3 Set the application back to normal mode:

- For Oracle databases, take the database (or selected table spaces) out of online back-up mode (ALTER DATABASE END BACKUP or ALTER TABLSPACE … END BACKUP).
- For DB2 UDB databases, issue the following command on all DB2 partitions residing on the filesystem to be thawed:

```
DB2 SET WRITE RESUME FOR DATABASE
```

4 Verify the integrity of the target volumes and filesystem (see section below).

SINGLE VOLUME FILESYSTEMS OR RAW DEVICES

For filesystems residing entirely on a single LUN (logs as well as data), point-in-time 'consistent' target copies can be created without the use of consistency groups. This technique can be useful, for instance, for storage subsystems that do not support consistency groups. It can also be used when raw devices, rather than filesystems, are used to contain user's data.

The following is a list of steps to be performed when this option is chosen:

1 Set the application to online back-up mode:

- For Oracle databases, put the database (or selected table spaces) into online back-up mode (ALTER DATABASE BEGIN BACKUP or ALTER TABLSPACE … BEGIN BACKUP).
- For DB2 UDB databases, issue the following command on all DB2 partitions residing on the filesystem to be frozen:

```
DB2 SET WRITE SUSPEND FOR DATABASE
```

2 Issue a point-in-time copy command for the LUN associated with a particular filesystem or set of raw devices. Wait for point-in-time copy operations to complete.

3 Repeat the previous steps serially or in parallel for any other single volume filesystems or set of raw devices that have to be copied.

4 Set the application back to normal mode:

- For Oracle databases, take the database (or selected table spaces) out of online back-up mode (ALTER DATABASE END BACKUP or ALTER TABLSPACE … END BACKUP).
- For DB2 UDB databases, issue the following command on all DB2 partitions residing on the file system to be thawed:

```
DB2 SET WRITE RESUME FOR DATABASE
```

5 Verify the integrity of the target volumes and filesystem (see section below).

VERIFY THE INTEGRITY OF THE COPIED FILESYSTEM

The following steps have to be performed prior to mounting and accessing the copied filesystem:

1 Execute command **logredo /dev/logvol** for the JFS logical volumes containing logs for each filesystem that has been copied. If the 'log wrap' error is displayed by the **logredo** command, or is recorded in the system error log, the filesystem(s) using the particular JFS log is not useful and should be discarded. In this case, the filesystem log for that filesystem needs to be extended, and a new point-in-time copy must be created.

The following is a typical sample of a JFS log wrap error condition. Please note that the problematic log file is identified by major and minor device numbers presented in hexadecimal notation.

```
---------------------------------------------------------------------
LABEL:            J2_LOG_WRAP_STOP
IDENTIFIER:       854D3B24

Date/Time:        Tue Jun 14 23:10:08 IDT
Sequence Number: 19956
Machine Id:       00579FFA4C00
Node Id:          testhost
Class:            O
Type:             INFO
Resource Name:    SYSJ2

Description
JFS2 LOGGING IS BACK TO NORMAL

Detail Data
JFS2 LOG MAJOR/MINOR DEVICE NUMBER
0039 0002
---------------------------------------------------------------------
LABEL:            J2_LOG_WRAP_START
IDENTIFIER:       4C41C0D0

Date/Time:        Tue Jun 14 23:10:06 IDT
Sequence Number: 19955
Machine Id:       00579FFA4C00
Node Id:          testhost
Class:            O
Type:             INFO
Resource Name:    SYSJ2

Description
JFS2 LOG RECORDS FORCED OVERWRITTEN

Probable Causes
LOG SIZE IS TOO SMALL

      Recommended Actions
      INCREASE THE SIZE OF LOG DEVICE

Detail Data
JFS2 LOG MAJOR/MINOR DEVICE NUMBER
0039 0002
---------------------------------------------------------------------
```

2 Execute a full **fsck** command on each of the copied filesystem(s). If **fsck** reports any uncorrectable errors, the copied filesystem should be discarded.

The above steps are necessary to ensure the integrity of the

filesystem(s) including metadata and user data. However, it is possible that data will not be consistent from the application point of view because some of the information has been held in filesystem cache or buffer cache managed by the application (for instance Oracle buffer cache) and was not written to the disk prior to the point-in-time copy operation. In this case additional, application-specific, recovery steps will be required.

For Oracle databases, for instance, a 'crash recovery' operation must be performed to bring the copy into a database-'consistent' state. Oracle 'crash recovery' requires redo log (online and/or archive) and control file information. Depending on the particular Oracle back-up/recovery scenarios being implemented, this may require that point-in-time copies of redo logs and/or control files be made in addition to point-in-time copies of Oracle .dbf files, so that necessary log and control file information is available for recovery.

REFERENCES

1 *Clarification of Supported and Unsupported Methodology for Flashcopy Backups of Mounted AIX Filesystems*, IBM Technote 1475.

2 *Requirements for Implementing an Online Copy of a DB2 UDB Database Utilizing IBM ESS TotalStorage FlashCopy on the AIX Platform*, IBM Technote 1191417.

3 *Using IBM ESS TotalStorage FlashCopy or Similar for Oracle 10g on AIX*, Oracle Bulletin 300225.1.

4 *A Unix Perspective on Oracle Archive Redo Log Files*, Mark Bole, *System Administration* magazine, July 2005.

Alex Polak
System Engineer
APS (Israel)

Understanding the calendar command

AIX has an interesting command to display, on a day-by-day basis, things you have scheduled for the day. The **calendar** command can help you determine the day's tasks, and will also show you the next day's tasks as well.

Various implementations of the command can also display entries in a master calendar, perhaps supplied by your system administrator, as well as automatically mail entries in other users' calendars to those users.

CALENDAR COMMAND BASICS

In its most basic form, entering **calendar** will search the current directory for a file called *calendar* and will write to the console any and all lines containing dates matching today's and tomorrow's date. If **calendar** is run on a Friday, you will get any available results for Friday, Saturday, Sunday, and Monday.

You can run **calendar** any time you would like to see today's and tomorrow's events, or you can run the command from a shell script that executes daily or whenever you log into your system.

CALENDAR DATE FORMATS

There are several acceptable date formats that the **calendar** command recognizes, as well as many on which the command will not report. The command will accept the following date formats, with the noted conditions and exceptions:

1 A numeric month and a numeric day with the forward slash between.

Examples: 4/18 05/23 11/05 12/3

- Either month or day designation may have one or two digits.

- The day may *not* precede the month, such as 25/03 for 25 March.
- Dates cannot contain parentheses, such as (4/18) or (05/23/05).

2 A three-character alphabetic abbreviation of the month and a numeric day.

Examples: apr 18 May. 23 nov.05 Dec/3

- The month need not have initial capitalization.
- The month cannot be all caps, such as JUL 12
- There need not be a space between the period (full stop), if used, and the day.
- A forward slash may be used between the month and day.
- Ordinal designations cannot be specified, such as Mar 17th, Apr 3rd, and Oct. 22nd.

3 Fully spelled out month and numeric day.

Examples: April 18 August 23 November 05 December 3

- The month *must* have initial capitalization.
- The month cannot be all caps, such as NOVEMBER 26.
- The day may *not* precede the month, such as 18 April.

4 Numeric month, day, and 2-digit year, with forward slashes.

Examples: 04/18/05 5/23/05 11/5/05

- The year may *not* have four digits, such as 04/18/2005.

A NOTE ABOUT YEARS

You can keep adding to a calendar file as your schedule grows, but be careful if your calendar contains numerals for

the year. Though the command can recognize that a two digit number following a day may be a valid and acceptable date format for its purposes, it does not make its determination of whether or not the year is valid to display its results. In other words, if today is 15 July 2005, and you have the following entries across a broad, multi-year calendar file, they will all display if the calendar command is executed today:

07/15/04 – Meeting with Bob – 10:00

07/15/05 – Dental appointment 2:45

07/15/06 – New project release to begin.

MULTIPLE DAYS IN ONE ENTRY

You can have multiple days in one line entry. Suppose a colleague was scheduled to be in a class on three consecutive Thursdays. You could have an entry as follows:

07/07 07/14 07/21 – Mary in class

You would see that line entry if you entered the **calendar** command on any of the following dates:

Wed July 6 or Thu July 7

Wed July 13 or Thu July 14

Wed July 20 or Thu July 21.

This can be useful if certain calendar entries recur over several dates and you do not want a line entry for each one.

USING THE WILDCARD CHARACTER

You can use the asterisk wildcard character in your calendar file to cause data to be displayed every month for that desired day each month.

Suppose you had a report due at the end of each month, for which you wanted a reminder on or about the 25th. You could have an entry that said:

*/25 - Prepare Monthly Progress Report

You would see that line displayed on the 24th and the 25th of each month, and as early as the 22nd if the desired day fell on a Monday or on a weekend.

You must use a slash between the asterisk and the numeric day (for example * 25 would not be valid.) Also, you cannot specify the day as the wildcard character, such as 03/*, intending to display a line on every day of March.

CALENDAR COMMAND EXAMPLES

Suppose your calendar file contained the following entries:

06/15 06/22 06/29 – Team meeting preparation

Wed 06/22 – Meeting with Douglas, 2:00

Thu 06/23 – Doctor appointment, 3:15

Fri 06/24 – Meet with boss

Sat 06/25 – Jimmy's softball game

Mon 06/27 – Project meeting, 9:00

*/22 – prepare for monthly Board meeting

Here are the results you would see if you entered the **calendar** command on the following days:

If entered on Wednesday, 06/22:

06/15 06/22 06/29 – Team meeting preparation

Wed 06/22 – Meeting with Douglas, 2:00

Thu 06/23 – Doctor appointment, 3:15

*/22 – prepare for monthly Board meeting

Note that you would see your Team meeting preparation because 06/22 is included in the multi-date entry. You would see your meeting with Douglas scheduled for today, and a reminder for your doctor appointment tomorrow. Finally, you

would see your monthly reminder to prepare for the Board meeting.

If entered on Thursday, 06/23:

Thu 06/23– Doctor appointment, 3:15

Fri 06/24 – Meet with Boss

Note that you would see your doctor appointment for today, and a reminder to meet your boss tomorrow.

If entered on Friday, 06/24:

Fri 06/24 – Meet with boss

Sat 06/25 – Jimmy's softball game

Mon 06/27 – Project meeting, 9:00

Note your meeting with the boss today, and, because it's Friday, you will see entries for Saturday through Monday; in this case, Jimmy's game and your Monday morning meeting.

FLAGS FOR THE CALENDAR COMMAND

The **calendar** command has one flag, the minus (-) character, which executes the command for all users who have a calendar file in their home directory. The minus flag is typically run by the system administrator, possibly from a daily script.

You must set your calendar file to be readable by the issuer of **calendar** in order to be reminded using this method.

When the minus flag is used, reminders are sent to the user by using the **mail** command rather than displaying them on their terminal.

EXAMPLES FOR THE MINUS FLAG

Suppose other users had calendar files in their HOME directories as follows. Note that each contains one of the supported calendar formats.

- /home/eddie/calendar

  6/21 manager's meeting 2:00

  6/22 Emily's surprise birthday party – rm. 207

  6/23 work from home
- /home/betty/calendar

  jun 21 project proposal due

  jun 22 Emily's surprise birthday party – rm. 207

  jun 23 car in shop
- /home/marty/calendar

  06/21/05 meet with Emily to discuss project 9:00

  06/22/05 Emily's surprise birthday party – rm. 207

  06/23/05 half day vacation – afternoon
- /home/emily/calendar

  June 21 meet with Marty to discuss project 9:00

  June 22 taking the day off for my birthday

  June 23 Diane's piano recital 3:00
- /home/tommy/calendar

  06/21 06/22 06/23 – in class all day

When the superuser issued the command **calendar -** (either manually or via a daily script,) on 21 June, each user's calendar file located in their home directory would get read individually, and data would be sent to each of them using the **mail** command to remind them of any entries they had for 21 June and also for the next day, 22 June.

The results would be as follows (for the command issued on 21 June):

1 For 21 June entries: Eddie would receive notice that he

has a manager's meeting, Betty would learn that her proposal was due, Marty and Emily would be reminded of their mutual meeting, and Tommy would learn of his all-day class.

2 For June 22 (tomorrow's) entries: all but Tommy would be reminded of the surprise party for Emily, who will be reminded to take off that day.

CROSS-USER CALENDARS

Your system administrator may have created a site calendar to help remind you of dates common to your organization, such as site holidays, general meetings, and anniversaries.

If so, ask your administrator for the name of that calendar file and put the following line as the first line of your local calendar file:

```
#include <admin_calendar>
```

where *admin_calendar* is the name of the file created for you. Be sure to surround the name by the < and > characters.

For example, suppose your administrator created a file called */home/allusers_calendar* containing the following lines:

06/27 – site meeting, Ace Hotel

06/28 – company 10-year anniversary picnic

06/29 – site holiday

To see these entries, add the following line to the top of your calendar file:

```
#include </home/allusers_calendar>
```

Then, if you were to execute the **calendar** command on 28 June, you would see a reminder of today's picnic and tomorrow's site holiday, as well as any other entries normally displayed in your own calendar.

SUMMARY

The **calendar** command can become a valuable tool to help keep track of your ever-changing schedules.

David Chakmakian
Software Engineer (USA)

New helpful AIX tools

Some useful but unknown commands are coming along with AIX 5.1 that make process information more trackable.

Freeware tools like lsof (list open files) are no longer necessary.

All the programs can be found in the content of the *bos.perf.proctools* LPP:

```
lslpp -f bos.perf.proctools

  Fileset               File
  ----------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.perf.proctools 5.3.0.0
                        /usr/bin/procsig
                        /usr/bin/procmap
                        /usr/bin/proccred
                        /usr/bin/procstack
                        /usr/bin/procstop
                        /usr/bin/procflags
                        /usr/bin/procwait
                        /usr/bin/procldd
                        /usr/bin/procrun
                        /usr/bin/proctree
                        /usr/bin/procwdx
                        /usr/bin/procfiles
```

DESCRIPTION

The source for the new tools is the new */proc* filesystem, which provides a mechanism to control processes. It gives access

to information about the current state of processes in binary form. The commands in *bos.perf.proctools* convert the information to ASCII reports.

Most of them take a list of process IDs as input. We can use the shell expansion */proc/** to call the tools.

The commands gather the information from the */proc* directory tree of the specified processes and display it on standardout. The proctools **procrun** and **procstop** start and stop a process using the new interface of the */proc* filesystem:

```
df -k /proc

Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/proc                   -         -    -         -     -  /proc
```

1 The following command displays the credentials of processes (effective, real, saved user IDs, and group IDs):

```
proccred [ ProcessID ] ...
```

Examples:

```
proccred 233628
233628: e/r/suid=288  e/r/sgid=1
```

2 This command reports information about all file descriptors opened by the processes:

```
procfiles [ -F ] [ -n ][ ProcessID ] ...
```

-n displays the names of the files referred to by file descriptors.

-F forces procfiles to take control of the target process even if another process has control.

Examples:

```
procfiles 233628
233628 : sshd: it_user@pts/0 A
Current rlimit: 2000 file descriptors
5: S_IFIFO mode:00 dev:65535,65535 ino:404464640 uid:288 gid:1 rdev:0,0
   O_RDONLY | O_NONBLOCK
6: S_IFIFO mode:00 dev:65535,65535 ino:404464640 uid:288 gid:1 rdev:0,0
```

```
   O_WRONLY | O_NONBLOCK

procfiles -n 418024
418024 : /usr/sbin/rsct/bin/ctcasd
Current rlimit: 2000 file descriptors
0: S_IFCHR mode:00 dev:10,5 ino:4150 uid:0 gid:0 rdev:2,2
   O_RDONLY  name:/dev/null
1: S_IFCHR mode:00 dev:10,5 ino:4150 uid:0 gid:0 rdev:2,2
   O_WRONLY  name:/dev/null
2: S_IFREG mode:0311 dev:10,8 ino:7 uid:0 gid:0 rdev:0,0
   O_WRONLY size:0  name:/tmp/.sec3994
```

3 The next command prints the */proc* tracing flags, the pending and held signals, and other */proc* status information for each thread in the specified processes:

```
procflags [ -r ] [ ProcessID ] ...
```

-r displays the current machine register's state if a process is stopped in an event of interest.

Examples:

```
procflags  233628
233628 : sshd: it_user@pts/0 A
data model = _ILP32 flags = PR_FORK
/757933: flags = PR_NOREGS
```

```
procflags -r 233628
233628 : sshd: it_user@pts/0 A
data model = _ILP32 flags = PR_FORK
/757933: flags = PR_ASLEEP | PR_NOREGS
Not stopped, can't show registers
```

4 The following command lists the dynamic libraries loaded by processes, including shared objects explicitly attached using dlopen():

```
procldd [ -F ] [ ProcessID ] ...
```

-F forces **procldd** to take control of the target process even if another process has control.

Examples:

```
procldd 233628
233628 : sshd: it_user@pts/0 A
sshd
/usr/lib/libC.a[shrcore.o]
/usr/lib/libC.a[ansicore_32.o]
```

```
/usr/lib/libC.a[shr.o]
/usr/lib/libcrypt.a[shr.o]
/usr/lib/libz.a[libz.so.1]
/opt/freeware/lib/libcrypto.a[libcrypto.so.0]
/usr/lib/libpthreads.a[shr_xpg5.o]
/usr/lib/libpthreads.a[shr_comm.o]
/usr/lib/libc.a[shr.o]
```

5 The following command shows the address space map of processes:

```
procmap [ -F ] [ ProcessID ] ...
```

Example:

```
procmap 565424
565424 : sqlplus -s TAPPS/****  @/oracle/test11iappl/ar/11.5.0/patch/
115/sql/sarhpg08.sql
10000000        4264K  read/exec         sqlplus
20000c88         267K  read/write        sqlplus
d007f0f8           2K  read/exec         /usr/lib/libcrypt.a
f0238508           0K  read/write        /usr/lib/libcrypt.a
d6e374e0           3K  read/exec         /usr/lib/libc_r.a
f1090cb8           0K  read/write        /usr/lib/libc_r.a
d004e000         195K  read/exec         /usr/lib/libpthreads.a
f02e5000          16K  read/write        /usr/lib/libpthreads.a
d004a000          14K  read/exec         /usr/lib/libpthreads.a
f02a2000         265K  read/write        /usr/lib/libpthreads.a
d021b6c0        2284K  read/exec         /usr/lib/libc_r.a
f019bb10         621K  read/write        /usr/lib/libc_r.a
   Total        7935K
```

6 This command prints the signal actions defined by processes:

```
procsig [ ProcessID ] ...
```

Examples:

```
procsig 233628
233628 : sshd: it_user@pts/0 A
HUP         default
INT         default
QUIT        default
ILL         default  RESTART
TRAP        default  RESTART
ABRT        default  RESTART
EMT         default  RESTART
FPE         default  RESTART
KILL        default
```

```
BUS         default  RESTART
SEGV        default  RESTART
SYS         default  RESTART
PIPE        ignored
ALRM        caught
TERM        default
URG         default
STOP        default
TSTP        default
CONT        default
CHLD        caught
TTIN        default
TTOU        default
IO          default
XCPU        default
XFSZ        default
MSG         default
WINCH       default
PWR         default
USR1        default
USR2        default
PROF        default
DANGER      default
VTALRM      default
MIGRATE     default
PRE         default  RESTART
VIRT        default
ALRM1       default
WAITING     default  RESTART
RECONFIG    default
CPUFAIL     default
KAP         default
RETRACT     default
SOUND       default
SAK         default
```

7 The next command stops processes on the PR_REQUESTED event:

```
procstop [ ProcessID ] ...
```

8 The following command starts a process that has stopped on the PR_REQUESTED event:

```
procrun [ ProcessID ] ...
```

Example:

First I start a **sleep** in the background to play with the process:

```
nohup sleep 220 &
[1]     18800
```

Check the ‘S’ column – it shows the state of the process or kernel thread. The status now shows it as active:

```
ps -l  -p 18800
      F S      UID   PID  PPID   C PRI NI ADDR    SZ    WCHAN    TTY
TIME CMD
  200001 A        0 18800 16774   0  68 24 4302   136 35e30898  pts/1
0:00 sle

procstop 18800
```

Check the ‘S’ column – it shows the status now as stopped:

```
ps -l  -p 18800

      F S      UID   PID  PPID   C PRI NI ADDR    SZ    WCHAN    TTY
TIME CMD
  200001 T        0 18800 16774   0  68 24 4302   136 35e30898  pts/1
0:00 sle

procrun  18800  # start it again

ps -l  -p 18800
      F S      UID   PID  PPID   C PRI NI ADDR    SZ    WCHAN    TTY
TIME CMD
  200001 A        0 18800 16774   0  68 24 4302   136 35e30898  pts/1
0:00 sle
```

9 The command:

```
procstack [ -F ] [ ProcessID ] ...
```

prints the hexadecimal addresses and symbolic names for all the threads in the process:

```
procstack 233628
233628 : sshd: it_user@pts/0 A
0xd02a9c54  __fd_select(?, ?, ?, ?, ?) + 0x9c
0x1003b1b8  ????????()
0x1003a2f4  ????????()
0x1003ab14  ????????()
0x10032640  ????????()
0x10034be0  ????????()
0x10001c98  ????????()
0x100001b0  __start() + 0x88
```

10 The following command shows the process trees containing the specified process IDs or users:

```
proctree [ -a ] [ { ProcessID | User } ]
```

-a will include children of process 0 in the display. The default is to exclude them.

Examples:

```
proctree 233628
188440    /usr/sbin/srcmstr
   147480    /usr/sbin/sshd A
      204996    sshd: it_user [priv] A
         233628    sshd: it_user@pts/0 A
            184330    -ksh
               282838    -ksh
                  323742    proctree 233628

proctree -a 233628
1    /etc/init
   188440    /usr/sbin/srcmstr
      147480    /usr/sbin/sshd A
         204996    sshd: it_user [priv] A
            233628    sshd: it_user@pts/0 A
               184330    -ksh
                  282838    -ksh
                     323744    proctree -a 233628
```

11 This command waits for all of the specified processes to terminate:

```
procwait [ -v ] [ ProcessID ] ...
```

Examples:

To show how the tool works, I first start a **sleep** in the background:

```
nohup sleep 120 &
[1]     16908

procwait -v 16908
```

after 120 seconds the procwait terminates and returns the exit status of the **sleep**.

```
16908 : terminated, exit status 0
```

12 This command prints the current working directory of the processes.:

```
procwdx [ -F ] [ ProcessID ] ...
```

-F forces procfiles to take control of the target process even if another process has control.

Examples:

```
procwdx   258226
258226: /usr/tivoli/tsm/client/ba/bin/
```

13 **Truss**. Another very useful command introduced with AIX 5.1 is the **truss** tool. We can find it in the *bos.sysmgt.serv_aid* LPP:

```
lslpp -f bos.sysmgt.serv_aid | grep /usr/bin
                        /usr/bin/errmsg
                        /usr/bin/sysdumpstart
                        /usr/bin/errlogger
                        /usr/bin/errclear
                        /usr/bin/errpt
                        /usr/bin/truss
```

Truss traces a process's system calls, dynamically loaded user- level function calls, received signals, and incurred machine faults.

Truss has many flags to customize the output. For the real meaning and description please see the man page:

```
truss [ -f] [ -c] [ -a] [ -l ] [ -d ] [ -D ] [ -e] [ -i] [ { -t |  -x}
  [!] Syscall [...] ] [ -s [!] Signal [...] ] [ { -m }[!] Fault [...]] [
  { -r |  -w} [!] FileDescriptor [...] ] [ { -u } [!]LibraryName [...]::
  [!]FunctionName [ ... ] ] [ -o Outfile] {Command| -p pid [. . .]}
```

The most often used combinations of the command are:

```
truss -leaf -p PID     # to look at a running process
```

or:

```
truss -leaf command
               # to find out what a process does till it gets stuck
```

Examples:

```
truss -leaf who am i
```

```
499922: 921655: execve("/usr/bin/who", 0x2FF22C58, 0x2FF22C68)  argc: 3
499922: 921655:  argv: who am i
499922: 921655:  envp: _=/usr/bin/truss LANG=C LOGIN=root SSH_TTY=/dev/
pts/1
499922: 921655:  PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/
sbin:/usr/java14/jre/bin:/usr/java14/bin:/usr/java131/jre/bin:/usr/
java131/bin
499922: 921655:  CT_TR_TRACE_LEVELS= LC__FASTMSG=true LOGNAME=root
499922: 921655:  MAIL=/usr/spool/mail/root LOCPATH=/usr/lib/nls/loc
499922: 921655:  PS1=hugo-srv $PWD # USER=root AUTHSTATE=compat
499922: 921655:  SHELL=/usr/bin/ksh ODMDIR=/etc/objrepos HOME=/
499922: 921655:  SSH_CONNECTION=172.28.17.170 2123 172.28.13.42 22
499922: 921655:  SSH_CLIENT=172.28.17.170 2123 22 TERM=xterm
499922: 921655:  MAILMSG=[YOU HAVE NEW MAIL] PWD=/
499922: 921655:  TZ=MET-1MEST,M3.5.0/02:00:00,M10.5.0/03:00:00 ENV=//
.kshrc
499922: 921655:  A__z=! LOGNAME
499922: 921655:  NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/
%N.cat
499922: 921655: kioctl(1, 22528, 0x00000000, 0x00000000) = 0
499922: 921655: getuidx(2)                              = 0
499922: 921655: sbrk(0x00000000)                        = 0x200010FC
499922: 921655: sbrk(0x00000004)                        = 0x200010FC
499922: 921655: __libc_sbrk(0x00000000)                 = 0x20001100
499922: 921655: open("/usr/lib/security/methods.cfg", O_RDONLY) = 3
499922: 921655: kioctl(3, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kfcntl(3, F_GETFD, 0x00000000)          = 0
499922: 921655: kfcntl(3, F_SETFD, 0x00000001)          = 0
499922: 921655: statx("/usr/lib/security/methods.cfg", 0x2FF218B0, 76,
0) = 0
499922: 921655: lseek(3, 0, 1)                          = 0
499922: 921655: kioctl(3, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kread(3, " *   @ ( # ) 7 8\t 1 . 5".., 4096) = 1703
499922: 921655: lseek(3, 0, 1)                          = 1703
499922: 921655: lseek(3, 0, 1)                          = 1703
499922: 921655: lseek(3, 0, 1)                          = 1703
499922: 921655: kread(3, " *   @ ( # ) 7 8\t 1 . 5".., 4096) = 0
499922: 921655: close(3)                                = 0
499922: 921655: open("/etc/passwd", O_RDONLY)           = 3
499922: 921655: kioctl(3, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kfcntl(3, F_GETFD, 0x00000000)          = 0
499922: 921655: kfcntl(3, F_SETFD, 0x00000001)          = 0
499922: 921655: accessx("/etc/security/passwd", 04, 0)  = 0
499922: 921655: open("/etc/security/passwd", O_RDONLY)  = 4
499922: 921655: kioctl(4, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kfcntl(4, F_GETFD, 0x00000000)          = 0
499922: 921655: kfcntl(4, F_SETFD, 0x00000001)          = 0
499922: 921655: accessx("/etc/passwd", 04, 0)           = 0
```

```
499922: 921655: _getpid()                                = 499922
499922: 921655: _getpid()                                = 499922
499922: 921655: open("/etc/passwd", O_RDWR)              = 5
499922: 921655: kioctl(5, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kfcntl(5, F_GETFD, 0x00000000)           = 0
499922: 921655: kfcntl(5, F_SETFD, 0x00000001)           = 0
499922: 921655: kfcntl(5, F_SETLKW, 0x2FF216B0)          = 0
499922: 921655: fstatx(5, 0x2FF218A0, 76, 0)             = 0
499922: 921655: fstatx(5, 0x2FF218A0, 76, 0)             = 0
499922: 921655: kfcntl(5, F_GETFL, 0x00000000)           = 2
499922: 921655: lseek(5, 0, 1)                           = 0
499922: 921655: open("/etc/passwd.nm.idx", O_RDWR)       Err#2  ENOENT
499922: 921655: open("/etc/passwd.nm.idx", O_RDONLY)     Err#2  ENOENT
499922: 921655: open("/etc/passwd.id.idx", O_RDWR)       Err#2  ENOENT
499922: 921655: open("/etc/passwd.id.idx", O_RDONLY)     Err#2  ENOENT
499922: 921655: kfcntl(5, F_GETFL, 0x00000000)           = 2
499922: 921655: lseek(5, 0, 0)                           = 0
499922: 921655: lseek(5, 0, 1)                           = 0
499922: 921655: kioctl(5, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kread(5, " r o o t : ! : 0 : 0 : :"., 4096) = 584
499922: 921655: kfcntl(5, F_SETLKW, 0x2FF217B0)          = 0
499922: 921655: close(5)                                 = 0
499922: 921655: _getpid()                                = 499922
499922: 921655: _getpid()                                = 499922
499922: 921655: open("/etc/passwd", O_RDWR)              = 5
499922: 921655: kioctl(5, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kfcntl(5, F_GETFD, 0x00000000)           = 0
499922: 921655: kfcntl(5, F_SETFD, 0x00000001)           = 0
499922: 921655: kfcntl(5, F_SETLKW, 0x2FF217E0)          = 0
499922: 921655: kfcntl(5, F_GETFL, 0x00000000)           = 2
499922: 921655: lseek(5, 0, 1)                           = 0
499922: 921655: kfcntl(5, F_GETFL, 0x00000000)           = 2
499922: 921655: lseek(5, 0, 0)                           = 0
499922: 921655: kfcntl(5, F_GETFL, 0x00000000)           = 2
499922: 921655: lseek(5, 0, 1)                           = 0
499922: 921655: kioctl(5, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kread(5, " r o o t : ! : 0 : 0 : :"., 4096) = 584
499922: 921655: lseek(5, 0, 1)                           = 584
499922: 921655: lseek(5, 0, 1)                           = 584
499922: 921655: lseek(5, 0, 0)                           = 0
499922: 921655: kfcntl(5, F_SETLKW, 0x2FF218E0)          = 0
499922: 921655: close(5)                                 = 0
499922: 921655: close(3)                                 = 0
499922: 921655: close(4)                                 = 0
499922: 921655: open("/etc/netsvc.conf", O_RDONLY)       = 3
499922: 921655: kioctl(3, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kioctl(3, 22528, 0x00000000, 0x00000000) Err#25 ENOTTY
499922: 921655: kread(3, " #   @ ( # ) 4 3        "., 4096) = 4096
499922: 921655: kread(3, " n l y   I P v 4   a d d"., 4096) = 604
499922: 921655: kread(3, " n l y   I P v 4   a d d"., 4096) = 0
```

```
499922: 921655: close(3)                                  = 0
499922: 921655: open("/etc/irs.conf", O_RDONLY)           Err#2  ENOENT
499922: 921655: getdomainname(0xF01F2E40, 1024)           = 0
499922: 921655: __libc_sbrk(0x00000000)                   = 0x20011110
499922: 921655: getdomainname(0xF01F2E40, 1024)           = 0
499922: 921655: open("/etc/hesiod.conf", O_RDONLY)        Err#2  ENOENT
499922: 921655: getdomainname(0xF01F2E40, 1024)           = 0
499922: 921655: getdomainname(0xF01F2E40, 1024)           = 0
499922: 921655: getdomainname(0xF01F2E40, 1024)           = 0
499922: 921655: kioctl(0, 22529, 0x2FF21734, 0x00000000) = 0
499922: 921655: statx("/dev/pts/1", 0x2FF21758, 76, 0)    = 0
499922: 921655: open("/etc/utmp", O_RDWR|O_CREAT)         = 3
499922: 921655: kioctl(3, -2147195266, 0x2FF219D0, 0x00000000) = 0
499922: 921655: kioctl(3, -2147195267, 0x2FF219D0, 0x00000000) = 0
499922: 921655: kfcntl(3, F_SETFL, 0x00000001)            = 0
499922: 921655: kread(3, "\0\0\0\0\0\0\0\0\0\0\0"..., 648) = 648
499922: 921655: lseek(3, 0, 1)                            = 648
499922: 921655: lseek(3, 0, 1)                            = 19440
root        pts/1       May 31 11:52     (it_adm1.bull)
499922: 921655: kwrite(1, " r o o t                 "..., 58) = 58
499922: 921655: kread(3, " r o o t\0\0\0\0\0\0\0"..., 648) = 0
499922: 921655: kfcntl(1, F_GETFL, 0x00000000)            = 67110914
499922: 921655: close(1)                                  = 0
499922: 921655: kfcntl(2, F_GETFL, 0x2FF22FFC)            = 67110914
499922: 921655: _exit(0)
```


Perl – a practical script language

Perl (Practical Extraction and Report Language), a programming language developed by Larry Wall, is designed especially for processing text. Because of its strong text processing abilities, Perl has become one of the most popular languages for writing CGI scripts. Perl is an interpretive language, which makes it easy to build and test simple programs.

Programs written in Perl are called Perl scripts.

Perl is a practical extraction and report language that is freely available for Unix, MVS, VMS, MS/DOS, Macintosh, OS/2, Amiga, and other operating systems.

Perl has powerful text-manipulation functions. It eclectically combines the features and purposes of many command languages. Perl has enjoyed recent popularity for programming World Wide Web electronic forms and generally as the glue and gateway between systems, databases, and users.

Perl is a compiled scripting language.

The same program, written in the Windows operating system, also works on the AIX operating system and on USS (Unix System Services in the z/OS operating system).

PERL MODULES

Perl modules provide a powerful way to extend the core features of the Perl language. They are easy to use.

What are the modules?

A module is one sequence of operations that can simply be used inside another Perl script calling the module.

Inside a Perl script, various modules can be called.

Usually, the modules have a name that finishes with ‘pm’ (eg English.pm, Apache.pm, and Socket.pm).

USER ENVIRONMENT

On z/OS, the user who develops Perl scripts must have the OMVS segment.

The OMVS segment is created, in the case of the RACF security system, with the following commands:

```
ALTUSER userid  OMVS(UID(1111) GID(2222) HOME('/u/userid') PROGRAM('/
bin/sh'))
```

where *1111* is the ID assigned to the user and *2222* is the ID assigned to the group.

OMVS ENVIRONMENT

To activate the user OMVS environment, log on to the TSO (Time Sharing Option) and from the command line type:

```
OMVS
```

The screen in Figure 1 will be shown.

```
===> _      this is the command prompt

To run the command, type the command from the command prompt:

==>  ls -la                    display files in the current directory
```

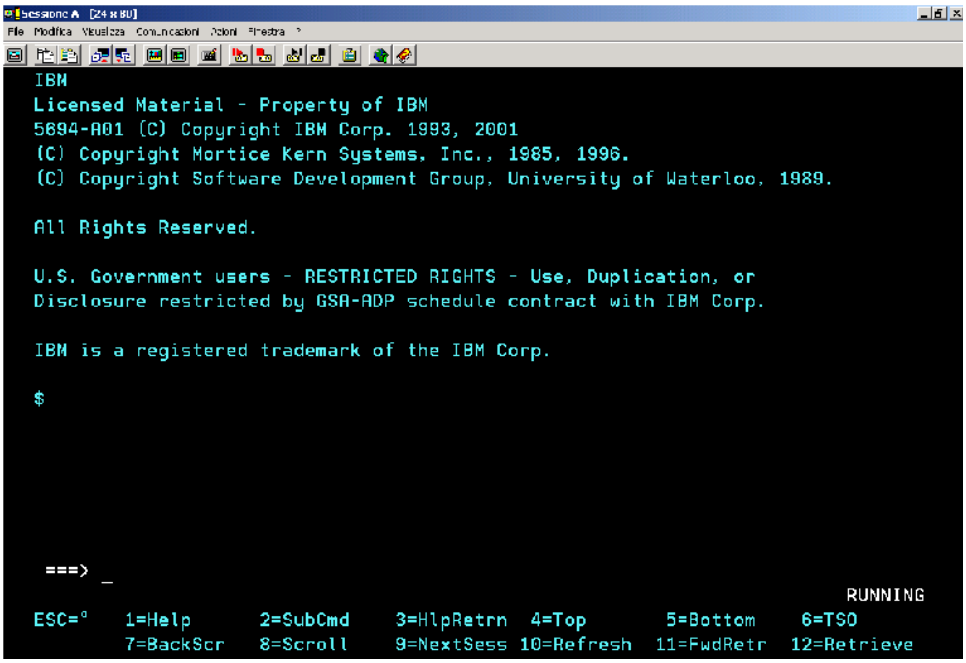

Figure 1: OMVS environment

PERL VERSION

To know which Perl version is installed on your system, type the following command from the command prompt:

```
Perl -version
```

The result can be the following:

```
This is Perl, version 5.004_03

Copyright 1987-1997, Larry Wall
Perl may be copied only under the terms of either the Artistic License
or the GNU General Public License, which may be found in the Perl 5.0
source kit.

In your system is installed the 5.004_03 Perl version.
```

UNIX, LINUX, AND WINDOWS FREE PERL DISTRIBUTION

You can download Perl for all operating systems from http://www.activestate.com/Products/ActivePerl.

IBM Z/OS (OS/390)

Since OS/390 R2.3, Perl 5.004_03 has shipped as a standard component.

If Perl is not installed in your system, you can download the latest version from http://www-1.ibm.com/servers/eserver/zseries/zos/unix/redbook/index.html#perl.

The instructions for the installation are at http://aspn.activestate.com//ASPN/Perl/Products/ActivePerl/lib/Pod/perlos390.html#modules_and_extensions_for_perl_on_os_390.

A module list can be found at http://www.perl.com/CPAN/modules/00modlist.long.html.

Instructions for the installation of the modules can be found at http://www.xav.com/perl/lib/Pod/perlmodinstall.html.

HOW PERL FINDS MODULES AND LIBRARIES

The way that you can find out what Perl is doing when it looks for modules is to print **@INC**, which is an array that Perl uses to determine the directories to search for libraries and modules.

From the command prompt type:

```
perl -e "print @INC";
```

The result can be following:

```
/usr/local/lib/perl5/s390/5.00403/usr/local/lib/perl5/usr/local/lib/
perl5/site_perl/s390/usr/local/lib/perl5/site_perl
```

The directories in the list are in the order that Perl will use when it searches the modules.

If the module isn't in one of those directories, Perl will give up after printing the message:

```
Can't locate MyModule.pm in @INC at line ….
```

The directories may be different on your computer, but whatever they are, the message will be the same.

HOW TO MODIFY @INC

Insert this line in the Perl script before you load the module:

```
use lib "/my/path/modules"; this line indicates where to find the module
use MyModule; this line loads the module
```

HOW TO USE PERL MODULES

A Perl module is a self-contained piece of Perl code that can be used by a Perl program or by other Perl modules. It is conceptually similar to a C link library, or a C++ class.

CPAN

The Comprehensive Perl Archive Network (CPAN) contains a huge collection of publicly available modules, and includes guidelines for module creation and use.

It can be found at http://www.perl.com/CPAN.

Module names

Each Perl module has a name. Module names must be unique. To minimize name space collisions, Perl provides a hierarchical name space for modules, similar to the name space for Java classes (for example Net::Ping).

Module files

Each module is contained in a single file. Module files are stored in a subdirectory hierarchy that parallels the module name hierarchy.

For example the module Ping is placed in:

```
/perl/lib/perl5/5.6.1/Net/Ping.pm
```

Ping module

A detailed directory of Perl modules can be found at http://www.perl.com/CPAN/modules/00modlist.long.html.

Each Perl module has the following syntax:

- NAME – module name and short description about the usage.
- SYNOPSIS – a practical example on the use of the module.
- DESCRIPTION: Functions – a detailed description about the module utilization and all its functions.
- NOTES – author notes.
- INSTALL – installation instructions.
- BUGS – notes about bugs.
- AUTHORS – authors list.
- COPYRIGHT – copyright notes.

Ping synopsis

```
use Net::Ping;
    $p = Net::Ping->new();
    print "$host is alive.\n" if $p->ping($host);
    $p->close();

    $p = Net::Ping->new("icmp");
    $p->bind($my_addr); # Specify source interface of pings
    foreach $host (@host_array)
    {
```

```
        print "$host is ";
        print "NOT " unless $p->ping($host, 2);
        print "reachable.\n";
        sleep(1);
    }
    $p->close();

    $p = Net::Ping->new("tcp", 2);
    # Try connecting to the www port instead of the echo port
    $p->{port_num} = getservbyname("http", "tcp");
    while ($stop_time > time())
    {
        print "$host not reachable ", scalar(localtime()), "\n"
            unless $p->ping($host);
        sleep(300);
    }
    undef($p);

    # Like tcp protocol, but with many hosts
    $p = Net::Ping->new("syn");
    $p->{port_num} = getservbyname("http", "tcp");
    foreach $host (@host_array) {
      $p->ping($host);
    }
    while (($host,$rtt,$ip) = $p->ack) {
      print "HOST: $host [$ip] ACKed in $rtt seconds.\n";
    }

    # High precision syntax (requires Time::HiRes)
    $p = Net::Ping->new();
    $p->hires();
    ($ret, $duration, $ip) = $p->ping($host, 5.5);
    printf("$host [ip: $ip] is alive (packet return time: %.2f ms)\n",
1000 * $duration)
      if $ret;
    $p->close();
```

Ping.pl

Below is a simple Perl script that uses the Ping module.

```
my @hosts = qw(
   serprd.mydom.us
   srvtest.mydom.en
   server.mydom.it
   10.10.100.34
);
$mytime1 = (localtime)[5]; # year
$mytime1 = $mytime1 + 1900;
$mytime2 = (localtime)[4]; # month
```

```
$mytime2 = $mytime2 + 1;
$mytime3 = (localtime)[3]; # day
$mytime4 = (localtime)[2]; # hour
$mytime5 = (localtime)[1]; # minute
$mytime6 = (localtime)[0]; # second
print "\n";
print "Date:$mytime1-$mytime2-$mytime3
Time:$mytime4:$mytime5:$mytime6\n";
print "\n";
use Net::Ping;
    $p = Net::Ping->new("tcp"); # to use the icmp protocol need root
auth
    foreach my $host(@hosts){
     if ($p->ping($host)) {
     print "$host is alive.\n"
     }
     else {
     print "$host unreachable.\n"
     }
     sleep(1);
    }
    $p->close();
```

where:

- *my @hosts = qw(* – array declaration. In this case the array contains the directory of the servers to which the **ping** command is being addressed.
- *$mytime1 = (localtime)[5];* – $mytime1= variable assignment.
- *(localtime)[5];* – it is the predefined Perl time variable. The fifth element contains the year.
- *print "\n";* – print instruction. In this case *"\n"* means new line.
- the instruction *print "Date:$mytime1-$mytime2-$mytime3 Time:$mytime4:$mytime5:$mytime6\n"* means print to the screen the date and time value.
- *use Net::Ping;* – load the Ping module.
- *$p = Net::Ping->new("icmp");* – create a new Ping object
- *foreach my $host(@hosts){* – list of hosts to be **ping**ed.

- *if ($p->ping($host)) {* – **ping** the remote host and wait for a response.
- *sleep(1);* – wait one second.
- *$p->close();* – close the network connection for this Ping object.

START PERL SCRIPT

There are two ways to execute a Perl script, depending on whether its first line contains a statement that specifies the absolute path where the Perl program is found:

1 Program script with Perl path specified:

- Unix or USS (Unix System Services) operating system:

```
sample.pl
#!/usr/local/lib/perl5/perl -w
print "This is my program"
. . . . .
```

To start this script from the command prompt type:

```
./sample.pl
```

- Windows operating system:

From the directory in which the Perl script is found, type the following command:

```
sample.pl
```

2 Program script without Perl path specified:

```
sample.pl

print "This is my program"
. . . . .
```

To start this script from the command prompt type:

```
perl ./sample.pl
```

HOW TO WRITE AND START SCRIPTS IN THE USS ENVIRONMENT

Before writing the script we must know the emulation codepage because the USS environment has codepage IBM-1047 and the Perl script must be converted to this codepage before execution.

Where to find the emulator codepage

From the emulator bar command choose *Communications* (see Figure 2).

Figure 2: Selecting Communications option

Then select *configure* and *session parameter*; what is shown in Figure 3 will be displayed.

Our emulation codepage is: IBM-1047.

I wrote a REXX program that copies the input script Perl program to a USS environment, converts the codepage to IBM-1047, executes it, and browses the output.

Here is the program:

```
/* rexx */
/*
The objective of this program is to convert
the Perl script in USS codepage,
and to execute it.
*/
/** Setup environment-start **/
tmpout ="tmpout"               /* temporary filename */
usscpage="IBM-280"             /* emulation Codepage */
```

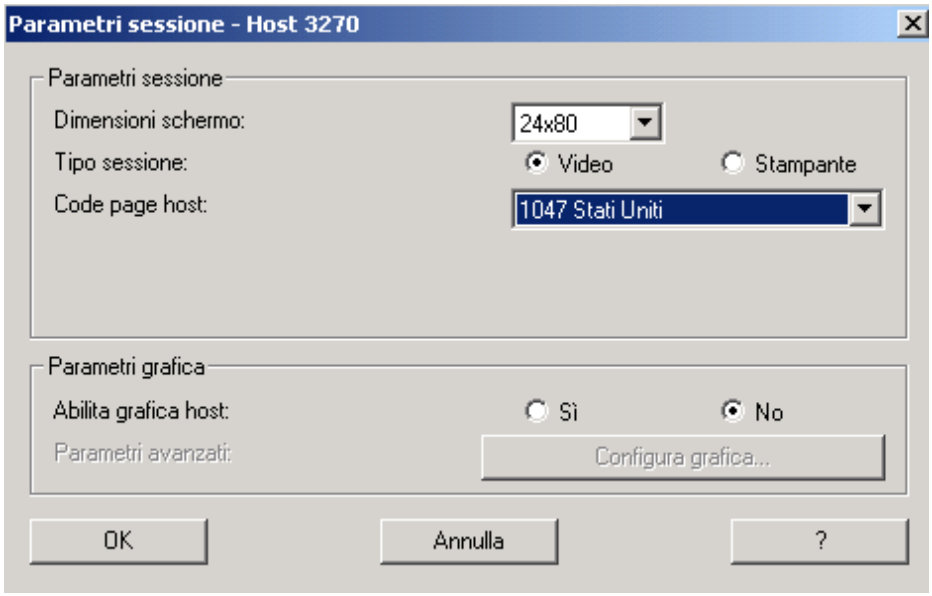

Figure 3: Displaying emulation codepage

```
ussibmcp="IBM-1047"            /* Uss Codepage */
ussp="750"                     /* chmod permit */
dsn_input="user.lib.test(firstpl)"
dsn_output="/u/userid/firstpl"
conv_comm ="iconv -f "usscpage ,
               " -t "ussibmcp" "dsn_output">"dsn_output"_cnv"
chmod_comm="chmod "ussp" "dsn_output"_cnv"
/** Setup environment-end **/
address tso
"free dd(pathname)"
"alloc dd(input) da("dsn_input") shr reu"
 "ALLOC DD(PATHNAME) PATH('"dsn_output"')" ,
 "PATHMODE(SIRWXU,SIXGRP,SIRGRP) PATHOPTS(OCREAT, OWRONLY)"
"OCOPY INDD(input) OUTDD(PATHNAME) TEXT"
say "Iconv routine ..."
if bpxwunix(conv_comm,,stack) ^= 0
                  then do
                         say "Error during iconv-RC= "rc
                         return
                        end
say "Iconv chmod command .."
if bpxwunix(chmod_comm,,stack) ^= 0
                  then do
                         say "Error during chmod-RC= "rc
```

```
                          return
                         end
"oedit "dsn_output"_cnv"
say "Execution .."
call bpxwunix dsn_output'_cnv  1>'tmpout '2>'tmpout,,out.
do while queued()>0      /* Empty the stack queue */
 pull x
end
address tso
"obrowse "tmpout
exit
```

Program set-up:

- *usscpage="IBM-280"* – emulator codepage.
- *ussibmcp="IBM-1047"* – USS codepage default.
- *ussp="750"* Script permission 750 means: user read,write,execute; group read,execute; other none.
- *dsn_input="user.lib.test(firstpl)"* – insert here your library and member name where you have written your Perl script.
- *dsn_output="/u/userid/firstpl"* – insert here your USS absolute path and filename.

PROGRAM INSTALLATION AND START UP

1 To install the REXX program copy it in user.lib.test library with an optional name (for example, myname).

2 To start the program from TSO option 6 type:

```
ex 'user.lib.test(myname)' ex
```

The program copies the dataset in the USS output directory and its filename, converts this script file to the IBM-1047 codepage, executes it, and browses the output file.

When programming with Perl modules, it is very important to read the Synopsis section found in the documentation of every module.

Magni Mauro
Systems Engineer (Italy)

AIX news

IBM has announced Version 5.3 of its High Availability Cluster MultiProcessing (HACMP) clustering software for AIX.

HACMP 5.3 now supports AIX 5L V5.3. Although the new Power5-based servers will run AIX V5.2, the virtualization features that are supported on Power5s require AIX V5.3.

HACMP 5.3 simplifies the integration of DB2 and Oracle databases in a cluster, supports Veritas filesystems as well as IBM's own filesystems for AIX, and supports up to 32 nodes.

There are also extended distance (XD) add-ons, allowing several different styles of clustering over large geographical distances.

For further information contact:
URL: www-1.ibm.com/support/docview.wss?rs=203&context=SW000&dc=D600&uid=tss1flash10326&loc=en_US&cs=UTF-8&lang=en.

* * *

Vision Solutions has announced ORION for AIX. This addition to the ORION family of products provides switched disk solution topology for applications running under AIX.

The product supports AIX-based applications on pSeries servers and also supports AIX on an iSeries hosted partition (which, the company claims, no one else does). Built on an autonomic architecture and clustering, ORION for AIX is designed to deliver application high availability between two or more servers connected to a common storage tower or storage subsystem; supporting both iASP and SCSI data storage. In the event of an application or server failure, ORION for AIX fails over the production server to a back-up server.

For further information contact:
URL: www.visionsolutions.com/solutions/ORION/default.asp.

* * *

AdventNet has announced an update to ManageEngine Applications Manager, its Web application management software that provides integrated application, server, and systems monitoring. The update now includes support for monitoring AIX and WebSphere 6.

Applications Manager monitors application servers (WebLogic, WebSphere, JBoss, Tomcat), databases (SQL Server, Oracle, MySQL, DB2), systems, custom Java, JMX, J2EE applications, and Web sites with fault management and notification capabilities.

For further information contact:
URL: www.appmanager.com/download.html.

* * *

Adobe has released details of a security problem with Acrobat Reader, which could allow attackers to seize control of a user's computer. The weakness could be exploited by e-mails containing malicious PDF files.

The solution for AIX platforms is to update to Adobe Reader 5.0.11.

For further information contact:
URL: www.adobe.com/support/techdocs/329121.html.