In this issue

3 Useful tools in disaster recovery testing
10 Those missing sar records
14 Monitoring the availability of networked systems
31 Curses programming – part 2
49 AIX news

© Xephon Inc 2004
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.
Useful tools in disaster recovery testing

The following three scripts are useful during disaster recovery testing on AIX machines. We were using them with EMC technology, such as SRDF (Symmetrix Remote Data Facility), and Timefinder technology. Let me remind you about these technologies. The EMC SRDF product is used to mirror data from a primary site to a remote site. The disks at the primary site are referred to as R1, and the disks at the remote site are referred to as R2. This solution is best if you have heterogeneous servers such as AIX, Sun, DecUnix, etc. The distance between the primary site and the remote site depends on the technology chosen. In our environment we use Cisco’s DWDM (Dense Wavelength Division Multiplexing) and we go up to 100km. Timefinder is another product from EMC that is useful during disaster recovery testing. It has a set of disks called BCV (Business Continuity Volumes). These disks can be attached to and detached from the R2 disks at the remote site. Business Continuity Volumes can also be used as point-in-time back-ups of the primary data centre. During disaster recovery testing, we detach these disks from R2 and mount them at the remote server. Users may bring up their applications from these disks and can do testing. However, R2 disks at the remote site are still being refreshed from the primary site. So even if a disaster happens at the primary site while disaster recovery testing is going on, we are still protected because the primary site data is being replicated to R2. During testing, a user might have corrupted the data on the BCV volumes. However, we can still bring up the application at the remote site from the R2 disks to the point when the disaster happened.

Three scripts written in Perl will perform a smooth test. These scripts use EMC solution enabler Version 5.2, along with Symmetric SRDF and Timefinder licence enabled.

Before using the Chk_bcv_srdf.pl, Splitbcv.pl, and Syncbcv.pl programs, we have to set up all R2 and BCV devices in /usr/ecc/srdf/symmalldisks and we have to create a single srdf group,
using the \texttt{symdg} command, and add all the disks in this group using the \texttt{symld} command. After creating the group, we have to make a relationship between R2 and BCV using the \texttt{symbcv} and \texttt{symmir} commands.

The format of the \texttt{/usr/ecc/srdf/symmalldisks} file would be:

\begin{verbatim}
0264  0522
00c8  0198
\end{verbatim}

The first four characters are R2 device addresses and the next four characters are BCV device addresses.

Program \texttt{chk_bcv_srdf.pl} monitors the status of all the R2 and BCV volumes. If hardware failure happens at the drive level, the status of the corresponding R2 and BCV disks will change it to \textquote{invalid}. We may schedule this script in cron to monitor every 5 or 10 minutes, so that the system administrator or storage administrator will be notified in a timely manner and will be able to take corrective action as soon as possible.

All three scripts are designed to send an e-mail or pager message to the person concerned.

Before we detach BCV volumes from R2 for our disaster recovery testing, we have to ensure that all the BCV volumes are fully synchronized with R2. Without checking this, if we split BCV from R2 while it is synchronizing, there is a high probability of having a corrupted file system in the Unix machine. Program \texttt{splitbcv.pl} will make sure that all BCV volumes are fully synchronized with R2; if not, it will wait until all BCV devices synchronize with R2. If there is a problem while/after splitting, it will notify the appropriate person by e-mail or by sending a page.

When we attach BCV disks to R2 at the remote site, we have to make sure that the R2 disks at the remote site are synchronized with the R1 disks at the primary site. If not, and if disaster strikes at the primary site, there won\’t be any usable data at the remote site. Program \texttt{syncbcv.pl} will make sure that all the R2 volumes are fully synchronized with R1. If they are not, it will wait until all R2 disks synchronize with R1 and will notify the appropriate person by e-mail or by sending a page if there is any problem.
#!/usr/bin/perl

use Shell;

##### VARS
$host='hostname';
chomp $host;

##### SUBS
sub notify
{
    system("telalertw STORSYS "$msg "");
    $d = 'date';
    chomp $d;
    print "$d : $msg\n";
}

sub emailstorage
{
    system("/bin/mailx -s "$msg "sysadmin@abcd.org </usr/ecc/srdf/email");
}

sub qrdf
{
    my @query='/usr/symcli/bin/symrdf list -R2';
    $n = @query;
    $n = $n - 13; # To eliminate last 13 lines
    for ( $i = 1; $i <= $n ; $i++ ) {
        # To Eliminate starting messages
        ($symdev,$rdev,$rdftyp,$sa,$ra,$lnk,$mda,$R1invtracks,$R2invtracks,$dev,$rdev,$pair)=
            split(' ',@query[$i] );
        # Check for in Invalid status of any of the EMC disk
        if ( ( $pair eq "Synchronized" ) ||
             ( $pair eq "SyncInProg" ) ||
             ( $pair eq "Partitioned" ) ||
             ( $pair eq "Split" ) ) &&
            ( $lnk eq "RW" )
        {
            print "RDF status for Symdev : $symdev is $pair\n";
        } else
        {
            my @query1='/usr/symcli/bin/symdev show $symdev';
            ( $a,$devgrp)=split(':',@query1[9] );
            if ( length($devgrp) < 6 ) { $devgrp = "MetaBody";
            } else
            {
                my @query2='/usr/symcli/bin/symdev show $symdev';
                ( $a,$devgrp)=split(':',@query2[9] );
                if ( length($devgrp) < 6 ) { $devgrp = "MetaBody";
                } else
                {
                    
            }
```perl
sub qbcv
{
  my @query2 = '/usr/symcli/bin/symmir -f /usr/ecc/srdf/symmalldisks query -sid 148';
  $n = @query2;
  $n = $n - 11;                  # To Eliminate last 13 lines
  for ( $i = 12; $i <= $n ; $i++ ) {
    ($stdlgcl,$r2dev,$stdinvtrks,$bcvlgcl,$bcvdev,$bcvintrks,$status) = split(' ',@query2[$i]);
    if  ( $status eq "Invalid" )
    {
      my @query3 = '/usr/symcli/bin/symdev show $bcvdev';
      ($a,$devgrp)=split(':'.$query3[9] );
      $msg = "Symm Dev $bcvdev on Group $devgrp Status is  $status";
      notify;
      emailstorage;
    }
  }
}

##### MAIN
qrdf;
qbcv;

SPLITBCV.PL
#!/usr/bin/perl
#
use Shell;
##### VARS
$fileid = "/usr/ecc/srdf/symmalldisks";
##### SUBS
sub notify
{
  system("telalertw STORSYS "$msg ");
  $d = 'date';
  print "$d : $msg\n";
}
sub emailstorage
{

```
sub idle {
    my $sleep='sleep $sleeptime'
}

sub splitbcv {
    Label1:
    my @query1='/usr/symcli/bin/symmir -f $fileid -sid 148 query';
    $n = @query1;
    $n = $n - 11; # To eliminate last 11 lines
    for ( $i = 12; $i <= n; $i++ ) {
        ($stdlgcl,$r2dev,$stdinvtrks,$bcvlgcl,$bcvdev,$bcvintrks,$status) = split(' ',query1[$i]);
        if ( $status eq "Synchronized") { ; } 
        else {
            $sleeptime=120;
            idle;
            goto Label1;
        }
    }
    my @groupspli='/usr/symcli/bin/symmir -$f $fileid split -consistent -sid 148 -noprompt';
    # Check the BCV split status for every device
    #
    my @query1='/usr/symcli/bin/symmir -f $fileid -sid 148 query';
    $n = @query1;
    $n = $n - 11; # To eliminate last 11 lines
    for ( $i = 12; $i <= n; $i++ ) {
        ($stdlgcl,$r2dev,$stdinvtrks,$bcvlgcl,$bcvdev,$bcvintrks,$status) = split(' ',query1[$i]);
        if ($status eq 'Split') { ; }
        else {
            @test1='/usr/symcli/bin/symmir -$f $fileid query -sid 148 | mailx -s'ALERT:BCVs did NOT split - Problem-EOM' sysadmin\@abcd.org';
            $msg = "RED ALERT:-BCVs did not split at BCC-EMC";
            notify;
            exit();
        }
    }
    @test1='/usr/symcli/bin/symmir -$f symmalldisks query -sid 148 | mailx -s'ALERT:BCVs did NOT split - Problem-EOM' sysadmin\@abcd.org';
    $msg = "BCVs split with Consistent at Remote - EMC";
    notify;
/usr/bin/perl
#
use Shell;

### VARS
$fileid = "/usr/ecc/srdf/symmalldisks";
$groupid = "symmalldisks"

### SUBS
sub notify
{
    system("telalertw STORSYS "$msg \n"");
    $d = 'date';
    print "$d : $msg\n";
}
sub emailstorage
{
    system("/bin/mailx -s "$msg \n" sysadmin@abcd.org </usr/ecc/srdf/email");
}
sub idle
{
    my @sleep='sleep $sleeptime'
}
sub syncbcv
{
    my @query1="/usr/symcli/bin/symrdf -g $groupid query";
    $n = @query1;
    $n = $n - 12;          # To eliminate last 13 information lines
    ($stdlgldev,$R2dev,$R2stat,$R1invtrks,$R2invtrks,$lnk,$R1dev,$R1stat,$R1invtrks,
    $R2invtrks,$mda,$rdfpair)=split(' ',@query1[17]);
    print "R2 device is $R2dev mda is $mda\n";
    if ( ( $mda eq "C.D" ) )
    {
        my @cmd1="/usr/symcli/bin/symrdf -g $groupid set mode sync
-noprompt";
        $sleeptime = 120;
        ###Wait until all disks synchronize with R1
        idle;
    } else 
    {
        $msg = "Disks are not in Adoptive Copy Mode-Check it -
        syncbcv.pl";
        notify;
    
}
my @query2='/usr/symcli/bin/symrdf -g $groupid query ';
$n = @query2;
$n = $n - 12;          # To elminate last 12 lines
for ( $i = 17; $i<=n; $i++ ) {

  ($symdev,$rdev,$rdftyp,$sa,$ra,$lnk,$mda,$R1invtracks,$R2invtracks,$dev,$rdev,$pair)=
split(' ',@query2[$i] );

  if ( $pair ne "Synchronized" ) {
    if ( $pair ne "SyncInProg" ) {
      $msg = "Device $symdev on BCC-DMX status is $pair -
      Check it out";
      notify;
    }
  }
}

Label1:              # Check all R2 device in Group sync with R1
my @query2='/usr/symcli/bin/symrdf -g $groupid query ';
$n = @query2;
$n = $n - 12;          # To elminate last 12 lines
for ( $i = 17; $i <= n; $i++ ) {

  ($symdev,$rdev,$rdftyp,$sa,$ra,$lnk,$mda,$R1invtracks,$R2invtracks,$dev,$rdev,$pair)=
split(' ',@query2[$i] );

  if ( $pair eq "Synchronized" ) { ; }
  else { $sleeptime = 120; idle; goto Label1; }
}

my @bcvsync='/usr/symcli/bin/symmir -f /usr/ecc/srdf/
symallidisks establish -sid 148 < /usr/ecc/srdf/yes';
$msg = "BCVs sync started on 148";
notify;
emailstorage;
exit;

##### MAIN
syncbcv;

K Muthukumar
Vector Consulting (USA)  © Xephon 2004
Those missing sar records

Performance monitoring is one of the most arduous tasks undertaken by a system administrator. The Unix-supplied tool sar is really helpful in initially diagnosing bottlenecks within a system (as well as being useful in producing pretty graphs for all those non-technical managers).

However, there is one slight problem with sar – when a system is rebooted, no sa1 data records will be written until the next time the entry in cron comes to be executed (usually on the hour). The following Perl script attempts to correct this shortcoming by using at to schedule the sa1 command to write the corresponding sar data after the reboot. It works out how regularly the data should be written and when, up until the next cron entry is executed. All that is required is for the script to be added to one of the start-up routines.

If you are using sar, don’t forget to uncomment the lines in the /etc/rc file.

```
RESTART_SAR.PL
#!/usr/bin/perl -w
##########################################################
# Perl script      :   restart_sar                      #
#                                                        #
# Purpose : To restart the sa1 program, if required,     #
#           after a reboot or similar                    #
##########################################################
$|=1;
use strict;
my ($lsec,$lmin,$lhr,$lmday,$lmon,$lyr,$today,$lyday,$lisdst)=localtime(time());
my $temp;
my $nextmin;
my $attime;
my $line;
my $cronline;
my $remainingintervals;
my $file;
my @days;
```
my @hours;
my @mins;
my @crontabs;
my $tomorrow="";
opendir(CD,"/var/spool/cron/crontabs");
while (defined($file=readdir(CD))) {
    next unless (-f "/var/spool/cron/crontabs/$file");
    open(FH,"/var/spool/cron/crontabs/$file");
    while ($cronline=<FH>) {
        next if ($cronline =~ /^#/);
        next unless ( grep /\s+usr\lib\sa1/, $cronline );
        $hr=sprintf("%02d",$hr);
        my ($a1,$a2,$a3,$a4,$a5,$a6,$a7,$a8,$a9)=split(/\s+/, $cronline);
        $_=$a5;
        SWITCH: {
            /\*/ and do {
                @days=(0,1,2,3,4,5,6);
                last;
            };
        };
        ^[0-9]/ and do {
            if ( grep /\s/, $a5 ) {
                my ($min,$max)=split(/\s/, $a5);
                for ($temp = $min; $temp <= $max; $temp++) {
                    push(@days,$temp);
                }
            } elsif ( grep /\s\s/, $a5 ) {
                @days=split(/\s\s/, $a5);
            } else {
                push(@days,$a5);
            }
            last;
        };
    }
} if ( grep /today/, @days ) {
    $_=$a2;
    SWITCH: {
        /\*/ and do {
            @hours=("00", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11",
            last;
        };
    };
    ^[0-9]/ and do {
        if ( grep /\s/, $a2 ) {
            my ($min,$max)=split(/\s/, $a2);
            if ( $min > $max ) {
        }
for ($temp = $min; $temp <= 23; $temp++) {
    push(@hours, sprintf("%02d", $temp));
}
for ($temp = 0; $temp <= $max; $temp++) {
    push(@hours, sprintf("%02d", $temp));
} else {
    for ($temp = $min; $temp <= $max; $temp++) {
        push(@hours, sprintf("%02d", $temp));
    }
}
else if ( grep /,/, $a2 ) {
    foreach (split(/,/, $a2)) {
        push(@hours, sprintf("%02d", $_));
    }
} else {
    push(@hours, sprintf("%02d", $a2));
} last;
;
;
} if ( grep /$lhr/, @hours ) {
    if ( $a1 == \/*\/) {
        # Cron deals with sa1 intervals not sa1 arguments
        # (ie one every minute)
    } else {
        # Need to work out if cron mins or sa1 args are in use
        if ( $a7 eq ";&\) {
            if ( $a1 =~ /^[0-9]$/ ) {
                close FH;
                close CD;
                exit 0;
            }
        }
    }
    # use arguments to sa1 to work out what samples are left to be taken
    my $everymin=$a7/60;
    if ( $lmin >= $a1 ) {
        $remainingintervals=$a8-int(($lmin-$a1)/$everymin)-1;
    } else {
        $remainingintervals=int(($a1-$lmin-1)/$everymin);
    }
    print "everymin = $everymin  a1 = $a1  a7 = $a7  a8 = $a8  remain = $remainingintervals  lmin = $lmin\n";
    if ( $remainingintervals == 0 ) {
        # Leave it until next hour
    } else {
        $nextmin=int(($lmin+$everymin)/$everymin)*$everymin;
        if ( $nextmin >= 60 ) {
            $nextmin = 60;
        }
$lhr++;  
if ($lhr == 24) {  
  $lhr = "00";  
  $tomorrow=" tomorrow";  
}

# Now write the command to at to continue sa1 processing
$sattime=sprintf("%02d",$lhr).sprintf("%02d",$nextmin).$tomorrow;
open(CM,"|/usr/bin/at $sattime");
print CM " /usr/lib/sa/sa1 $a7 $remainingintervals";
close CM;
close FH;
close CD;
exit 0;

@hours=();


refine END OF IS_HOUR FUNCTION


@days=();


refine END OF IS_DAY FUNCTION


close FH;
close CD;

Phil Pollard
Unix and TSM Administrator (UK)  © Xephon 2004

AIX Update on the Web

Code from individual articles of AIX Update, and complete issues in Acrobat PDF format, can be accessed on our Web site, at:

http://www.xephon.com/aix

You will be asked to enter a word from the printed issue.
Monitoring the availability of networked systems

Essentially, this is a tool to monitor the availability of networked systems, on the LAN or on the Web. It checks that the systems are pingable, and it can even check that certain listener services are functioning, by doing a socket connect to the port(s) that require monitoring.

NETMON

Netmon is the main script. Documentation is contained in its header. It should be set up in cron to run around the clock, as shown in the documentation.

```
#!/bin/ksh
#
# . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
# netmon
#
# Mike Stanton
# Montvale, NJ
# U.S.A.
#
# stanton@mbusa.com
# 10/16/03
#
# . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
#
# Check selected hosts (eg on the Web, www.abcdefg.com, or on the LAN,
# mylocalserver, etc) and e-mail/page as required if a site appears to
# be unresponsive to pinging.
#
# Connect to selected listener ports on the servers to monitor services
# that should be running, and report if there is a problem.
# A history file will be maintained to record all events.
#
# Notifications will be sent by e-mail (default) and if desired, also by
# pager. For machines that warrant paging, the hostname should be
# entered in the "pagefile" as shown below. It is important to include a
# colon after the hostname.
#
# This job should be run from cron, as frequently as required (eg every
# minute, every 5 minutes, etc).
#
# eg * * * * * /SYSMGR/netping >> /SYSMGR/LOGS/netping.log 2>&1
```

# In order to check any listening ports on a given machine, the C
# program sockck.c does a socket connect to the host and checks whether
# the port in question is listening or not. For example, it can check
# whether port 80 is responding for a webserver machine. Any in-house
# programs that listen on other ports can also be monitored, as desired.
# In order to specify which ports need to be monitored for a given
# server (if any), a file with the name "someserver.ports" should be
# created. Inside this file, simply
# list the port number and application name, delimited by a colon.
# For example: 80:HTTP
# See the myserver1.ports file as an example. As many ports as desired
# can be monitored.
#
# The script "networks.sh" can be used to selectively turn off/on
# paging. In the event that system maintenance is scheduled and
# excessive paging would result, this script can turn off the paging
# feature, and when the maintenance is done, the paging
# can be turned back on again.
#----------------------------------
# Modification History:
#---------------------
#
#--------------------------------------------------------------------------------

# SEND_MAIL ()
#
# trap ' ' ERR
msgoptions=

case $1 in
  "PING_NOTIFY")
    echo "$(cat ${logfile})" > ${MAIL_FILE}
    typeset -i totalnum=0 downnum=0 upnum=0
    typeset -s downmsg="" upmsg=""
    totalnum=$(grep -c responding ${logfile})
    downnum=$(grep -c "NOT" ${logfile})
    if [ ${downnum} -gt 0 ]
    then
      downmsg="${downnum} nodes Unreachable 
    fi
    upnum=$(grep -c "NOW" ${logfile})
    if [ ${upnum} -gt 0 ]
    then
      upmsg="${upnum} nodes Reachable 
    fi
MAIL_SUBJECT="${CURR_SHELL}: (${totalnum}) Events; ${downmsg}${upmsg}"

"PORT_NOTIFY"
  cat << EOF > ${MAIL_FILE}
  Port ${port} (${application}) of server ${hostname} is not responding. Please check if there is a problem.
EOF

MAIL_SUBJECT="${CURR_SHELL}: Problem with port ${port} (${application}) on ${hostname}"

"PORT_UP"
  cat << EOF > ${MAIL_FILE}
  Port ${port} (${application}) of server ${hostname} is now responding again.
EOF

MAIL_SUBJECT="${CURR_SHELL}: ${hostname} port ${port} (${application}) is now responding again"

"LOCK_FOUND"
  cat << EOF > ${MAIL_FILE}
  There was a netmon.lock lock file found, so netmon is exiting. Normally this means that the previous netmon job was still running, longer than usual.

  NOTE: netmon will not work again if the lock file remains. If the previous running netmon finishes shortly, it should delete the lockfile when it finishes up.
EOF

MAIL_SUBJECT="${CURR_SHELL}: Overlapping netmon jobs detected"

")
echo " Message Source: $CURR_SHELL" >> $MAIL_FILE
MAIL_FILE
MAIL_SUBJECT="$CURR_SHELL: INVALID PARAMETER - PROCESSING TERMINATED"

esac

case $1 in
  "LOCK_FOUND"
  # techs_dis is a mail distribution list stored in the aliases file of
  # the sendmail server...
  mail -s "${MAIL_SUBJECT}" "techs_dis" < ${API_FILE}
  #mail to techsupt only
  ;;
  "PING_NOTIFY" | "PORT_NOTIFY" | "PORT_UP")

  # netmon_dis is a mail distribution list stored in the aliases file of
  # the sendmail server...
  mail -s "${MAIL_SUBJECT}" "netmon_dis" < ${API_FILE}
  #mail to all on list
  ;;
  esac
  return

*************************************************************************
# START OF MAIN SHELL BODY
*************************************************************************

/etc/profile
MAIL_FILE="$MAIL_API/api$$.bdy"
API_FILE="$(basename $MAIL_FILE | sed 's/\..*$//')"
CURR_SHELL="$(basename $Ø)"

echo "***** netmon procedure beginning at $(date)...."
#echo "... First check whether or not there is a running netmon script at
# this time:"
lockfile=/SYSMGR/netmon.lock

if [ -f ${lockfile} ]
then
  echo "Previous running netmon detected (${lockfile} exists)."
  SEND_MAIL LOCK_FOUND
  echo "Exiting this script...."
  exit Ø
else
  touch ${lockfile} #create a lock file
fi

normal_ping=3
re_ping=1

# for double-checking: it failed already,
# multiple pings won't help, timeout might...
short_ping=3  # normal times to ping
long_ping=${long_ping:=9}
  # just to double-check that it's really down
echo "Using a long_ping value of: ${long_ping}."

export hostsfile="/SYSMGR/netmon_hosts.dat"
export pagefile="/SYSMGR/netmon_pagenodes.dat"
export logfile="/SYSMGR/ping_log_file"
export badfile="/SYSMGR/ping_badfile"
export stillbadfile="/SYSMGR/ping_badtmpfile"
export historyfile="/SYSMGR/ping_historyfile"
export port_badfile="/SYSMGR/netmon_port_badfile"
export portstillbadfile="/SYSMGR/port_badtmpfile"
export nopage_flag="/home/netmon/netmon.nopage_flag"
export page_exceptions="/SYSMGR/netmon_page_exceptions.dat"

# cat /dev/null > ${logfile}
[[ ! -s ${badfile} ]] && cat /dev/null > ${badfile}
# touch empty one
[[ ! -s ${port_badfile} ]] && cat /dev/null > ${port_badfile}
# touch empty one
#---------------------------------------------------------------
IFS=":
# Internal Field Separator
while read hostname
# reading from ${hostsfile}
do
  grep -q "${hostname}:" ${badfile}
      # Trailing colon to delimit ending.
      if [[ $? -eq 0 ]]
        then
          echo "Temporarily skipping host ${hostname} because it was
          found in ping_badfile..."
          continue
          # Skip this record
        fi

  # print "$(date +%Y/%m/%d %T): Trying to ping ${hostname}"
  # ping -c${normal_ping} -f -w${short_ping} ${hostname} > /dev/null 2> /
  # dev/null
  ping_status=0
  if [ ! ${ping_status} -ne 0 ]
    then
      DAY="$(date +%a)"
      if [[ [ $DAY = "Sun" ]]]
        then


HR=$(date +%H)

# get the current hour
if [[ ${HR} -ge "22" ]]
then
    continue
# Sunday 22-midnight ignore
fi
fi

# echo "Short ping failed for ${hostname}; trying long ping now..."
ping -c${re_ping} -w${long_ping} ${hostname} > /dev/null 2> /dev/null
long_ping_status=$?
#
if [ ${long_ping_status} -ne Ø ]
then
    echo "Long ping also failed for ${hostname}"
    echo "$(date) ${hostname} is NOT responding\n"
>> ${logfile}
echo "${hostname}:
" >> ${badfile}

###... Add the entry in the historyfile as well.
echo "$(date)"
>> ${historyfile}
echo "${hostname} is NOT responding" >> ${historyfile}
${historyfile}
#
host=$(echo ${hostname} | cut -f1 -d'.')
# get name before the '.'
grep "${host}:" ${page_exceptions}
# check for skipping page
if [[ $? -eq Ø ]]
# during certain times
then
    HR=$(date +%H)
    # get the current hour
    if [[ ${HR} = "05" || ${HR} = "06" ]]
    then
        continue
    # ignore 5-6am timeframe
    fi

    if [[ ! -f ${nopage_flag} ]]
    then
        grep -q "${hostname}:" ${pagefile}
    fi

# if in here, we'll page
if [[ $? -eq Ø ]]
then
  echo "$(date) "
  MAIL_SUBJECT="DOWN: ${hostname} is not responding"
  mail -s "${MAIL_SUBJECT}" "netmonpage_dis" < ${API_FILE}
fi
fi
continue
# read next record
fi
#
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#...Now check the ports on the server that just successfully responded
# to ping...
if [[ -s /SYSMGR/${hostname}.ports ]]
# any ports for this host will appear in its .ports file
then
  while read port application
  do
    print -n "$(date '+%Y/%m/%d %T'): Checking port
${port} (${application}) on server ${hostname}..."
    /SYSMGR/sockck ${hostname} ${port}
    if [[ $? -ne Ø ]]
      then
        host=$(echo ${hostname} | cut -f1 -d'.')
        # get name before the '.'
        grep -q "${hostname}:${port}:" ${port_badfile}
        # known bad already?
        if [[ $? -eq Ø ]]
          then
            echo "Skipping check of
${hostname}:${port} since it's in the port_badfile.."
            continue
        # check the next port..
        else
          echo "Problem with port ${port} on server ${hostname}!"
          echo "$(date)"
    >> ${historyfile}
    echo "${hostname}:${port} is NOT responding"
  >> ${historyfile}
  echo "*******************************"
    >> ${historyfile}
  echo "${hostname}:${port}:${application}"  >> ${port_badfile}
  SEND_MAIL PORT_NOTIFY
#...initialize the $MAIL_FILE since the call to PORT_NOTIFY would # have removed it..  
    echo "$(date)"
>> $MAIL_FILE

host=$(echo ${hostname} | cut -f1 -d'.')

grep "$({host}):" ${page_exceptions}

if [[ $? -eq Ø ]]
#during certain times
then
    HR=$(date +%H)
    if [[ ${HR} = "05" || ${HR} = "06" ]]
    then
        continue
    fi
fi

#ignore 5-6am timeframe
  fi

#  
  if [[ ! -f ${nopage_flag} ]]
then
    grep -q "${hostname}:" ${pagefile}
    if [[ $? -eq Ø ]]
then
        MAIL_SUBJECT="${hostname} port
${port} (${application}) is not responding"
        mail -s "${MAIL_SUBJECT}"
        "net_monpage_dis" < ${API_FILE}
    fi
fi

fi
done < /SYSMGR/${hostname}.ports
fi

done < ${hostsfile}
#
#==================================================================================
if [[ -s ${badfile} ]]
then
    while read hostname
    do
        echo "Retrying host from badfile: ${hostname}"
        ping -c${re_ping} -w${long_ping} ${hostname} > /dev/null 2> /dev/null
    done < ${badfile}
fi

ping_status=${?}
if [[ ${ping_status} -ne 0 ]]
#...node is still unreachable
then
    echo "${hostname}:
    echo "${hostname} is still unreachable"
else
#...node is reachable again
    echo "${date} ${hostname} is NOW responding\n" >> ${logfile}
#
# Add the entry in the historyfile as well.
#
    echo "${date}"
    echo "${hostname} is NOW responding" >> ${logfile}
    echo "********************************************** " >> ${logfile}
grep "${host}:" ${page_exceptions}
#check for skipping page
    if [[ $? -eq 0 ]]
    #during certain times
    then
        HR=${date +%H}
    #get the current hour
    if [[ ${HR} == "05" || ${HR} == "06" ]]
        then
            continue
    fi
    fi
    if [[ ! -f ${nopage_flag} ]]
    then
        grep -q "${hostname}:" ${pagefile}
# if in here, we'll page
    if [[ $? -eq 0 ]]
    then
        echo "${date}"
    MAIL_SUBJECT="UP:${hostname} is now responding again"
    mail -s "$MAIL_SUBJECT" "netmonpage_dis" < ${API_FILE}
fi
fi
done < ${badfile}
fi
#
#*****************************************************************************
if [[ -s ${port_badfile} ]]               #if we had at least one bad port...
  then
    while read hostname      port application
      # (reading from ${badfile})
      do
      ping -c${normal_ping} -w${short_ping} ${hostname} > /dev/null
      2> /dev/null
      ping_status=$?
      if [ ${ping_status} -ne Ø ]
      then
        continue
        # no sense checking port if host is down...
      fi
      echo "Retrying ${hostname}:${port}..."
      # try the port again
      /SYSMGR/sockck ${hostname} ${port}
      # see if it's still bad
      port_status=$?
      if [[ ${port_status} -ne Ø ]]
      #...node is still unreachable
      then
        echo "${hostname}:${port}:${application}"
        >> ${portstillbadfile}
      else
      #...node is reachable again
        echo "${(date)}: ${hostname} ${port} (${application}) is NOW responding"
        # Add the entry in the historyfile.
        #
        echo "${(date)}"
        >> ${historyfile}
        echo "${hostname}:${port} is NOW responding" >> ${historyfile}
        echo "*************************************************************************** "
        >> ${historyfile}
      SEND_MAIL PORT_UP
      grep "${host}:" ${page_exceptions}
      # check for skipping page
      if [[ $? -eq Ø ]]
      # during certain times
      then
HR=$(date +\%H)
if [[ $HR = "05" || $HR = "06" ]]
then
    continue
fi

# ignore 5-6am timeframe
if [[ ! -f $nopage_flag ]]
then
    grep -q "$hostname:" $pagefile
fi

# if in here, we'll page
if [[ $? = 0 ]]
then
    #...initialize the $MAIL_FILE since the call to PORT_UP would have
    #    removed it..
    echo "$(date)"
    MAIL_SUBJECT="$hostname port ${port} (${application}) is now responding again"
    mail -s "MAIL_SUBJECT" "netmonpage_dis" < $API_FILE
fi
fi

done < $port_badfile
fi

# #===============================================================================
# # if [ ! -s $stillbadfile ]
# then
#     rm $badfile
# else
#     mv $stillbadfile $badfile
# fi
# #===============================================================================
# if [ ! -s $portstillbadfile ]
# then
#     rm $port_badfile
# else
#     mv $portstillbadfile $port_badfile
# fi
# If there was at least one "Up" or "Down" event, we will send mail...
grep "responding" '${logfile}' > /dev/null 2> /dev/null
if [ $? = 0 ]
then
    SEND_MAIL PING_NOTIFY
fi

# Remove the lockfile now that we're done
rm -e '${lockfile}'

SOCKCK.C
Sockck.c is a C program that is used by the netmon script, and it will do a socket connect to a given system to check on listener services to be monitored.

/* sockck
   * Connects to a socket on a given system and checks for a valid response.
   * M. Stanton             August 2003
   *
   * Modification History:
   * ---------------------
   * 
   */

#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <netdb.h>
#include <signal.h>
#include <setjmp.h>
#define HOSTSIZE 80
#define DEFAULT_TIMEOUT 20

extern int errno;
jmp_buf env;           /* Buffer to save the process environment in */
char hostname[HOSTSIZE];
char *timeout_str;
int timeout_val;
main (int argc, char *argv[]) {
    register int s;
    int port = Ø;
    FILE *fp;
    struct hostent *hp;
    struct sockaddr_in sin;
    char *getenv();
    extern void timeout();
    char c;
    if (argc != 3) {       /* Check for correct number of arguments */
        fprintf (stderr, "Usage: %s <hostname> <port>\n", argv[0]);
        exit (1);
    }
    timeout_str = getenv("SOCKCK_TIMEOUT");
    if (timeout_str == NULL) {
        /* DEBUG fprintf (stdout, "Value was not set. Setting it to
         default of 20 secs...\n"); */
        timeout_val=DEFAULT_TIMEOUT;
    } else {
        /* DEBUG fprintf (stdout, "Value was defined. It is %s.\n", timeout_str); */
        timeout_val = atoi(timeout_str);
    }
    /* Put 1st arg into hostname variable and
    convert port arg to an integer */
    strcpy (hostname, argv[1]);
    /* Copy host argument into hostname variable */
    port = atoi(argv[2]);
    /* Convert port argument into an integer */
    /* Check if hostname is known to DNS */
    if ((hp = gethostbyname (hostname)) == NULL) {
        fprintf (stderr, "%s: unknown host.\n", hostname);
        exit (1);
    }
    /* Create a socket */
    if ((s = socket (PF_INET, SOCK_STREAM, Ø)) < Ø) {
        perror ("Error during socket call");
        exit (1);
    }
    /* Fill in the necessary information into the data structure */
    sin.sin_len = sizeof (struct sockaddr_in);
    sin.sin_family = PF_INET;
    sin.sin_port = htons(port);
    bcopy (hp->h_addr, &sin.sin_addr, hp->h_length);
    signal (SIGALRM, timeout);
    /* Set and alarm to execute timeout routine */
}

if (setjmp(env) == Ø) { /* Save the process context */
    alarm(timeout_val); /* Set alarm */
    timer_in_seconds */
    /*DEBUG fprintf (stdout, "Trying to Connect to %s using port %d
now.\n", hostname, port); */
    /* Connect to the socket on the remote system */
    if (connect (s, (struct sockaddr *) &sin, sizeof(sin)) < Ø) {
        perror ("Error during connect");
        alarm(Ø); /* Turn off the alarm */
        close (s); /* Close the socket */
        exit (1); /* Exit with an error condition */
    }
    fprintf (stdout, "Connect worked.\n");
} else { /* We get here if the timer expired before the connect operation
completed */
    fprintf(stdout, "Connection timed out.\n");
    close (s); /* Close the socket */
    exit (1); /* Exit with an error condition */
}
close (s); /* Close the socket */
exit (Ø);
}
void timeout (sig) /* Timeout routine */
int sig; /* One argument is passed */
{
    signal (sig, SIG_IGN); /* Ignore the signal; no action
need be taken */
    signal (SIGHUP, timeout); /* Reset the alarm */
    longjmp(env, 1);
    /* Restore the process context returning a value of 1 */
}

NETMON_HOSTS.DAT

Netmon_hosts.dat is a sample file containing the names of all
the servers to be monitored.

myserver1:
myserver2:
myserver3:
myserver4:
myserver5:
myserver6:
myserver7:
myserver8:
**NETMON_PAGENODES.DAT**

Netmon_pagenodes.dat is a sample file containing all those hosts that are important enough to need page alerts as well as e-mail alerts.

```markdown
myserver1:
myserver2:
myserver3:
myserver4:
myserver5:
myserver6:
myserver7:
myserver8:
```

**NETWORKS.SH**

Networks.sh is a script that can be used to selectively turn on/off paging. It is useful during known system maintenance periods.

```bash
#!/bin/ksh
#
# AUTHOR : MIKE STANTON
#
# This script is for the Networks Services people so that
# they can turn off/on the paging from the Netping script.
#
#set -xv
#
stty -isig
stty erase '^H'

flag_file="/home/netmon.nopage_flag"
historyfile="/home/netmon.history"
setting=""
while true
do
    if [[ -f ${flag_file} ]]
        then
            setting="OFF"
        else
            setting="ON"
    fi
    clear
    echo "\n\tRS/6000 Netping Options"
    echo "\t"$setting"
    echo "Netping paging is currently: ${setting}\n\n"
```

```
echo "1. Turn paging OFF"
echo "2. Turn paging ON"
echo "3. Generate e-mail Status Report of 'down' nodes"
echo "4. EXIT this program."
read choice
case ${choice} in
  ")
    clear
    continue
  ;;
1)                                         # turn paging off
    if [[ -f ${flag_file} ]]
      then
        echo "$Paging was already turned off."
      else
        touch ${flag_file}
        echo "$Paging has been turned off."
        echo "$$(date) Paging has been turned off." >> ${historyfile}
    fi
    echo "$Press <Enter> to continue....";
  ;;
2)                                         # turn paging on
    if [[ -f ${flag_file} ]]
      then
        rm -f ${flag_file}
        echo "$Paging has been turned on."
        echo "$$(date) Paging has been turned on." >> ${historyfile}
    else
      echo "$Paging was already turned on."
    fi
    echo "$Press <Enter> to continue....";
  ;;
3)                                          # generate e-mail Status Report
    if [[ -s /MBSYSMGR/ping_badfile ]]
      then
        mail -s "Netping Status - 'Down' nodes" < /SYSMGR/ping_badfile
          person1, person2, person3
      else
        echo "There are no 'down' nodes at this time." > ./nonedown.dat
        mail -s "Netping Status - No 'Down' nodes at this time" < ./nonedown.dat
        person1, person2, person3
    fi
  ;;
4)                                          # exit the program
```
break
;
*)
clear
;
 esac
#
done
stty isig
clear
logout

**MYSERVER1.PORTS**

Myserver1.ports is a sample file that would contain a list of ports to monitor, for an imaginary server with a hostname of ‘myserver1’.

```
80: HTTP
7010: MyApplication1
7042: MyApplication2
```

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

© Xephon 2004

If you have ever experienced any difficulties with AIX, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon’s wide range of products and services, simply by telling us all about it.

More information about contributing an article to a Xephon *Update*, and an explanation of the terms and conditions under which we publish articles, can be found at [http://www.xephon.com/nfc](http://www.xephon.com/nfc).

Articles, or article proposals, can be sent to the editor, Trevor Eddolls, at any of the addresses shown on page 2. Alternatively, you can e-mail him at trevore@xephon.com
This month we continue our look at curses programming.

```c
if ( nc == Ø )
{
    /* error may have occurred */
    nc = read ( w[i].err[Ø], buffer, BUFSIZE - 1 );
    buffer[nc]='Ø' ;
    waddstr ( w[i].win, buffer );
}
/* if we read something, update screen */
if ( nc > Ø )
    wnoutrefresh(w[i].win );
}/* end for */
/* read from keyboard */
i = Ø ;
while ( ( c = getch () ) > Ø )
{
    switch (c)
    {
        case Øx1b:  /* ESCAPY  key           */
            /* for switching window */
            if ( cwin == MAX_WIN - 1 )
            {
                cwin = Ø ;
                DisplayWindowTitle ( "Top Window" );
            }
            else
            {
                cwin ++ ;
                DisplayWindowTitle ( "Bottom Window" );
            }
            break ;
        case  4:   /* ctrl-D      */
            /* end program */
            DisplayError("Quitting",__LINE__);  
            default:
            /* add the character to screen buffer */
            buffer[i++] = c ;
            /* write it to pipe that will be read */
            /* by waiting shell via out[Ø] */
            write(w[cwin].out[1],&c,1);
    }
}/* end while */
if (i)
{
```
buffer[i] = '\0';
  waddstr(w[cwin].win, buffer);
}
/* update the screen */
wnoutrefresh(w[cwin].win);
doupdate();
} /* end while */
endwin();
return ( SUCCESS );
}
/* update the screen */
wnoutrefresh(w[cwin].win);
doupdate();
} /* end while */
endwin();
return ( SUCCESS );
}
int CreateWindows (void)
{
  short i;
  WINDOW *wptr;
  /*
   * initialise curses
   */
initscr();
  for (i = 0; i < MAX_WIN; i++)
  {
    wptr = newwin (pos[i].lines, pos[i].cols, pos[i].begy, pos[i].begx);
    if (wptr == (WINDOW *) NULL)
    {
      printf("ERROR: Failed to create window\n");
      return ( FAILURE );
    }
    /*
     * store this pointer to window in array
     */
    w[i].win = wptr;
    scrollok (w[i].win, TRUE);
  }
  return ( SUCCESS );
}

int CreateShells (void)
{
  int rc; /* function return code */
  int i;
}
for ( i=0; i < MAX_WIN; i++ )
{
    
    /* first set of pipes */
    rc = pipe ( w[i].out );
    if ( rc < 0 )
    {
        DisplayError("Failed to create pipes",__LINE__ );
        return ( FAILURE );
    }

    /* second set of pipes */
    rc = pipe ( w[i].err );
    if ( rc < 0 )
    {
        DisplayError("Failed to create pipes",__LINE__ );
        return ( FAILURE );
    }

    /* third set of pipes */
    rc = pipe ( w[i].in );
    if ( rc < 0 )
    {
        DisplayError("Failed to create pipes",__LINE__ );
        return ( FAILURE );
    }

    /* create a child process */
    pid = fork () ;
    if ( pid == -1 )
    {
        DisplayError("Failed to create child process",__LINE__ );
        return ( FAILURE );
    }

    /* handle child process */
    /* a copy of sh will be running here */
    
    if ( pid == 0 )
    {

    }
out[1] open for writing; output goes to ----> out[Ø] for reading
err[1] open for writing; output goes to ----> err[Ø] for reading
in[1] open for writing; output goes to ----> in[Ø] for reading
*/
/*
* close file descriptor for stdin
*/
close(Ø);
/*
* copy file descriptor used for reading the command by sh
*/
dup[w[i].out[Ø]] ;
/*
* close file descriptor for stdout
*/
close(1);
/*
* copy file descriptor used for writing the command output by sh
*/
dup[w[i].in[1]] ;
/*
* close file descriptor for stderr
*/
close(2);
/*
* copy file descriptor used for writing the errors by sh
*/
dup[w[i].err[1]] ;
/*
* close all original file descriptors
*/
close( w[i].in[Ø] ) ;
close( w[i].in[1] ) ;
close( w[i].out[Ø] ) ;
close( w[i].out[1] ) ;
close( w[i].err[Ø] ) ;
close( w[i].err[1] ) ;
/*
* start the shell
*/
execlp("/usr/bin/sh","sh","-i",Ø);
}
/*
* handle parent process
*/
{
/*
* close unwanted file descriptors
*/
}
close( w[i].out[0] );
close( w[i].in[1] );
close( w[i].err[1] );
/
* exercise file control
*/
rc = fcntl ( w[i].in[0], F_GETFL, 0 );
r0 = fcntl ( w[i].in[0], F_SETFL, rc | O_NDELAY );
r0 = fcntl ( w[i].err[0], F_GETFL, 0 );
r0 = fcntl ( w[i].err[0], F_SETFL, rc | O_NDELAY );
r0 = fcntl ( w[i].out[1], F_GETFL, 0 );
r0 = fcntl ( w[i].out[1], F_SETFL, rc | O_NDELAY );
/
* parent process read from in[0] any output generated by sh
* parent process write to out[1] for sh to read from
* parent process read error from err[0]
*/
}
}
}
return ( SUCCESS );
}/*******************************************************
* Name : DisplayError
* Overview : The function displays an error message on the screen.
* Notes :
*******************************************************/
void DisplayError ( char *msg, int line_no )
{
/*
* end window
*/
endwin ( );
clear ( );
printf( "Pid = %d
", pid );
printf("%s
Line No = %d\n", msg, line_no);
exit (1);
}/*******************************************************
* Name : DisplayWindowTitle
* Overview : The function displays window title
* Notes :
*******************************************************/
void DisplayWindowTitle ( char *msg )
{
standout ( );
mvprintw(12, 0, "%- 79s", msg );
standend ( );
refresh ( );
}
* Name : HandleSignal ( )  
* Overview : The function handles signal.  
* Notes : 

***************************************************************************************/

#include <signal.h>

void HandleSignal ( int signo )
{
  DisplayError("Child died ", __LINE__); 
}

Curses functions used

• **initscr ( )** – the function is used to perform all the required initialization for any curses programs. It must be invoked at the beginning of any other invocation of a curses function. It should be called only once in the program.

• **endwin ( )** – the function is used to end a curses program. The function must be invoked before exiting the program; otherwise, the program might leave the terminal in an unpredictable state.

• **nodelay (WINDOW *win, bool boolean_state)**  
  If *boolean_state* is TRUE, the function makes the input function **getch ( )** a non-blocking call – that is, if no input is ready when **getch ( )** is called, **getch()** will not wait for input but will return an error.

• **getch ( )** – this is a macro that has been defined as:

   ```c
   #define getch () 
   wgetch (stdscr) 
   ```

   **wgetch (stdscr)** is used to get a character input from stdscr.

• **noecho ( )** – in this mode, characters are not added to the window by **wgetch ( )**. It is up to you to add them to it. This is normally done by **waddch ( )** followed by **wrefresh ( )**. This mode is often required by programs that want to examine each character before displaying them.

• **WINDOW *newwin (lines, cols, ycor, xcor)**  
  This function creates a new curses window whose dimensions are specified in the arguments.
• **raw ( )** – the terminal is set into a raw mode. All characters typed are passed directly to the program without being interpreted.

• **wrefresh (WINDOW *win)**

The function refreshes the terminal screen with the contents of curscr. window. If the window does not represent the entire screen, only that part covered by the window is updated.

• **waddstr (WINDOW *win, char *str)**

The function adds the string, str in the window pointed to by win.

• **wnoutrefresh (WINDOW *win)**

The function updates the curses virtual screen, curscr, with the contents of the window pointed to by win. No actual update is done to the physical screen.

• **doupdate ( )** – the function updates the physical terminal screen. It compares the virtual screen, curscr, with the physical screen. It updates the parts of the physical screen that have changed.

• **standout ( )** – this function sets the video attribute standout to the window pointed to by **stdscr**.

• **mvprintw (WINDOW *win, y, x, format , char *msg)**

The function is equivalent to **printf ( )** for the curses window.

• **standend ( )** – this function clears the video attribute standout to the window pointed to by **stdscr**.

• **refresh ( )** – the function refreshes the physical screen with the contents of **stdscr**.

• **scrolllok (win, boolean_state)**

The function manipulates the scroll/on scroll/off toggle on the specified window, win.

• **mvcur (oldline, oldcol, newlinw, newcol)**
The function moves the cursor from the old location to the new location.

**TASK BAR WIDGET**

The program creates a task bar widget (displaying the progress of a job) on the terminal screen.

```c
#include <stdio.h>
#include <curses.h>
#include <unistd.h>
#include <time.h>
#include <fcntl.h>

short MakeTaskbarWidget   ( void );
short UpdateTaskbar       (short percent);
short DisplayMessage      (char *msg);
short RemoveTaskbarWidget (void);
short GetTime             (char *time);
void DisplayCompletionTime(void);
```

**MODULE CONSTANT**

```c
#define TRUE            1
#define FALSE           0
#define DONE            3
#define SUCCESS         1
#define FAILURE         0
#define UNIX_SUCCESS    0
```
#define UNIX_FAILURE 1
#define WINXCOR 10 /* details of primary window */
#define WINYCOR 5 /* in absolute coordinates */
#define WINHEIGHT 15
#define WINWIDTH 50
#define TBXCOR 10 /* details of subwindow for displaying */
#define TBYCOR 11 /* taskbar */
#define TBHEIGHT 3
#define TBWIDTH 50
#define MWXCOR 10 /* details of subwindow for displaying */
#define MWYCOR 18 /* message */
#define MWHEIGHT 1
#define MWWIDTH 50
#define CW1XCOR 11 /* details of subwindow for displaying */
#define CW1YCOR 7 /* starting time */
#define CW1HEIGHT 1
#define CW1WIDTH 50
#define CW2XCOR 38 /* details of subwindow for displaying */
#define CW2YCOR 7 /* completion time */
#define CW2HEIGHT 1
#define CW2WIDTH 21
#define HWXCOR 26 /* details of subwindow for displaying */
#define HWYCOR 5 /* heading */
#define HWHEIGHT 1
#define HWWIDTH 15

GLOBAL VARIABLES
*****************************************************************************
WINDOW *wptr; /* pointer to the main window structure */
WINDOW *tbptr; /* pointer to taskbar window structure */
WINDOW *mwptr; /* pointer to message window structure */
WINDOW *cw1ptr; /* pointer to starting clock window structure */
WINDOW *cw2ptr; /* pointer to ending clock window structure */
WINDOW *hwptr; /* pointer to heading window structure */
*****************************************************************************

Name : MakeTaskbarWidget
* Overview : The function creates the taskbar widget and its
  associated components.
* Returns : SUCCESS, FAILURE
* Notes : 1. All the window coordinates are held in symbolic
  constants.
  2. The following components are also created by this function:
    o heading window
    o start time display window
    o message display window
*****************************************************************************
short MakeTaskbarWidget ( )
{
int i;
char msg[40];
char time_now[10];
/*
 * initialize the screen
 */
initscr();
/*
 * create main window
 */
wptr = newwin(WINHEIGHT, WINWIDTH, WINYCOR, WINXCOR);
if ( wptr == (WINDOW *) NULL )
{
    printf("%s:%d:ERROR: Failed to create the
window\n", __FILE__, __LINE__);
    return FAILURE;
}
/* reverse the video for the whole window */
wattron ( wptr, A_REVERSE);
for ( i = 0; i < (WINHEIGHT * WINWIDTH); i++ )
    waddstr(wptr,"");
wrefresh(wptr);
/*
 * make sub-window and display heading
 */
wattroff ( wptr, A_REVERSE);
hwptr = subwin(wptr, HWHEIGHT, HWWIDTH, HWYCOR, HWXCOR);
for ( i = 0; i < (HWHEIGHT * HWWIDTH) ; i++ )
    waddstr(hwptr," ");
wxmove(hwptr,0,0);
waddstr(hwptr,"Task Bar Widget");
wrefresh(hwptr);
/*
 * make the subwindow for taskbar
 */
wattroff ( wptr, A_REVERSE);
tbptr = subwin(wptr, TBHEIGHT, TBWIDTH, TBYCOR, TBXCOR);
for ( i = 0; i < (TBHEIGHT * TBWIDTH) ; i++ )
    waddstr(tbptr," ");
wrefresh(tbptr);
/*
 * make the subwindow for message
 */
wattroff ( wptr, A_REVERSE);
mwptr = subwin(wptr, MWHEIGHT, MWWIDTH, MWYCOR, MWXCOR);
for ( i = 0; i < (MWHEIGHT * MWWIDTH) ; i++ )
waddstr(mwptr," ");
wrefresh(mwptr);
/*
 * make the subwindow for displaying starting time
 */
wat troff (wptr, A_REVERSE);
cw1ptr = subwin (wptr, CW1HEIGHT, CW1WIDTH, CW1YCOR, CW1XCOR);
for (i = 0; i < (CW1HEIGHT * CW1WIDTH); i++)
waddstr(cw1ptr," ");
wrefresh(cw1ptr);
/*
 * display starting time
 */
strcpy(msg,"Started at ");
GetTime(time_now);
strcat(msg,time_now);
wmove(cw1ptr,0,0);
wat troff (cw1ptr, A_REVERSE);
waddstr(cw1ptr,msg);
/*
 * update the screen
 */
wrefresh(cw1ptr);
}

 /******************************************************************************
 * Name : UpdateTaskbar
 * Input : Percentage (short )
 * Returns : SUCCESS
 * Description : The function updates the taskbar to reflect
 * the percentage of task being completed.
 * Notes : 1. For percentage less than 2 and greater than 100, the function
does not do anything.
 * 2. For percentage 100, it displays the completion time in addition to
 updating the taskbar.
 ******************************************************/
short UpdateTaskbar ( short percent )
{
static short cur_xcorval ;
static short i;
static char msg[30];
static task_completed = FALSE;
/*
 * determine the x-coordinate value for the percent provided
 * the width of thermometer is 50 spaces which represents 100 percent.
 */
cur_xcorval=percent / 2 ;
if ( cur_xcorval < 1 || cur_xcorval > 50 )
    return SUCCESS ;
if ( task_completed == TRUE )
return SUCCESS;

/*
 * highlight the percentage done
 */
wmove(tbptr, 1, 0);
wattron(tbptr, A_REVERSE);
for ( i = 0; i < cur_xcorval; i ++ )
   waddstr(tbptr," ");

/*
 * write percentage done message
*/
memset(msg,'\0',30);
strftime(msg,"%d",percent);
strcat(msg,"% done");
wmove(tbptr, 1, 20);
waddstr(tbptr,msg);

/*
 * update the screen
*/
wrefresh(tbptr);

/*
 * display completion time
*/
if ( cur_xcorval == 50 )
{
   task_completed = TRUE;
   DisplayCompletionTime();
}
return SUCCESS;
}
******************************************************************************
* Name        : DisplayMessage                                       *
* Input       : Pinter to message                                   *
* Returns     : SUCCESS                                              *
* Description : The function displays a given message.              *
* Notes       : 1. The message length must be 50 or less. The function*
*                truncates the message to 50 characters.               *
******************************************************************************
short DisplayMessage( char *msg )
{
  static char message[51];
  static short len, i;

  /*
  * copy first 50 characters of the message
  */
  memset(message,'\0',51);
  strncpy(message,msg,50);
  /*
  * rightpad the message
  */
len = strlen(message);
for ( i = len; i < 50; i++ )
    message[i] = ' ';
message[i] = '\0';
/*
 * move the pointer to the beginning of message window
 */
wmove(mwptr,0,0);
wattroff(mwptr,A_REVERSE);
waddstr(mwptr,message);
/*
 * update the screen
 */
refresh(mwptr);
return SUCCESS;
}

/***********************************************************************
*   Name        : EndTaskbarWidget                                     *
*   Returns     : SUCCESS                                              *
*   Description : The function removes the window structure from the   *
*                   memory.                                            *
*   Notes       :                                                      *
***********************************************************************/
short EndTaskbarWidget ( void )
{
    /*
     * remove the window structure
     */
    endwin();
    return SUCCESS;
}

/***********************************************************************
*   Name        : DisplayCompletionTime                                *
*   Returns     : SUCCESS                                              *
*   Description : The function removes the window structure from the   *
*                   memory.                                            *
*   Notes       :                                                      *
***********************************************************************
void DisplayCompletionTime (void)
{
    char time_now[10];
    char msg[40];
    short i;
    /*
     * make window for clock
     */
    wattron(wp ptr, A_REVERSE);
cw2ptr = subwin(wp ptr, CW2HEIGHT, CW2WIDTH, CW2YCOR, CW2XCOR);
    for ( i = 0; i < (CW2HEIGHT * CW2WIDTH); i++ )
        waddstr(cw2ptr, " ");
```

wrefresh(cw2ptr);
GetTime(time_now);
strcpy(msg,"Completed at ");
strcat(msg,time_now);
wmovex(cw2ptr,0,0);
wattron(cw2ptr, A_REVERSE);
waddstr(cw2ptr,msg);
/*
 * update the screen
 */
wrefresh(cw2ptr);
}

/*******************************************************************************/
/* Name        : GetTime                                              */
/* Input       : Address of a character array                         */
/* Returns     : SUCCESS                                              */
/* Description : The function retrieves the current time and writes it to the address given. */
/*******************************************************************************/
short GetTime (char *l_time )
{
struct tm *ptm;    /* pointer to time structure tm */
long int_time;    /* current time in seconds returned by time() */
ptm = localtime(&int_time);
strftime(l_time,"%02d:%02d:%02d",ptm->tm_hour,ptm->tm_min,ptm->tm_sec);
return SUCCESS;
}

/*******************************************************************************/
/* Name      : task.c                                                   */
/* Overview  : The program illustrates the usage of program taskbar.c. */
/* Notes     : 1. The following interface functions from taskbar.c are called from this program:
   o MakeTaskbarWidget ()
   o UpdateTaskbar ()
   o DisplayMessage ()
   o EndTaskbarWidget ()
   2. Must call EndTaskbarWidget () to re-instate the terminal.
   3. Compile the program as follows:
      cc -o taskbar task.c taskbar.c /usr/lib/libcurses.a
******************************************************************************/

#include <stdio.h>

/*******************************************************************************/
/* INCLUDE FILES                                                             */
*******************************************************************************/
#include <stdio.h>

/*******************************************************************************/
/* FUNCTION PROTOTYPES                                                      */
*******************************************************************************/
```
void main (void);

/******************************************************************************
*                          MODULE CONSTANT
*******************************************************************************/
#define TRUE 1
#define FALSE 0
#define SUCCESS 1
#define FAILURE 0
#define UNIX_SUCCESS 0
#define UNIX_FAILURE 1

/******************************************************************************
*                          GLOBAL VARIABLES
******************************************************************************

/******************************************************************************
*   Name        : main
*   Returns     : SUCCESS
*   Description : The function displays the progress from start to
*                 completion of a specific task using taskbar widget.
*   Notes       :
******************************************************************************

void main ( void )
{
  /* create taskbar widget */
  MakeTaskbarWidget ( );
  /* simulate part of task with system command */
  DisplayMessage("Starting Report 1");
  system("sleep 5");
  /* display 2 percent completed on taskbar */
  UpdateTaskbar (20);
  DisplayMessage("Report 2 completed");
  /* simulate part of task with system command */
  DisplayMessage("Starting Report 2");
  system("sleep 5");
  /* display 10 percent completed on taskbar */
  UpdateTaskbar (40);
  DisplayMessage("Report 2 completed");
  /* simulate part of task with system command */
DisplayMessage("Starting Report 3" );
system("sleep 5");
/*
 * display 50 percent completed on taskbar
 */
DisplayMessage("Report 3 completed" );
UpdateTaskbar (60) ;
/*
 * simulate part of task with system command
 */
DisplayMessage("Starting Report 4" );
system("sleep 5");
/*
 * display 80 percent completed on taskbar
 */
DisplayMessage("Report 4 completed" );
UpdateTaskbar (80) ;
/*
 * simulate part of task with system command
 */
DisplayMessage("Starting Report 5" );
system("sleep 5");
/*
 * display 100 percent completed on taskbar
 */
DisplayMessage("Report 5 completed" );
UpdateTaskbar (100) ;
/*
 * completed the task; remove the taskbar widget
 */
EndTaskbarWidget ( );
}

VIDEO ATTRIBUTES AND CURSES

What exactly is a video attribute? As far as curses is concerned, it is the capability to draw a character on the terminal screen in some way that makes it stand out differently from other characters being displayed. Characters in reverse video, for example, are displayed dark on a light background. This, by the way, is the default highlight mode.

As long as the terminal supports a particular attribute, characters can be displayed in this pseudo-graphic form on the terminal screen using that attribute. It is a bit like being able to add characters to the screen with a paint brush, where the attributes
represent the colours of the paint.

Attributes are represented by constants defined in the include file <curses.h>. These constants are preceded by ‘A_’. Attributes may be combined by ORing them together so that they can be turned on or off as required.

They are as follows:

- **A_STANDOUT** – curses refers to this attribute as being the terminal’s best highlight mode.
- **A_UNDERLINE** – character is displayed on the screen underlined.
- **A_REVERSE** – character is displayed inverse (dark on a bright background). This mode is often referred to as reverse or inverse video mode.
- **A_BLINK** – character added with this attribute will blink on screen. It is often referred to as flash mode.
- **A_DIM** – character added with this attribute is displayed in half-intensity mode.
- **A_BOLD** – character is displayed in high-intensity mode and is brighter than normal.
- **A_NORMAL** – turns off all attributes; character is displayed in normal intensity without underlining.

**RUN-TIME COMPLICATION**

If the program dumps core, make sure you have placed the `initscr()` function in your source code before any other curses function calls.

If the program runs but leaves the terminal in a ‘funny’ state when the program returns to the shell, check that you have called `endwin()` before exiting the program.

Curses takes full control of the terminal that it is running on. If you want to do any special I/O, the curses package provides routines
that will do it for you. This means that you should not use the stdio package directly. If you try to do anything outside curses’ control, it may leave your terminal in an unpredictable state.

FURTHER REFERENCES
The Man pages for curses.

Leaving? You don’t have to give up AIX Update

You don’t have to lose your subscription when you move to another location – let us know your new address, and the name of your successor at your current address, and we will send AIX Update to both of you, for the duration of your subscription. There is no charge for the additional copies.
Veritas Software has announced NetBackup 5.0, Data Lifecycle Manager 5.0, and CommandCentral Service 3.5.

NetBackup 5.0 has new features aimed at speeding up the back-up and data recovery process, including synthetic back-up, which combines smaller back-ups into one reducing recovery time and tape media usage. The Desktop and Laptop Option enables the protection of data on laptops and desktops outside the data centre. Users can also restore their own data.

Data Lifecycle Manager 5.0 is compliance-specific software, designed to handle e-mail and file archiving in Microsoft Exchange and NTFS formats. The software automates the placement and management of data in virtual archives that can span online, nearline, and offline storage media.

CommandCentral Service 3.5 is integrated with NetBackup and Backup Exec applications to create a single management interface. The Web-based portal allows an IT manager to define levels of storage service based on user needs and reports back on those systems for chargeback purposes.

NetBackup 5.0 runs on AIX, HP-UX, Linux, Solaris and Windows platform.

For further information contact: Veritas, 350 Ellis Street, Mountain View, CA 94043, USA. Tel: (650) 527 8000. URL: http://www.veritas.com/products/category/ProductDetail.jhtml?productId=nbupro.

GPFS allows users to run a file system across numerous servers. It provides shared access to the files regardless of what server in particular they are on.

IBM has also added new storage features to GPFS. Users can create a logical copy (or snapshot) of a GPFS file system. In addition, both Linux and AIX clients can tap into features of IBM’s Tivoli SANergy software.

For further information contact your local IBM representative. URL: http://www-1.ibm.com/servers/eserver/pseries/software/whitepapers/gpfs_primer.html.

SeeBeyond Technology has announced a scaled-down version of its eInsight Business Process Manager platform for Web services orchestration. The eInsight Enterprise Service Bus (ESB) 5.0 offering combines native support for Web services, synchronous remote procedure calls, and asynchronous messaging with a publish/subscribe model and standards-based transformation and content-based routing.

The eInsight Enterprise Service Bus is available for AIX, Windows, and other platforms.

For further information contact: SeeBeyond Technology, 800 E Royal Oaks Drive, Monrovia, CA 91016-6347, USA. Tel: 650 622 2100. URL: http://ir.seebe...