103

AIX update

May 2004

In this issue

© Xephon Inc 2004

AIX Update

Published by
Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

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

Editor
Trevor Eddolls
E-mail: trevore@xephon.com

Publisher
Nicole Thomas
E-mail: nicole@xephon.com

Subscriptions and back-issues
A year’s subscription to *AIX Update*, comprising twelve monthly issues, costs $275.00 in the USA and Canada; £180.00 in the UK; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 2000 issue, are available separately to subscribers for $24.00 (£16.00) each including postage.

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

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

© Xephon Inc 2004. 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.

Printed in England.

Administering users in an HACMP cluster

This article tries to address some of the issues that system administrators encounter while administering an HACMP cluster.

All of the issues listed have been encountered in a production environment and some fixes have been created quickly to address issues promptly, so perhaps could be improved on.

At our site there are two nodes running the HACMP V4.4 software – I have tried to take care of sites using HACMP/ES by adding appropriate variables although commenting them out.

The users access the same application running independently on each node. Each user can access both nodes, but may not need to do so until the event of a failover; then the users must be able to access the node hosting all of the resources. This provides us with two problems – password security and password consistency.

Auditors' requirements on most systems are very tight and the combination of HACMP and AIX does not provide much for us to use when dealing with the attributes set for users within a cluster environment. One such attribute that we address is the MAXAGE attribute. We have a requirement to ensure that users change passwords every four weeks.

Once the four weeks have passed, the login command checks the MAXAGE attribute and identifies that the user must change his password. This then invokes a low-level function of the **passwd** command, and the user must change his password.

While this is fine on single log-on systems, our user now has inconsistent passwords across the cluster. Users will also get inconsistent passwords if they decide to change their password by invoking the */usr/bin/passwd* command.

Most scripts in use here use a library function that I have created for determining the IP addresses of nodes in the cluster for generic usage across any cluster without the need for hard-coding server names or IP addresses. The library is called getclusterinfo and resides in the */usr/local/bin/libs* directory.

INCONSISTENT PASSWORDS

First, to address the inconsistent password problem, we created a script that would execute the cluster version of the **passwd** command. The script mimics some of the functionality that the AIX **passwd** command enjoys, such as accepting the user id as an argument, as well as checking whether the user executing the script is actually root or an admin user with the ability to change other users' passwords.

```
#!/bin/ksh
#Script Name    : chclpwd
#Script Version : 1.4
#Written By     : Stewart Robinson (Ø2/11/Ø1)
#Cron Time      : not cron'd
#Description    : runs cluster passwd command on node
#Amendments     : SR Ø7/12/Ø1 V1.1 :
#                     : Change script to get PID and cmd associated
#                     : so that we do not access passwd run from the
#                     : HACMP scripts -
#                     : IBM should be more vigilant in their use of
#                     : fully-qualified paths for commands
#                     : SR 25/Ø1/Ø2 V1.2 : Change made to carry out
#                     : security checks on users allowed to change each
#                     : other's passwords this includes checks
#                     : against root
#                     : SR Ø1/Ø2/Ø2 V1.3 : Change check as above (1.1)
#                     : to simplify for generic HACMP Builds
#                     : SR 24/2/Ø4 V1.4 : Change way script obtains IP
#                     : addresses - use shell script library
#
#
#***********************************************************************
PATH=:/usr/sbin/cluster:/usr/sbin/cluster/cspoc:/usr/sbin/cluster/
sbin:$PATH
#PATH=:/usr/es/sbin/cluster:/usr/es/sbin/cluster/cspoc:/usr/es/sbin/
cluster/sbin:$PATH
export PATH
#export library variables
. getclusterinfo
```

© 2004. Xephon USA telephone (214) 340 5690, fax (214) 341 7081.

```
CURPROC=$Ø
USERID=$1
LOGFILE="/root/file/cluster_passwd_changes"

log_to_file(){
echo "`date` : chclpwd : $* " >> $LOGFILE
}

who am i |awk '{print $1}' |read RUNUSER

if [ $# != 1 ]
then
  #no args or more than one arg - assign USERID to user process running
        USERID=$RUNUSER
else
        #Check if we are actually root (check against su) - to see if
        #we can change anyone's passwd
        if [ $RUNUSER != root ]
        then
                #Check UID is 'whoami' if not exit...
                if [ $USERID != $RUNUSER ]
                then
                        #user is trying to change passwd for someone
else -
                        #check if member of security group
                grep security /etc/group |grep $RUNUSER |wc -l |read OK
                        if [ x$USERID = xroot ]
then
                                OK=Ø
                        fi
                        case $OK in
                                Ø ) echo "You do not have authority to
change the Password for $USERID "
                                exit 2 ;;
                                * ) continue ;;
                        esac
                fi
        fi
fi


change_via_smit(){
echo "\nPlease Wait - checking user account details ...\n"

#cl_chpasswd command flags -cspoc -f forces a skip in the verification
#that all clusters are active - need to copy passwd files back when
#node starts again

log_to_file "Checking other node available ..."
```

```
IPLABELS='echo $REMOTEPLABELS |sort -r'
check_active(){
ISALIVE=Ø
for NAME in $IPLABELS
do
       ping -c3 $NAME >>/dev/null
       rc=$?
       if [ $rc = 1 ]
       then
                echo " cannot ping adapter $NAME" >>/dev/null 2>&1
       else
                 ISALIVE='expr $ISALIVE + 1'
                 echo " can ping adapter $NAME" >>/dev/null 2>&1
                 break
        fi
 done
if [ $ISALIVE -ge 1 ]
then
    log_to_file "Other node available - running cluster passwd command"
        log_to_file "User $USERID changing passwd ..."
  /usr/sbin/cluster/cspoc/fix_args nop cl_chpasswd -cspoc -f -k $USERID
        if [ $? -ne Ø ]
        then
                log_to_file "User $USERID passwd change failed ..."
                exit 1
        else
        log_to_file "User $USERID passwd change successful"
        exit Ø
        fi
else
        log_to_file "Cannot access other node - attempting to run /usr/
sbin/passwd command "
        /usr/bin/passwd $USERID
fi
}

check_active

}

###############################################
#Script starts here
###############################################
PID=$$
AIXCLPWD="/usr/sbin/cluster/sbin/cl_chpasswd"
#AIXCLPWD="/usr/es/sbin/cluster/sbin/cl_chpasswd"

if [ ,ps -ef |grep $PPID |grep $AIXCLPWD |grep -v grep | wc -l, = 1 ]
then
        #cluster command running this script - run /usr/bin/passwd
        log_to_file "running /usr/bin/passwd"
        /usr/bin/passwd $USERID
else
        #normal usage of script - run cluster password command
        log_to_file "running cluster util"
        change_via_smit
fi
```

After creating the script we needed some way of running it as root (for the cluster command to work) and also ensuring that users used this script and not the */usr/bin/passwd* command, which creates inconsistent cluster passwords. This is achieved by compiling a C program (so that users would not be able to manipulate the script) and naming it passwd in the */usr/local/bin/commands/* directory.

The source is very simple:

```
serverØ2:/usr/local/bin/source> cat passwd.c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

main(int argc, char **argv[])
{

        setuid (Ø);

        execlp("/usr/local/bin/command/chclpwd", "chclpwd", argv[1],
(char *) Ø);
}
```

The compiled program should have the permissions set so that the cluster commands can run:

```
serverØ2:/usr/local/bin/command> ls -l passwd
-rwsr-xr-x   1 root     system      4415 25 Jan 2ØØ2  passwd
```

To set the permissions:

```
chmod 4755 passwd
```

To ensure that all users run this binary before the */usr/bin/passwd* command, we modified several system files and edited the PATH variable so that */usr/local/bin/commands* was resolved before */usr/bin*. The following example shows the last entry in */etc/profile* to illustrate the point:

```
serverØ2:/> tail -1 /etc/profile
export PATH="/usr/local/bin/command:"$PATH
```

Other files that need modifying in the same way are:

- */etc/security/.profile* (default .profile copied to new users' home directories).
- All users' .profile files.

We are now in a position whereby we can be sure that passwords will not become inconsistent between the nodes of the cluster whilst HACMP is active on both nodes.

We experienced errors when a node was physically inactive in the cluster (inaccessible to network traffic) and this caused:

- The user to see an error that we already knew would exist.
- The cluster password to become inconsistent because it would not update the following files on the inactive node:
  - */etc/passwd*.
  - */etc/security/passwd*.
  - */etc/security/user*.

We addressed the issue of error messages being reported by having a check built into the chclpwd script ensuring that if a node is physically inaccessible it will run the */usr/bin/passwd* command and not the cluster password.

To address the second issue, when the failed node becomes accessible again, we check the files for consistency, and, if any are found, copy these files to the failed node when the system boots. This can be automated via the script copy_in_passwdfiles and the **inittab**.

The script uses the **rdist** command and as we need to push the files using **rdist** we need to execute the script on the opposite node within the cluster – the script caters for this by being able to detect whether it is running on a failed node just coming up and then remotely running the script on the active node with a parameter for the copy of the files – see below:

```
serverØ2:/> lsitab cppasswd
cppasswd:2:once:/usr/local/bin/script/copy_in_passwdfiles
```

copy_in_passwdfiles:

```
#!/bin/ksh
#Script Name    : copy_in_passwdfiles
#Script Version : 1.4
#Written By     : S.P.Robinson
#Cron Time      : N/A
#Description    : Copies passwd files from active node if this node has
#               : been Powered down. Ensures that passwords are
#               : synchronized between nodes
#Amendments     : SR V1.Ø Needs to rdist files onto boot address
#               :of down machine
#               : SR V1.1 Could be either boot / service address to
#               : receive files from
#               : SR V1.2 Change file list to include groups
#          : 1Ø/Ø9/Ø3 - SR V1.3 Change script to log more information
#          : 2Ø/Ø2/Ø4 - SR V1.4 Change script to use HACMP library info
#
#**********************************************************************
#get library variables for cluster
. /usr/local/bin/libs/getclusterinfo

log_to_file(){
echo "`date` : $*" >>"/tmp/copy_in_passwdfiles.log"
}

create_rdist_file(){
RDISTF="/root/file/rdist.file"
FILELIST="/etc/passwd /etc/opasswd /etc/security/passwd /etc/security/
opasswd /etc/group /etc/ogroup /etc/security/group /etc/security/ogroup
/etc/security/user /etc/security/ouser"
echo "HOSTS= ( $DEST )" >$RDISTF
echo "FILES= ( $FILELIST )" >>$RDISTF
echo "\${FILES} ->\${HOSTS}" >>$RDISTF
}

ping_func(){
NOPING=$1

if [ x$NOPING = x ]
then
log_to_file "Ping test to establish which adapter to send to… test
$REMOTE_BOOT_ADDRESS – boot address"
ping -c3 $REMOTE_BOOT_ADDRESS  >>/dev/null
            rc=$?
            if [ $rc = 1 ]
then
```

```
                  log_to_file "Ping failed to $REMOTE_BOOT_ADDRESS –
boot address – trying service address …"
                        ping -c3 $REMOTE_SVC_ADDRESS  >>/dev/null
                        rc=$?
                        if [ $rc = 1 ]
                        then
                  log_to_file "Ping failed to $REMOTE_SVC_ADDRESS –
service address – trying standby  …"
                  ping -c3 $REMOTE_STB_ADDRESS  >>/dev/null
                              rc=$?
                              if [ $rc = 1 ]
                              then
                        log_to_file "Ping failed to $REMOTE_STB_ADDRESS
– standby adapter address – cannot contact node $REMOTENODE – exiting
…"
                                exit 3
                              else
                        export SOURCE=$REMOTE_STB_ADDRESS
                  fi
            else
                  export SOURCE=$REMOTE_SVC_ADDRESS
            fi
      else
            export SOURCE=$REMOTE_BOOT_ADDRESS
       fi
fi

}

log_to_file "Script starting"

DEST=$1
if [ x$DEST = "x" ]
then
      log_to_file "rsh this script to opposite node as this is the host
on which we want the files and rdist supports only pushing files, not
pulling…"
              ping_func noping
      rsh $REMOTE_SVC_ADDRESS /usr/local/bin/script/copy_in_passwdfiles
$LOCAL_BOOT_ADDRESS
        rc=$?
        if [ $rc != Ø ]
        then
            log_to_file "Script failed on remote node with boot address
-- trying service"             rsh $REMOTE_SVC_ADDRESS /usr/local/bin/
script/copy_in_passwdfiles $LOCAL_SVC_ADDRESS
            rc=$?
            if [ $rc != Ø ]
             then
                  log_to_file "Script failed on remote node –
```

```
investigate"
                  mail –s "Passwd copy script failed on remote node
$SOURCE – investigate"  SysAdmin@YourCompany.com
                  exit 1
            fi
        else
                #script worked - exit Ø
                exit Ø
        fi
else

       ping_func
      #rdist flags -f file to use; -y prevent newer files being replaced
by older ones
      create_rdist_file
      log_to_file "rdist being executed on this node"
      rdist -f $RDISTF -y
      rc=$?
      if [ $rc != Ø ]
      then
            log_to_file "Script failed on local node – investigate"
            mail -s "Passwd copy script failed on local node $LOCALNODE
- investigate" SysAdmin@YourCompany.com
                      exit 2
              else
                      #script worked - exit Ø
                      exit Ø
              fi
fi
```

As can be seen, the script will take care of all of the addresses assigned to each node in the attempt to copy the files. This script also copies the Unix group files in the case of a new user being added while a node is down.

USER PASSWORD EXPIRATION

As previously mentioned we have quite strict audit requirements for ensuring that users change their passwords every 28 days. As we cannot address this with HACMP we have built a solution around it.

Instead of setting the MAXAGE variable to 4 (time in weeks) we decided to change it to 8 and have a script check the time of last password change and enforce a change after four weeks. This allows us some spare time in case a user goes on holiday and we do not want the account locking. This will ensure that we do not end up with inconsistent passwords.

The issues here are that in each user's profile we need to insert the line */usr/local/bin/command/checkpwd* and also ensure that this is in the default user profile (as listed earlier in the article).

The script checkpwd uses other scripts to obtain information. It uses the script **getval.sh** to obtain the user's last password change time in epoch format and also the MAXAGE value. However, this invokes the **lssec** command, which requires a user to be a member of the security group – so the work-around is a C program with appropriate permissions set so that the script can run as root.

The source code for the C program is:

```
serverØ1:/usr/local/bin/source> cat getval.c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

main(int argc, char *argv[])
{
setuid(Ø);
execlp("/usr/local/bin/command/getval.sh", "getval.sh", argv[1], Ø);
}
```

Compile with cc getval.c getval.

Change the permissions to appropriate values:

```
serverØ1:/usr/local/bin/command> ls -l getval*
-rwsr-xr-x   1 root     system      4415 25 Jan 2ØØ4  getval
-rwsr-xr-x   1 root     system      4415 25 Jan 2ØØ4  getval.sh
```

To set the permissions:

```
chmod 4755 getval*
```

The code of the script that the getval program runs is:

```
#!/bin/ksh
#Script Name    : getval.sh
#Script Version : 1.Ø
#Cron Time      : N/A
```

© 2004. Xephon USA telephone (214) 340 5690, fax (214) 341 7081.

```
#Description    : Retrieves users maxage or lastupdate value
#************************************************************************

case $1 in
      maxage) lssec –f /etc/security/user –s $LOGNAME –a $1 | awk –F '='
'{print $2} ' ;;
      lastupdate) lssec –f /etc/security/passwd –s $LOGNAME –a $1 | awk
–F '=' '{print $2} ' ;;
esac
```

The script code for checkpwd is:

```
#!/bin/ksh
#Script Name    : checkpwd
#Script Version : 1.Ø
#Cron Time      : N/A
#Description    : Checks whether it is time for the user to change
#                 their password.
#
#************************************************************************
trap "" 1 2 3

#set cluster library variables
. /usr/local/bin/libs/getclusterinfo

l_ogfile=/root/file/checkpwd.log

Checkfailover () {
netstat -i | grep $LOCAL_SVC_ADDRESS 1> /dev/null 2>&1
r_code1=$?
netstat -i | grep $REMOTE_SVC_ADDRESS 1> /dev/null 2>&1
r_code2=$?
if [ $r_code1 = Ø -a $r_code2 = Ø ]
  then
    f_ailover=y
  else
    f_ailover=n
fi
}

Checkmaxage() {
m_axage='/usr/local/bin/command/getval "maxage"'
if [ $? != Ø ]
  then
    log_to_file "maxage check failed for user $LOGNAME"
    m_axage=Ø
fi
}

Checklastupdateflag() {
```

```
l_udflag='/usr/local/bin/command/getval "lastupdate"'
if [ $? = Ø ]
  then
    c_hangetime='expr $l_udflag + 24192ØØ'
    if [ $c_hangetime -lt '/usr/local/bin/command/now_epoch' ]
      then
        c_flag=1
      else
        c_flag=Ø
    fi
  else
    log_to_file "password age check failed for user $LOGNAME"
    c_flag=Ø
fi
}

log_to_file() {
echo "`date`: $*" >> $l_ogfile
}

#
# Execution starts here
#
Checkfailover
if [ $f_ailover = n ]
  then
    Checkmaxage
    if [ $m_axage != Ø ]
      then
        c_ount=Ø
        Checklastupdateflag
        while [ $c_flag = 1 -a $c_ount -lt 2 ]
          do
            echo "\nPassword at least 28 days old. Starting change
password procedure........"
            /root/pilkcommand/passwd $LOGNAME
            Checklastupdateflag
            c_ount='expr $c_ount + 1'
          done
        if [ $c_flag = 1 ]
          then
            log_to_file "user $LOGNAME logged out because they chose not
to change password"
            logout
        fi
    fi
fi
trap 1 2 3
```

The script checkpwd uses the now_epoch C program to obtain the date in epoch format. The C source code is:

```
/*
 * FUNCTION:
 * Displays epoch time
 *
 * SYNTAX:
 * epoch
 *
 * ARGUMENTS:
 * none
 *
 * RETURNS:
 * The time in seconds since start of 197Ø
 * Compile with:
 * cc -o now_epoch now_epoch.c
 */

#include <time.h>
#include        <stdio.h>
#include        <stdlib.h>
#include        <sys/types.h>
#include        <sys/timeb.h>
#include        <sys/time.h>
#include        <sys/types.h>

main() {

int rc;
struct timeb timeb;

rc = ftime(&timeb);
printf( "%d\n", timeb.time);

}
```

Finally the source code for the script that obtains the cluster information (this may be handy to use for other scripts as well as in our examples here) is:

```
#!/bin/ksh
#Script Name    : getclusterinfo
#Script Version : 1.1
#Written By     : Stewart Robinson 25/Ø2/Ø4
#Cron Time      : N/A
#Description    : Gets info from odm regarding cluster - I.E. IP
addresses
#               : and nodenames as variables for use in scripts
#               : Assumption is a two node cluster
#
```

```
#*************************************************************************
export LOCALNODE='odmget HACMPcluster |grep nodename |awk -F '"' '{print
$2}''

export ALLNODES='odmget -q "object=VERBOSE_LOGGING" HACMPnode |grep
name|awk -F '"' '{print $2}''

export REMOTENODE='echo $ALLNODES |sed "s/$LOCALNODE//g" |sed "s/ //g" '

export LOCAL_SVC_ADDRESS='odmget -q "function=service and type=ether and
nodename=$LOCALNODE" HACMPadapter |grep identifier |awk -F '"' '{print
$2}' '

export LOCAL_BOOT_ADDRESS='odmget -q "function=boot and type=ether and
nodename=$LOCALNODE" HACMPadapter |grep identifier |awk -F '"' '{print
$2}' '

export LOCAL_STB_ADDRESS='odmget -q "function=standby and type=ether and
nodename=$LOCALNODE" HACMPadapter |grep identifier |awk -F '"' '{print
$2}' '

export REMOTE_SVC_ADDRESS='odmget -q "function=service and type=ether
and nodename=$REMOTENODE" HACMPadapter |grep identifier |awk -F '"'
'{print $2}' '

export REMOTE_BOOT_ADDRESS='odmget -q "function=boot and type=ether and
nodename=$REMOTENODE" HACMPadapter |grep identifier |awk -F '"' '{print
$2}' '

export REMOTE_STB_ADDRESS='odmget -q "function=standby and type=ether
and nodename=$REMOTENODE" HACMPadapter |grep identifier |awk -F '"'
'{print $2}' '

export ALLIPLABELS='odmget -q "type = ether" HACMPadapter |grep ip_label
|awk -F
 \" '{print $2}' `

export REMOTEIPLABELS='odmget -q "type = ether and nodename !=
$LOCALNODE" HACMPadapter |grep ip_label |awk -F \" '{print $2}' '

export LOCALIPLABELS='odmget -q "type = ether and nodename = $LOCALNODE"
HACMPadapter |grep ip_label |awk -F \" '{print $2}' '
```

Stewart Robinson
Technical Services Manager
Pilkington (UK)

© Xephon 2004

AIX mail in the mailbox on your PC

INTRODUCTION

Every AIX administrator will have configured the mail system in AIX. Somewhere in the login sequence they will have placed some lines to do with the mail – that's why you get the message 'YOU HAVE NEW MAIL'. If you want to send mail, the file */etc/sendmail.cf* is configured and the mailserver is specified in a line somewhere.

How it works in practice is: if you are logged on, you will get the message 'YOU HAVE NEW MAIL', and if you aren't too busy, you will read it – otherwise you will read it another time. In our company we have lots of AIX servers, and I am not logged on to all of these servers everyday, so I miss a lot of mail. If I decide to log on to one of the less-frequently used (by me) servers, I sometimes have lots of mail – and sometimes no mail. Sometimes this mail is (was) important!

I can make two choices about how to read all my mail (and mail from other users that looks interesting). Either, once a day (or week) I can log in to all the AIX servers and spend (waste) a lot of time logging in and checking whether there is mail on all these servers. My second choice is to deliver the mail and other stuff to the mailbox on my PC.

I chose the last option. Users now get their mail on their PCs a quarter of an hour after it is delivered to an AIX box. If it is important then we can react much quicker to the mail than with the first option. You can choose to get the mail in a special format, so you can see whether the mail is sent by an AIX user or not.

IMPLEMENTATION STEPS

To implement the delivery of the AIX mail to my PC mailbox I had to do a few things on the AIX machine:

© 2004. Reproduction prohibited. Please inform Xephon of any infringement.

1 Configure the mail system.

2 Check whether there is mail.

3 Send the AIX mail to my PC.

Step 1: configure the mailserver on AIX

The first thing is easy; there is a configuration file in */etc* called *sendmail.cf*. In this file you can specify the network mail system name where the mail will be delivered. In the file, check for the next word beginning on the line:

```
^DS - DS
```

If the whole line is just the two letters DS, the system is not configured for sendmail. To configure the mail server, place the name of this server right after the DS, for example:

```
Dsmailserver.domain_name.company
```

At this point you can deliver mail to your mailbox on your PC, and you can check it with the command:

```
mail -s "This is a test" your_name@company  <$HOME/.profile
```

This will send your .profile to your PC.

Steps two and three: check for the existence of mail on AIX and mail it

I decided to write a script that would be executed every 15 minutes. To start it up I use the crontab of a user who is a member of the group mail. These times can be configured however you like.

Example:

```
Ø,15,3Ø,45 * * * * /usr/local/bin/mail2myPC
```

MAIL2MYPC SCRIPT

This script uses two files:

- mail_addressbook

© 2004. Xephon USA telephone (214) 340 5690, fax (214) 341 7081.

This file is used to translate the AIX user to a known user, group of users, or a list of users on the network.

- mail_unknown

  This file is used to filter users not mentioned in the first file.

mail2myPC code

```
# Name           : /home/oper/mail2myPC
# Last change    : 16 February 2004 Teun Post created the script
# Purpose        : mail files in /var/spool/mail to mailboxes on the LAN
#-----------------------------------------------------------------------

# Define the AIX user administrator
user_administrator=Your.UserAdministrator@yourCompany

# Define files
unknown_users=/user_data/mail_unknown
mail_addressbook=/user_data/mail_addressbook

# Send mail
send_mail () {
 mail_address=$1
 mail -s "Mail from user $file at $hostname" ${mail_address} <$file
 shift 1
 mail_list=$*
}

# Get the hostname once, it is much easier if you know where the mail is
# generated
hostname='hostname'

# create temp file to send
echo "Please configure user at $hostname in $mail_addressbook" >/tmp/
mail2myPC.tmp
echo "After configuring user: remove user in $unknown_users" >>/tmp/
mail2myPC.tmp

# go the the mail directory
cd /var/spool/mail
# list all mail
for file in 'ls'
     do
     # get the mail address in $mail_addressbook : the second part
     # explication grep: it must begin with the user AND do not use
     # parts of a user which begins with the same name
    mail_list='cat $mail_addressbook | grep "^$file " | cut -f2- -d" "'

     # Mail only non-empty files!
     if [ ! -s $file ]
     then rm $file
          continue
          fi

     if [ "$mail_list" ]
     then # address(es) found: you've got mail
          echo $mail_list | while [ "$mail_list" ]
               do
               # Send the mail to the mailing list
               send_mail $mail_list
               done
          rm $file
     else # address not found: send request to deal with the user
          already_send='cat $unknown_users | grep "^$file "'
          if [ ! "$already_send" ]
          then # Address not found, send mail to administrate user
               mail -s "Unknown mail address $file in $hostname"
$user_administrator </tmp/mail2myPC.tmp
               echo "$file ">>$unknown_users
               fi
          fi
     done

# exit script
```

The file mail_addressbook

The mail_addressbook file is used to transform AIX users to valid network mail users or groups.

Example file mail_addressbook:

```
# Format: aixuser valid_mail_address valid_mail_address …
oracle dba_users@your_company.country
root aix_administrators@your_company.country
usera lonely_me@your_company.country
user1 userA@yourcompany.country userB@your_company.country
user2 application_group1@your_company.country
user3 application_group2@your_company.country
user4 usera.support.fr
```

Explanation: if the AIX user oracle has mail, it will be sent to all the database administrators.

All mail for usera will be sent to lonely_me. The users userA and userB will get the mail from user user1, and the support group in France will get the mail of user4.

The file mail_unknown

The file mail_unknown contains users whose network mail address is not known or who were just forgotten to be included. This file is created when one user who has mail cannot be found in the file mail_addressbook. Once a user is in this file, no messages like ‘User user1 not found in file mail_addressbook, please include the user’ will be generated.

Example file mail_unknown:

```
# Format aixuser
userA
user99
applic1
OS$appl
```

Examples output:

1 Output of send mail:

Heading in Outlook:

```
Mail_user@unixa.domain_name Mail from user userA at unixa ma 16-2-04
11:03
```

Mail:

```
   From daemon Sun Feb  8 13:00:00 2004
   Received: (from daemon@localhost)
             by unixa.domain_name (AIX4.3/8.9.3p2/8.9.3) id NAA259792
             for userA; Sun, 8 Feb 2004 13:00:00 +0100
   Date: Sun, 8 Feb 2004 13:00:00 +0100
   From: daemon
   Message-Id: <200402081200.NAA259792@unixa.domain_name>
   To: userA

   userA 151110 164748 0 12:59:24 - 0:00 ftp -v
other_aix.other.domain.name


   *************************************************
   Cron: The previous message is the standard output
        and standard error of one of your cron commands.
```

2 Output from unknown mail address:

Heading in Outlook:

```
Mail_user@unixa.domain.name Unknown mail address root in unixa ma 16-2-
Ø4 1Ø:55
```

Mail:

```
   Please configure user at unixa in /user_data/mail_addressbook
   After configuring user: remove user in /user_data/mail_unknown
```

After configuring the mail delivery system, you can implement other applications such as:

- Error reporting.
- Security.

Just get the data out of the application and place the output in */var/spool/mail*. It can be an unused user id like error_reports and (when configured) it will be forwarded to the right person(s), and it will be delivered on time.

Below is an example of such a proposed extension to the application. Here is some output that would be sent to a PC user, using this application from user error_report at aftaxp03:

```
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
BEØAØ3E5   Ø12Ø23Ø3Ø4 P H sysplanarØ     ENVIRONMENTAL PROBLEM
BEØAØ3E5   Ø21822Ø4Ø4 P H sysplanarØ     ENVIRONMENTAL PROBLEM
BFE4CØ25   Ø21Ø1123Ø4 P H sysplanarØ     UNDETERMINED ERROR
BFE4CØ25   Ø21Ø1541Ø4 P H sysplanarØ     UNDETERMINED ERROR
```

This will be described in more detail in a future article.

Teun Post
AIX Administrator/Oracle DBA
Schuitema NV (The Netherlands)

© Xephon 2004

AIX system cloning AIX 4.3.3

Novice AIX system administrators often run into trouble after a cloned system starts up.

Why?

1. They forget to disconnect the cloned system from the network. The system starts with the same IP address as the original system. This leads to duplicate IP addresses talking and listening on the same network – if the source system is a production machine – we get lots of trouble with the availability of the delivered service.
2. Monitoring scripts and application scripts, started automatically via the cron or a subsystem, generate the wrong information and confuse the operators or Help Desk people.
3. E-mails that were queued at the time the system back-up was taken are sent twice. Users are unhappy and start complaining.

Before you start cloning a system a detailed analysis has to be done. Actions must be planned – what configuration decisions should be taken and how.

There are two primary decisions. You can do it manually with **mksysb** tape and CD-ROM, or within an existing NIM environment.

The first way assumes that both systems have the same types of tape drive and the target system has a CD-ROM drive available.

The second way needs a NIM environment set up – and works only if the source operating system level is lower than the NIM master level. If it is not, the NIM master has to be upgraded to this level before starting.

STARTING POINT

Write down the following:

- The new IP address that the system will get.
- The new default gateway if the clone system should work in a network other than the one for the source system.
- New hostname.
- The disk layout (partition size, count, and size of disks) of the **rootvg**. If your clone system has the same number of disks, and all the disks are at least the same size, the restore/install process will not run into any trouble. If you like to clone from a mirrored system to a non-mirrored system, or want to make any changes to parts of the **rootvg** layout (eg reduced filesytem sizes, more disks but less disk space), cloning is still possible. Some tricks and adjustments will have to be done.
- All non-root volume groups steps can be carried out after the clone is on-line.

METHOD 1 – USING TAPE

If you plan to use the tape method, you have to prepare a floppy disk.

Put it in the source system. Format it:

```
mkdir /tmp/floppy

echo data > /tmp/floppy/signature

cp /bosinst.data /tmp/floppy
cp /image.data /tmp/floppy
cd /tmp/floppy
```

Create a script that does the customizing after the system install has finished.

Configure the IP address and hostname.

vi custom.ksh:

```
# sample code of custom.ksh

#!/bin/ksh
log=/setupnic.log
cat >> /etc/firstboot <<EOF
# cron:2:respawn:/usr/sbin/cron
rmitab cron # remove cron from inittab
#
# to activate cron again after cronjob review
#
# mkitab "cron:2:respawn:/usr/sbin/cron"
#
mv /etc/sendmail.cf /etc/_sendmail.cf # prevent sending of old mail
#
# to activate sendmail again after mailqueue review
#
# mv /etc/_sendmail.cf /etc/sendmail.cf
#
#
cfgmgr > $log 2>&1
#
echo creating new ip address + hostname
#
/usr/sbin/mktcpip -h'New_clone' -a'172.24.19Ø.99' -m'255.255.255.Ø' -
i'en1' -n'
172.24.19Ø.19Ø' -d'sony' -g'172.24.19Ø.1Ø' -t'N/A' >> $log 2>&1
#
EOF
#
# end of script custom.ksh
```

Edit the file *bosinst.data*. This file includes the install directives for the base operating system installation steps, and references to the customization script.

For a detailed description of the parameters take a look in the *AIX 4.3.3. Installation Guide*, page 30.

control_flow:

```
CONSOLE = /dev/ttyØ
INSTALL_METHOD = overwrite
PROMPT = yes
EXISTING_SYSTEM_OVERWRITE = yes
INSTALL_X_IF_ADAPTER = yes
RUN_STARTUP = yes
RM_INST_ROOTS = no
ERROR_EXIT =
CUSTOMIZATION_FILE = /../custom.ksh
```

```
TCB = no
INSTALL_TYPE = full
BUNDLES =
SWITCH_TO_PRODUCT_TAPE = yes
RECOVER_DEVICES = yes
BOSINST_DEBUG = no
```

For each disk to install to, you need a *target_disk_data:* stanza.

Edit the *image.data* file. It contains all the logical volumes and filesystems parameters. This is the point where you can switch from a mirrored to an un-mirrored system or change the partition or filesystem sizes.

Useful parameters to change in the *vg_data* stanza are *PPSIZE* (Physical Partition Size) and the *VG_SOURCE_DISK_LIST* if you want to use more or fewer hard disks.

vg_data:

```
VGNAME= rootvg
PPSIZE= 16
VARYON= yes
VG_SOURCE_DISK_LIST= hdiskØ hdisk1
QUORUM= 2
```

Useful parameters to change in the *lv_data* stanza are *LV_SOURCE_DISK_LIST* if you like to mirror, unmirror, or distribute the logical volume, and *COPIES*, *LPs*, and *PP*, which correspond to parameters of the smit mklv screen. There you can access help information.

COPIES, *LPs*, *PP*, and *LV_MIN_LPS* build a relationship. This sample is from a */ lvm hd4* without a mirror.

lv_data:

```
VOLUME_GROUP= rootvg
LV_SOURCE_DISK_LIST= hdiskØ
LV_IDENTIFIER= ØØ129Ø279e81Ø293.4
LOGICAL_VOLUME= hd4
VG_STAT= active/complete
TYPE= jfs
MAX_LPS= 512
COPIES= 1
LPs= 1
```

© 2004. Xephon USA telephone (214) 340 5690, fax (214) 341 7081.

```
STALE_PPs= Ø
INTER_POLICY= minimum
INTRA_POLICY= center
MOUNT_POINT= /
MIRROR_WRITE_CONSISTENCY= on
LV_SEPARATE_PV= yes
PERMISSION= read/write
LV_STATE= opened/syncd
WRITE_VERIFY= off
PP_SIZE= 16
SCHED_POLICY= parallel
PP= 1
BB_POLICY= relocatable
RELOCATABLE= yes
UPPER_BOUND= 32
LABEL= /
MAPFILE=
LV_MIN_LPS= 1
STRIPE_WIDTH=
STRIPE_SIZE=
```

This sample is from a */ lvm hd4* that is mirrored.

lv_data:

```
VOLUME_GROUP= rootvg
LV_SOURCE_DISK_LIST= hdisk1 hdiskØ
LV_IDENTIFIER= ØØ424cbafdafabØf.1
LOGICAL_VOLUME= hd5
VG_STAT= active/complete
TYPE= boot
MAX_LPS= 512
COPIES= 2
LPs= 8
STALE_PPs= Ø
INTER_POLICY= minimum
INTRA_POLICY= edge
MOUNT_POINT=
MIRROR_WRITE_CONSISTENCY= on
LV_SEPARATE_PV= yes
PERMISSION= read/write
LV_STATE= closed/syncd
WRITE_VERIFY= off
PP_SIZE= 16
SCHED_POLICY= parallel
PP= 16
BB_POLICY= relocatable
RELOCATABLE= no
UPPER_BOUND= 32
LABEL=
MAPFILE=
LV_MIN_LPS= 8
STRIPE_WIDTH=
STRIPE_SIZE=
```

When you are finished, back up the content of the */tmp/floppy* directory to a floppy disk:

```
ls ./bosinst.data ./image.data ./custom.ksh ./signature | backup -qv
```

Now create the back-up on the source system.

Remove the network adapter definitions from the odm. This prevents the clone system from starting up with duplicate IP addresses.

You do not need this step if the clone system is to be installed on a network other than the source system.

Here is a sample with en1 as the primary interface:

```
netstat -in
Name  Mtu   Network     Address           Ipkts Ierrs  Opkts Oerrs  Coll
en1   15ØØ  link#2      Ø.4.ac.de.7b.Ø    14415    Ø      62     Ø     Ø
en1   15ØØ  172.25.19Ø  172.24.19Ø.17     14415    Ø      62     Ø     Ø
loØ   16896 link#1                          119    Ø     141     Ø     Ø
loØ   16896 127         127.Ø.Ø.1           119    Ø     141     Ø     Ø
loØ   16896 ::1                             119    Ø     141     Ø     Ø

odmget -q name=en1 CuAt

CuAt:
name = "en1"
attribute = "netaddr"
value = "172.25.19Ø.17"
type = "R"
generic = "DU"
rep = "s"
nls_index = 4

CuAt:
name = "en1"
attribute = "netmask"
value = "255.255.255.Ø"
type = "R"
generic = "DU"
rep = "s"
nls_index = 8
```

```
CuAt:
name = "en1"
attribute = "state"
value = "up"
type = "R"
generic = "DU"
rep = "sl"
nls_index = 5
#
# Save the network configuration - for later restore.

odmget -q name=en1 CuAt > /en1.save

# Delete the network configuration.

odmdelete -o CuAt -q name=en1
Ø518-3Ø7 odmdelete: 3 objects deleted.
```

Take a system back-up from the source system to tape.

After the back-up has finished, restore the configuration on the source system with:

```
odmput /en1.save
```

On the target system, load the appropriate AIX install CD-ROM.

Load the **mksysb** tape.

Insert the newly-created floppy disk.

Boot from the CD-ROM.

If you still have root access you can modify the bootlist with:

```
bootlist -m normal cdØ
```

Verify this with:

```
bootlist -m normal -o
```

Reboot.

If the target system is down, start it up. Go to the system console.

On PCI-Systems enter *1* on the serial console or F1 on the graphic console when the keyboard symbol shows up on the screen.

Define the bootdevice cd0.

Exit from the menu to boot the system from this device.

After booting, the following screen appears on the system console. Make your selection in the following way:

```
******* Please define the System Console. *******

Type a 1 and press Enter to use this terminal as the system console.
```

Type 1 and press *Enter* to have English during the install.

```
To reboot in normal mode, turn the key to normal (if necessary) and
press reset.
>>> 1 Continue with Install
```

You will see:

```
                     Welcome to Base Operating System
                     Installation and Maintenance

Type the number of your choice and press Enter.  Choice is indicated by
>>>.

>>> 1 Start Install Now with Default Settings
   2 Change/Show Installation Settings and Install
   3 Start Maintenance Mode for System Recovery



   88  Help ?
   99  Previous Menu

>>> Choice [1]: 3
```

Choose 3, you then see:

```
                   Maintenance


Type the number of your choice and press Enter.


>>> 1 Access a Root Volume Group

   2 Copy a System Dump to Removable Media
   3 Access Advanced Maintenance Functions

   4 Install from a System Backup




   88  Help ?

   99  Previous Menu


>>> Choice [1]: 4
```

Choose 4, and you get:

```
                   Choose mksysb Device

Type the number of the device containing the system backup to be
installed and press Enter.

     Device Name                    Path Name

>>>   1 tape/scsi/ost                /dev/rmtØ
     2 cdrom/scsi/scsd              /dev/cdØ



   88  Help ?
   99  Previous Menu

>>> Choice [1]: 1
New volume on /dev/rfdØ:
Cluster 9216 bytes (18 blocks).
   Volume number 1
   Date of backup: Mon Feb 16 Ø8:59:Ø3 2ØØ4
   Files backed up by name
   User root
   files archived: 4
\
```

Next is:

```
To reboot in normal mode, turn key to normal (if necessary)
and press reset.

>>> 1 Continue with Install


   88  Help ?
```

```
>>> Choice [1]:
```

Then:

```
                System Backup Installation and Settings

Either type Ø and press Enter to install with the current settings, or
type the
number of the setting you want to change and press Enter.

     Setting:                                 Current Choice(s):

   1 Disk(s) where you want to install ...... hdiskØ...
        Use Maps............................. No
   2 Shrink File Systems..................... No


>>> Ø Install with the settings listed above.



                 +------------------------------------------------------
   88  Help ?    |    WARNING: Base Operating System Installation will
                 |    destroy or impair recovery of ALL data on the
                 |    destination disk hdiskØ.
>>> Choice [Ø]:
```

Finally:

```
                       Installing Base Operating System

If you used the system key to select SERVICE mode,
turn the system key to the NORMAL position any time before the
installation ends.

       Please wait...



       Approximate     Elapsed time
    % tasks complete   (in minutes)


         1               Ø      Making boot logical volume.
```

After the installation/restore process the system reboots and the configuration from the customization script is taken.

The second way, cloning with the help of the NIM environment, will be discussed in a follow-up article.

Imhotep
Unix Systems Administrator (Austria) © Xephon 2004

Displaying memory allocation – part 2

This month we conclude the article looking at displaying memory allocation.

```
#####################################################################
#  Name     : DisplayListOfValues
#  Overview : The function displays a list of values for system
#             processes, and extracts the process id from the
#             selected record.
#  Notes    :
#####################################################################
DisplayListOfValues ( )
{
echo "          List of values for System Processes "  > \
                                                ${TEMP_FILE_1}
echo "          =================================== " >> \
                                                     ${TEMP_FILE_1}
echo "          Select Value by Deleting Line and Saving File\n">> \
                                            ${TEMP_FILE_1}
ps  -eaf  >>  ${TEMP_FILE_1}
cp ${TEMP_FILE_1}  ${TEMP_FILE_2}
view ${TEMP_FILE_1}
SELECTED_PID='diff ${TEMP_FILE_1} ${TEMP_FILE_2} | tail -1 | awk {'print
$3'}'
}
#####################################################################
#  Name     : GetProcessId
#  Overview : The function gets a process id from the user.
#  Notes    :
#####################################################################
GetProcessId ()
{
while  true
do
 clear
 echo "Enter Process Id (l=list of values q=quit):\c"
 read SELECTED_PID
 case  $SELECTED_PID  in
    l|L ) DisplayListOfValues ;
          if [ "${SELECTED_PID}" = "" ]
          then
                   : ;
          else
                 break ;
          fi ;;

    ""  ) DisplayMessage E "${INVALID_ENTRY}" ;;
      q ) return $FALSE  ;;

     *  ) break ;;
 esac
done
#
}
######################################################################
#   Name     : BuildChildProcessTree
#   Overview : The function establishes all the child proceses for
#              the selected process id.
#   Notes    :
######################################################################
BuildChildProcessTree ()
{
ListAllProcesses
# retrieve command for selcted process id
#
COMM=""
COMM='grep "|pid=${SELECTED_PID}|" ${PROC_FILE} | grep -v "grep" |cut -
d'|' -f4 | cut -c6-'
#
if [ "${COMM}"  = "" ]
then
    DisplayMessage  E "${INVALID_PID}" N
    return $FALSE
fi
# populate first element in array with selected pid details
#
# determine rss and vsz for selected pid
#
RSSIKB='ps -o rss  -p $SELECTED_PID | tail -1'
VSZIKB='ps -o vsz  -p $SELECTED_PID | tail -1'
#
export CH_INDEX=Ø                            #  initialize to Ø
export CHILD_HIERARCHY[$CH_INDEX]="Pid=${SELECTED_PID}|rss=${RSSIKB}|
                                          vsz=${VSZIKB}|Command=${COMM}"
#
# initialize required variables before invoking  GetAllChildProcessIds
()
```

© 2004. Xephon USA telephone (214) 340 5690, fax (214) 341 7081.

```
#
export LEVEL=1
SEARCH_PID=${SELECTED_PID}
#
GetAllChildProcessIds
#
}
######################################################################
#   Name     : CalculateMemoryAllocatedForAllProcesses ()
#   Overview : The function calculates memory allocation for all
#              established processes (ie the parent and all its
#              child processes)
#   Notes    : 1. The function requires root access to execute the
#                 bootinfo command.
######################################################################
CalculateMemoryAllocatedForAllProcesses ()
{
# retrieve amount ( in MB) for   total physical memory
#
TPMIM='su -c "bootinfo -r"'
# total physical memory in bytes
#
TPMIB='expr ${TPMIM}  \* 1Ø24ØØØ'
# initialize total physicaL memory allocated to this process
#
TPMA=Ø
#initialize total virtual memory allocated to this process
#
TVMA=Ø
# timestamp this snapshot
#
DATETIME='date "+%d/%m/%Y at %H:%M:%S"'
REPORT_LINE="${DATETIME}\n"
echo "${REPORT_LINE}"  >> ${REPORT_FILE}
#
CH_INDEX=Ø
#
while [ "${CHILD_HIERARCHY[$CH_INDEX]}" !=  ""  ]
do
  LEVEL_IND='echo ${CHILD_HIERARCHY[$CH_INDEX]} | cut -d'|' -f1 | cut -
d'=' -f1'
  PID='echo ${CHILD_HIERARCHY[$CH_INDEX]}       | cut -d'|' -f1 | cut -
d'=' -f2'
  RSSIKB='echo ${CHILD_HIERARCHY[$CH_INDEX]}    | cut -d'|' -f2 | cut -
d'=' -f2'
  VSZIKB='echo ${CHILD_HIERARCHY[$CH_INDEX]}    | cut -d'|' -f3 | cut -
d'=' -f2'
  COMM='echo ${CHILD_HIERARCHY[$CH_INDEX]}      | cut -d'|' -f4 | cut -
c9-'
  PROG_NAME='echo ${CHILD_HIERARCHY[$CH_INDEX]} | cut -d'|' -f4 | cut -
```

```
c9- | awk {'print $1'}'
  #
  # resident set size in bytes
  #
  RSSIB='bc  <<!
 ( ${RSSIKB} * 1Ø24 )
!'
  # virtual size in bytes
  #
  VSZIB='bc  <<!
 ( ${VSZIKB} * 1Ø24 )
!'
  # calculate running total memory allocated
  # if the same program is invoked more than once
  # for the second invocation, subtract the text size from rss
  #
  if  grep "${PROG_NAME}"  ${REPORT_FILE}  > /dev/null 2>&1
  then
      # get the text  size in bytes  for this program
      #
      TSZIB='size ${PROG_NAME} | awk {'print $1'}'
      #
      # establish data size in bytes
      #
      DSZIB='expr $RSSIB - $TSZIB'
      #
      #
      TPMA='bc <<!
   $TPMA + ${DSZIB}
!'
 else
  #
  TPMA='bc <<!
   $TPMA + ${RSSIB}
!'
  #
fi
  #
  TVMA='bc <<!
   $TVMA + ${VSZIB}
!'
  #
  REPORT_LINE="${LEVEL_IND}=${PID} Rmem=${RSSIB} Vmem=${VSZIB}
Command=${COMM}"
  FormatReportLine "${REPORT_LINE}"
  echo "${REPORT_LINE}"  >> ${REPORT_FILE}
  CH_INDEX='expr ${CH_INDEX} + 1'
done
# percentage of memory allocated
#
```

```
PMA='bc <<!
scale=4
($TPMA * 100 )/${TPMIB}
!'
#
echo "\nTotal Physical Memory =${TPMIB}"  >> ${REPORT_FILE}
echo "Total Memory allocated  =${TPMA}"    >> ${REPORT_FILE}
echo "Percentage of Total Memory allocated=${PMA}%\n"    >>
${REPORT_FILE}
}
####################################################################
#  Name      : ListAllProcesses
#  Overview  : The function lists all the procesess into $PROC_FILE
#              file.
#  Notes     :
###################################################################
ListAllProcesses ( )
{
DisplayMessage I "${FORMATTING_PROCESS_IDS}"  N
#
> ${PROC_FILE}
# get all the process details minus the header
#
ps  -eaf | grep -v "UID"  > ${TEMP_FILE_1}
#
cat ${TEMP_FILE_1} |  while  read  LINE
do
   # format uid, pid and ppid
   #
   FORMATTED_PS_OUTPUT='echo ${LINE} |  awk  {'print
"uid="$1"|pid="$2"|ppid="$3'}'
   # add the command to the formatted output
   #
   TIMESTAMP='echo "${LINE}" | awk {'print $5'}'
   if echo "${TIMESTAMP}" |  grep ":"  1> /dev/null
   then
       COMMAND='echo "${LINE}" | awk {'print $8" "$9" "$10" "$11'}'
   else
       COMMAND='echo "${LINE}" | awk {'print $9" "$10" "$11" "$12'}'
   fi
   #
   FORMATTED_PS_OUTPUT="${FORMATTED_PS_OUTPUT}|comm=${COMMAND}"
   # store formatted output in the file
   #
   echo "${FORMATTED_PS_OUTPUT}"  >> $PROC_FILE
done
#
}
####################################################################
#  Name      : GetAllChildProcessIds
#  Overview  : The function retrieves all child process ids for the
#              selected process id.
#  Notes     : 1. The function calls itself recursively in order to
#                 build child hierarchy (ie parent, child, grand
#                 child etc).
#              2. Recursive calls are made within the current shell.
#####################################################################
GetAllChildProcessIds ()
{
export INDEX=Ø
export CHILD_FILE="/tmp/scamp_ptl_$LEVEL.tmp"  #  output file for
current level
# select all processes  spanwed by this process
#
cat ${PROC_FILE} | grep  "ppid=${SEARCH_PID}"  | grep  -v "grep"    >
${CHILD_FILE}  2>&1
#
cat  ${CHILD_FILE}  |  while  read  LINE
do
    SEARCH_PID='echo "${LINE}" |  cut -d'|' -f2 | cut -d'=' -f2'
    #
    COM='echo "${LINE}" |  cut -d'|' -f4 | cut -c6-'
    RSSIKB='ps vg |  grep "${SEARCH_PID}" | grep -v "grep" | awk {'print
$7'}'
    if [ "${RSSIKB}" = ""  ]
    then
        # process disappeared
        #
          continue
    fi
    #
    INDENT=
    while [  $INDEX  -lt  $LEVEL  ]
    do
        INDENT="$INDENT....."
        INDEX='expr $INDEX + 1'
    done
    CH_INDEX='expr $CH_INDEX + 1'
    LEVEL='expr $LEVEL + 1'
    VSZIKB='ps -o vsz -p $SEARCH_PID | tail -1'
   CHILD_HIERARCHY[$CH_INDEX]="${INDENT}Pid=${SEARCH_PID}|rss=${RSSIKB}|
                                          vsz=${VSZIKB}|Command=$COM"
    #
    GetAllChildProcessIds
done
# finished processing current level
# return to previous level
#
LEVEL='expr $LEVEL - 1'
return $TRUE
```

```
}
######################################################################
#   Name     : ProcessExit
#   Overview : The function removes any temporary files and makes
#              a graceful exit.
#   Input    : Exit Code
#   Notes    :
######################################################################
ProcessExit ()
{
EXIT_CODE="$1"
clear
#
rm -f  ${TEMP_FILE_1}
rm -f  ${TEMP_FILE_2}
rm -f  ${TEMP_FILE_3}
#
rm -f ${PROC_FILE}
# remove all child process ids file
#
rm -f /tmp/scamp*.tmp
#
exit ${EXIT_CODE}
}
######################################################################
#   Name     : Displayhelp
#   Overview : The function displays help for the script
#   Notes    :
######################################################################
DisplayHelp  ()
{
#
cat  <<! >  ${TEMP_FILE_1}
scamp stands for show current memory allocation for a process
The script runs in interactive mode but can be run at the background
using & operator

For the selected process and all its child processes
the script displays rss (the amount of real memory in bytes,  a process
has acquired)
                    vsz (the amount of total memory ( real + virtual) in
bytes, a process is allocated

Total  amount of real memory in the system
Total  amount of real memory allocated to selected and all its  child
processes
Percentage of total amount of real memory allocated to selected and all
its child processes

The script accepts a process id either in interactive or batch mode and
```

© 2004. Reproduction prohibited. Please inform Xephon of any infringement.

```
then shows current allocation of real memory in bytes to this process
and all of its child's processes recursively.

The options to the script are as follows:

ns=no of samples (how many times the snapshot will be taken)
   any positive integer
   optional
   defaults to 1

ti=time interval in seconds
   any posive integer
   optional
   defaults to 1  second
   only meaningful if ns > 1

help=describes this help
     accepted value is Y or y

pid=process id for a given process
    mandatory for background execution


Examples
========
scamp.ksh                      executes once in interactive mode
scamp.ksh help=y               display help
scamp.ksh ti=2                 executes once in interactive mode
scamp.ksh ns=4                 executes 4 times with time interval of 1
                                              second in interactive mode
scamp.ksh ns=4   ti=2          executes 4 times with time interval Øf 2
                                             seconds in interactive mode
scamp.ksh ns=4   ti=6Ø  pid=3456 &   executes 4 times with time
                               interval Øf 6Ø seconds at the background


!
more  ${TEMP_FILE_1}
}
######################################################################
#   Name     : ViewReport
#   Overview : The function displays the report file and gives an
#              option to print the report.
#   Notes    :
######################################################################
ViewReport  ()
{
#
view ${REPORT_FILE}
PrintReport
```

© 2004. Xephon USA telephone (214) 340 5690, fax (214) 341 7081.

```
}
#####################################################################
#  Name     : ArgumentSpecifierValid
#  Overview : The function performs a syntax check for a specific
#             argument whose value must be specified using =
#             operator.
#  Input    : An argument (string)
#  Returns  : $TRUE
#             $FALSE if argument specifier (=) not found
#  Notes    :
#####################################################################
ArgumentSpecifierValid ()
{
# assign parameter
#
P_ARG="$1"
#
ARG_LEN='echo "${P_ARG}\c" | wc  -c'
#
FROM_POS=1
TO_POS=1
#
while  true
do
  NEXTCHAR='echo "${P_ARG}" | cut -c${FROM_POS}-${TO_POS}'
  if [  "${NEXTCHAR}" = "="  ]
  then
     if [ $FROM_POS -eq 1 ]
     then
         return $FALSE
     else
         return $TRUE
     fi
  fi
  #
  if  [  $TO_POS -eq  $ARG_LEN ]
  then
       return $FALSE
  fi
  #
  FROM_POS='expr $FROM_POS + 1'
  TO_POS='expr $TO_POS + 1'
done
#
}
#####################################################################
#  Name     : ParseCommandLine
#  Overview : The function parses the command line for the following
#             parameters:
#                           o NS
#                           o TI
#                           o HELP
#                           o LOG
#                           o PID
#  Returns  : $TRUE
#             $FALSE if arguments are incorrectly supplied.
#  Notes    : 1. The parameters in the command line are specified as
#                follows:
#                 HELP=Y  LOG=memory.dat  NC=1Ø TI=2
#              2. The values of following parameter are assigned
#                  here:
#                   LOG     --------> P_LOG
#                     NS    --------> P_NS
#                     TI    --------> P_TI
#                   HELP    --------> P_HELP
#                   PID     --------> P_PID
#             3. The function calls the following function:
#                       o  ArgumentSpecifierValid
#####################################################################
ParseCommandLine ()
{
#
P_LOG=""
P_NS=""
export P_TI=""
P_HELP=""
P_PID=""
#
if [ $ARGC -eq Ø -a "${MODE}" =   ""   ]
then
   # no arguments have been suppiled
   #
   P_LOG=/tmp/${REPORT_FILE}
   P_NS=1
   P_TI=1
   P_HELP=""
   P_PID=""
   return $TRUE
fi
# process argument line (eg  NS = 3  LOG=memalloc.log)
# get rid off unwanted white spaces
#
ARGL='echo "${ARGL}" | sed -e "s, *= *,=,g"'
# append end of line argument to safeguard against
# single parameter being put in the command line, such as alfa.ksh LOG
#
ARGL="${ARGL} EOL=YES"
#
INDEX=1
ARG_ARRAY[$INDEX]=""
```

© 2004. Xephon USA telephone (214) 340 5690, fax (214) 341 7081.

```
# extract and store argument details from command line
#
while true
do
  #
  ARG='echo "${ARGL}" | cut -d' ' -f${INDEX}'
  #
  if [ "${ARG}" =  ""  ]
  then
        break
  fi
  #
  ARG_ARRAY[$INDEX]="${ARG}"
  #
  INDEX='expr $INDEX + 1'
  #
done
# extract and process argument type
#
INDEX=1
while [ "${ARG_ARRAY[$INDEX]}"  != ""  ]
do
  ARG="${ARG_ARRAY[$INDEX]}"
  #
  ARG_TYPE='echo ${ARG_ARRAY[$INDEX]} | cut -d'=' -f1'
  ARG_TYPE_UPPER='echo ${ARG_ARRAY[$INDEX]} | cut -d'=' -f1 | tr '[a-z]'
'[A-Z]''
  #
  case  "${ARG_TYPE_UPPER}"  in
       LOG ) if ! ArgumentSpecifierValid  "${ARG}"
             then
                 DisplayMessage  E "${INVALID_ARG_SPECIFIER}"  N ;
                 return $FALSE ;
              fi ;
              #
              if [ "${P_LOG}" != "" ]
              then
                     DisplayMessage  E "${DUPLICATE_ARG_TYPE}"  N ;
                     return $FALSE ;
              fi ;
              #
              P_LOG='echo ${ARG_ARRAY[$INDEX]} | cut -d'=' -f2' ;;

       HELP)  if ! ArgumentSpecifierValid  "${ARG}"
              then
                  DisplayMessage  E "${INVALID_ARG_SPECIFIER}"  N ;
                  return $FALSE ;
              fi ;
              #
              if [ "${P_HELP}" != "" ]
              then
                     DisplayMessage  E "${DUPLICATE_ARG_TYPE}"  N ;
                     return $FALSE ;
              fi ;
              #
              P_HELP='echo ${ARG_ARRAY[$INDEX]} | cut -d'=' -f2 | tr
'[a-z]' '[A-Z]'' ;
              #
              if [ "${P_HELP}" != "Y" -a "${P_HELP}" !=  "N"  ]
              then
                  DisplayMessage  E "${INVALID_VALUE}"  N ;
                  return $FALSE ;
              fi ;;

        NS)  if ! ArgumentSpecifierValid  "${ARG}"
              then
                  DisplayMessage  E "${INVALID_ARG_SPECIFIER}"  N ;
                  return $FALSE ;
             fi ;
             #
             if [ "${P_NS}" != "" ]
             then
                     DisplayMessage  E "${DUPLICATE_ARG_TYPE}"  N ;
                     return $FALSE ;
             fi ;
             #
             P_NS='echo ${ARG_ARRAY[$INDEX]} | cut -d'=' -f2 | tr '[a-
z]' '[A-Z]'' ;
             #
             if ( [ 'expr ${P_NS} + Ø' -eq ${P_NS} ] ) >/dev/null  2>&1
             then
                if [ $P_NS  -lt 1 ]
                then
                     DisplayMessage  E "${POS_INT_REQ}"  N ;
                     return $FALSE ;
                fi ;

             else
                     DisplayMessage  E "${POS_INT_REQ}"  N ;
                     return $FALSE ;
             fi ;;

       TI)   if ! ArgumentSpecifierValid  "${ARG}"
             then
                  DisplayMessage  E "${INVALID_ARG_SPECIFIER}"  N ;
                  return $FALSE ;
             fi ;
             #
             if [ "${P_TI}" != "" ]
             then
```

© 2004. Xephon USA telephone (214) 340 5690, fax (214) 341 7081.

```
                     DisplayMessage  E "${DUPLICATE_ARG_TYPE}"  N ;
                     return $FALSE ;
             fi ;
             #
             P_TI='echo ${ARG_ARRAY[$INDEX]} | cut -d'=' -f2 | tr '[a-
z]' '[A-Z]'' ;
             #
             if ( [ 'expr ${P_TI} + Ø' -eq ${P_TI} ] ) >/dev/null  2>&1
             then
                if [ $P_TI  -lt 1 ]
                then
                     DisplayMessage  E "${POS_INT_REQ}"  N ;
                     return $FALSE ;
                fi ;

             else
                     DisplayMessage  E "${POS_INT_REQ}"  N ;
                     return $FALSE ;
             fi ;;

     PID)   if ! ArgumentSpecifierValid  "${ARG}"
            then
                  DisplayMessage  E "${INVALID_ARG_SPECIFIER}"  N ;
                  return $FALSE ;
             fi ;
             #
             if [ "${P_PID}" != "" ]
             then
                     DisplayMessage  E "${DUPLICATE_ARG_TYPE}"  N ;
                     return $FALSE ;
             fi ;
             #
             P_PID='echo ${ARG_ARRAY[$INDEX]} | cut -d'=' -f2 | tr '[a-
z]' '[A-Z]'' ;;
             #

      EOL ) : ;;

       *  ) DisplayMessage  E "${INVALID_ARG_TYPE}" N ;
            return $FALSE ;;
  esac
  #
  INDEX='expr $INDEX + 1'
done
# remove EOL argument from the command line
#
ARGL='echo "${ARGL}" | sed s/EOL.*//'
# default parameters
#
if [  "${P_NS}" = ""  ]
```

```
then
    P_NS=1
fi
#
if [  "${P_TI}" = ""  ]
then
    P_TI=1
fi
#
if [ "${MODE}" = "B"  -a "${P_PID}" = "" ]
then
     DisplayMessage E "${NULL_PID}" N
     return $FALSE
else
    SELECTED_PID="${P_PID}"
fi
#
if [ "${P_LOG}" != ""  ]
then
    REPORT_FILE="${P_LOG}"
fi
return $TRUE
}
#####################################################################
#  Name     : HandleTerminalIoInBackgroundInterrupt
#  Overview : The function starts an instance of the script at the
#              background using the & operator.
#  Returns  :
#  Notes    :
#####################################################################
HandleTerminalIoInBackgroundInterrupt ()
{
# job started at the background by the user
# re-start another instance of the script at the background
#
export MODE="B"
nohup  scamp.ksh  ${ARGL}  &
#
exit Ø
}
#####################################################################
#  Name     : PerformTestForBackGroundJob
#  Overview : The function establishes whether or not the user
#             executed the script at the background.
#  Returns  :
#  Notes    : 1. It performs that test by trying to read from the
#                stdio. If the script was at the background, a signal
#                is raised and the function
#                HandleTerminalIoInBackgroundInterrupt () is invoked
#####################################################################
```

```
PerformTestForBackGroundJob ()
{
if [ "${MODE}"  !=  "B"  ]
then
  echo "Press Enter..."
  read DUMMY
fi
}
#####################################################################
#   Name     : main
#   Overview : The function invokes all the other functions.
#   Notes    : 1. The function invokes the following functions:
#                        o InitialiseVariables
#                        o ParseCommandLine
#                        o InitialiseReportFile
#                        o PerformTestForBackGroundJob
#                        o GetProcessId
#                        o BuildChildProcessTree
#                        o ViewReport
#####################################################################
main ()
{
InitialiseVariables
#
if ! ParseCommandLine
then
   ProcessExit $FEC
fi
#
PerformTestForBackGroundJob
#
DisplayMessage I "${WORKING}" N
#
if [ "${P_HELP}" = "Y"  ]
then
     DisplayHelp
     ProcessExit $SEC
fi
#
if [ "${MODE}" != "B"  ]
then
   if ! GetProcessId
   then
      ProcessExit $FEC
   fi
fi
#
InitialiseReportFile
#
export NS=$P_NS
#
while [ $NS -gt Ø ]
do
   if ! BuildChildProcessTree
   then
      ProcessExit $FEC
   fi
   #
   CalculateMemoryAllocatedForAllProcesses
   #
   NS='expr  $NS  - 1'
   #
   #sleep for time interval
   sleep ${P_TI}
   #
done
#
if [ "${MODE}" != "B"  ]
then
   ViewReport
fi
#
ProcessExit $SEC

}
# define trap
#
trap "HandleInterrupt "  1  2  15
# define trap to trap read while job is running at the background
#
trap "HandleTerminalIoInBackgroundInterrupt "  TTIN
# package the comand line
#
ARGL="$@"
ARGC="$#"
# perform background job test if the job has not been started at the
# background by this script
#
# invoke main
#
main
```

SAMPLE REPORT

```
Memory Allocation (in bytes) for Pid(11613) and Child Processes
===============================================================
Mem=Ø indicates that the process is sleeping and not resident in memory
Rmem= indicates the amount of real memory the process has acquired
Vmem= indicates the amount of total (real + virtual ) memory the process
```

```
has acquired"
Total memory allocation is adjusted for multiple invocation of same
program by adding the memory allocation for text segment once for that
program


Ø8/Ø2/2ØØ4 at 11:51:11

Pid=11613 Rmem=1163264 Vmem=1851392 Command=ksh do_batch.ksh
.....Pid=11614 Rmem=1163264 Vmem=1851392 Command=process_batch
..........Pid=11615 Rmem=1163264 Vmem=1851392 Command=job_batchØØ1

Total Physical Memory =2Ø97152ØØØ
Total Memory allocated  =3489792
Percentage of Total Memory allocated=.1664%
```

Arif Zaman
DBA/Developer (UK)

© Xephon 2004

AIX news

MetiLinx has announced MetiLinx Connector for Microsoft Operations Manager (MOM). The product extends MOM by enabling it to monitor and automate heterogeneous server environments, including AIX, Linux, Solaris, HP-UX, and Windows NT, as well as Windows 2000 and Windows Server 2003.

The MetiLinx Connector for MOM helps customers reduce the complexity of managing IT systems by extending the manageability inherent in the Windows platform to heterogeneous environments, the company says.

For further information contact:
MetiLinx, 999 Baker Way, Suite 410, San Mateo, CA 94404, USA.
Tel: (650) 292 9200.
URL: http://www.metilinx.com/company/USpress03_13.htm.

* * *

XOsoft has announced that it will work with IBM to develop real-time software migration tools for the IBM TotalStorage SAN File System on AIX. These tools will help customers migrate data without taking the servers offline or rebooting systems, helping reduce downtime and lost data.

XOsoft's WANSync will provide IBM with a non-disruptive data migration feature through its soft-installation technology, which allows administrators to deploy the software transparently.

XOsoft will port WANSync to IBM's AIX operating environment, optimizing the software to support the IBM TotalStorage SAN File System. In addition to using WANSync for data migration, IBM's customers will have the ability to activate WANSync for disaster recovery and high availability solutions for their mission critical servers.

WANSync's Rewind technology allows administrators to instantly roll back to the actual event before corruption took place, avoiding data loss entirely.

For further information contact:
Xosoft, 35 Corporate Drive, Suite 400, Burlington, MA 01803, USA.
Tel: (781) 685 4965.
URL: http://www.xosoft.com/press/f_new9.htm.

* * *

Watercloud at Xfocus has reported some vulnerabilities in AIX 4.3.3, which can be exploited by malicious, local users to gain escalated privileges on a vulnerable system.

A boundary error exists within the make utility when handling the argument for the 'CC' option, which is used for specifying the compiler that should be used for C compilations. This can be exploited to cause a buffer overflow by supplying an overly long string. Successful exploitation may grant a malicious, local user root group privileges.

Boundary errors exist within the getlvcb and putlvcb utilities when handling command line options. This can be exploited to cause a buffer overflow by passing an overly long command line option to either of the utilities. Successful exploitation may grant root user privileges to a malicious, local user but requires that the user already has root group privileges (eg by exploiting the first vulnerability).

Their solution is to grant only trusted users access to affected systems.

For further information contact:
URL: http://www.xfocus.org/exploits/200403/31.html.

xephon