59 AIX update

September 2000

In this issue

Published by

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

Harold Lewis

Disclaimer

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

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £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).

Printed in England.

Understanding file archiving

There may be occasions when you have to send a large number of files to another user, or you may want to conserve space or to consolidate data. Working with individual files can tax resources beyond their capacity – a problem that's compounded if the files are spread across a complex directory structure that must be preserved.

AIX provides a method to accomplish these tasks. You can use the **tar** command to create archives that contain hundreds of files, and you can process the archive files further with the **compress**, **uncompress**, and **zcat** commands, which help to reduce the total space used by the files.

THE TAR COMMAND

The **tar** command, which originally stood for 'tape archive', allows you to create a new archive, add files to an existing archive, extract files from an archive, or just view the contents of an archive. The **tar** command can process simple directories and complex directory structures.

The syntax of the command is as follows:

```
tar required flags optional flags file|directory
```

The **tar** command has two kinds of flag: required and optional.

Required flags

The **tar** command must have one of the following required flags to indicate the operation that is to be performed.

-c Create an archive.

-t List the path of files in an archive.

-u Add files to an archive.

-x Extract files from an archive.

Optional flags

The following optional flags specify how the data is to be processed:

-f Specifies the archive file to be read or written. The default is system-dependent, such as the file */dev/rmt0*. If a minus sign ('-') is specified as the archive file, standard input or standard output is used.

-v Displays file names as they are processed.

Creating archive files

Suppose your current directory has 200 files named:

```
dev001.doc
dev002.doc
dev003.doc
   ...
dev198.doc
dev199.doc
dev200.doc
```

If you want to archive all 200 in a single archive, you can enter:

```
tar -cvf filegroup.tar dev*.doc
```

This creates the archive file *filegroup.tar*, which contains all files matching the specification *dev*.doc*. The **-c** flag tells **tar** to create an archive file, **-v** causes it to display all the file names as they are processed (this can be used as a progress indicator), **-f** tells it the name of the archive file (*filegroup.tar*), and the file specification tells it which files to process.

Note that it's not mandatory that the names of archive files end in *.tar* – this is just a convention to help identify them.

You can also archive directories and subdirectories, including all their files. Suppose *userdata* is a subdirectory of your current directory and that it contains the following files:

```
userdata/admins/jjones.data
userdata/admins/kmartin.data
userdata/admins/rsmith.data
userdata/user/areese.fil
userdata/user/lkern.fil
userdata/user/swebber.fil
userdata/user/wmather.fil
```

If you want to archive all files in *userdata*, including the *admins* and *user* subdirectories and all their files, you enter:

```
tar -cvf dirstruc.tar userdata
```

This creates archive file *dirstruc.tar*, which includes all specified directories, subdirectories, and files. Specifying a directory instead of a file tells **tar** to archive all subdirectories and files in that directory.

Adding files to an archive file

You can add files to an existing archive using the **-u** flag. Suppose you have an archive named *filegroup.tar* and you want to add the file *dev201.doc* to it. If you enter:

```
tar -uvf filegroup.tar dev201.doc
```

tar first inspects the archive to determine whether *dev201.doc* is already in it. If it isn't, **tar** opens the archive *filegroup.tar* and appends the file to it.

Updating existing archived files

The **-u** flag can also be used to update an archive with the latest version of a file that's already in the archive. If the current version of a file on your workstation has a modification date later than that of the file in the archive, **tar** will append the later version to the archive.

If there is no change in the modification data and time of the specified file, the file is not appended to the archive.

tar doesn't have the ability to replace a file in an archive – all it can do is append a later version of the file to the end of the archive. This means that an archive may contain multiple versions of a file.

When extracting, **tar** will overlay an existing file with one being extracted. This happens even if the existing file is the same as the one being extracted, and it means that the version of a particular file that's closest to the end of an archive will become the current version when the file is extracted.

Having multiple versions of a file in an archive may or may not be troublesome. If you change a file frequently and append the revised version of it to your existing archive each time, the archive will

contain many copies of the file. However, only the last one is accessible via extraction. Hence, if you reduce the size of a particular archived file and use the **-u** flag to update the archive, it will still contain both the larger and the smaller version of the file, though only the smaller one is utilized.

To get around this, you can rebuild archives completely rather than altering them. This saves space in your archives.

Extracting archive files

You can choose to extract all files from an archive or just selected ones. If an archive contains directories, you can extract the entire structure or only some subdirectories and their files.

The **tar** command will preserve the owner and group of files in your archive as long as your user-id has the authority to do this when the command is issued (this may require *root* authority). If not, the command extracts the files and sets their owner and group ids to those of the user running the **tar** command.

To extract all files contained in the archive *filegroup.tar*, enter:

```
tar -xvf filegroup.tar
```

This results in all archived files being extracted to the current directory. The **-x** flag tells **tar** to extract and **-v** tells it to display each file name as it is extracted from the archive. Without **-v**, **tar** would just return to the command prompt at the end of processing. For very large archives, it is desirable to use the **-v** flag as a progress indicator. The **-f** flag is used to indicate the archive file to extract (*filegroup.tar*, in this instance).

To extract all directories and files in the archive *dirstruc.tar*, enter:

```
tar -xvf dirstruc.tar
```

This creates the directories in the current directory, if they don't already exist, and then places files in their appropriate directories on your system.

Suppose you want to extract only one file (*dev112.doc*) that you know is in a particular archive (*filegroup.tar*), rather than all the archive's contents. To do this, enter:

```
tar -xvf filegroup.tar dev112.doc
```

This would place the specified file in the current directory and ignore all other files in the archive. If you want to extract only the subdirectory *userdata/admins* and its contents from *dirstruc.tar*, enter:

```
tar -xvf dirstruc.tar userdata/admins
```

This creates the *userdata/admins* subdirectory in the current directory, if it doesn't already exist, and extracts all relevant files to it. Specifying a directory tells **tar** to extract the directory and all its subdirectories and files recursively.

Viewing archive files

Suppose you don't want to extract files from an archive, but want to determine which files the archive contains, perhaps passing the file names to another process or filter, such as **grep** or **sort**. To do this, use the **-t** flag – if you enter:

```
tar -tvf dirstruc.tar
```

tar displays all directories, subdirectories, and files in the *dirstruc.tar* archive file. The **-t** flag tells **tar** to display the names of files and directories and to do no further processing on the archive files. Note that, when the **-t** and the **-v** flags are used together, **tar** displays additional information, such as the file size, modification time, user and group ids, and file permissions of each entry in the archive.

As can be seen, **tar** is a very powerful utility that allows you to manage your resources through creating, viewing, updating, and extracting either all or some archived files. However, **tar** is only part of the file archiving process.

THE COMPRESS COMMAND

The **compress** command can reduce the size of archive files. It takes as input the name of the file that's to be compressed, appending a '.Z' to the name of the processed file to indicate that it's compressed. If you enter:

```
compress dirstruc.tar
```

compress compresses *dirstruc.tar* and renames it *dirstruc.tar.Z.*

The **compress** command can be very processor-intensive and a large archive file can take several minutes to compress. Note that your user-id must have the correct authority to compress each file, and that the **compress** command works with many types of file, not just archives.

THE UNCOMPRESS COMMAND

The counterpart of the **compress** command is **uncompress**. With a few exceptions, there's not much one can do with a compressed file without first uncompressing it.

The **uncompress** command takes as input the name of the compressed file, removing the '.Z' extension from the expanded file. If you enter:

```
uncompress dirstruc.tar.Z
```

uncompress expands the file, removes the '.Z' extension, and replace the input file. Like **compress**, **uncompress** can take several minutes to expand a large archive file.

Note that, if a file is compressed by a root user and is then uncompressed by another user who doesn't have root privileges, the file will acquire the owner and group ids of the user who uncompresses the file.

THE ZCAT COMMAND

zcat is another interesting utility. Although its output goes to the display, there is a clever way of using it to process archive files that is described later.

zcat takes a compressed file as input and displays the contents of the file on the screen, without creating an uncompressed file. To view the contents of the compressed file *filegroup.tar.Z*, enter:

```
zcat filegroup.tar.Z
```

zcat gives the results immediately, avoiding the potentially lengthy delay of an **uncompress** operation. More importantly, it prevents the utilization of resources which would be needed to contain an uncompressed file.

Note that **zcat** displays the entire contents of each archived file, not just the file names. If the files are large, **zcat** will display a lot of data.

For this reason, you may want to pipe the output of the **zcat** command to **more** for viewing a screen at a time.

PUTTING IT ALL TOGETHER

The **zcat** command is useful if you want to inspect the contents of a compressed file quickly. However, you can use **zcat** to view file names or extract data from a compressed archive without uncompressing it first.

Consider the following command:

```
zcat filegroup.tar.Z | tar -xvf -
```

The first part of it processes a compressed archive using **zcat**. The output of the **zcat** command – the contents of the archive file – are then piped to the **tar** command. The **-x** flag tells **tar** to extract and the **-v** flag tells it to display each file name as it's processed. The **-f** flag specifies the archive file to be processed, which is taken from the standard input, as indicated by the minus sign. In this example, standard input is the output of the **zcat** command.

The result is that files in the compressed archive are expanded into the current directory without using the resources needed to process and store the uncompressed archive.

The **zcat** and **tar** commands can thus be used to extract specific files and directories from compressed archives by specifying the name of the file or directory after the minus sign, as explained in the section above on the **tar** command.

Note that **zcat** uses information found in the compression header prior to each file in the archive to determine the file name to pass to the **tar** command.

You can also use **zcat** and **tar** to view the list of compressed files in an archive. Consider the following command:

```
zcat filegroup.tar.Z | tar -tvf -
```

In this case, piping the output of **zcat** to **tar** (invoked with the **-t** flag) causes it to display the path names of all files archived and compressed in *filegroup.tar.Z* on the screen.

There may be occasions when the only way you can process a compressed archive file is using **zcat**. If your file system is nearing capacity, you may find yourself unable to uncompress a file in order to process it. This is when **zcat** is a very valuable resource.

A PRACTICAL EXAMPLE

Here is a practical example of archiving operations in use.

Let us say you have a product named *GreatJob* that performs a very important task for your users. *GreatJob* can be built in one of several different ways, using different options and in several languages.

Assumptions:

1. You have the following directory trees on your build machine:

```
GreatJob.base
GreatJob.option1
GreatJob.option2
GreatJob.option3
```

2 You also have a directory tree for each language in which *GreatJob* is available:

```
GreatJob.French
GreatJob.German
GreatJob.Japanese
GreatJob.Spanish
```

3 Each directory tree contains many subdirectories and hundreds of files.

4 The product can be built by compiling *GreatJob.base* with no, some, or all the options. It can be built in English, if no language directory is specified, or in another language by specifying one of the language directories.

Implementation:

1 To create your deliverable file, you first create an archive using **tar -cvf** for each of the directory structures listed above. Then, after using the **compress** command, you would have created the following eight compressed archives:

```
GreatJob.base.tar.Z
GreatJob.option1.tar.Z
GreatJob.option2.tar.Z
GreatJob.option3.tar.Z
GreatJob.French.tar.Z
GreatJob.German.tar.Z
GreatJob.Japanese.tar.Z
GreatJob.Spanish.tar.Z
```

2 Now you can write a build script that builds a version of the product at any option level in any language specified by the user. Let's name the build script **DoaGreatJob.Script**.

3 You could now create an archive containing the eight compressed archives and then add the build script.

```
tar -cvf GreatJobBuild.tar GreatJob.*.tar.Z
tar -uvf GreatJobBuild.tar DoaGreatJob.Script
```

You now have one file named *GreatJobBuild.tar* that you can store in a library or to deliver to users.

Usage:

1 When someone needs to build a version of your product, first they extract the files from your archive by entering:

```
tar -xvf GreatJobBuild.tar
```

which would write the eight compressed archive files, plus the build script, to their workstation.

They can then run the build script, specifying the desired options and languages. Suppose they wanted to build a Japanese version with options 1 and 3; they might enter:

```
DoaGreatJob.Script -Japanese -Option1 -Option3
```

2 The build script, using various combinations of the **zcat** and **tar** commands, would expand and extract only the following base, option, and language compressed archives:

```
GreatJob.base.tar.Z
GreatJob.option1.tar.Z
GreatJob.option3.tar.Z
GreatJob.Japanese.tar.Z
```

The other four archives (option 2 and the other three languages)

would be available but not expanded, as they are not used for that particular build of the product.

Fewer resources are needed to compile any version of the product than if you first have to expand all the files in all the options and language directories.

You have just seen how an administrator can use not only **tar** and **compress** to create individual archives, but also to create archives of existing archived files to simplify further the packaging of deliverables.

EXERCISES

The following exercises help you understand file archiving by giving you hands-on experience with the commands described in this article.

Create subdirectory *archexer* off your home directory. Where the exercise directs you to create files to be archived, you may use existing files on your workstation. However, be careful to isolate them from your environment, as you may create several copies of them during the exercise.

1 CREATING, VIEWING, AND EXTRACTING FILE ARCHIVES

Step 1 Create files *test001.txt*, *test002.txt*, *test003.txt*, *test004.txt*, and *test005.txt* in your exercise directory. Avoid using **touch** to create them as the exercises benefit from the files containing data. You may redirect the output of **ls -l** to create the files or copy existing ones from your workstation.

Step 2 Enter:

```
tar -cvf filegroup.tar test*.txt
```

tar creates the archive *filegroup.tar* in your directory.

Step 3 Enter:

```
tar -tvf filegroup.tar
```

tar displays the files found in the archive on your screen.

Step 4 Create subdirectory *exergroup*, copy *filegroup.tar* to it, and **cd** to the directory.

Step 5 From *archexer/exergroup*, enter:

```
tar -xvf filegroup.tar test003.txt
```

tar extracts only *test003.txt* from your archive.

Step 6 Enter:

```
tar -xvf filegroup.tar
```

tar extracts all the files from the archive, overlaying existing files of the same name – so use caution!

2 ADDING FILES TO AND UPDATING AN EXISTING ARCHIVE

Step 1 Return to the exercise directory and create a new file named *addthis.fil*.

Step 2 From *archexer*, enter:

```
tar -uvf filegroup.tar addthis.fil
```

tar appends *addthis.fil* to the archive.

Step 3 Enter:

```
tar -tvf filegroup.tar
```

tar displays the names of files in the archive, including the new file, on your screen.

Step 4 Edit *test002.txt* to modify its time-stamp.

Step 5 Enter:

```
tar -uvf filegroup.tar test002.txt
```

tar appends the revised *test002.txt* to the archive.

Step 6 Enter:

```
tar -tvf filegroup.tar
```

tar displays the names of files in the archive, including the revised version of *test002.txt*. Note that the older version of *test002.txt* is also listed. **tar** will overlay the first version of *test002.txt* with the second one when the file is extracted from the archive.

Step 7 Enter:

```
tar -uvf filegroup.tar test001.txt
```

As *test001.txt* has not been modified since it was archived, you just get your prompt back, indicating that the archive has not been updated.

3 CREATING, VIEWING, AND EXTRACTING DIRECTORIES

Step 1 Create *exerstruc* as a subdirectory of *archexer* and **cd** to it.

Step 2 Create three directories (*dir1*, *dir2*, and *dir3*) in *exerstruc*.

Step 3 Copy the **.txt* files in *archexer* to *dir1*, *dir2*, and *dir3*.

You now have the following directory structure:

```
archexer/exerstruc/dir1
archexer/exerstruc/dir2
archexer/exerstruc/dir3
```

Each *dir** directory contains a copy of the five test files.

Step 4 Return to *archexer* and enter:

```
tar -cvf dirstruc.tar exerstruc
```

tar creates the archive *dirstruc.tar*, which contains all the subdirectories and files in the *exerstruc* directory tree.

Step 5 Enter:

```
tar -tvf dirstruc.tar
```

tar displays the paths of all files in the archive, plus some other information, such as permissions and time-stamps.

Step 6 From *archexer* enter:

```
rm -R exerstruc
```

This erases the contents of *exerstruc* and removes the subdirectories in preparation for extraction.

Step 7 While still in *archexer*, enter:

```
tar -xvf dirstruc.tar exerstruc/dir2
```

tar creates the directory *exerstruc/dir2* in *archexer* and extracts the five test files to it, ignoring the other directories in the archive.

Step 8 Return to *archexer* and enter:

```
tar -xvf dirstruc.tar
```

tar creates directories *exerstruc/dir1* and *exerstruc/dir3*. (*exerstruc/dir2* was created in step 7). The command also extracts data from the archive to the subdirectories.

4 COMPRESSING AND UNCOMPRESSING ARCHIVES

Step 1 Return to *archexer* and note the size of *dirstruc.tar*.

Step 2 Enter:

```
compress dirstruc.tar
```

compress reduces the size of the archive, and renames it *dirstruc.tar.Z*.

Step 3 Enter:

```
uncompress dirstruc.tar.Z
```

uncompress expands the compressed archive and restores its name to *dirstruc.tar*, removing the extension '.Z'.

5 USING ZCAT

Step 1 Before the exercise, recreate the compressed archive from Exercise 4. From *archexer*, enter:

```
compress dirstruc.tar
```

This recreates *dirstruc.tar.Z*.

Step 2 Enter:

```
zcat dirstruc.tar.Z | more
```

zcat displays the contents of the compressed file, preceding each file name with a compression header. Note that the command does not change the compressed file. If necessary, press 'q' exit **more**.

Step 3 Enter:

```
zcat dirstruc.tar.Z | tar -tvf -
```

1 **zcat** expands the compressed archive and passes the output to **tar**.

2 **tar** displays the path names of all subdirectories and files found in *dirstruc.tar.Z* on the screen.

Step 4 Create subdirectory *exerzcat*, copy *dirsruc.tar.Z* to it, and **cd** to the directory.

Step 5 From *exerzcat*, enter:

```
zcat dirstruc.tar.Z | tar -xvf -
```

1 **zcat** expands the compressed archive and passes its output to **tar**.

2 **tar** extracts the directory structure, creating subdirectories:

```
exerstruc/dir1
exerstruc/dir2
exerstruc/dir3
```

3 **tar** extracts files in their proper directories according to the compressed archive.

Note that **zcat** performs all operations without using the resources needed to uncompress or store an uncompressed archive file.

POST EXERCISE CLEAN UP

To delete all files created during the exercise, return to *archexer*'s parent directory, and enter:

```
rm -R archexer
```

This deletes all directories and files recursively.

David Chakmakian
Programmer (USA)

Hot-plug on F80s, H80s and M80s

Hot-plug support for selected PCI adapters is one of the features of the new F80, H80, and M80 servers. Its purpose is to improve both the reliability and versatility of RS/6000 servers by enabling the replacement of failed or unused PCI interface cards with ones that are functioning properly and required – without interrupting the operation of the server. Both hardware and software modifications were needed to implement this feature.

In order to support hot-plug hardware and software, the following characteristics are needed: the ability to disconnect and re-connect the electrical supply to the card, the ability to identify the slot containing the card, and the ability to fasten the card into its slot securely without affecting other hardware.

The I/O drawers of the new machines contain special plastic separators that are designed to prevent electrical short-circuits and damage while the adapters are added or removed. The hot-plug adapters are also secured with special retainer clips, located on the top of the slots – this eliminates the possibility of retainer screws being dropped while the adapter is being removed.

LEDs	PCI slot status	Message
Off	Off	Slot power is off. It is safe to remove or replace adapters.
On	On	Slot power is on. Do not remove or replace adapters.
Slow flashing (one a second)	Identify	The slot has been identified by the software; do not remove or replace adapters at this time.
Quick flashing (six to eight a second)	Action	The slot is ready for removing or replacing adapters.

Figure 1: Summary of hot-plug LED signals

Special hot-plug LEDs indicate the state of the adapter, enabling its secure identification and replacement. The lights are visible from both the outside and inside of the I/O drawer. Figure 1 summarizes the LED signals.

Not all PCI adapters are hot-pluggable – you should always check the manual *PCI Adapter Placement Reference Guide* (SA38-0538) for information about specific adapters. You also should be aware that adapters that are used to support certain hardware features, such as system disks, I/O controllers, or graphics cards connected to the console, cannot be replaced without shutting down the system.

Software support for manipulating slot power was introduced in AIX 4.3.3 via the AIX 4330-03 maintenance package (APAR IY09047), which is included on all pre-installed systems and also on the April 2000 'Update CD' that ships with AIX 4.3.3. In addition, APAR IY09814, which includes additional fixes that where not available before the AIX 4330-03 package shipped, also has to be installed.

Support has been implemented via two new commands, **lsslot** and **drslot**, which are also incorporated in **smit** and WebSM. **lsslot** is used to list slots and their characteristics and **drslot** is used to perform the dynamic reconfiguration of system slots.

The following scenario illustrates the procedure for adding an adapter to a live system.

1 The **drslot** command is used to transfer the chosen slot to the 'identify' state (the LED flashes slowly to indicate it's in this state). This allows the administrator to verify that the right slot was selected.

2 The system administrator is then prompted to press *Enter*, which causes the LED to change its state to 'action' (LED flashes quickly).

3 The adapter can now be added to the system and all external devices connected to it can be powered on.

4 Once this is done, the administrator should press *Enter* again to turn the slot's power on. The hot-plug LED will change its state

to ‘on’. The adapter is now integrated into the system and can be configured using AIX’s configuration manager, **cfmgr**.

To remove an adapter, you must first remove its configuration from AIX. Once the adapter is removed from the ODM, its power supply should be turned off and the adapter should be physically removed from the system.

Note that the addition of an adapter that wasn’t previously installed on a particular system requires the installation of the adapter’s device drivers.

REFERENCES

1 *RS/6000 Enterprise Server Model M80 Installation Guide*, SA38-0576

2 *RS/6000 Enterprise Server Model M80 User’s Guide*, SA38-2537

3 *RS/6000 Enterprise Server Model M80 Service Guide*, SA38-2538

4 *PCI Adapter Placement Reference Guide*, SA38-0538

5 *Supplemental Information for the PCI Adapter Placement Reference*, SA32-9077

6 *PCI Hot Plug Management, AIX Version 4.3 System Management Concepts, Operating Systems and Devices*

7 *Managing Hot Plug Connectors, AIX Version 4.3 System Management Guide, Operating Systems and Devices.*

Alex Polak
System Engineer
APS (Israel)

AIX printing on MVS AFP printers (part 1)

INTRODUCTION

Our company has migrated some MVS applications to AIX. As some of these applications require JES2 services, particularly for printing AFP documents on central, high throughput printers and distributed enterprise printers, we have written an application that allows AIX to send documents to the JES2 spool using AIX-specified spool parameters. (AFP is IBM's Advanced Function Presentation – a software technology that allows sophisticated output on AFP-capable mainframe printers.)

The application has two components: one, which runs under MVS, is an LU6.2-generic transaction server running as a started task, and the other, which runs under AIX, is a printing back-end program. Using this application, you simply use AIX's **lp** print command to print an AIX document under MVS, specifying the print queue (option **-d**) and JES2 spool parameters (option **-o**), which are in the form: *'class, dest, form, formdef, pagedef'*. For example, an AFP document with the following JES2 spool parameters:

```
(CLASS=B, DEST=XPTO, PAGEDEF=PD99, FORMDEF=FD99)
```

may be printed using the command:

```
lp -dlaser -o'B,XPTO,,PD99,FD99' afp_document_name
```

A regular document with the following spool parameters:

```
(CLASS=B, DEST=XPTO, FORM=XER1)
```

may be printed using the following command:

```
lp -dlaser -o'B,XPTO,XER1' non_afp_document_name
```

('laser' is the name of the AIX print queue.)

The printer back-end program associated with the print queue establishes an LU 6.2 session with the MVS started task. The MVS started task accepts the LU6.2 request, extracts the transaction program name specific to this service, and schedules a TCB to process the

request. The scheduled program then authenticates the AIX user-id using RACF (the IBM MVS security product), and dynamically allocates a SYSOUT dataset with the required characteristics (*class*, *dest*, *form* or *pagedef*, and *formdef*) to the user. After successful allocation, the document is transmitted and written to the spool. In case a transmission error occurs or the request is cancelled, the document is deleted from the spool. On successful execution, JES2 selects a printer based on the specified *class* and *dest.* The spooled document is then printed according to the specified *form*, or *formdef* and *pagedef*. The last two are AFP parameters.

This application was tested and certified for AIX (4.1 and 4.2) and MVS 4.3 to OS/390 2.4.

THE AIX COMPONENT

We created a printer back-end program called 'laser', which is scheduled by the print subsystem after the print command executes. On start-up, it receives two parameters as input: the **-o** option from the AIX **lp** command and the name and full path of the document file. After setting defaults, parsing and validating the input parameters, and retrieving the user-id of the document owner and the number of copies to print, the document file is scanned and the maximum line size (a requirement for JES2 spool allocation) and the number of lines are evaluated for transmission control and reporting. An LU6.2 session request is sent to the MVS transaction server requesting the program 'WRT2SPL' to be executed. After successful LU6.2 allocation, a block of data containing the JES2 options, the number of copies, the user-id, and the maximum line length are transmitted. A confirmation request is sent and, after successful notification, the file is sent line by line. After each block of 512 bytes of data is transferred, a confirmation is requested. On successful completion, the program ends with return code '0' and the input file is deleted from the AIX print queue. In case of an error, the program ends with return code '-2', which stops the print queue and retains the input file in the queue.

During processing, the **laser** program reports on the status of the print job and the percentage of the file transmitted to the AIX print subsystem (this may be viewed with the AIX command **lpq**). The

status may be ‘CONNECT’ (when requesting the LU6.2 session), ‘SENDING’ (when transmitting spool parameters), and ‘RUNNING’ (when transmitting the document data).

When an error is detected, an e-mail is sent to the document’s owner and to the *root* user (this is hard-coded in the program as ‘ADM_USER’, see *common.h*). An example message to *root* is shown below.

```
From p17112 Wed Nov 24 09:54:01 1999
Date: Wed, 24 Nov 1999 09:54:00 GMT
From: xpto <p7112>
To: root@aixprod.intranet

A message from queuing system:

Stop signal received. Print job cancelled!


******* Print job description*******
Host name       : aixprod
Printer queue   : xerox
Job submitted at : Wed Nov 24 09:53:43 1999
Job Number      : 794
Userid          : p7112
Options         : E
File            : /tmp/xerox.out
Backend program : /sys/bin/wrt2spl
Submit command  : -Plaser-j--
```

For non-AFP documents, which have no *pagedef* and *formdef* specified, form-feed characters (‘\f’) are converted to JES2 channel commands. For AFP documents, the first character of each line is assumed to be a JES2 channel command.

THE MVS COMPONENT

We coded a generic MVS LU 6.2 transaction server that accepts an LU6.2 request and extracts the requested transaction program name and input parameters from the conversation request. A TCB is then scheduled for the requested program.

When started, the transaction server:

- Opens an LU6.2 ACB.
- Schedules a timer for a periodic verification routine. On a VTAM

connection failure resulting from, for example, a VTAM restart or LU6.2 'INACT/ACT' operator command, the routine closes the ACB and periodically attempts to re-open it.

- Waits for three different types of event on an ECB list:
  - *Timer expired*. The verification routine runs and the timer is rescheduled
  - *Console command*. If an MVS stop command request is received, the server stops. The *MODIFY* MVS command is also accepted, but currently has no function
  - *LU6.2 request received*. VTAM schedules an exit routine (*ATTNEXIT*) for each request. This exit increments a 'pending request counter' and posts an ECB for the server. The server wakes up and executes the dispatch program for each LU6.2 pending request.

The dispatch program allocates a working area, receives the queued LU6.2 conversation requests, and retrieves the transaction program name and parameters (if present). Next it attaches TCP to the requested program using the working area as the input parameter. This area contains the LU6.2 conversation control blocks and the program input parameters. An 'ESTAI' recovery routine is also established. In case the transaction ends abnormally, an LU6.2 *DEALLOC/ABNDPROG* verb is sent and the working area is released. The initial TCB program is *TCBINIT*. Once running, *TCBINIT* calls the requested transaction program, which releases the working area allocated by the dispatch program when it terminates. The number of dispatched TCBs is limited by the SNA mode definition.

We wrote the *WRT2SPL* transaction program for this print service. When scheduled, this allocates and initializes dynamic memory and receives the first data block from the AIX back-end print program. This block contains the JES2 *SYSOUT* parameters and the user name of the document's owner. The transaction program then calls the *CHUSER* program to change the TCB *ACEE* for that user name. The user should have a valid account in RACF that's not revoked. A spool *SYSOUT* dataset is then dynamically allocated with the correct owner and an initial disposition 'DEL' – this ensures the dataset is

automatically deleted by JES2 in case of abnormal server termination. The data is then received and written to the JES2 *SYSOUT* dataset. The data received is translated from ASCII to EBCDIC using the table *CONVTBL*. On successful transmit/write, the *SYSOUT* dataset *DISP* is changed from *DEL* to *KEEP* and closed.

BUILDING AND INSTALLING THE AIX COMPONENTS:

First compile the module *common.c*. This contains general-purpose routines.

```
cc common.c -c -o common.o
```

Then compile and link the program *laser.c*.

```
cc laser.c common.o -o laser -g -l qb -l sna
```

DEFINING THE PRINT QUEUE

Using **smit**, choose the 'Print Spooling' option followed by 'Add a Print Queue'.

Specify *QUEUE*, *QUEUE DEVICE*, *BACKEND PROGRAM pathname*, and *BACKEND OUTPUT FILE*. For example:

```
                                Add a Print Queue

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Name of QUEUE to add                               [laser]
* Name of QUEUE DEVICE to add                        [laser]
* BACKEND PROGRAM pathname                           [/<exec_dir>/laser]
  ACTIVATE the queue?                                 yes                      +
  Should this become the DEFAULT queue?               no                       +
  Queuing DISCIPLINE                                  first come first serve   +
  ACCOUNTING FILE pathname                            []                       /
  HOSTNAME of remote server                           []
  Name of QUEUE on remote server                      []
  Pathname of the SHORT FORM FILTER for queue         []                       +/
   status output
  Pathname of the LONG FORM FILTER for queue          []                       +/
   status output
  BACKEND OUTPUT FILE pathname                        [/<log_dir>/laser.log]   /
  ACCESS MODE of backend output file                  write only               +
```

```
Print HEADER pages?                              never                +
Print TRAILER pages?                             never                +
ALIGN page if printer has been idle?             yes                  +
Number of FORM FEEDS when printer goes idle      [0]
```

<*exec_dir*> is the directory where the compiled laser program is located and <*log_dir*> the directory of the printer log.

AIX SNA SERVER CONFIGURATION

Two LU6.2-independent LUs are needed, one for the AIX back-end printer program and another for the MVS started task. The following parameters need to be modified when you define the AIX SNA server profiles below:

- *AIXPROD1* – your independent local LU6.2 name.
- *MVSPROD1* – the independent MVS LU6.2 name.
- *NETA*– your SNA network name.
- *CPNAME* – the partner LU that owns the control point name.

LOCAL_LU_LU6.2

```
prof_name                                = 'AIXPROD1'
local_lu_name                            = 'AIXPROD1'
local_lu_alias                           = 'AIXPROD1'
local_lu_dependent                       = no
local_lu_address                         =
sscp_id                                  = *
link_station_prof_name                   = ''
conversation_security_list_profile_name  = ''
rrm_enabled                              = no
comments                                 = ''
```

PARTNER_LU6.2

```
prof_name                                = 'MVSPROD1'
fq_partner_lu_name                       = 'NETA.MVSPROD1'
partner_lu_alias                         = 'MVSPROD1'
session_security_supp                    = no
parallel_session_supp                    = yes
conversation_security_level              = none
comments                                 = ''
```

PARTNER_LU6.2_LOCATION

```
prof_name                                  = 'MVSPROD1'
fq_partner_lu_name                         = 'NETA.MVSPROD1'
partner_location_method                    = owning_cp
fq_partner_owning_cp_name                  = 'NETA.CPNAME'
local_node_is_network_server_for_len_node = no
fq_node_server_name                        = ''
local_lu_name                              = ''
link_station_profile_name                  = ''
comments                                   = ''
```

SIDE_INFO

```
prof_name                                  = 'MVSPROD1'
local_lu_or_control_pt_alias               = 'AIXPROD1'
partner_lu_alias                           = 'MVSPROD1'
fq_partner_lu_name                         = ''
mode_name                                  = 'LU62CONV'
remote_tp_name_in_hex                      = no
remote_tp_name                             = ''
comments                                   = 'AIX to MVS printing'
```

MODE

```
prof_name                                  = 'LU62CONV'
mode_name                                  = 'LU62CONV'
max_sessions                               = 50
min_conwinner_sessions                     = 25
min_conloser_sessions                      = 25
auto_activate_limit                        = 5
max_adaptive_receive_pacing_window         = 16
receive_pacing_window                      = 7
max_ru_size                                = 512
min_ru_size                                = 128
class_of_service_name                      = '#CONNECT'
comments                                   = ''
```

Below are files that provide a sample VTAM configuration.

SAMPLE LU6.2 DEFINITION

```
*********************************************************************
**                                                                 **
**       VTAM APPLICATIONS                                         **
**       HOST APPLS: MVS DEVELOPMENT APPLICATION                   **
**                                                                 **
*********************************************************************
         VBUILD TYPE=APPL
```

```
MVSPROD1 APPL  APPC=YES,SRBEXIT=NO,                                    X
               DSESLIM=20,DMINWNL=10,DMINWNR=10,                       X
               MODETAB=APPCTAB
```

SAMPLE APPCTAB MODETAB DEFINITION

```
LOGLU62  MODEENT LOGMODE=LOGLU62,                                      *
               FMPROF=X'12',                                           *
               TSPROF=X'04',                                           *
               PRIPROT=X'B1',                                          *
               SECPROT=X'B1',                                          *
               COMPROT=X'70A0',                                        *
               RUSIZES=X'8686',                                        *
               PSNDPAC=X'03',                                          *
               SRCVPAC=X'04',                                          *
               SSNDPAC=X'05',                                          *
               PSERVIC=X'06003800000C380000000000'
```

Below are files that provide a sample MVS configuration. Note that the output load PDS should be an APF-authorized library.

SAMPLE JCL FOR THE MVS SERVER ASSEMBLY

```
//UNIXSRV JOB (0001),'ASM',CLASS=B,
//             NOTIFY=&SYSUID,
//             MSGCLASS=X
//*
//ASM        PROC MBR=               *MEMBER NAME
//ASM    EXEC PGM=ASMA90,REGION=4M,
//            PARM='DECK,NOOBJECT,ALIGN'
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//         DD DSN=server_source_pds,DISP=SHR
//SYSUT1   DD UNIT=WORK,SPACE=(1700,(400,400))
//SYSUT2   DD UNIT=WORK,SPACE=(1700,(400,400))
//SYSUT3   DD UNIT=WORK,SPACE=(1700,(400,400))
//SYSPUNCH DD DSN=&&LOADSET,UNIT=WORK,DISP=(,PASS),
//           SPACE=(400,(100,100,1))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD DSN=server_source_pds(&MBR),DISP=SHR
//*
//LKED   EXEC PGM=IEWL,COND=(7,LT,ASM),
//            PARM='AC=1,LIST,LET,XREF,MAP'
//SYSLIB   DD  DSN=MVSSPPC.UNIXSRV.LOAD,DISP=SHR
//SYSLIN   DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//SYSUT1   DD UNIT=WORK,SPACE=(1024,(200,20))
//SYSLMOD  DD DSN=authorized_server_load(&MBR),DISP=SHR
//SYSPRINT DD SYSOUT=*
```

```
//   PEND
//*
//CHUSER   EXEC ASM,MBR=CHUSER
//WRT2SPL  EXEC ASM,MBR=WRT2SPL
//TCBINIT   EXEC ASM,MBR=TCBINIT
//DISPATCH EXEC ASM,MBR=DISPATCH
//UNIXSRV EXEC ASM,MBR=UNIXSRV
```

SAMPLE JCL PROCEDURE TO RUN THE MVS SERVER

```
//UNIXSRV   PROC
//*
//UNIXSRV  EXEC PGM=UNIXSRV,TIME=1440,REGION=0K
//STEPLIB  DD  DISP=SHR,DSN=authorized_server_load
```

The application's source code is listed below.

AIX PROGRAM LASER.C

```
/*********************************************************************/
/* PROGRAM:  laser.c                                                 */
/*                                                                   */
/* REQUIREMENTS:  SNA Services/6000                                  */
/*                LU6.2 Connection to remote system.                 */
/*                                                                   */
/*********************************************************************/

/* Include Files              */
#include 'common.h'

int     sfd;
long    rid;
char    hostname[64];
int     total_bytes_written;
char    prog_id[80];
char    adm_user[10];
char    file_name[80];
char    options[80];

struct passwd *user;

int main(argc, argv, envp)
 int     argc;
 char    *argv[];
 char    *envp[];

{
 FILE *fptr;
 char message[200];
```

```
unsigned int  i,j,step,count,percent,page_count,numparms;
char mode;
char buffer [512];
int  parmpos[20];

struct {
       char    userid[8];
       char    class;
       char    dest[18];
       char    form[8];
       char    formdef[6];
       char    pagedef[6];
       char    chars[4];
       char    filler;
       short   reg_len;
       short   n_lines;
       short   n_copies;
       } header;

log_init();
log_status(INIT);
sprintf(prog_id,'%s',argv[0]);
user=getpwuid(getuid());
sprintf(file_name,'%s',argv[argc-1]);
if (argc > 2)
   sprintf(options,'%s',argv[1]);
 else
   sprintf(options,'*default parameters*');

gethostname(hostname,sizeof(hostname));
sprintf(adm_user,ADM_USER);

signal(SIGUSR1, (void (*)(int)) abender); /* catch sna signals */
signal(SIGTERM, (void (*)(int)) abender); /* catch software signals */

header.class=' ';
for(i=0;i<sizeof(header.userid);i++)  header.userid[i]=' ';
for(i=0;i<sizeof(header.dest);i++)    header.dest[i]=' ';
for(i=0;i<sizeof(header.form);i++)    header.form[i]=' ';
for(i=0;i<sizeof(header.formdef);i++) header.formdef[i]=' ';
for(i=0;i<sizeof(header.pagedef);i++) header.pagedef[i]=' ';
for(i=0;i<sizeof(header.chars);i++)   header.chars[i]=' ';

get_userid(header.userid);
if (argc == 3 )
 {
  j=0;
  for(i=0;argv[1][i]!='\0';i++)
    if (argv[1][i]!=' ') {buffer[j]=argv[1][i];j++;}
  buffer[j]='\0';
  numparms=get_parm_pos(buffer,parmpos);
```

```
  if (numparms>0)
    get_parm(buffer,&(header.class),1);
  if (numparms>1)
    get_parm(&(buffer[parmpos[0]]),header.dest,sizeof(header.dest));
  if (numparms>2)
    get_parm(&(buffer[parmpos[1]]),header.form,sizeof(header.form));
  if (numparms>3)
    get_parm(&(buffer[parmpos[2]]),header.formdef,sizeof(header.formdef));
  if (numparms>4)
    get_parm(&(buffer[parmpos[3]]),header.pagedef,sizeof(header.pagedef));
  if (numparms>5)
     get_parm(&(buffer[parmpos[4]]),header.chars,sizeof(header.chars));
    }
fptr = fopen (argv[argc-1], 'r');
if (fptr==NULL) call_file_not_found();
header.reg_len=0;
header.n_lines=0;
fgets (buffer, sizeof(buffer), fptr);
while (!feof(fptr))
 {
  header.n_lines+=1;
  i=strlen(buffer);
  if (i>header.reg_len) {header.reg_len=i;};
    fgets (buffer, sizeof(buffer), fptr);
 }
fclose(fptr);

header.n_copies=get_copies();

call_snaopen('MVSPROD1');
log_status(CONNECT);
call_snalloc('','WRT2SPL',SYNC_CONF,WHEN_SESSION_ALLOC,SECUR_NONE,'','','M');
log_status(SENDING);
call_snawrit(sizeof(header), 0, (char *)&header, 'M');
call_flush('M');
call_confirm('M');
log_status(RUNNING);
fptr = fopen (argv[argc-1], 'r');
if (fptr==NULL) call_file_not_found();
page_count=1;
log_pages(page_count);
count=0;
percent=0;
step=header.n_lines/100;
if (step==0) step=1;

if ((header.pagedef[0]==' ')&&(header.formdef[0]==' '))
  {
   fgets (&(buffer[1]), sizeof(buffer)-1, fptr);
   buffer[0]=' ';
   while (!feof(fptr))
```

```
     {
      buffer[strlen(buffer)-1]=' ';
      if (buffer[1]=='\f')  /* formfeed test */
       {
        page_count++;
        log_pages(page_count);
        buffer[0]='1';
        buffer[1]=' ';
       }
      call_snawrit(strlen(buffer),0,buffer,'M');
      count++;
      if (percent<count)
       {
        log_percent((char)((percent*100)/header.n_lines));
        percent=count+step;
       }
      if (total_bytes_written>512)
       {
        total_bytes_written=0;
        call_confirm('M');
       }
      buffer[0]=' ';
      fgets (&(buffer[1]), sizeof(buffer)-1, fptr);
     }
   }
  else
   {
    fgets ((buffer), sizeof(buffer), fptr);
    while (!feof(fptr))
     {
      buffer[strlen(buffer)-1]=' ';
      call_snawrit(strlen(buffer),0,buffer,'M');
      count++;
      if (percent<count)
       {
        log_percent((char)((percent*100)/header.n_lines));
        percent=count+step;
       }
      if (total_bytes_written>512)
       {
        total_bytes_written=0;
        call_confirm('M');
       }
      fgets ((buffer), sizeof(buffer), fptr);
     }
   }
 fclose (fptr);
 call_confirm('M');
 call_snadeal(DEAL_DEFAULT,DISCARD,'M');
 call_snaclse();
} /* End Main */
```

AIX PROGRAM COMMON.H

```
/* Include Files               */
#include <fcntl.h>
#include <luxsna.h>
#include <errno.h>
#include <sys/signal.h>
#include <sys/time.h>
#include <sys/select.h>
#include <NLxio.h>
#include <stdio.h>
#include <pwd.h>
#include <IN/backend.h>
#include <IN/standard.h>
#include <unistd.h>

/*  Global Defines  */
#define OK              0
#define YES             1
#define NO              0
#define ERROR          -1
#define SUCCESS         0
#define CONV_ID_LEN     8
#define MAX_LU_LEN      9
#define MAXPARMS       20

#define ADM_USER    'root'

void call_file_not_found();
int  get_parm(char *,char *,int);
int  get_parm_pos(char *,int *);
int  get_userid(char *);
int  notify(char *);
int  abender(int);
int  handle_errors(int);
int  call_snaopen(char *);
int  call_snaclse();
int  call_snalloc(char *,char *,int,int,int,char *,char *,int);
int  call_snadeal(int,int,int);
int  call_snawrit(int,int,char *,int);

int  call_flush(int);
int  call_confirm(int);
```

AIX PROGRAM COMMON.C

```
/*********************************************************************/
/* PROGRAM:  Common programs of application UNIXSERV                 */
/*                                                                   */
/* REQUIREMENTS:  SNA Services/6000 Version 1.1.101.220 (or later)   */
```

```
/*                LU6.2 Connection to remote system.                    */
/*                                                                      */

#include ‘common.h’

extern  int        sfd;
extern  long       rid;
extern  int        errno;
extern  int        a64l();
extern  char       prog_id[80];
extern  int        total_bytes_written;
extern  char       hostname[64];
extern  char       adm_user[10];
extern  char       file_name[80];
extern  char       options[80];
extern  struct     passwd *user;

void call_file_not_found()
{
  char message[200];
  struct passwd *user;
  user=getpwuid(getuid());
  sprintf(message,’\n\n\nUnable to open input file !\n’);
  notify(message);
  exit(EXITBAD);
}

int  get_parm(parm,var,size)
char *parm,*var;
int  size;
{
  unsigned int i;
  for(i=0;((parm[i]!='\0')&&(parm[i]!=',')&&(i<size));i++) var[i]=parm[i];
  for(i=i;i<size;i++) var[i]=' ';
}

int get_parm_pos(str,parmpos)
char *str;
int  *parmpos;
{
  int i,j;
  for (i=0;i<MAXPARMS;i++) parmpos[i]=0;
  j=0;
  for (i=0;((str[i]!='\0')&&(j<MAXPARMS));i++) if (str[i]==',')
    {parmpos[j]=i+1;j++;};
  return j+1;
}

get_userid(str)
char *str;
{
```

```
  struct passwd *user;
  unsigned int  i;
  user=getpwuid(getuid());
  for(i=0;((i<8)&&(user->pw_name[i]!='\0'));i++) str[i]=user->pw_name[i];
}

int notify(message)
char *message;
{
  char adm_msg[1024];
  sprintf(adm_msg,'%s\n******* Print job description*******\n',
          message,adm_msg);
  sprintf(adm_msg,'%sHost name       : %s\n',adm_msg,hostname);
  sprintf(adm_msg,'%sPrinter queue   : %s\n',adm_msg,get_queue_name());
  sprintf(adm_msg,'%sJob submited at : %s\n',adm_msg,get_qdate());
  sprintf(adm_msg,'%sJob number      : %d\n',adm_msg,get_job_number());
  sprintf(adm_msg,'%sUser id         : %s\n',adm_msg,user->pw_name);
  sprintf(adm_msg,'%sOptions         : %s\n',adm_msg,options);
  sprintf(adm_msg,'%sFile            : %s\n',adm_msg,file_name);
  sprintf(adm_msg,'%sBackend program : %s\n',adm_msg,prog_id);
  sprintf(adm_msg,'%sSubmit command  : %s\n',adm_msg,get_cmd_line());
  sysnot(adm_user,hostname,adm_msg,DOMAIL);
  sprintf(adm_msg,'%s\n\nPlease contact your technical support.\n',adm_msg);
  sysnot(user->pw_name,hostname,adm_msg,DOMAIL);
  printf('%s\n',adm_msg);
}

int abender(signo)
int signo;
{
  struct passwd *user;
  char message[200];
  sprintf(message,'\nStop signal received. Print job cancelled!\n\n');
  notify(message);
  if (signo == SIGUSR1)
    printf('%s: Error, received signal 1 from link station!\n',prog_id);
  else
    if (signo == SIGTERM)
      printf('%s: Error, received software signal kill!\n',prog_id);
  printf('%s: Attempting de-allocate with type = ABEND! \n',prog_id);
  call_snadeal(DEAL_ABEND,0,0);
  printf('%s: About to send a SIGKILL to myself, bye! \n',prog_id);
  kill(getpid(), SIGKILL);
}

int handle_errors(aix_err)
int aix_err;
{
  struct passwd *user;
  unsigned int  i;
  char message[200];
```

```
  user=getpwuid(getuid());
  sprintf(message,'\nComunication failure with the MVS server.\n');
  sprintf(message,'%sUnable to print\n',message);
  notify(message);
  printf('%s: Error = %d\n',prog_id,errno);
  if ((aix_err == SNA_PGMDEAL) || (aix_err == SNA_NRMDEAL))
    printf('%s: Conversation was de-allocated by the remote TP\n',prog_id);
  printf('%s: Attempting deallocate with type = ABEND! \n',prog_id);
  call_snadeal(DEAL_ABEND,0,0);
  printf('%s: Exiting program\n',prog_id);
  exit(EXITBAD);
}

int call_snaopen(side_info)
unsigned char side_info[30];
{
  sfd = snaopen(side_info);
  if (sfd == ERROR)
  {
    printf('%s: Open of side_info (snaopen) %s failed !\n',prog_id,side_info);
    handle_errors(errno);
  }
  return(SUCCESS);
}

int call_snaclse()
{
  int return_code;
  return_code = snaclse(sfd);
  if (return_code == ERROR)
  {
    printf('%s: Close file descriptor (snaclse) failed!\n',prog_id);
    handle_errors(errno);
  }
  return(SUCCESS);
}

int call_snalloc(mode_name, tp_name, sync_level, ret_ctl, security,
                 uid, pw, c_type)
char mode_name[9];
char tp_name[65];
int  sync_level;
int  ret_ctl;
int  security;
char uid[SECUR_USERID_LEN];
char pw[SECUR_PASSWD_LEN];
int  c_type;
{
  int return_code;           /* return code           */
  struct allo_str allo_str;  /* allocate parameters   */
  memset(&allo_str, 0, sizeof(allo_str));
```

```
  allo_str.rid = rid;
  strcpy(allo_str.mode_name, mode_name);
  strcpy(allo_str.tpn, tp_name);
  allo_str.sync_level = sync_level;
  allo_str.return_control = ret_ctl;
  strncpy(allo_str.pgm.user_id, uid, SECUR_USERID_LEN-1);
  strncpy(allo_str.pgm.passwd, pw, SECUR_PASSWD_LEN-1);
  allo_str.type = (c_type=='B') ? BASIC_CONV : MAPPED_CONV;
  rid = snalloc(sfd, &allo_str, c_type);
  if (return_code == ERROR)
  {
    printf('%s: Conversation Allocation (snalloc) failed !\n',prog_id);
    handle_errors(errno);
  }
  return(SUCCESS);
}

int call_snadeal(deal_type, deal_flag, c_type)
int  deal_type;
int  deal_flag;
int  c_type;
{
  int return_code;          /* return code           */
  struct deal_str deal_str; /* allocate parameters   */
  memset(&deal_str, 0, sizeof(deal_str));
  deal_str.rid = rid;
  deal_str.type = deal_type;
  deal_str.deal_flag = deal_flag;
  rid = snadeal(sfd, &deal_str, c_type);
  if (return_code == ERROR)
  {
    printf('%s: Conversation deallocation (snadeal) failed!\n',prog_id);
    handle_errors(errno);
  }
  return(SUCCESS);
}

int call_snawrit(length, translate, ascii_buf, c_type)
int  length;
int  translate;
char ascii_buf[512];        /* temporary storage   */
int  c_type;
{
  int bytes_written;          /* Bytes written       */
  char buf[512];              /* Input buffer        */
  char *data_ptr;             /* Temporary data ptr  */
  struct write_out write_out; /* Allocate parameters */
  memset(buf, 0, sizeof(buf));
  memset(&write_out, 0, sizeof(write_out));
  if (length > 0)
  {
```

```
    if (c_type == 'M')
    {
      if (translate == YES)
        length = NLxout(buf, ascii_buf, length);
      else
        memcpy(&buf,ascii_buf,length);
    }
    else
    {
      if (translate == YES)
        length = NLxout(&buf[2], ascii_buf, length);
      else
        memcpy(&buf,ascii_buf,length);
      buf[0] = 0;
      length=+2;
      buf[1] = length;
    }
  }
  bytes_written = snawrit(sfd, buf, length, rid, &write_out, c_type);
  if (bytes_written == ERROR)
  {
    printf('%s: Write Data (snawrit) failed !\n',prog_id);
    handle_errors(errno);
  }
  total_bytes_written+=bytes_written;
  return(SUCCESS);
}

int call_flush(c_type)
int c_type;
{
  int return_code;            /* return code        */
  struct flush_str flush_str; /* flush parameters   */
  memset(&flush_str, 0, sizeof(flush_str));
  flush_str.rid = rid;
  return_code = snactl(sfd, FLUSH, (int) &flush_str, c_type);
  if (return_code == ERROR)
  {
    printf('%s: Flush (snactl(FLUSH)) failed !\n',prog_id);
    handle_errors(errno);
  }
  return(SUCCESS);
}

int call_confirm(c_type)
int c_type;
{
  int return_code;               /* return code         */
  struct confirm_str confirm_str; /* confirm parameters  */
  memset(&confirm_str, 0, sizeof(confirm_str));
  confirm_str.rid = rid;
```

```
  return_code = snactl(sfd, CONFIRM, (int) &confirm_str, c_type);
  if (return_code == ERROR)
  {
    printf('%s: Confirm (snactl(CONFIRM)) failed !\n',prog_id);
    handle_errors(errno);
  }
  return(SUCCESS);
}
```

MVS PROGRAM UNIXSRV

```
         TITLE ' MVS COMMAND SERVER '
         ENTRY UNIXSRV
UNIXSRV  RMODE 24                required for open acb macro
UNIXSRV  AMODE 31
UNIXSRV  CSECT
MAIN     BAKR  R14,0
         LA    R10,0(R15,0)
         USING UNIXSRV,R10
         LA    R13,SAVEAREA
         MVC   4(4,R13),=C'F1SA'
*  test if already running
         ENQ   (QNAME,RNAME,E,L'RNAME,SYSTEM),MF=(E,ENQLIST)
         LTR   R15,R15
         BZ    UNIQUE
*
         WTO   'UNIXSRV- ALREADY ACTIVE'
         SR    R15,R15
         PR
*
UNIQUE   LA    R9,MAINRPL             Establish addressability
         USING IFGRPL,R9                 of MAIN routine RPL
         LA    R8,MAINRPL6            Establish addressability
         USING ISTRPL6X,R8               of MAIN routine RPL Extension
*
*********************************************************************
*                                                                   *
*    Issue OPEN macro instruction to identify this program to VTAM.  *
*    The OPEN macro instruction references a VTAM ACB with label     *
*    ACB. The ACB is VTAM's representation of the LU. All VTAM       *
*    macros reference this ACB.                                      *
*                                                                   *
*********************************************************************
*
         XR    R15,R15                Initialize register 15 = 0
         OPEN  ACB
         LTR   R15,R15                Test OPEN
         BZ    OPENOK                 Branch if successful
         WTO   'UNIXSRV- UNABLE TO ESTABLISH COMMUNICATIONS'
         B     MAINCODE               Branch to MAIN return
```

```
*
*********************************************************************
*                                                                   *
*    Issue SETLOGON macro instruction to enable VTAM to accept      *
*    LU 6.2 session initiation request on behalf of the application *
*    program. The RPL operand of this and other macro instructions  *
*    specifies the request parameter list (RPL) that is used to send *
*    and receive VTAM information about macro instructions.         *
*                                                                   *
*********************************************************************
*
OPENOK   DS    0H
         SETLOGON RPL=MAINRPL,OPTCD=START
         CLI   RPLRTNCD,USFAOK         RTNCD = X'00' ?
         BNE   SETLOGFL                branch if no
         CLI   RPLFDB2,USFAOOK         FDB2 = X'00'
         BNE   SETLOGFL                branch if yes
         WTO   'UNIXSRV- UP AND RUNNING'
         B     MAINCODE                branch if yes
SETLOGFL DS    0H
         WTO   'UNIXSRV- UNABLE TO ESTABLISH COMMUNICATIONS'
         B     MAINCODE
*
*********************************************************************
*                                                                   *
*    Any user application will include some mechanism that is able  *
*    to initiate a transaction program. That mechanism will be      *
*    triggered by a reply from the following WTOR macro request.    *
*    A reply of 'START' launches code to start a conversation.      *
*    A reply of 'CLOSE' launches code to terminate the application. *
*                                                                   *
*********************************************************************
*
MAINCODE DS    0H
*
         LA    R1,0
         STH   R1,REQCOUNT       set request count to 0
*
         STIMER  REAL,TIMER_EXIT,DINTVL=INTERVAL
         LA    R6,COMADDR       get comm area address
         EXTRACT (R6),FIELDS=COMM,MF=(E,EXTRACT)
*
         L     R6,COMADDR       get address of the area
         USING COM,R6           R9 used as a base reg for comm area
         ICM   R7,15,COMCIBPT   get cib address from comm area
*
         QEDIT ORIGIN=COMCIBPT,CIBCTR=5
         L     R1,COMECBPT      get address of the communications ECB
         ST    R1,@COMECB       put addr of modify ecb in list
*
         LA      R1,TIMERECB        get addr of FM5 received ECB
```

```
         ST      R1,@TIMERECB
         LA      R1,FMH5ECB         get addr of FM5 received ECB
         O       R1,=X'80000000'    set high bit - last ecb in list
         ST      R1,@FMH5ECB
*
LOOP     DS    OH
         L     R2,WAITFMH5
         LTR   R2,R2
         BNZ   NEXTFMH5
*
         WAIT  ECBLIST=ECBS       wait for a message
*
NEXTFMH5 DS    OH
         WTO   'UNIXSRV- WAKEUP TIME'
         L     R4,WAITFMH5
         CVD   R4,WORKAREA
         UNPK  WORKAREA(5),WORKAREA+4(4)
         OI    WORKAREA+4,X'F0'
         LA    R1,WORKAREA
         MVC   NUMFMH5(5),WORKAREA
         WTO   TEXT=WTOWAIT         * Issue statistics message
*****************************************************************
*
         L     R1,TIMERECB        get timer ecb
         N     R1,=X'40000000'    check for post
         BNZ   TIME_OUT           set, process timer
         XC    TIMERECB,TIMERECB  clear message ecb
*
         L     R1,@COMECB         Get CIB ECB  address
         L     R1,0(R1)           get cib ecb
         N     R1,=X'40000000'    check for post
         BNZ   CHK_CIB            not set, check for oper cmd
*
         XC    FMH5ECB,FMH5ECB    clear message ecb
         L     R2,WAITFMH5        test if any FMH5 is pending
         LTR   R2,R2
         BZ    LOOP
*
         WTO   'UNIXSRV- CONVERSATION REQUEST ACCEPTED'
         XC    FMH5ECB,FMH5ECB    clear message ecb
*
         LA    R1,1
DECLOOP  LR    R3,R2
         SR    R3,R1
         CS    R2,R3,WAITFMH5
         BC    4,DECLOOP
*
         LH    R1,REQCOUNT       load,increment and save request count
         LA    R1,1(R1)
         STH   R1,REQCOUNT
*
```

```
         CALL  DISPATCH,(MODLRPL,MODLRPL6,REQCOUNT)
         B     LOOP
*
TIME_OUT DS    0H
         WTO   'UNIXSRV- TIMER ROUTINE SCHEDULED'
         XC    TIMERECB,TIMERECB
         STIMER  REAL,TIMER_EXIT,DINTVL=INTERVAL
*
         SETLOGON RPL=MAINRPL,OPTCD=START   * Test VTAM link
         LTR   R15,R15
         BNZ   OPEN_ACB
         B     LOOP
*
OPEN_ACB DS    0H
         OPEN  ACB
         LTR   R15,R15
         BNZ   COMM_ERR
*
         SETLOGON RPL=MAINRPL,OPTCD=START
         CLI   RPLRTNCD,USFAOK         RTNCD = X'00' ?
         BNE   COMM_ERR                branch if no
         CLI   RPLFDB2,USFAOOK         FDB2 = X'00'
         BNE   COMM_ERR                branch if yes
         WTO   'UNIXSRV- COMMUNICATIONS RE-ESTABLISHED'
         WTO   'UNIXSRV- UP AND RUNNING'
         B     LOOP
*
COMM_ERR DS    0H
         WTO   'UNIXSRV- UNABLE TO ESTABLISH COMMUNICATIONS'
         B     LOOP
*
CHK_CIB  DS    0H
         WTO   'UNIXSRV- CONSOLE COMMUNICATION ROUTINE SCHEDULED'
*
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R7) free de cib
         L     R6,COMADDR       get address of the area
         USING COM,R6           R9 used as a base reg. for comm area
         ICM   R7,15,COMCIBPT   get cib address from comm area
         USING CIB,R7           base cib mapping
         CLI   CIBVERB,CIBMODFY was it a modify?
         BE    MODIFY           yes, process it
         CLI   CIBVERB,CIBSTOP  was it a stop
         BNE   LOOP             no, then loop
         WTO   'UNIXSRV- STOP COMMAND ACCEPTED'
         B     MAINEND
*
MODIFY   DS    0H
         B     LOOP
*
*********************************************************************
*                                                                   *
```

```
*    This termination routine makes the assumption that all          *
*    conversations have completed successfully. Therefore, the tasks *
*    remaining are to CNOS the LOGMODE mode LU62CONV session limits  *
*    to zero and then CNOS the SNASVCMG limits to zero. Once         *
*    the CNOSes have completed, the ACB is CLOSED and the            *
*    application returns to the system.                              *
*                                                                    *
**********************************************************************
*
MAINEND  DS    0H
         CLI   LUSTATE,LUCNOSD         have session limits been set?
         BNE   DOCLOSE                 branch if not
         MVC   RPL6MODE,LOGMODE        set RPL extension LOGMODE
         XC    MAINAREA,MAINAREA       initializing the AREA to zero
*                                      will produce a CNOS structure
*                                      with session limits set to zero
         LA    R1,MAINAREA             address AREA storage
*
         WTO   'UNIXSRV- RESETTING SESSION LIMITS FOR LU62CONV MODE'
*
         APPCCMD CONTROL=OPRCNTL,QUALIFY=CNOS,                         X
               ACB=ACB,RPL=MAINRPL,AAREA=MAINRPL6,                     X
               AREA=MAINAREA,RECLEN=SLCLEN,                            X
               OPTCD=SYN
*
         MVC   RPL6MODE,=CL8'SNASVCMG' Set to CNOS the SNASVCMG mode
*
         WTO   'UNIXSRV- RESETTING SESSION LIMITS FOR SNASVCMG MODE'
*
         APPCCMD CONTROL=OPRCNTL,QUALIFY=CNOS,                         X
               ACB=ACB,RPL=MAINRPL,AAREA=MAINRPL6,                     X
               OPTCD=SYN
*
DOCLOSE  DS    0H

         CLOSE ACB                     Close ACB
MAINRETN DS    0H
         DEQ   (QNAME,RNAME,L'RNAME,SYSTEM),MF=(E,DEQLIST)
         PR
**********************************************************************
*                                                                    *
*        ATTN EXIT Routine                                           *
*                                                                    *
*  ON ENTRY:                                                         *
*       R1 - address of a 6-word parameter list, as documented in    *
*            in the LU 6.2 programming manual                        *
*       R14 - Return address when processing is finished             *
*       R15 - Address of this ATTN exit.                             *
*                                                                    *
**********************************************************************
*
```

```
TPENDEXIT DS   0H
         USING TPENDEXIT,R15          temporary base for this routine
         CNOP  0,4                    full word alignment
         BAL   R13,*+76               branch around save area
         DC    18F'0'                 set up save area chain
         STM   R14,R12,12(R13)        store current registers
         LR    R12,R15                establish normal base register
         DROP  R15                       for this
         USING TPENDEXIT,R12                ATTN exit
         CNOP  0,4                    full word alignment
         BAL   R15,*+76               branch around save area
         DC    18F'0'                 save area for called routines
*                                     and macro requests
         ST    R15,8(0,R13)           chain save areas
         ST    R13,4(0,R15)              together
         LR    R13,R15                set R13 to second savearea
         LR    R11,R1                 preserve parameter address
*
         L     R9,16(R11)             load address of read-only RPL
         USING IFGRPL,R9              establish base register
         L     R8,RPLAAREA            load address of read only RPL6
         USING ISTRPL6X,R8            establish base register
*
         WTO   'UNIXSRV- TPEND EXIT SCHEDULED, COMUNICATIONS ALERT'
*
         L     R13,4(0,R13)           load address of first savearea
         LM    R14,R12,12(R13)        load original registers
         BR    R14                    branch back to VTAM
*
ATTNEXIT DS    0H
         USING ATTNEXIT,R15           temporary base for this routine
         CNOP  0,4                    full word alignment
         BAL   R13,*+76               branch around save area
         DC    18F'0'                 set up save area chain
         STM   R14,R12,12(R13)        store current registers
         LR    R12,R15                establish normal base register
         DROP  R15                       for this
         USING ATTNEXIT,R12                ATTN exit
         CNOP  0,4                    full word alignment
         BAL   R15,*+76               branch around save area
         DC    18F'0'                 save area for called routines
*                                     and macro requests
         ST    R15,8(0,R13)           chain save areas
         ST    R13,4(0,R15)              together
         LR    R13,R15                set R13 to second savearea
         LR    R11,R1                 preserve parameter address
*
         L     R9,16(R11)             load address of read only RPL
         USING IFGRPL,R9              establish base register
         L     R8,RPLAAREA            load address of read only RPL6
         USING ISTRPL6X,R8            establish base register
```

```
*
LUOK     DS    0H
         CLC   RPL6MODE,LOGMODE        is LOGMODE = LU62CONV
         BE    MODEOK                  branch if yes
         CLC   RPL6MODE,=CL8'SNASVCMG' is this SNASVCMG LOGMODE
         BE    MODEOK                  branch if yes
         WTO   'UNIXSRV- #####DEFINITON ERROR#####'
         B     ATTNRETN                branch to EXIT return
*
MODEOK   DS    0H
         CLC   12(4,R11),=CL4'CNOS'    Is this a CNOS event?
         BE    ATTNCNOS                branch if yes to CNOS process
         CLC   12(4,R11),=CL4'FMH5'    Is this an FMH-5 reception ?
         BE    ATTNFMH5                branch if yes to FMH-5 process
         CLC   12(4,R11),=CL4'LOSS'    Is this an UNBIND session
*                                       event?
         BE    ATTNLOSS                branch if yes to LOSS process
*
ATTNCNOS DS    0H
*
         L     R7,RPLAREA              establish addressability to
         USING ISTSLCNS,R7                CNOS structure
         CLC   SLCSESSL,=H'0'          Is the a Reset Session limits?
         BE    RESET                   branch if yes
         MVI   LUSTATE,LUCNOSD         else this request is setting
*                                       limits which this sample
*                                       program will just make note
*                                       session limits have been CNOSed
         B     ATTNRETN                branch to return processing
RESET    DS    0H
         MVI   LUSTATE,LUNCNOS         make note that session limits
*                                       have been reset and that no
*                                       limits are established.
         B     ATTNRETN                branch to return processing
         DROP  R7                      remove basing to CNOS
*                                       structure
*
ATTNFMH5 DS    0H
         WTO   'UNIXSRV- CONVERSATION REQUEST RECEIVED AND QUEUED'
*
         L     R4,=A(WAITFMH5)
         L     R2,0(R4)
         LA    R1,1
INCLOOP  LR    R3,R2
         AR    R3,R1
         CS    R2,R3,0(R4)
         BC    4,INCLOOP
*
         L     R2,=A(FMH5ECB)
         POST  (R2),1
         B     ATTNRETN                Branch to EXIT return
```

```
*
*********************************************************************
*                                                                   *
*        LOSS Processing Routine                                    *
*                                                                   *
*********************************************************************
*
ATTNLOSS DS    0H
         WTO   'UNIXSRV- SESSION LOST DETECTED'
         B     ATTNRETN
*
*********************************************************************
*                                                                   *
*        ATTN return processing                                     *
*                                                                   *
*********************************************************************
*
ATTNRETN DS    0H
         L     R13,4(0,R13)            Load address of first savearea
         LM    R14,R12,12(R13)         Load original registers
         BR    R14                     Branch back to VTAM
*
         DROP  R8,R9,R12          remove basing for passed
*                                       structure
*
*  Exit scheduled by the timer when expired
*
TIMER_EXIT DS    0H
*
         USING TIMER_EXIT,R15
         STM   14,12,12(13)
         LR    R12,R15
         USING TIMER_EXIT,R12
*
         L     R2,=A(TIMERECB)
         POST  (R2),1
*
         L     R14,12(0,R13)
         LM    R0,R12,20(R13)
         BR    R14
         DROP  R12
*********************************************************************
         DS    0D        * Double word work area for mathematics
WORKAREA DS    CL8
*
WTOWAIT   DC    AL2(WTOWAITL)
          DC    C'UNIXSRV- WAITING FMH5 IN QUEUE '
NUMFMH5   DS    CL5
WTOWAITL  EQU   *-WTOWAIT-2
*********************************************************************
*    MAIN routine data area                                         *
```

```
*********************************************************************
*
REQCOUNT DS    AL2
LUENTRY  DS    0D
LUSTATE  DC    XL1'00'
LUNCNOS  EQU   X'00'                  session limits not established
LUCNOSD  EQU   X'80'                  session limits established
*
LOGMODE  DC    CL8'LU62CONV'          LOGMODE name
SAVEAREA DS    20A
INTERVAL DC    CL8'00010000'          wake up every minute
WAITFMH5 DC    F'0'
ECBS     DS    0H
@COMECB  DS    A
@TIMERECB DS    A
@FMH5ECB DS    A
*
TIMERECB DC    F'0'
FMH5ECB  DC    F'0'
*                                     FMH-5
COMADDR  DS    F               comm area address from extract
EXTRACT  EXTRACT MF=L          extract parameter list
*
QNAME    DC    CL8'UNIXSRV'
RNAME    DC    C'MVS COMMAND UNIXSRV'
DEQLIST  DEQ   (,,,SYSTEM),RET=NONE,MF=L
ENQLIST  ENQ   (,,,,SYSTEM),RET=USE,MF=L
*
*********************************************************************
*                                                                   *
*    VTAM and LU6.2-specific variable areas                         *
*                                                                   *
*********************************************************************
*
ACB      ACB   AM=VTAM,                                             X
               EXLST=EXLST,                                        X
               APPLID=APPLID
EXLST    EXLST AM=VTAM,                                             X
               ATTN=ATTNEXIT,                                      X
               TPEND=TPENDEXIT
*
APPLID   DC    AL1(8)
         DC    CL8'MVSPROD1'
*
MAINRPL  RPL   AM=VTAM,ACB=ACB         MAIN task RPL
MAINRPL6 ISTRPL6                       MAIN task RPL6
MAINAREA DC    XL(SLCLEN)'00'          MAIN task AREA
*
         DS    0F
MODLRPL  RPL   AM=VTAM,ACB=ACB         Model RPL
MODLRPL6 ISTRPL6 CONMODE=CS,FILL=LL,LOGMODE=LU62CONV   Model RPL6
```

```
         LTORG
*
*************************************************************************
*                                                                       *
*    Program DSECTs                                                     *
*                                                                       *
*************************************************************************
*
         IFGRPL AM=VTAM
         ISTFM5
         ISTSLCNS
SLCLEN   EQU    SLCEND-ISTSLCNS
         ISTUSFBC
         IFGACB AM=VTAM
*
COM      DSECT
         IEZCOM   ,            com area
CIB      DSECT
         IEZCIB                cib
*
*************************************************************************
*                                                                       *
*    Register EQUATES                                                   *
*                                                                       *
*************************************************************************
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

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

Fernando Manuel Carvalho Nunes
System Programmer
Companhia de Seguros Bonanca (Portugal)

A script to resize a filesystem

In order to take the drudgery out of resizing filesystems, I wrote a script that allows an administrator simply to specify the desired increase in megabytes. The script takes the PP size of the volume group to which the target filesystem belongs into account and advises how many available PPs would be consumed by the specified increase before proceeding. The script also handles circumstances in which the *MAX LPs* parameter needs to be increased to accommodate the request. Finally, it displays the before and after statistics on disk space.

EXAMPLE:

In order to increase the */tmp* filesystem by 32 megabytes, use the following command:

```
CHFS 32 /tmp
```

CHFS

```
#!/bin/ksh
#
# CHFS
#
# Michael Stanton
# Mercedes-Benz USA, LLC.
# 9/15/99
#
# The CHFS script allows the size of a named filesystem to be increased
# by a specified number of megabytes.
#
# Sample usage:
#    To add 32 MB of space to the /tmp filesystem, issue the command:
#    CHFS 32 /tmp
#
# Modification History
#=====================
# 01/31/2000 MGS Handle cases where the MAX LPs need to be increased.
#
#============================================================================
#
# Check that two arguments were supplied
if [[ $# -ne 2 ]]
then
    echo "Usage: CHFS <size in MB> <filesystem>"
```

```
    echo "Example: CHFS 32 /tmp (increase /tmp by 32MB)"
    exit 1
fi
#-----------------------------------------------------------------------
size=$1
FS=$2
#
#-----------------------------------------------------------------------
# Check that the filesystem specified exists
grep -q "${FS}:" /etc/filesystems
if [[ $? -ne 0 ]]
then
    echo "\nThe filesystem ${FS} does not exist. Please check."
    exit 1
fi
#-----------------------------------------------------------------------

echo "Working..."

#-----------------------------------------------------------------------
# Set some variables
typeset -i MAXLPS=0 LPS=0 PPSIZE=0 PPSLEFT=0 PPSTOADD=0
typeset -i FREEMBS=0 VGPPSAVAIL=0 DEFICIT=0
typeset -s LVNAME="" VGNAME="" answer=""

#-----------------------------------------------------------------------
#
# Get settings for LVs, VGs, PPs, etc.
LVNAME=$(lsfs ${FS}|tail -1|awk '{print $1}'|cut -f3 -d '/')
MAXLPS=$(lslv ${LVNAME}|grep "MAX LPs"|awk '{print $3}')
LPS=$(lslv ${LVNAME}|grep "LPs"|grep -v MAX|awk '{print $2}')
PPSIZE=$(lslv ${LVNAME}|grep "MAX LPs"|awk '{print $6}')
VGNAME=$(lslv ${LVNAME}|head -1|awk '{print $6}')
VGPPSAVAIL=$(lsvg ${VGNAME}|grep "FREE PPs"|awk '{print $6}')
PPSLEFT=$((MAXLPS-LPS))
PPSTOADD=$((size/PPSIZE))
FREEMBS=$((VGPPSAVAIL * PPSIZE))
#
#-----------------------------------------------------------------------
if ((PPSTOADD > VGPPSAVAIL))
then
    echo "Not enough free PPs to satisfy your request for ${size} MBs."
    echo "There are only ${VGPPSAVAIL} PPs left (${FREEMBS} MBs) in
         ➤  ${VGNAME}."
    exit 0
fi

#-----------------------------------------------------------------------
echo "\nThe number of Free PPs in that VG (${VGNAME}) is:
     ➤  ${VGPPSAVAIL}."
echo "Your request of ${size} MBs will use up ${PPSTOADD} of those.\n"
```

```
answer="x"
while [[ "${answer}" = "x" ]]
do
   echo "Do you want to continue? (y/n): \c"
   read answer rest
   case ${answer} in
   "y"|"Y")
   :                                  # Continue
   ;;
   *)
    echo "Exiting procedure."        # Exit the script
    exit 0
   ;;
   esac
done

#--------------------------------------------------------------------
if ((PPSTOADD > PPSLEFT))
then
    DEFICIT=$((PPSTOADD - PPSLEFT))   # Calculate new value of MAXLPs

    echo "\nThe present MAX LPs for ${LVNAME} is set to: ${MAXLPS}."
    echo "To add ${size} MBs, the MAX LPs parameter must be increased"
    echo "for this logical volume (${LVNAME}) by ${DEFICIT} PPs.\n"

    answer="x"

    while [[ "${answer}" = "x" ]]
    do
        echo "Increase the MAX LPs by ${DEFICIT} PPs? (y/n): \c"
        read answer rest
        case ${answer} in
            "y"|"Y")
                    echo "Executing: chlv -x ${DEFICIT} ${LVNAME}"
                    chlv -x $((DEFICIT + LPS)) ${LVNAME}
                    if [ $? -eq 0 ]
                    then
                        echo " MAX LPs has been increased."
                    fi
                    ;;
                *)
                    echo "Exiting procedure."
                    exit 0
                    ;;
        esac
    done
fi

#--------------------------------------------------------------------
before=$(df -k ${FS})
```

```
answer=""
echo "\nIncrease ${FS} by ${size} MB? (y/n): \c"
read answer extra
case ${answer} in
     "y"|"Y")
               echo "Working..."
               ;;
           *)
               echo "No changes will be made."
               exit 0
               ;;
esac

#-----------------------------------------------------------------------

chfs -a size="+$((2 * ${size}000))" ${FS}   # Increase the filesystem

#-----------------------------------------------------------------------

after=$(df -k ${FS})
#
echo "\n(BEFORE)\n${before}"                # Show the difference
echo "\n(AFTER) \n${after}\n"
#
#eoj
```

Michael G Stanton
Supervisor, Midrange Systems
Mercedes-Benz (USA)

Mailto – HTML help file

The code and documentation for Mailto were published in issues 57 and 58 of *AIX Update*, and it had been our intention to publish the HTML help file in this issue. However, in view of the fact that this is very similar to the printed documentation, we've decided not to include it in this issue – you can download the file from:

```
http://www.xephon.com/extras/mailto.htm
```

AIX news

IBM has unveiled its WebSphere Edge Server, which sits at local and remote network boundaries and provides services on behalf of back-end servers to improve Web application performance, availability, and scalability. It provides dynamic load balancing across Web servers and geographic sites, including content-based routing for service differentiation among incoming requests. It also provides Web content caching and screening and filtering of non-productive or undesirable URLs.

The integrated system includes enhanced versions of proxy caching, a new streamlined install/configuration process, and a network dispatcher, providing high-availability back-up, QoS routing based on Type of Service (ToS) rules, Lotus Domino Server scalability for POP3 and IMAP4 clients, and cross-port affinity. It runs on AIX, NT, Windows 2000, Solaris 7, and Linux. Out now, it costs US$8,000 for both 56-bit and 128-bit encryption versions.

Separately, the company announced Version 2.2 of its WebSphere Host Publisher for AIX. It supports applications written for 3270, 5250, VT, Java, and databases that provide a JDBC interface and it supports any HTML-based browser.

Out now, it costs US$15,000.

For further information, contact your local IBM representative.

* * *

StorageTek has announced that its 9500 Shared Virtual Array (SVA) 1.5 disk system and Virtual Power Suite software now support AIX, Solaris, HP-UX, and Windows NT/2000 with Fibre Channel direct-attach connection.

The Virtual Power Suite now has easier centralized management, comprehensive reporting, greater availability, and SnapShot data duplication software. Specifically, it now includes SVA Reporter, providing information and reports from the 9500 SVA; SVA NMP, allowing communication with CA Unicenter; SVA Path, providing fail-over path-balancing, and Power Peer-to-Peer Remote Copy (PPRC) for real-time backup, now with increased distance capabilities, higher performance and throughput, and faster initial synchronization (from five minutes to two minutes).

Availability and pricing weren't announced.

For further information contact:
Storage Technology Corp, 2270 S 88th St, Louisville, CO 80028, USA
Tel: +1 303 673 5151
Fax: +1 303 673 8876
Web: http://www.storagetek.com

Storage Technology Ltd, Storage Tek House, Woking Business Park, Albert Drive, Woking, Surrey GU21 5JY, UK
Tel: +44 1483 737333
Fax: +44 1483 737222

* * *