97

AIX update

November 2003

In this issue

3 Updating anti-virus driver files

12 Brocade FC network port investigation

20 The awk command

35 Manage FTP process

53 Character to hex

54 AIX news

© Xephon plc 2003

AIX Update

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

North American office
Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

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 1999 issue, are available separately to subscribers for £16.00 ($24.00) 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.

Editor
Trevor Eddolls

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.

Contributions

When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) 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.

© Xephon plc 2003. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.

Updating anti-virus driver files

One of the most important issues is to protect your system against viruses. Generally, viruses spread by e-mails. This means it is vital to ensure that your mail gateways (MTAs – Mail Transfer Agents) have the latest virus driver files to protect your system. The virus driver files are produced by companies for their appropriate anti-virus products and the anti-virus API used by the attachment filter's virus scanner. They are updated frequently, usually whenever a new virus threat is identified. Therefore, to avoid being infected by the new viruses it is very important to put the updates on the mail servers as soon as a newer version virus driver is available. I have written a script that works on our MTAs at regular intervals using the system scheduling tool, cron. In our company the script works every hour and downloads and updates the virus driver files when it finds a new package available on the vendor's FTP server.

Cron entry:

```
0 * * * *  /usr/local/av/ftp_av.sh > /dev/null 2>&1
```

Steps of the script:

1 It checks whether an FTP process is hung from a previous execution of the script. If yes, it is killed in order to keep the system clear of the hanged processes.

2 It first FTPs an information file (update.ini). This small information file contains the version number of the latest virus driver files. That version number is compared with the version of the virus drivers in the MTAs. If they differ, the script will FTP the new virus driver package.

   An information file looks like this:

```
UPDATE.INI

[SuperDat-IA32]
EngineVersion=4260
DATVersion=4285
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
FileName=sdat4285.exe
FileSize=5569972
Checksum=BB1E,BDF2

[ZIP]
EngineVersion=0
DATVersion=4285
FileName=dat-4285.zip
FilePath=/pub/antivirus/datfiles/4.x/
FileSize=3227863
Checksum=4FF2,D26D
MD5=a8d394e5fec24209bdca2a60ad2b569b

[Incremental]
EngineVersion=0
DATVersion=4285
FileName=delta.ini
FileSize=1303
Checksum=7083,697F

[Engine-LINUX]
EngineVersion=4240
FileName=elnx4240.zip
FileSize=1006785
Checksum=781E,B2DE
MD5=d461201cf97ce2ef79b91f6cc34d3202
FilePath=/pub/antivirus/engine/4.x/

[Engine-NETWARE]
EngineVersion=4240
FileName=nw4240.zip
FileSize=3303598
Checksum=1C67,E3DA
MD5=de2e9111236428905ce867ddfab41ee7
FilePath=/pub/antivirus/engine/4.x/
```

In this information file (update.ini), it is important to get the DAT Version number:

```
DATVersion=4285
```

4285 is the newest version number of the virus driver, which has to be compared with the current version number of the virus driver on the mail server.

```
# /usr/local/sendmail/smmfilter-2.4/bin/avupdate –c

avupdate: DAT version in /var/db/mime-filter/drivers.4284 is 4284
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

Since the DAT version on the vendor FTP site is greater than the local mail server, the new package (dat-4285.tar) has to be downloaded. 4285 should be the highest number in the directory; in other words, the latest virus driver available.

3 After the new package is successfully downloaded, it is installed. After successful installation of the new package, the script sends an e-mail to the sendmail administrators saying that a new package has been received with an attached log file. If not successfully completed, it sends an e-mail to the sendmail administrators with the error text and the log file in order to examine what went wrong.

An example of an e-mail sent by a script is:

```
Wed Aug 13 22:31:43 CEST 2003

STARTING FTP (GET) .....

Connected to ftp.nai.com.
220 sncwebftp5 Microsoft FTP Service (Version 5.0).
331 Anonymous access allowed, send identity (e-mail name) as password.
230-You are connected to ftp.nai.com.
230-Your use is subject to the terms and
230-conditions in Legal.TXT and Usage.TXT files.
230-*Mirror sites at FTPEUR.NAI.COM and FTPDE.NAI.COM*
230 Anonymous user logged in.
Interactive mode off.
200 Type set to I.
250 CWD command successful.
Hash mark printing on (1024 bytes/hash mark).
200 PORT command successful.
150 Opening BINARY mode data connection for update.ini(748 bytes).
#
226 Transfer complete.
748 bytes received in 0.000111 seconds (6581 Kbytes/s)
local: update.ini remote: update.ini
221   Thanks for using ftp.nai.com!

INFO:  update.ini file RECEIVED!

AVUPDATE

Connected to ftp.nai.com.
220 sncwebftp6 Microsoft FTP Service (Version 5.0).
331 Anonymous access allowed, send identity (e-mail name) as password.
230-You are connected to ftp.nai.com.
230-Your use is subject to the terms and conditions
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
230-in Legal.TXT and Usage.TXT files.
230-*Mirror sites at FTPEUR.NAI.COM and FTPDE.NAI.COM*
230 Anonymous user logged in.
Interactive mode off.
200 Type set to I.
250 CWD command successful.
Hash mark printing on (1024 bytes/hash mark).
200 PORT command successful.
150 Opening BINARY mode data connection for dat-4285.tar(3717120 bytes).
226 Transfer complete.
3717120 bytes received in 745.9 seconds (4.867 Kbytes/s)
local: dat-4285.tar remote: dat-4285.tar
221    Thanks for using ftp.nai.com!
INFO:  dat-4285.tar FILE RECEIVED!
<<< 220 venus FTP server (Version 4.1 Thu Sep 12 23:46:23 CDT 2002)
ready.
>>> USER smadmin
<<< 331 Password required for smadmin.
>>> PASS sendmail
<<< 230-Last unsuccessful login: Wed Jul 16 22:06:18 CEST 2003 on ftp
<<< 230-Last login: Tue Aug 12 01:07:09 CEST 2003 on ftp
<<< 230 User smadmin logged in.
>>> CWD /var/db/mime-filter/ftp
<<< 250 CWD command successful.
>>> TYPE A
<<< 200 Type set to A; form set to N.
>>> PASV
<<< 227 Entering Passive Mode (217,110,62,85,249,189)
>>> RETR update.ini
<<< 150 Opening data connection for update.ini (748 bytes).
<<< 226 Transfer complete.
>>> CWD /pub/antivirus/datfiles/4.x
<<< 250 CWD command successful.
>>> TYPE I
<<< 200 Type set to I.
>>> PASV
<<< 227 Entering Passive Mode (217,110,62,85,249,191)
>>> RETR dat-4285.tar
<<< 150 Opening data connection for dat-4285.tar (3717120 bytes).
<<< 226 Transfer complete.
>>> QUIT
<<< 221 Goodbye.
avupdate: current driver directory is /var/db/mime-filter/drivers.4284
avupdate: DAT version in /var/db/mime-filter/drivers.4284 is 4284
avupdate: venus: trying 217.101.55.55...
avupdate: connected to venus [217.101.55.55]
avupdate: DAT version at venus is 4285
avupdate: downloading temporary copy to /var/db/mime-filter/new/dat-
4285.tar
avupdate: unpacking /var/db/mime-filter/new/dat-4285.tar
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
avupdate: renaming driver directory: drivers.4285, keeping Ø old drivers
avupdate: drivers updated to version 4285

INFO: AVUPDATE successfully executed ...
```

We see that in this case the new package, 4285, is successfully installed and Version 4284 is now updated to Version 4285. Avupdate tries to start the anti-virus engine using the driver files currently in */var/db/mime-filter/drivers*. The attachment filter regularly samples the modification time of one of the driver files it knows to exist to see whether that timestamp has changed. When a discrepancy is discovered, it will close and re-open the anti-virus engine to begin making use of the new virus driver files.

FTP_AV.SH

```
#!/bin/ksh
#
# Adnan Akbas , Turkcell , 13.Ø5.2ØØ2
#
# The script transfers and installs virus driver files to the Mail
# Transfer Agents. Works from Crontab every hour and makes FTP
# and update, if a new package is available on the vendor's FTP
# server.

# Variables

av_srv=ftp.nai.com
av_user=anonymous
av_pwd=venus@turkcell.com
av_dir=/pub/antivirus/datfiles/4.x
local_dir=/var/db/mime-filter/ftp
logfile=${local_dir}/log/av_'date +%H.%M'.log

# Check if a ftp process is hanged before. If yes kill!

pid1='ps -ef | grep "ftp_av.sh" | grep -v grep | grep -v "$$" | grep -v
"view ftp_av.sh" | awk '{print $2}''
pid2='ps -ef | grep "ftp.nai.com" | grep -v grep | awk '{print $2}''

if [[ $pid1 = +([Ø-9]) ]] && [[ $pid1 > 1 ]] ; then
  kill $pid1 > /dev/null 2>&1
fi

if [[ $pid2 = +([Ø-9]) ]] && [[ $pid2 > 1 ]] ; then
  kill $pid2 > /dev/null 2>&1
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
fi

# Delete old files

rm $local_dir/dat-*.tar | tee -a $logfile 2>&1
rm $local_dir/update.ini  | tee -a $logfile 2>&1

# Check if we are in the right directory in local server

cd ${local_dir} > $logfile
if [ $(pwd) != ${local_dir} ]
then
   echo "ERROR: Cannot change to ${local_dir} ... FILE NOT RECEIVED" |
tee $logfile
   exit 8
fi

# Start FTP Job for the info-file. This info-file contains the version
# number of the latest virus driver files.

print  | tee -a $logfile
echo "$(date)" | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile
echo "STARTING FTP (GET) ..... " | tee -a $logfile
print | tee -a $logfile

ftp -v -n ${av_srv} << ! | tee -a $logfile
user $av_user $av_pwd
prompt
bin
cd ${av_dir}
hash
get update.ini
bye
!

print  | tee -a $logfile

# Check if successfully connected to ftp.nai.com

cat $logfile | grep "Connected to ftp.nai.com" > /dev/null 2>&1
if [ $? != 0 ]
then
     echo "ERROR: Cannot connect to ${av_srv} ... FILE NOT RECEIVED !!!"
| tee -a $logfile
     exit 9
fi
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
# Checking if the user could logged in.

cat $logfile | grep "Login failed"  > /dev/null 2>&1
if [ $? -eq 0 ]
then
     echo "ERROR: Login failed, check user and password ... FILE NOT
RECEIVED !!!" | tee -a $logfile
     exit 10
fi

# Checking if the target directory and file exists.

cat $logfile | grep "does not exist"  > /dev/null 2>&1
if [ $? -eq 0 ]
then
     echo "ERROR: Target file does not exist. ... FILE NOT RECEIVED !!!"
| tee -a $logfile
     exit 11
fi

# Checking if ftp is successful

cat $logfile | grep "bytes received in"
if [ $? -eq 0 ]
then
    echo "INFO:  update.ini file RECEIVED!" | tee -a $logfile
else
    echo "ERROR: update.ini file NOT RECEIVED!" | tee -a $logfile
    exit 14
fi

# Getting the version number in the info-file that is downloaded.

let dat_ver_nai='cat update.ini | grep DATVersion | sort | uniq | awk -
F= '{print $2}''

# Getting the version number of the current virus driver installed.

let dat_ver_local='/usr/local/sendmail/smmfilter-2.4/bin/avupdate -c |
awk '{print $7}''

# Comparing the version numbers to determine if a newer version exists
# to be installed.

if [[ ${dat_ver_nai} -eq ${dat_ver_local} ]]
then
   print | tee -a $logfile
   echo "Local dat-file version and remote dat-file versions are the
same - ${dat_ver_nai}" | tee -a $logfile
   print | tee -a $logfile
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
   touch /home/libk/rzxoper/log/check_'hostname'.log
   exit Ø
else

print | tee -a $logfile
echo "AVUPDATE" | tee -a $logfile
print | tee -a $logfile

# Here seen that virus driver is old therefore FTP the new package.

ftp -v -n ${av_srv} << ! | tee -a $logfile
user $av_user $av_pwd
prompt
bin
cd ${av_dir}
hash
get dat-${dat_ver_nai}.tar
bye
!

# Checking if ftp is successful

nfiles='cat $logfile | grep "bytes received in" | wc -l'
if [ $nfiles -eq 2 ]
then
     echo "INFO:  dat-${dat_ver_nai}.tar FILE RECEIVED!" | tee -a
$logfile
     cp -p ${local_dir}/dat-*.tar /pub/antivirus/datfiles/4.x >>
$logfile 2>&1
     cp -p ${local_dir}/update.ini /pub/antivirus/datfiles/4.x >>
$logfile 2>&1
else
     echo "ERROR: Unknown failure ...dat-${dat_ver_nai}.tar  FILE NOT
RECEIVED !!!" | tee -a $logfile
     exit 12
fi

rm -r /var/db/mime-filter/new > /dev/null 2>&1

# Performing the virus driver update locally.

/usr/local/sendmail/smmfilter-2.4/bin/avupdate -s $(hostname) -U smadmin
-P sendmail -u ${local_dir}  -vvv >> $logfile 2>&1

if [ $? -eq Ø ]
then
     print | tee -a $logfile
     echo "INFO: AVUPDATE successfully executed ..." | tee -a $logfile
     /usr/bin/mailx -s "new av paket is received! -  ${dat_ver_nai}"
sendmailadmins@turkcell.com <  $logfile > /dev/null 2>&1
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
else
     print | tee -a $logfile
     echo "ERROR: AVUPDATE failure !!!" | tee -a $logfile
     exit 13
fi

rm /pub/antivirus/datfiles/4.x/dat-*.tar >> $logfile 2>&1
rm /pub/antivirus/datfiles/4.x/update.ini >> $logfile 2>&1

fi

print | tee -a $logfile
echo "FINISHED: $(date)" | tee -a $logfile
print | tee -a $logfile
"ERROR: update.ini file NOT RECEIVED!"
```

The exit codes from the script are:

- Exit 8: “ERROR: Cannot change to local directory ... FILE NOT RECEIVED”.
- Exit 9: “ERROR: Cannot connect to FTP Server ... FILE NOT RECEIVED !!!”.
- Exit 10:“ERROR: Login failed, check user and password ... FILE NOT RECEIVED !!!”.
- Exit 11:“ERROR: Target file does not exist. ... FILE NOT RECEIVED !!!”.
- Exit 12:“ERROR: Unknown failure ...dat-????.tar FILE NOT RECEIVED !!!”.
- Exit 13:“ERROR: AVUPDATE failure !!!”.
- Exit 14:“ERROR: update.ini file NOT RECEIVED!” .

Adnan Akbas
System Administrator
TURKCELL (Germany)
© Xephon 2003

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

Brocade FC network port investigation

When using storage area networks, tools are necessary to manage the interconnected environment. Although there are powerful tools coming from several Fibre Channel (FC) switch vendors and others, there is only basic integration for doing this on the AIX command line. Therefore this document shows a way to get the information directly from a Brocade FC network to AIX. A script is given to list all the devices that are connected to active ports of the network.

FC NETWORKS

Brocade is one of the IBM-supported manufacturers that builds FC switch devices for Storage Area Networks (SAN). Since IBM is an OEM vendor of Brocade, the FC switches can wear an IBM logo. There are also some other well-known OEM vendors of Brocade. All SAN switches can be recognized reliably by their operating system, which, in this case, is Brocade's Fabric OS. The following way of checking this is a first glimpse of what we can do to find out much more. The following AIX command line asks the switch, via SNMP and through the network, about its ISO object identification:

```
snmpinfo -c public -h switch 1.3.6.1.2.1.1.2.0
```

It starts with 1.3.6.1.4.1.1588 if a Fabric OS responds.

For the sake of simplicity we assume there is an existing and working SAN. If several switches are interconnected with Inter-Switch Links (ISLs), you call it a network (a 'fabric' in Brocade's parlance). The only prerequisite for a working network that I want to name here is that every switch must have a domain identification (DID) that is unique within one network.

Devices in arbitrated loops are identified by a three-byte code that consists of a DID, the port number, and the loop device number. This means they are identified mainly by the port they are connected to.

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

Since networks allow several hundreds or thousands of ports, the identification of the devices is done not on a physical but on a logical basis. Each device that allows point-to-point connections, also known as network logins, has a unique 8-byte address that is called its World-Wide Port Name (WWPN). Since it contains some vendor information, it is often compared to the MAC addresses in LANs.

Assigning disks or grouping devices together (called zoning) is mostly done using those WWPNs. They simplify the cabling since the exact port does not matter as long as it is within the network. On the other hand, finding the port where a fibre channel is connected is getting harder – especially when using several patch panels or several connections within one trunk – the fibre cables can easily be confused. It is usually not easy to double-check the accuracy.

During my work with SANs, I sometimes missed the ability to verify, using a simple command, where the local machines's FC adapters are connected. The script to list the devices connected to the ports of a network was what I needed. I added an interpretation option because I was tired of comparing FCs and WWPNs manually.

```
#!/usr/bin/sh
#
#        Name:  lssw  -  list switch port connections
#
# Description:  list ports of a brocade FC switch network connections
#               - Domain ID of the Switch where the port is
#               - Number of the port where a device is connected to
#               - the WWPN of the adapter or device that is seen
#               - the symbol offered by the connected device
#
#     Command:  lssw [-r] <brocade-switch-IP-address> [<community>]
#                    [-r]            enable WWPN interpretation
#                   <brocade-switch-IP-address>
#                                    an IP-address or name of a switch
#                                    within a brocade network
#                   [<community>]   community string if NOT "public"
#
#       Hints:  - If the symbolic name is printed in hex digits
#                 please add at least the following two lines of
#                 information to the file:/etc/mib.defs .  Please
#                 start in the very first coulmn of that file:
```

```
#
#                swNsLocalEntry           enterprises.1588.2.1.1.1.7.2.1
#        swNsPortSymb swNsLocalEntry.5 DisplayString read-only mandatory
#
#                - All switches within a network
#                  need to have the same community
#                  otherwise this command will fail
#
#-----------------------------------------------------------------------
#
# check correct usage
#
if [ $# -eq 0 ]
then
    echo "Usage:$0 [-r] <brocade-switch-IP-address> [<community>]"
    exit
fi
#
# check for symbol interpretation option
#
interpretation=0
if [ "x$1" = "x-r" ]
then
    interpretation=1
    shift
fi
#
# take whatever the switch is named
#
SWITCH=$1
#
# check for community string or use default
#
COMMUNITY=public
if [ $# -ge 2 ]
then
    COMMUNITY="$2"
fi
#
# routine to extract the port number
#
porttranslated()

    m=$(echo $1|sed -e 's:^..*::')
    m=$(expr $m - 1)
    n=$(echo $1|sed -e 's:^...*::' -e 's:a:10:' -e 's:b:11:' -e
's:c:12:' -e 's:d:13:' -e 's:e:14:' -e 's:f:15:')
    n=$(expr "$m" '*' "16" "+" "$n")
    if [ $n -lt 10 ]
    then
        echo " "$n
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
    else
        echo $n
    fi

#
# routine to do the symbol translation
#
fcstype()

    #
    # assign the given WWPN in uppercase hex letters
    #
    wwpn=$(echo $1|sed -e 's:[ "]::g'| tr 'a-f' 'A-F')
    #
    # try to find a local interface with the given WWPN
    #
    for fcs in $(lscfg|grep fcs)
    do
        if [ "$wwpn" = "$(lscfg -vl $fcs | sed -ne '/Network Address/s/
^.*................/:::::::/p' )"  ]
        then
            #
            # successful, print a formatted string
            # stop further interpretation
            #
            (echo $(hostname)"" && lscfg -l $fcs ) |\line            awk
'printf" IBM %12s %4s  %5s",$1,$2,$3'
            return
        fi
    done
    #
    # the first digits of a wwpn name the manufacturer
    # show at least this information
    #
    case  $wwpn in
    10:00:00:00:C9:2E:E3:7E) echo " IBM     m80_prod fcs1  P2-I6" ;;
    10:00:00*)  echo " EMULEX  Adapter        @???" ;;
    21:00:00*)  echo " QLOGIC  Adapter        @???" ;;
    50:05:00*)  echo " IBM     2105??0        .???" ;;
    esac
    #
    # please extend this list if necessary
    #
    return

#
#-------------------------------------------------------------------------
#               #    #    #     ###   #    #
#               ##  ##   # #     #    ##   #
#               # ## #  #   #    #    # #  #
#               #    # #######   #    #  # #
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
#               #    # #     #   #    #   ##
#               #    # #     #  ###   #    #
#------------------------------------------------------------------------
#
# print Headline
#
echo "DID Port  connected to WWPName       Symbol of connected system"
#
# ask for the switches that belong to the network
# and loop the actions over all of them
#
for SWITCH in $(snmpinfo -v -m dump -h $SWITCH
1.3.6.1.4.1.1588.2.1.1.1.2.10.1.4 | cut -d' ' -f3)
do
    #
    # do query switches only if the switch is reachable
    #
    ping -c 1 $SWITCH >/dev/null 2>&1
    if [ $? -eq 0 ]
    then
        #
        # get all FC name server entries from that switch
        #
        NSCOUNT=$(snmpinfo -v -h $SWITCH -c $COMMUNITY
1.3.6.1.4.1.1588.2.1.1.1.7.1.0 | cut -d= -f2)
        n=0
        while [ $n -lt $NSCOUNT ]
        do
            n=$(expr $n + 1)
            #
            # collect the relevant information
            #
            ALPA=$(snmpinfo -v -h $SWITCH -c $COMMUNITY
1.3.6.1.4.1.1588.2.1.1.1.7.2.1.2.$n | cut -d= -f2)
            TYPE=$(snmpinfo -v -h $SWITCH -c $COMMUNITY
1.3.6.1.4.1.1588.2.1.1.1.7.2.1.3.$n | cut -d= -f2)
            NAME=$(snmpinfo -v -h $SWITCH -c $COMMUNITY
1.3.6.1.4.1.1588.2.1.1.1.7.2.1.4.$n | cut -d= -f2)
            SYMB=$(snmpinfo -v -h $SWITCH -c $COMMUNITY
1.3.6.1.4.1.1588.2.1.1.1.7.2.1.5.$n | cut -d= -f2)
            COS=$(snmpinfo -v -h $SWITCH -c $COMMUNITY
1.3.6.1.4.1.1588.2.1.1.1.7.2.1.10.$n | cut -d= -f2)
            #
            # analyse the information found
            #
            DID=$(echo $ALPA | cut -d: -f2)
            PORT=$(echo $ALPA | cut -d: -f3)
            PORT=$(porttranslated $PORT)
            FCAL=$(echo $ALPA | cut -d: -f4)
            #
            # symbol interpretation if needed and enabled
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
            #
            if [ "$SYMB" = " "  -a $interpretation = 1 ]
            then
                SYMB="$(fcstype $NAME)"
            fi
            #
            # print collected information
            #
            echo "$DID   $PORT  $NAME  $SYMB"
        done
    fi
done
#
# all done well ? good bye
#
exit Ø
#------------------------------------------------------------------------
```

ABOUT SNMP

Communication is needed to investigate another device. There are two means of transport to reach devices within a SAN from one of its nodes. The most obvious is using the Fibre Channel links, which are used for data transfer between CPUs and disks or tapes. An Ethernet connection to the management interface of a SAN switch within the network concerned is not always present; but the Ethernet-based management interface offers the most interesting information. The simplest way to access it is by using the Simple Network Management Protocol (SNMP).

AIX provides the command **/usr/sbin/snmpinfo** to access other networking devices via SNMP. This command is part of the installation if the *bos.net.tcp.server* file set is present. Its definition file, */etc/mib.defs,* comes with the file set *bos.net.tcp.client*. The full Management Information Base (MIB) for resolving Brocade switch MIB trees is available from the Brocade Web server after being registered.

It is not necessary to download and install the MIBs for Brocade to use the network investigation script, but, for easier readability of the command output, I recommend adding the following two lines to the end of the file */etc/mib.defs*:

```
swNsLocalEntry   enterprises.1588.2.1.1.1.7.2.1
```

```
swNsPortSymb      swNsLocalEntry.5   DisplayString   read-only   mandatory
```

I do not recommend downloading the Brocade MIBs because they are written in ASN.1, while AIX uses a proprietary notation. (ASN.1 is the standard notation that most network management stations use – see *Bibliography*.)

The Brocade switches usually have SNMP enabled by default, with all MIBs being accessible read-only. You can easily change the settings on a Brocade system from the switch's command line using **agtcfgSet**, which will interactively query all parameters. The command **agtcfgshow** lists the settings. It is not necessary to enter any trap recipient or read-write access.

The port investigation script uses MIBs that are specific to Brocade and are not shared with other vendors. Therefore, it is likely to fail with other SAN switches.

INVESTIGATING PORTS

When calling the network investigation script, it is important to know one accessible network address of one SAN switch from the network you are interested in. First the script retrieves the other switches' addresses and later it will try to get their information as well. The output below shows a single switch, which you can easily determine by the DID (which is always 01). The port number 1 is unused or the device connected to it is switched off. Therefore this port is not listed here. The last column shows symbolic names, which are mostly empty. Usually only the storage targets answer with symbolic names.

```
# lssw.sh  10.20.10.24
DID Port  connected to WWPName      Symbol of connected system
01   0    50:05:07:63:00:cc:9b:f0  "IBM     2105800          .136"
01   2    10:00:00:00:c9:2e:c7:7c  ""
01   3    21:00:00:e0:8b:03:06:bd  ""
01   4    10:00:00:00:c9:2e:e3:7e  ""
01   5    21:00:00:e0:8b:03:dd:92  ""
01   6    50:05:07:63:00:c8:9b:f0  "IBM     2105800          .136"
01   7    10:00:00:00:c9:23:33:85  ""
```

Mostly I am interested less in knowing the symbolic names (where they exist) than in the devices that are behind them. So

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

the interpretation function compares all local FC interfaces with the WWPNs that are connected to the ports and replaces an empty symbol string with one that shows system name, the adapter name, and its slot. The slot helps particularly in LPAR environments to avoid confusion.

There is also a section within the script that allows you to add entries in a case list. This can be used to identify manufacturers (as here, but not too accurately!) or to translate the WWPNs to definitions that contain whatever you want. You simply need to extend the list with the most exact entries at the beginning as shown with the entry that converts the symbol of the WWPN at port 4. Calling the script with this interpretation function leads to more verbose output:

```
# ./lssw.sh -r 10.20.10.24
DID Port  connected to WWPName       Symbol of connected system
01    0   50:05:07:63:00:cc:9b:f0   "IBM     2105800           .136"
01    2   10:00:00:00:c9:2e:c7:7c   "EMULEX  Adapter           @???"
01    3   21:00:00:e0:8b:03:06:bd   "QLOGIC  Adapter           @???"
01    4   10:00:00:00:c9:2e:e3:7e   "IBM     m80_test fcs1  P2-I6"
01    5   21:00:00:e0:8b:03:dd:92   "QLOGIC  Adapter           @???"
01    6   50:05:07:63:00:c8:9b:f0   "IBM     2105800           .136"
01    7   10:00:00:00:c9:23:33:85   "IBM     f50_test fcs0  P2-I1"
```

Be careful when using the interpretation function, since wrong modifications of the script or later changes to adapters may cause mis-interpretations. These are most likely in dynamically-changing LPAR environments.

When investigating whole networks there are not many differences. Obviously you have to select the switch using the DID before you concentrate on the port. Reading the remaining output does not change. There is another helpful feature of the Brocade switches. If you do not have network access via a LAN to all FC switches, you can route IP traffic via the SAN for the SNMP queries. But you still need network access to at least one SAN switch of each network, because AIX does not support IP within the SAN at the moment.

CONCLUSION

The script given here can be used as is to retrieve from Brocade

FC networks the WWPNs of all active connected ports. Using its interpretation option, the local FC adapters are resolved with name and location code in the listing.

The SNMP strings and the composed commands given in this script can be used outside the script for quicker retrieval. The installation of MIBs is not needed, but, for a readable character representation, at least a two-line extension to the file */etc/mib.defs* is given and recommended.

The script does not provide zoning information, but does allow an unrestricted view to the ports of the network.

REFERENCES

The Fibre Channel Management Integration MIB.

Subsystem Device Driver (SDD for AIX).

Brocade Web server.

Andreas Neuper
PROFI Engineering Systems (Germany)

© Dr Andreas Neuper 2003

The awk command

awk is one of the most important commands you should master in shell programming since it can perform many of the functions available in other commands, and often in a more efficient manner since a single command performing multiple functions is usually faster than multiple commands each performing a single function. Learning the rules, construction, and syntax used in **awk** is no easy task, however, and it will probably require extensive practice before you are totally familiar with its capabilities.

This article will introduce you to the standard **awk** functions, and a future article will introduce more complex **awk** operations.

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

CAPABILITIES OF AWK

Amongst other things, **awk** is a text selection and alteration tool that has its own powerful pattern matching and programming language with which to specify a variety of complex text processing operations. The syntax is similar in many ways to C, and it uses variables and programming constructs such as testing and looping.

If you have a file containing data held in the form of text strings, you can use **awk** to process the data to change the layout and perform comparisons and calculations at the same time. **awk** can also be used to filter out unwanted lines in a file, or to check that a file conforms to a particular layout specification.

No changes are made to the original file on which **awk** operates and by default its output is sent to standard output. As a text manipulator, it can do everything that **sed** can do, and more, but you may find that the construction of the command is more complex than that of **sed**, not in its syntax (of which **sed** is the undoubted master), but in its logic, and will usually require a greater number of lines of **awk** coding.

You may ask **awk** to extract, manipulate, and perform computations with information from a file. The structure of the information in the file must be consistent, and for lines that are similar, each piece of information must be in the same position in each line, or record, of the file. The **awk** program treats each line of a file as a set of fields, normally separated by spaces or tabs, although you can specify any separator you wish.

CONSTRUCTION OF AWK COMMANDS

At its simplest, **awk** selects a line from a file according to selection criteria, which can be text patterns as in **grep** or **sed**. When **awk** operates on a file, the selection and action process can be represented by:

```
awk pattern {action} file(s)
```

This means that every line which matches the specified pattern

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

should have the action performed on it. You can run the same pattern and action combination on multiple files at the same time; in this case **awk** concatenates the output of all files and treats it as a single file input.

Both the pattern and action are optional, and a pattern with no corresponding action simply selects the matched record for display on standard output. An action with no associated pattern is performed on all records in the file. In other words, a missing pattern matches all lines in the file.

The search pattern can be a simple text string or it can be a regular expression as used in **sed**, although there are additional operators available in **awk** that cannot be used in **sed**.

As an example of a simple search pattern, we can find all the lines containing **Smith** in the **stars** file by using:

```
awk /Smith/ stars
```

The default action is to print only the selected lines. If you don't specify any pattern, **awk** will perform the specified action on all lines in the file, and in this case you must specify an action, even if all you want to do is print the lines. For example:

```
awk {print} stars
```

The actual action, or program (**{print}**), to be used must be the first argument to the command, and the remaining arguments are the files to be processed. We can use both a pattern and action together with:

```
awk /Smith/{print} stars
```

although in this particular case the result would be the same if we omitted the **{print}**. In the syntax used above there must be no spaces between the pattern and action, nor anywhere else in this combination.

As **awk**'s capabilities are expanded, the programs will become more complex and many will contain metacharacters. These must be enclosed in quotes to make sure that **awk** sees them, and it is good practice to use single quotes since in many cases

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

double quotes will not do. If both a pattern and action are included as arguments to **awk** and either contains metacharacters, you must quote both within a single set of quotes. You will also discover that it is particularly difficult, if not almost impossible, to quote single quotes within the **awk** script.

There is also an option to put **awk** subcommands in a file and then use this file as a command line option.

REFERENCING FIELDS

One of the things that **awk** can do is operate on individual fields within a line. Every line or record in a file is composed of fields, which are separated by field separators. These are normally spaces or tabs but can be changed to whatever you like by using the **FS** variable, or the **-F** option to **awk**.

Fields are referenced by the notation **$n**, where **n** is the number of the field you want. **$1** is the first field on the line, **$2** the second, and so on, and **$0** refers to the whole line. The maximum number of fields you can have in a line is **99**.

The default value of **FS** is blank, so that by default fields are separated by one or more spaces or tabs. If **FS** contains a single non-blank character, then fields are separated by that single character. If **FS** contains more than one character, then fields are separated by patterns matching this regular expression.

Suppose we want to find all the **Smiths** in the **stars** file, but we want to see only the first name and the contestant number. We can do this by using:

```
awk '/Smith/ {print $1 " - " $3}' stars
```

Note that we had to specifically insert space characters (and also in this case the minus character) in the output by enclosing them in quoted strings in the **print** statement. If you used a **print** statement such as:

```
print $1,$3
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

then the fields would appear on the output with a single space between them.

By using this method of referencing fields on a line it is possible to rearrange your output into a different order. For example, if we wanted to reorder the names in the **stars** file by putting the last name first, followed by the first name, this could be achieved with:

```
awk '{print $2 ", " $1 "^I" $3}' stars
```

which has an action but no pattern. In this case the missing pattern selects every record in the file.

USING AWK COMMANDS IN A FILE

If in the above example we wished to perform this exercise on a number of files, this could best be achieved by putting the **awk** program into a file and using the **-f** option.

Suppose we create a file called **swap**, which contains the **awk** program:

```
{print $2 ", " $1 "^I" $3}
```

The **awk** program is now simply a line in a file. The single quotes surrounding the command are no longer necessary. This program can now be run on any file that needs rearranging:

```
awk -f swap popstars > popstars.new
awk -f swap tenstars > tenstars.new
awk -f swap footiestars > footiestars.new
```

BUILT-IN PATTERNS

There are two special built-in patterns within **awk**, called **BEGIN** and **END**. If **BEGIN** appears as a pattern, it matches the beginning of the file, so that you can execute a number of subcommands before processing starts on the lines of input. Similarly, the pattern **END** matches the end of the file, and you can then execute further commands after processing has been completed on all the lines of input.

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

You could use these to put a header and a trailer on a report you are producing. As a simple example, let's produce a list of our competitors, with the first and last names swapped. First we construct a source file, called **report**, containing the following **awk** commands (the **^I**s represent tabs):

```
BEGIN {
     print
     print "Competitor ^I^INumber"
     print
}

{ print $2, $1 "^I^I" $3}

END {
     print
     print "Number of competitors = " NR }
```

awk is very flexible about syntax. Braces can be located either within the **BEGIN** or **END** sections or in the body of the program. They can be on lines of their own, or both on the same line with a command in between, or the starting brace can be on the same line as the **BEGIN** or **END**. There is also no requirement for spaces on either side of the braces. You can, if you wish, place a semi-colon at the end of each line, as if you were programming in C. There is not much point in doing this normally unless you wanted to include two or more commands on the same line.

In the **BEGIN** section, all the actions between the opening **{** and the closing **}** are performed before the data file is processed. After the **BEGIN** section follows the body of the program, which in this case consists of only the single **print** statement; this statement operates on all lines since no pattern is specified. The last section is the **END** section, which will be executed after all the lines have been processed.

This program can be run with, for example:

```
awk -f report popstars tenstars footiestars
```

In the last line of the program is a reference to the variable **NR**. This is a built-in **awk** variable that in this example contains the number of records processed. Because it is not quoted in our

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

print statement, it is understood by **awk** to be a variable and so its value is substituted into the output.

FORMATTING PRINTED OUTPUT

In our example above, instead of using **print**, which has fairly limited formatting capabilities, we can use the **printf** subcommand, which gives much greater control over the final output. The formatting expressions are very similar to that for the **printf** command. By using **printf** we could shorten the **report** program to:

```
BEGIN {printf "\nCompetitor ^I^INumber\n\n"}

{ print $2, $1 "^I^I" $3}

END {printf "\nNumber of competitors = " NR "\n"}
```

printf does not automatically supply a newline, which must be explicitly specified as **\n**. Note that we must enclose the final **\n** on our last line in quotes since it comes after the **NR** variable, which is outside our quotes. Another commonly used escape sequence is **\t** for a tab.

Generally speaking, **printf** is not used just to shorten **awk** source files. By far its most important use is in formatting the output from shell scripts so that fields are aligned, thus making the output easier to read. This alignment can be achieved by using the full syntax of **printf**:

```
printf ("format expression", arg1 [,arg2, . .])
```

As we have seen, the parentheses and **format expression** followed by the comma are not essential. The format specifications are each preceded by a per cent sign, and the following characters are most commonly used to define the output:

- **c** - single ASCII character
- **d** - decimal integer
- **f** - floating point format
- **o** - octal number

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

- **s** - string of characters
- **x** - hexadecimal number
- **%** - the % character

Whenever you have a format specification, you must have a corresponding argument, which can be either a variable name, a number, or a quoted string. If an argument is missing, you will get an **awk** error message when you try to run your script. If there are more arguments than there are format specifications, the argument will be ignored. You can also specify the width of the field using the following format:

```
%[justifier]width[.precision]specifier
```

For example:

```
printf ("|%-10s|\t|%8.2fp per |%3d Kg\n", "Apples", 12.2, $3)
```

where **$3** is a variable whose value is **2**, will produce output:

```
|Apples    |    |      12.20p per |  2 Kg
```

The vertical bars have been introduced to give some idea where the fields start or finish. By default, if no justifier is specified, the text will be right-justified; a '**-**' justifier will left-justify. When using floating point format, the first number is the total width of the field, and the second integer specifies the number of decimal places; remember that the decimal point itself occupies one of the characters specified by the width.

Fields are automatically padded out with spaces if the number of characters is less than the width. If the number of characters is greater than the width specification, then all the characters will be printed out, which may spoil your formatting. The exception to this is when a number is in floating point format and only the specified number of decimal places are printed, even though the non-decimal part may contain more characters than the total width of the field.

FORMATTING AWK COMMANDS IN SCRIPTS

So far we have mainly discussed using **awk** with a source file

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

containing all the subcommands, but this may not always be desirable, and indeed some shell programmers do not like this approach, preferring instead to write their scripts as self-contained entities.

When you write scripts containing all the **awk** subcommands, you should be careful to format the script so that it is easy to read, remembering that at some future date someone other than yourself may have to read and understand your script, and, more importantly, make alterations to it. This will be much simpler if you have adopted a consistent formatting style.

Using source scripts does not require the commands to be surrounded by quotes, but when included in a shell script these must be quoted otherwise you will get syntax errors. The easiest to read formatting could look something like the following:

```
awk '
     BEGIN {
          command1
          command2 . . .
     }

     {
          command3
          command4 . . .
     }

     END {
          command5
          command6 . . .
     }
' filename
```

By indenting the subcommands in the **BEGIN**, **END**, and main sections, and by separating the sections with blank lines, it is much easier to see where the commands belong, and consequently make debugging easier. If you do not have a **BEGIN** and **END** section, you may also see **awk** commands written like:

```
cat filename |
awk '{
     command1
     command2 . . .
}'
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

In this example we have piped the output to **awk** and placed the command itself at the start of the next line. This type of format may be preferable if you have a long pipeline to produce input for **awk** so that it is then relatively simple to determine where the command starts and finishes.

USING VARIABLES

There are three different types of variable used within **awk**. These are:

- Built-in **awk** variables.
- User variables defined within **awk**.
- Variables passed in to **awk**.

Built-in variables

We have already come across one built-in variable, **NR**, which actually stands for the number of the current input line. We used **NR** as a count of the total number of lines, but this is not always accurate since it assumes that every line in the file contains information we want to process, which is not always the case. If we had had blank lines in our file, then using **NR** to represent the total number of competitors would not have been correct.

Another predefined variable is **NF**, which contains the number of fields on the current line or record. This variable could be used to check the syntax of a file using a conditional expression (which is discussed later). You can do this by adding a further field to any of the lines of the **stars** file and then running:

```
awk 'NF > 3 {print "Too many fields on line " NR}' stars
```

The **NF** variable is frequently used to print the last field on a line where you are uncertain of the number of fields since this may vary from line to line. Instead of using **$3**, say, you would instead use **$NF**, since this would extract the value associated with the last field on the line. For example:

```
awk '{print $NF}' stars
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

A variable which is also quite useful is the **FILENAME** variable, which is set equal to the name of the file currently being processed by **awk**. Suppose we wished to match patterns in a number of files, then this could be achieved with:

```
awk '/Smith/ {print FILENAME ":" $Ø}' *stars
```

You should be aware that when you are processing a number of files as above, then the **NR** variable gets incremented for each record that is read sequentially through all the files. It is not reset to 1 when the **FILENAME** variable changes. You could overcome this problem by using the variable **FNR**, which gives the current input line in the current file. You can see the difference by running the following:

```
awk '/Smith/ {print FILENAME ":" $Ø " : line " NR}' *stars
awk '/Smith/ {print FILENAME ":" $Ø " : line " FNR}' *stars
```

By default **awk** assumes that fields are separated by spaces or tabs. If the file you want to process has some other field separator, you should set the value of the **FS** variable to that separator. Alternatively you could use the **-F** option on the **awk** command line, but you must set the value in the **BEGIN** section before the first input line is read.

For example, to print the names of all users who have the Korn shell as the initial shell, enter:

```
awk -F: '/ksh/ {print $1}' /etc/passwd
```

or using a statement within the **awk** source file:

```
BEGIN { FS = ":" }
```

You may often see multiple **awk** commands with different **-F** options piped into each other in order to extract fields from input lines. For example, if we wanted to display the number of free megabytes in **rootvg**, we could use the pipeline:

```
lsvg rootvg | grep FREE | awk -F: '{print $3}' |
awk -F"(" '{print $2}' | tr -d ")"
```

Similarly, output produced by the **print** statement has a default field separator of one space. If you want a different separator, you should change the value of the output field separator, **OFS**.

 © 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

There are two other built-in variables similar to **FS** and **OFS**. These are **RS**, the input record separator, and **ORS**, the output record separator. By default they are both set to the new-line characters, so that records and lines equate to the same thing on input, and print statements write each record as a separate line of output.

If you wish to change any of these built-in variables, it is best to do so in the **BEGIN** section.

User-defined variables

You can define your own variables for use in **awk**; the name can consist of letters, digits, and underscores, but it cannot start with a digit. A variable is defined the first time it is mentioned, and **awk** initializes it to zero, or to a null string. You don't have to specify what type of variable it is since **awk** will figure it out from the context. You should be careful not to use variable names which may be the same as an **awk** subcommand. If in doubt, use upper-case variable names.

Suppose that the **stars** file contained scores in a competition instead of telephone numbers, and we wished to calculate the average score. To do this we could first create an **awk** program in a file called **average**:

```
{ total = total + $3 }
END { print "Average score is ", total / NR }
```

To compute the average with **awk**, we use:

```
awk -f average stars
```

This is a very simple program and because variables are automatically initialized we don't need a **BEGIN** section. The first line of the program adds the value of the third field of each record of the data file to the variable called **total**.

The second line contains a **print** statement in the **END** section, so that it is done after all lines in the file have been processed. First we print the message **Average score is**, and then on the same line we print the value of **total** divided by **NR** to show the average score.

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

In this program we have assumed that the input file has the correct layout: that there are no blank lines, and that each line does have a score in its third field. This is a very sloppy program and we should have checked for such things. We can ensure that our output is more accurate by using:

```
/^$/ { blank = blank + 1 }
{ total = total + $NF }
END { print "Average score is " , total / (NR - blank) }
```

The first line now allows us to check for completely blank lines (you could modify this to check for spaces and tabs also), and if one is encountered then we add **1** to the variable **blank**, which we finally subtract from **NR** to give an accurate count of the number of lines. We have used the **$NF** variable to extract the value of the last field since this will allow us to have contestants with more than a first and last name. It still doesn't check that this field contains valid numbers, but you will shortly see how this also can be achieved.

Variables passed to awk

It is sometimes necessary to pass variables from the main part of your script to the **awk** command. You can do this by using the **-v** option to **awk**, which must be specified before any **-f** argument. By using this option, you specify variables that will be available in the **BEGIN** section before any input line has been read, and also throughout the rest of the **awk** script.

As an example, let us assume that we want the ability to print the average scores for all contestants in our **stars** file with the same last name. First we produce our **average** script, which contains lines like the following:

```
{ if (match ($2,name) != 0) {
     total = total + $NF
     nameline = nameline + 1
     }
}
END {
     print "Average score is " , total / nameline
}
```

 © 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

For the time being, let us ignore the command construction of the first line and its corresponding syntax. What the first line does is make a string comparison between **$2** and the variable **name**, which we pass into the script from the command line. If there is a match, then we increment the **total** and **nameline** variables to allow us to produce our final average score; if there are no matches then the next line of input is processed. Although we have not done so here, we should really check that **nameline** does not have a value 0, which means we should exit without printing since there are no matches.

We call this script from the command line using:

```
awk -v name=Smith -f average stars
```

You can read in any number of variables to **awk** by using multiple **-v** options. If you are reading in a variable which is set to another variable's value within your shell script and which contains spaces or other metacharacters, you must quote this variable with, say, **-v name="$NAME"**.

MATCHING USING CONDITIONAL EXPRESSIONS

We can use conditional expressions within **awk** to match simple character strings, or to determine whether the values of numeric fields satisfy certain criteria. These expressions are really just shorthand notations for the **if** subcommand. As an example, consider:

```
awk '$NF >= 200' stars
```

This will print out the lines with the contestant numbers greater than or equal to **200**. Similarly:

```
awk '$NF >= 100 && $NF < 200' stars
```

or:

```
awk '($NF >= 100) && ($NF < 200)' stars
```

will print out the lines with contestant numbers greater than or equal to **100** and less than **200**. The double **&&** is the conditional operator and indicates that both conditions must be met for the

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

action to be taken, which in this case is the default action, ie print out the line. The second of the examples shows how you can increase the readability by enclosing your conditional expressions inside brackets – this will certainly be a requirement when your conditional expressions become complex constructions of **&&** and **||** operators.

You should be aware that when you want to check that something is equal to a particular value, you must use the double equals notation, otherwise prepare yourself for endless hours of fun debugging shell scripts where you use the single equals instead! For example:

```
awk '$NR == 5' stars
```

Now suppose that we wanted to print out the lines containing the name **Smith**, this can be achieved with:

```
awk '$2 ~ /Smith/' stars
```

The **~** means 'match', that is, test whether field 2 contains the pattern enclosed between the slashes; it does not necessarily have to match the whole field and the pattern **mith** would achieve the same result. You can exclude lines containing this pattern with **!~**. For example, you can print out all lines which do not have a contestant number starting with 3 by using:

```
awk '$NF !~ /3../' stars
```

We could add further conditions by requesting that only the **Smith**s with contestant numbers less than **300** be printed:

```
awk '$2 ~ /Smith/ && $NF < 300' stars
```

If two patterns are separated by a comma, the action will be performed on all lines between an occurrence of the first pattern and the next occurrence of the second pattern. For example:

```
awk '$2 ~ /Becks/, /Seles/' stars
```

will print out all the lines from **Becks** to **Seles** inclusive.

If there is more than one occurrence of the first pattern, then all the lines from the first occurrence of the first pattern to the first occurrence of the second pattern, inclusive, will be printed, and

 © 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

then all lines from the next occurrence of the first pattern to the next occurrence of the second pattern will be also be printed; and so on. If there are no further occurrences of the second pattern, then all the lines from the last occurrence of the first pattern to the end of the file will be printed.

Tonto Kowalski
Guru (UAE) © Xephon 2003

Manage FTP process

Manage FTP process (mfp) is a utility that automates the whole process of performing FTP. Some organizations do a great deal of file transfer via ftp throughout the day at various times and without any automated processing, which can be quite time consuming and prone to errors. This utility allows configuration records to be defined for each FTP task and, once it is defined, the daemon processing the mfp utility will continue FTPing at regular intervals. The utility provides a lot of flexibility in managing FTP configuration records (ie commenting out existing entries, un-commenting commented out entries, logically deleting entries, purging all logically-deleted entries, etc).

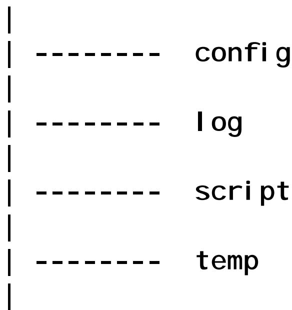
Any changes made to configuration files must be followed up by executing the appropriate option in the menu to reload the configuration file, otherwise the changes in the configuration file will not take effect. The biggest drawback with this utility is that it performs FTP in serial mode, that's to say that it reads an FTP configuration record, does the FTP, and then moves on to the second configuration record, etc.

INSTALLATION

mfp.sh requires the following directory structure:

```
$MFP_ROOT_DIR
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.



Define $MFP_ROOT_DIR in the script *mfp.sh* and create these sub-directories before attempting to run *mfp.sh*.

If the script is run without this directory structure in place, it will display the appropriate error message.

SECURITY

mfp.ksh can be run only by $AUTHORISED_USER, which is currently set to root. Because the configuration file, log file, and temporary files may contain passwords, the umask for all file creation is set to 077, which gives read and write access to $AUTHORISED_USER only.

TUNABLE PARAMETERS

Tunable parameters are:

1 $FTP_INTERVAL in *pfari.ksh* – this controls the duration of sleep between each pass of configuration file. It has been set to 900 seconds (15 minutes). This can be adjusted from *mfp.ksh* during runtime.

2 $WAIT_ON_CONFIG_INFO in *pfari.ksh* – this controls the duration of sleep while looking to establish any available configuration information. This can be adjusted in *pfari.ksh*.

The debug option can be switched on and off during runtime from *mfp.ksh*.

Once *pfari.ksh* is started, any changes to the configuration file will take effect only if the configuration files are reloaded after they have been changed.

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

Do not hand-edit the configuration file because some intricate logic is applied via *mfp.ksh* to maintain this file.

MFP.KSH

```
#!   /usr/bin/ksh
########################################################################
#   Name     : mfp.ksh (manage FTP process)
#   Overview : The shell script provides a menu-based interface to
#          manage all aspects of a process which FTPs files from
#          a host server to one or many nominated servers at an
#          adjustable regular interval.
#   Notes    : 1.  The  script  contains  following   functions:
#                   o InitialiseVariables
#                   o InitialiseLogFile
#                   o DisplayMessage
#                   o LogMessage
#                   o HandleInterrupt
#                   o MoveCursor
#                   o ProcessExit
#                   o DisplayMenu
#                   o ProcessMenuOption
#                   o InstanceCheck
#                   o GetRemoteHostName
#                   o GetRemoteDirectoryName
#                   o GetRemoteFileName
#                   o GetRemoteUserId
#                   o GetRemotePassword
#                   o GetLocalDirectoryName
#                   o GetLocalFileName
#                   o GetAction
#                   o GetComment
#                   o IsFTPDaemonRunning
#                   o PerformSanityCheck
#                   o AddEntryToConfigFile
#                   o ModifyConfigFileEntry
#                   o ReplaceEntryInConfigFile
#                   o StartFTPDaemon
#                   o KillFTPDaemon
#                   o StopFTPDaemon
#                   o SwitchOffDebug
#                   o SwitchOnDebug
#                   o AdjustFtpInterval
#                   o ReloadConfigurationFile
#                   o PurgeDeletedEntriesFromConfigFile
#                   o ViewLogFile
#                   o ViewConfigFile
#                   o TrimLogFile
#                   o ShowRemoteServerStatus
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
#                    o main
#              3. Host server is defined as the server on which this
#                 script is running.
#              4. The script starts a daemon process called pfari.ksh
#                 to perform the FTP at a regular interval.
#        5. The following directory structure is required by the script:
#                          $MFP_ROOT_DIR
#                               |
#                               |------ log
#                               |
#                               |------ script
#                               |
#                               |------ config
#                               |
#                               |------ temp
#                               |
#                               |
########################################################################
#  Name     : InitialiseVariables
#  Overview : The function initializes all module constants and
#             working variables.
#  Notes    :
########################################################################
InitialiseVariables  ()
{
AUTHORISED_USER="root"
USER='id  | cut -d'(' -f2 | cut -d')' -f1'
#  define root  directory
MFP_ROOT_DIR="${HOME}/ftp"        ; export MFP_ROOT_DIR
MFP_LOG_DIR="${MFP_ROOT_DIR}/log" ; export MFP_LOG_DIR
MFP_LOG_FILE="${MFP_LOG_DIR}/mfp.log" ; export MFP_LOG_FILE
# temporary directory
MFP_TEMP_DIR="${MFP_ROOT_DIR}/temp" ; export MFP_TEMP_DIR
MFP_FTP_INTERVAL_FILE="${MFP_TEMP_DIR}/ftp_interval.tmp" ; export
MFP_FTP_INTERVAL_FILE
TEMP_FILE_1="${MFP_TEMP_DIR}/mfp_1.tmp"
TEMP_FILE_2="${MFP_TEMP_DIR}/mfp_2.tmp"
# configuration file
MFP_CONFIG_DIR="${MFP_ROOT_DIR}/config" ; export MFP_CONFIG_DIR
MFP_CONFIG_FILE="${MFP_CONFIG_DIR}/pfari.config" ; export
MFP_CONFIG_FILE
# define script directory
MFP_SCRIPT_DIR="${MFP_ROOT_DIR}/script" ; export MFP_SCRIPT_DIR
FTP_INTERVAL=4 ; export FTP_INTERVAL
# terminal capabilities
BOLDON='tput smso'  ; export BOLDON
BOLDOFF='tput rmso' ; export BOLDOFF
ESC="\0033["
# menu title
MENU_TITLE="${BOLDON}Maintain FTP Process${BOLDOFF}"
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
ERROR="mfp.ksh:ERROR:"
INFO="mfp.ksh:INFO:"
# fuction return values
TRUE=0
FALSE=1
DEBUG_FLAG="${FALSE}"  ; export DEBUG_FLAG
# exit status
SEC=0
FEC=1
# sleep duration
SLEEP_DURATION=3
# an integer variable for numeric test
integer DUMMY_INT=0 ; export DUMMY_INT
DATETIME='date "+%d/%m/%Y at %H:%M:%S"'
# messages
INTERRUPT="Program Interrupted\; Quitting early"
WORKING="Working"
INVALID_ENTRY="Invalid Entry"
OS_ERROR="\${ERR_MSG}"
FTP_DAEMON_ALREADY_RUNNING="Daemon script \${MFP_SCRIPT_DIR}/pfari.ksh
is already running"
FTP_DAEMON_NOT_RUNNING="Daemon script \${MFP_SCRIPT_DIR}/pfari.ksh is
not running"
FTP_DAEMON_KILLED="Successfully killed executing daemon script
\${MFP_SCRIPT_DIR}/pfari.ksh"
FTP_DAEMON_STARTED="Successfully started daemon script
\${MFP_SCRIPT_DIR}/pfari.ksh"
FTP_DAEMON_NOT_STARTED="Failed to start daemon script
\${MFP_SCRIPT_DIR}/pfari.ksh"
VARIABLE_NOT_SET="Variable \${VAR} is not defined"
DIR_NOT_EXIST="Directory  \${DIR} does not exist"
SCRIPT_NOT_FOUND="Script \${SCRIPT} not found"
NO_CONFIG_FILE="\${MFP_CONFIG_FILE} does not exist"
ENTRY_EXISTS="Entry already exists in configuration file"
CONFIG_FILE_NOT_EXISTS="Configuration file \${MFP_CONFIG_FILE} does not
exist"
NO_CONFIG_FILE_ENTRIES="Configuration file \${MFP_CONFIG_FILE} does not
have any entries"
INVALID_ACTION_PARAM="\${P_ACTION}, is an invalid action for
ModifyConfigFileEntry\(\)"
NO_ENTRY_TO_REMOVE="No configuration file entries found for removal"
NO_ENTRY_TO_COMMENT_OUT="No configuration file entries found for
commenting out"
NO_ENTRY_TO_UNCOMMENT="No configuration file entries found for
uncommenting"
NO_ENTRY_TO_REPLACE="No configuration file entries found for replacing"
NO_CHANGE="No change made"
NO_LOG_FILE="Log file, \${MFP_LOG_FILE}  does not exist"
INSTANCE_RUNNING="An instance of mfp.ksh is already running"
NO_DELETED_ENTRIES_TO_PURGE="No deleted entries found for purging"
DELETED_ENTRIES_PURGED="Successfully purged \${NO_DELETED_ENTRIES}
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
deleted entries"
DAEMON_NOTIFIED="Successfully notified daemon"
INVALID_CONFIG_RECORD="Invalid config record, \${BAD_CONFIG_RECORD}"
NO_ENTRIES_FOUND="No log records found for
\${LOG_TRIM_FORMATTED_FROM_DATE} or for later date"
LOG_NOT_TRIMMED="Failed to trim log file"
DATE_NOT_NUMERIC="Date, \${LOG_TRIM_FROM_DATE} not numeric"
INVALID_MONTH="Month, \${MONTH} is invalid"
ENTER_DATE="Must enter a date \(DDMMYYYY\) "
INVALID_DATE_LENGTH="Date, \${LOG_TRIM_FROM_DATE} has invalid length"
INTERVAL_NOT_NUMERIC="FTP interval, \${FTP_INTERVAL} not numeric"
ZERO_INTERVAL="FTP interval must be greater than zero"
INVALID_DAY="\${DAY}, is invalid day for month, \${MONTH}"
RELOAD_CONFIG_FILE="Must reload configuration file"
SERVER_REACHABLE="Server, \${REMOTE_SERVER} is  reachable"
SERVER_NOT_REACHABLE="Server, \${REMOTE_SERVER} is  not reachable"
PINGING_SERVER="Pinging remote server, \${REMOTE_SERVER}"
NOT_AUTHORISED_USER="Must be \${AUTHORISED_USER} to run the script"
}
########################################################################
#  Name     : InitialiseLogFile
#  Overview : The function initializes the global log file.
#  Notes    : 1. This log file is also written to by the daemon process
#                pfari.ksh.
########################################################################
InitialiseLogFile ()
{
if  [ !  -s  "${MFP_LOG_FILE}"  ]
then
    #  initialise the file
    echo "               FTP Log File On ${DATETIME} "       >
${MFP_LOG_FILE}
    echo "               ======================================="  >> \
                                                 ${MFP_LOG_FILE}
fi
return $TRUE
}
########################################################################
#  Name     : HandleInterrupt
#  Overview : The function calls ProcessExit.
#  Notes    :
########################################################################
HandleInterrupt ()
{
DisplayMessage I "${INTERRUPT}"  N
ProcessExit  $FEC
}
########################################################################
#    Name     : MoveCursor
#    Input    : Y and X coordinates
#    Returns  : None
```

 © 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
#    Overview : It moves the cursor to the required location (Y,X).
#    Notes    :
##########################################################################
MoveCursor  ( )
{
YCOR=$1
XCOR=$2
echo     "${ESC}${YCOR};${XCOR}H"
}
##########################################################################
#  Name     : ProcessExit
#  Overview : The function calls ProcessExit.
#  Input    : Exit Code
#  Notes    :
##########################################################################
ProcessExit ()
{
# assign parameter
EXIT_CODE="$1"
rm -f ${TEMP_FILE_1}
rm -f ${TEMP_FILE_2}
exit ${EXIT_CODE}
}
##########################################################################
#  Name     : LogMessage
#  Overview : The function writes a message into the log file
#  Input    : 1. Message type (E = Error, I = Informative)
#             2. Error Code as defined in DefineMessages ().
#  Notes    :
##########################################################################
LogMessage ( )
{
MESSAGE_TYPE=$1
MESSAGE_TEXT='eval echo "${2}"'
if !  InitialiseLogFile
then
   return $FALSE
fi
DATETIME='date "+%d/%m/%Y %H:%M:%S"'
if [ "${MESSAGE_TYPE}" = "E" ]
then
    echo  "${ERROR}${DATETIME}:${MESSAGE_TEXT}"  >>  ${MFP_LOG_FILE}
else
    echo  "${INFO}${DATETIME}:${MESSAGE_TEXT}"   >>  ${MFP_LOG_FILE}
fi
}
##########################################################################
#  Name     : DisplayMessage
#  Overview : The function displays a message
#  Input    : 1. Message type (E = Error, I = Informative)
#             2. Error Code as defined in DefineMessages ().
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
#              3. Message to be acknowledged flag  (Y=yes N=no)
#  Notes    :
###########################################################################
DisplayMessage ( )
{
MESSAGE_TYPE=$1
MESSAGE_TEXT='eval echo $2'
ACKNOWLEDGE_FLAG="$3"
# default the message acknowledge flag
if [ "${ACKNOWLEDGE_FLAG}" = ""  ]
then
  ACKNOWLEDGE_FLAG="Y"
fi
#clear
MoveCursor 24 1
if [ "${MESSAGE_TYPE}" = "E" ]
then
  if [ "${ACKNOWLEDGE_FLAG}" = "Y"  ]
  then
     echo  "${BOLDON}${ERROR}${MESSAGE_TEXT}${BOLDOFF}\c"
  else
     echo  "${BOLDON}${ERROR}${MESSAGE_TEXT}...${BOLDOFF}\c"
  fi
else
  if [ "${ACKNOWLEDGE_FLAG}" = "Y"  ]
  then
     echo  "${BOLDON}${INFO}${MESSAGE_TEXT}${BOLDOFF}\c"
  else
     echo  "${BOLDON}${INFO}${MESSAGE_TEXT}...${BOLDOFF}\c"
  fi
fi
# examine message acknowledge flag
if [ "${ACKNOWLEDGE_FLAG}"  =  "Y"  ]
then
      read  DUMMY
else
      sleep ${SLEEP_DURATION}
fi
return ${TRUE}
}
###########################################################################
#  Name     : DisplayMenu
#  Overview : The function displays a menu.
#  Notes    :
###########################################################################
DisplayMenu ()
{
    clear
    echo "
        #########################################################
        #            ${MENU_TITLE}                            #
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
        #                                                       #
        #    5.  Start  FTP  daemon                             #
        #   10.  Stop   FTP  daemon                             #
        #   15.  Kill   FTP  daemon                             #
        #   20.  View FTP Configuration File                    #
        #   25.  Add Entry to Configuration File                #
        #   30.  Delete Entry from Configuration File           #
        #   35.  Comment Out Entry in Configuration File        #
        #   40.  Uncomment Entry in Configuration File          #
        #   45.  Replace Entry in Configuration File            #
        #   50.  Purge Deleted Entries in Configuration File    #
        #   55.  Reload Configuration File                      #
        #   60.  Adjust FTP Interval                            #
        #   65.  Switch On DEBUG for FTP                        #
        #   70.  Switch Off DEBUG for FTP                       #
        #   75.  View FTP Log File                              #
        #   80.  Trim Log File                                  #
        #   85.  Display Remote Server Status                   #
        #                                                       #
        #   99.  Exit                                           #
        #########################################################
                 Enter Option ----> \c
      "
  read  OPTION
}
#########################################################################
#  Name     : StartFTPDaemon
#  Overview : The function starts the FTP daemon, pfari.ksh, in the
#             background.
#  Notes    :
#########################################################################
StartFTPDaemon ()
{
if  IsFTPDaemonRunning
then
    DisplayMessage E  "${FTP_DAEMON_ALREADY_RUNNING}" N
    return ${FALSE}
fi
nohup ${MFP_SCRIPT_DIR}/pfari.ksh  > ${TEMP_FILE_1} 2>&1     &
if !  IsFTPDaemonRunning
then
    DisplayMessage E  "${FTP_DAEMON_NOT_STARTED}" N
    sed s/*//g  ${TEMP_FILE_1} > ${TEMP_FILE_2}
    ERR_MSG='cat ${TEMP_FILE_2}'
    DisplayMessage E  "${OS_ERROR}" N
    return ${FALSE}
fi
#
LogMessage  I  "${FTP_DAEMON_STARTED}"
echo "\n"  >>  ${MFP_LOG_FILE}
DisplayMessage I "${FTP_DAEMON_STARTED}"  N
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
return $TRUE
}
################################################################################
#  Name     : KillFTPDaemon
#  Overview : The function kills the FTP daemon using kill -9 <pid>
#             command.
#  Notes    :
################################################################################
KillFTPDaemon ()
{
if ! IsFTPDaemonRunning
then
    DisplayMessage E  "${FTP_DAEMON_NOT_RUNNING}" N
    return ${FALSE}
fi
# get the process  id
FTP_DAEMON_PID='ps -eaf | grep -v "grep" | \
                      grep "${MFP_SCRIPT_DIR}/pfari.ksh" | awk {'print
$2'}`
while true
do
  # send kill signal
  kill  -KILL ${FTP_DAEMON_PID}  2> /dev/null
  if !  IsFTPDaemonRunning
  then
    LogMessage     I  "${FTP_DAEMON_KILLED}"
    DisplayMessage E  "${FTP_DAEMON_KILLED}" N
    return ${TRUE}
  fi
done
}
################################################################################
#  Name     : StopFTPDaemon
#  Overview : The function stops the FTP daemon using kill -15 <pid>
#              command.
#  Notes    : 1. The FTP daemon terminates after the current FTP is
#                 complete; otherwise it terminates straight away.
################################################################################
StopFTPDaemon ()
{
if ! IsFTPDaemonRunning
then
    DisplayMessage E  "${FTP_DAEMON_NOT_RUNNING}" N
    return ${FALSE}
fi
# get the process  id
FTP_DAEMON_PID='ps -eaf | grep -v "grep" | \
                      grep "${MFP_SCRIPT_DIR}/pfari.ksh" | awk {'print
$2'}`
# send kill signal
kill  -TERM ${FTP_DAEMON_PID}  2> /dev/null
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
DisplayMessage I  "${DAEMON_NOTIFIED}" N
}
##########################################################################
#  Name     : ViewConfigFile
#  Overview : The function allows users to view the FTP config file
#  Notes    :
##########################################################################
ViewConfigFile ()
{
if [ ! -f  ${MFP_CONFIG_FILE}  ]
then
      DisplayMessage  E  "${NO_CONFIG_FILE}"   N
      return $FALSE
fi
cp  ${MFP_CONFIG_FILE}   ${TEMP_FILE_1}
view ${TEMP_FILE_1}
}
##########################################################################
#  Name     : ViewLogFile
#  Overview : The function allows users to view the FTP log file
#  Notes    :
##########################################################################
ViewLogFile ()
{
if [ ! -f  ${MFP_LOG_FILE}  ]
then
      DisplayMessage  E  "${NO_LOG_FILE}"   N
      return $FALSE
fi
cp  ${MFP_LOG_FILE}   ${TEMP_FILE_1}
view ${TEMP_FILE_1}
}
##########################################################################
#  Name     : ValidateDate
#  Overview : The function validates a date for DDMMYYYY format.
#  Notes    :1. Having validated the date, it formats the input date in
#    DD/MM/YYYY format and stores it in ${LOG_TRIM_FORMATTED_FROM_DATE}.
##########################################################################
ValidateDate ()
{
# supplied date is ${LOG_TRIM_FROM_DATE}
LEAP_YEAR=$FALSE
# validate for non numeric
integer DUMMY_INT
(DUMMY_INT=${LOG_TRIM_FROM_DATE}) 2> /dev/null
if [ $?  -ne 0 ]
then
    DisplayMessage E "${DATE_NOT_NUMERIC}" N
    return $FALSE
fi
# valiadate for length
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
DATE_LENGTH='echo "${LOG_TRIM_FROM_DATE}\c" | wc -c'
if [ ${DATE_LENGTH} -ne  8 ]
then
    DisplayMessage E "${INVALID_DATE_LENGTH}" N
    return $FALSE
fi
# validate  day
DAY='echo ${LOG_TRIM_FROM_DATE}   | cut -c1-2'
MONTH='echo ${LOG_TRIM_FROM_DATE} | cut -c3-4'
YEAR='echo ${LOG_TRIM_FROM_DATE}  | cut -c5-8'
# establish leap year
if [ "`expr ${YEAR} % 4`" -eq 0 ]
then
     LEAP_YEAR=${TRUE}
fi
# validate month
if [ "${MONTH}" !=  "01" -a "${MONTH}" != "02" -a "${MONTH}" != "03"  -a
\
   "${MONTH}" !=  "04" -a "${MONTH}" != "05" -a "${MONTH}" != "06"  -a \
   "${MONTH}" !=  "07" -a "${MONTH}" != "08" -a "${MONTH}" != "09"  -a \
     "${MONTH}" !=  "10" -a "${MONTH}" != "11" -a "${MONTH}" != "12" ]
then
    DisplayMessage E "${INVALID_MONTH}" N
    return $FALSE
fi
if [ "${MONTH}" =  "01" -o "${MONTH}" = "03" -o "${MONTH}" = "05"  -o \
     "${MONTH}" =  "07" -o "${MONTH}" = "08" -o "${MONTH}" != "10" -o \
     "${MONTH}" =  "12" ]
then
     if [[  $DAY -lt 1 || $DAY -gt 31 ]]
     then
         DisplayMessage E "${INVALID_DAY}" N
         return $FALSE
     fi
elif [ "${MONTH}" =  "04" -o "${MONTH}" = "06" -o "${MONTH}" = "09"  -o
\
       "${MONTH}" =  "11" ]
then
     if [[  $DAY -lt 1 || $DAY -gt 30 ]]
     then
         DisplayMessage E "${INVALID_DAY}" N
         return $FALSE
     fi
elif [ "${MONTH}" =  "02" -a "${LEAP_YEAR}" = "${TRUE}" ]
then
     if [[  $DAY -lt 1 || $DAY -gt 29 ]]
     then
         DisplayMessage E "${INVALID_DAY}" N
         return $FALSE
     fi
elif [ "${MONTH}" =  "02" -a "${LEAP_YEAR}" = "${FALSE}" ]
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
then
     if [[  $DAY -lt 1 || $DAY -gt 28 ]]
     then
         DisplayMessage E "${INVALID_DAY}" N
         return $FALSE
     fi
fi
LOG_TRIM_FORMATTED_FROM_DATE="${DAY}/${MONTH}/${YEAR}"
return $TRUE
}
##########################################################################
#  Name     : TrimLogFile
#  Overview : The function allows users to trim the log file
#  Notes    :
##########################################################################
TrimLogFile ()
{
if [ ! -f  ${MFP_LOG_FILE}  ]
then
      DisplayMessage  E  "${NO_LOG_FILE}"   N
      return $FALSE
fi
while  true
do
  tput clear
  echo "Enter date ( DDMMYYYY ) to trim log file from (q to quit):\c"
  read LOG_TRIM_FROM_DATE
  case ${LOG_TRIM_FROM_DATE} in
     q) return $FALSE ;;
    "") DisplayMessage E "${ENTER_DATE}" N  ;;
    * ) if  ValidateDate
        then
             break ;
        fi ;;
  esac
done
# get the starting line number to start trimming  from
LOG_TRIM_START_LINE_NO='grep -n ${LOG_TRIM_FORMATTED_FROM_DATE} \
           ${MFP_LOG_FILE} | cut -d':' -f1 | head -1'
if  [ "${LOG_TRIM_START_LINE_NO}" = ""  ]
then
    # log file does not contain any entries for the suppiled date and
    #  later date
    DisplayMessage I "${NO_ENTRIES_FOUND}" N
    return $FALSE
fi
# delete all records from the log file for supplied and later date
# adjust the line number for three header records (in case supplied
# date has covered the header records
if [  $LOG_TRIM_START_LINE_NO -eq  1  ]
then
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
   LOG_TRIM_START_LINE_NO='expr $LOG_TRIM_START_LINE_NO + 2'
fi
ed <<! ${MFP_LOG_FILE} > ${TEMP_FILE_1}  2>&1
${LOG_TRIM_START_LINE_NO},\$d
w
q
!
if [  $?  -ne  0  ]
then
    DisplayMessage I "${LOG_NOT_TRIMMED}" N
    return $FALSE
fi
}
########################################################################
#  Name     : ShowRemoteServerStatus
#  Overview : The function allows users to find out a remote server's
#             status.
#  Notes    :
########################################################################
ShowRemoteServerStatus ()
{
# get remote server name
while true
do
   tput clear
   echo "Enter remote server name(q to quit ):\c"
   read REMOTE_SERVER
   case ${REMOTE_SERVER}  in

     "" ) DisplayMessage E "${INVALID_ENTRY}"  N  ;;
      q ) return $FALSE ;;
      * ) #
          # ping the server
          #
          DisplayMessage I "${PINGING_SERVER}"  N ;
          ping ${REMOTE_SERVER} > ${TEMP_FILE_1}  2>&1  ;
          if [ $? -eq 0 ]
          then
              DisplayMessage I "${SERVER_REACHABLE}"  N ;
              return $TRUE ;
          else
              DisplayMessage E "${SERVER_NOT_REACHABLE}"  N ;
              sed s/\*// ${TEMP_FILE_1} > ${TEMP_FILE_2} ;
              ERR_MSG='cat ${TEMP_FILE_2}' ;
              DisplayMessage E "${OS_ERROR}"  N ;
              return $FALSE ;
          fi ;;
    esac
done
}
########################################################################
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
#  Name     : PurgeDeletedEntriesFromConfigFile
#  Overview : The function allows a user to purge the configuration
#             file of all deleted  entries
#  Notes    : 1. Records marked for deletion earlier will be removed.
#        2. Lines that need to be deleted have the following structures:
#                # Entry History       <-- history record heading
#                # <date>  Entry Deleted
#                #D#<configigutation record>    <-- deleted entry
#                #D##C#<configigutation record> <-- deleted entry
##########################################################################
PurgeDeletedEntriesFromConfigFile   ()
{
# get line numbers for all deleted entries
grep -n ^#D#  ${MFP_CONFIG_FILE} | cut -d':' -f1 > ${TEMP_FILE_1}
# establish no of deleted entries
NO_DELETED_ENTRIES='cat ${TEMP_FILE_1} | wc -l'
if [  $NO_DELETED_ENTRIES  -eq 0  ]
then
    DisplayMessage I "${NO_DELETED_ENTRIES_TO_PURGE}" N
    return $FALSE
fi
while true
do
#{
  # get next entry line no to be deleted
  ENTRY_LINE_NO='grep -n ^#D#  ${MFP_CONFIG_FILE} | cut -d':' -f1 | head
-1'
  if [ "${ENTRY_LINE_NO}" = ""  ]
  then
      # purged all delete entries
       break
  fi
  # establish the line no for history records' heading
  # initialise the line no to be substracted from the
  # deleted entry no to get to history header record
  LINE_ABOVE=1
  while  true
  do
     HISTORY_REC_HEADING_LINE_NO='expr $ENTRY_LINE_NO - ${LINE_ABOVE}'
     # print this line and compare
     HISTORY_REC_HEADING='sed -n ${HISTORY_REC_HEADING_LINE_NO}p
${MFP_CONFIG_FILE}'
     if [ "${HISTORY_REC_HEADING}" = "# Entry History" ]
     then
           # found the line for history header record
           # decrement by 1 to  get  the comment line above
           HISTORY_REC_HEADING_LINE_NO='expr
${HISTORY_REC_HEADING_LINE_NO} - 1'
           break
     else
          # increment the line no to be substracted
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
          LINE_ABOVE='expr ${LINE_ABOVE} + 1'
     fi
  done
  # remove  the  deleted entry together with its history
  ed <<! ${MFP_CONFIG_FILE} > /dev/null
${HISTORY_REC_HEADING_LINE_NO},${ENTRY_LINE_NO}d
w
q
!
done
#}
DisplayMessage I "${DELETED_ENTRIES_PURGED}" N
}
########################################################################
#  Name     : ReloadConfigurationFile
#  Overview : The function allows users to instruct the FTP daemon
#             process to reload the config file.
#  Notes    : 1. The function sends a PIPE signal to the running
#                pfari.ksh process.
########################################################################
ReloadConfigurationFile ()
{
if ! IsFTPDaemonRunning
then
    DisplayMessage E  "${FTP_DAEMON_NOT_RUNNING}" N
    return ${FALSE}
fi
# get the process  id
FTP_DAEMON_PID='ps -eaf | grep -v "grep" | \
                      grep "${MFP_SCRIPT_DIR}/pfari.ksh" | awk {'print
$2'}`
# send  signal
kill  -PIPE  ${FTP_DAEMON_PID}  2> /dev/null
DisplayMessage I  "${DAEMON_NOTIFIED}" N
}
########################################################################
#  Name     :  AdjustFtpInterval
#  Overview : The function allows users to instruct the FTP daemon
#             process to adjust FTP interval.
#  Notes    : 1. The function sends an EMT signal to the running
#                pfari.ksh process.
########################################################################
AdjustFtpInterval ()
{
if ! IsFTPDaemonRunning
then
    DisplayMessage E  "${FTP_DAEMON_NOT_RUNNING}" N
    return ${FALSE}
fi
# get the FTP interval from user
integer DUMMY_INT
```


© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
while true
do
  tput clear
  echo "Enter required FTP interval ( in seconds or q to quit):\c"
  read FTP_INTERVAL
  case ${FTP_INTERVAL}  in
    q)  return $FALSE ;;
   "") DisplayMessage "${INVALID_ENTRY}" N ;;
    *) #
       # validate  for numaric
      (DUMMY_INT=${FTP_INTERVAL}) 2> /dev/null ;
      if [ $?  -ne Ø ]
      then
         DisplayMessage E "${INTERVAL_NOT_NUMERIC}" N ;
      elif [ $FTP_INTERVAL -le Ø ]
         # must be greater than Ø
         then
             DisplayMessage E "${ZERO_INTERVAL}" N ;
      else
          break ;
      fi ;;
  esac
done
echo "${FTP_INTERVAL}"  > ${MFP_FTP_INTERVAL_FILE}
# get the process  id
FTP_DAEMON_PID='ps -eaf | grep -v "grep" | \
                      grep "${MFP_SCRIPT_DIR}/pfari.ksh" | awk {'print
$2'}`
# send  signal
kill  -EMT   ${FTP_DAEMON_PID}  2> /dev/null
sleep 4
DisplayMessage I  "${DAEMON_NOTIFIED}" N
}
##########################################################################
#  Name     : SwitchOnDebug
#  Overview : The function allows users to instruct the FTP daemon
#             process to switch on debug.
#  Notes    : 1. The function sends a USR1 signal to the running
#                pfari.ksh process.
##########################################################################
SwitchOnDebug ()
{
if ! IsFTPDaemonRunning
then
    DisplayMessage E  "${FTP_DAEMON_NOT_RUNNING}" N
    return ${FALSE}
fi
# get the process  id
FTP_DAEMON_PID='ps -eaf | grep -v "grep" | \
                      grep "${MFP_SCRIPT_DIR}/pfari.ksh" | awk {'print
$2'}`
```

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

```
# send signal
kill  -USR1   ${FTP_DAEMON_PID}  2> /dev/null
DisplayMessage I  "${DAEMON_NOTIFIED}" N
}
#########################################################################
#  Name     : SwitchOffDebug
#  Overview : The function allows users to instruct the FTP daemon
#             process to switch off debug.
#  Notes    : 1. The function sends a USR2 signal to the running
#                pfari.ksh process.
#########################################################################
SwitchOffDebug ()
{
if ! IsFTPDaemonRunning
then
    DisplayMessage E  "${FTP_DAEMON_NOT_RUNNING}" N
    return ${FALSE}
fi
# get the process  id
FTP_DAEMON_PID='ps -eaf | grep -v "grep" | \
                      grep "${MFP_SCRIPT_DIR}/pfari.ksh" | awk {'print
$2'}`
# send signal
kill  -USR2  ${FTP_DAEMON_PID}  2> /dev/null
DisplayMessage I  "${DAEMON_NOTIFIED}" N
}
#########################################################################
#  Name     : ReplaceEntryInConfigFile
#  Overview : The function allows users to replace an entry in the
#             configuration file.
#  Notes    :1. The function calls the following functions:
#                          o GetRemoteHostName
#                          o GetRemoteDirectoryName
#                          o GetRemoteFileName
#                          o GetRemoteUserId
#                          o GetRemotePassword
#                          o GetLocalDirectoryName
#                          o GetLocalFileName
#                          o GetAction
#                          o GetComment
#           2. The function adds a history record
#########################################################################
ReplaceEntryInConfigFile ()
{
# prepare  header for list of values file
# header consists of 4 lines
echo  "Configuration File Entries on ${DATETIME}"   >  ${TEMP_FILE_1}
echo  "==================================================" >>
${TEMP_FILE_1}
echo  "To Replace an Entry Delete Corresponding Line and Save the
File\n" >> \
```

 © 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
                                                          ${TEMP_FILE_1}
LOV_FILE_NO_HEADER_LINES=4
# find all entries except  marked for commented and deletion
grep "Remote_Host=" ${MFP_CONFIG_FILE} | grep -v "^#C#" | grep -v "^#D#"
>> ${TEMP_FILE_1}
# does the list of values file have any entries
LOV_FILE_LINES='wc -l ${TEMP_FILE_1} | awk {'print $1'}'
if [ ${LOV_FILE_LINES} -le  ${LOV_FILE_NO_HEADER_LINES}   ]
then
     DisplayMessage E "${NO_ENTRY_TO_REPLACE}" N
     return $FALSE
fi
```

Editor's note: this article will be concluded next month.

Arif Zaman
DBA/Developer (UK) © Xephon 2003

Character to hex

I recently needed to give a user group a way of converting characters to hex. I came up with two solutions – one a command and the other a little shell script.

The command is:

```
perl -e 'printf "%X\n", ord("A");'
```

which will return the hexadecimal representation of the ASCII character 'A' (41).

My simple (ksh) script was:

```
#!/bin/sh
export CHAR="B"
HEX=$(perl -e 'printf "%X\n", ord($ENV{CHAR});')
print "$CHAR = $HEX"
```

Stephen Hare
AIX Systems Expert (USA) © Xephon 2003

© 2003. Reproduction prohibited. Please inform Xephon of any infringement.

AIX news

Embarcadero has announced Job Scheduler 3.0, which provides database administrators with centralized control of cross-platform, enterprise-wide job scheduling. The latest version provides support for the companys data integration solution, Embarcadero DT/Studio, and introduces a new Java-based scheduler server to support AIX 5.1, Solaris 2.7/2.8, HP-UX 11.00, and Red Hat Linux 7.1 or later, and Windows.

Built-in database maintenance wizards, instant HTML reporting and documentation, and an easy-to-use centralized console help to further streamline job scheduling.

This latest version provides wizards that assist users in integrating DT/Studio tasks with other enterprise operations. Users can employ the integration of business processes and data with the ease of managing everything from a single interface.

For further information contact:
Embarcadero Technologies, 425 Market Street, Suite 425, San Francisco, CA 94105, USA.
Tel: (415) 834 3131.
URL: http://www.embarcadero.com/products/jobscheduler/index.asp.

* * *

Sybase has announced that Version 4.0 of Integration Orchestrator, a business process management and enterprise application integration solution, is now available on AIX and HP-UX, in addition to Windows and Solaris.

Sybase Integration Orchestrator's open, standards-based technology enables incremental deployment of projects, speeding overall time-to-deployment and reducing project risk, says the company. It also supports store-and-forward/message queuing and targeted messaging – a feature that delivers rapid execution of time-critical transactions.

For further information contact:
Sybase, One Sybase Drive, Dublin, CA 94568, USA.
Tel: (925) 236 5000.
URL: http://www.sybase.com/products.

* * *

Advanced Digital Information Corporation (ADIC) has announced Version 2.2 of StorNext Management Suite (SNMS), its data management software for open system SANs.

The product now supports AIX, SuSE Linux, and the latest versions of Windows XP and Windows 2003 server. Support has also been extended to new tape-based storage systems, including AIT WORM drive technology, StorageTek 9940B drives, and ADIC's new Scalar i2000 automated tape library.

With SNMS, users define data management policies that both place newly-created data on appropriate disk types and automatically move existing data between expensive, enterprise disk systems, low-cost ATA arrays, and tape media. Locating data automatically on different media types based on quality of service, access time, and protection requirements allows IT managers to reduce their total ownership costs while managing the life-cycle of data.

For further information contact:
ADIC, 11431 Willows Road NE, Redmond, WA 98052, USA.
Tel: (425) 881 8004.
URL: http://www.adic.com/stornext.