The Progress Electronic Magazine

In this issue:

Publisher's Statement:

On the day I sent out word that the E-Zine would require subscriptions, I received the "Emailing with PINE" article from Darrell Woodard. Since he sent it in under the perspective of the E-Zine being freely distributed, I have made this issue of the E-Zine freely distributable.

Darrell's article is a method of using PINE on Tru64 (UNIX) to email messages from a Progress CHUI application. Just think, a year or so ago this was a big problem to people with many questions on how to do it and now there seems to be so many ways to accomplish this!

Also included in the E-Zine is how to set up your Progress database and application server to run as a service in the most UNIX sense of a service. The article will show you how to provide commands commonly used by system administrators and how to automatically start and stop your progress services based on machine reboots, starts, halts, and run-level changes.

To your success,
Scott Auge
Founder, Amduus Information Works, Inc.
sauge@amduus.com

Coding Article: Using PINE to send emails from within CHUI Progress on UNIX

Written by Darrell Woodard

I work for a Utility Organization that provides Electricity and Water to our metro area of about 100,000 customers. We have approximately 35 in-house CHUI Progress databases on Compaq Tru64 UNIX. There are approximately 600 employees and of those approximately 350 of them use at least 1 of these databases and most use multiple databases. Each database may produce anywhere from a few to dozens to hundreds of reports. In order to save a report and view it in a text editor in Windows and/or email it to others a user was required to perform several cumbersome tasks including running the report and saving to a file, running an FTP program to send the file from UNIX to the users PC, and attaching the file to an email and sending the email. The IS director decided that a more user-friendly method was needed. The task of researching and developing a method for including email as an output option for reports from our CHUI Progress databases was assigned to me. Though I had been programming in COBOL and Visual Basic for about 4 years I was relatively new to Progress. This project would allow me to learn a lot through my research and design.

We were running Progress version 8 (now 9.1c) on a Compaq Tru64 UNIX server.

I started looking at several possibilities including sendmail and mailX. These programs however were not very easily adapted to send emails from a command line, send to multiple users, AND include multiple file attachments. I discovered that PINE was already loaded on our Operating System (OS) and decided to investigate it's possibilities.

Pine® (copyright and registered trademark of the University of Washington) - a **P**rogram for **I**nternet **N**ews & **E**mail - is a tool for reading, sending, and managing electronic messages. Pine was developed by Computing & Communications at the University of Washington. It is a popular email program for UNIX and is included with many UNIX and LINUX packages. It is also available for free download from http://www.washington.edu/pine/.

Pine (PC-Pine) is available for Windows PCs as well as Unix but I have not attempted to use the Windows version from within Progress.

The more I read about PINE the more I liked. I found that it was a very robust program and could be configured and manipulated rather easily to fit our needs. The source code was freely available and PINE encouraged users to tweak it (although I later found out that I would not need to do that).

My first problem was that I discovered that the original version of PINE does not allow email to be sent from the command line. However, PINE allows, even encourages, third party patches to be applied to their program. After a little research I found a FREE patch, which fixed the problem.

Links to PINE patches:
http://www.math.washington.edu/~chappa/pine/ Includes links to pages to download patches including the "Send from Command Line" patch.
http://www.math.washington.edu/~chappa/pine/info/outgoing.html - command-line patch web page.

If you are applying this patch for version 4.20 or earlier, you **must** first compile pine, then apply the patch and then compile pine again. Starting from version 4.21, you can apply the patch and then compile Pine. For this reason I recommend installing and compiling version 4.21 or later (of course it is preferable to always use the latest version) before attempting to install the patch.
The PINE command will accept quite a few command-line arguments.
Some PINE arguments overlap with variables in the PINE configuration file. If there is a difference, then a flag set in the command line takes precedence. PINE expects command line arguments (other than addresses) to be preceded by the "-" (dash) as normally used by UNIX programs. I also found the sequence of the command-line arguments to be crucial in some, but not all, cases.
The arguments I found important and useful to my program were:
(As seen at http://www.washington.edu/pine/man/#pine)

[addresses]

Send-to: If you give PINE an argument or arguments which do not begin with a dash, PINE treats them as email addresses. Multiple addresses should be separated with a space between them. Addresses are placed in the "To" field only.

< *file*

PINE will start with *file* read into the body of the message. I eventually allowed the user to enter the body of the message into an editor and then saved this to *file*

-*option*=*value*

Assign *value* to the config option *option*.

OPTIONS USED:

-feature-list=allow-changing-from

feature-list is a list of the many features (options),
which may be turned on or off.
allow-changing-from allows the FROM: header to be manipulated

-default-composer-hdrs=

Show only these headers (by default) when composing a message.

This list may include headers defined in the *customized-hdrs* list.

-customized-hdrs=

Add custom headers when composing.
Also possible to add default values to custom headers or to any of the standard headers. This is a list variable. Each entry in the list is a header name (the actual header name that will appear in the message) followed by an optional colon and value.
Leaving the optional value out allows the Progress Program to fill it in when composing a message.

-I "^X",y

-I = Initial Keystrokes: PINE will execute this comma-separated sequence of commands upon startup. Control keys are two character sequences beginning with ``^", such as ``^X". This particular sequence ("^X",y) tells PINE to send the email from the command-line rather than open the composer.

-subject

Sets the Subject line

-attach *file*

Name a file or files to be attached to message

By using Progress to create user or system updateable variables and create files and the files contents from those variables I would be able to manipulate each of these options/settings to send email(s) from the UNIX command-line using PINE.

I proceeded to run test command-line email sends. One problem I discovered rather quickly had to do with the configuration of PINE. Users who had never used PINE (which was everybody) would be prompted to create certain configuration files. When trying to use the Progress program to send emails this would lock up the user because the user could not respond to the UNIX prompt from within Progress.
The prompt received was:
Press Enter or PF1 to Send Email
[Address book .addressbook doesn't exist, creating]
Folder "sent-mail" doesn't exist. Create?
Y [Yes]
N [No]

The configuration information referred to is stored either in a primary configuration file called .pinerc or in other configuration files in a directory named mail that is created in the users home directory.

My DBA wrote a UNIX script, which solved the problem by copying his PINE configuration file and mail directory (and its files and subdirectories) to each users home directory and then changing the ownership of the files to that user.
Below is an example of the script:

```
/* Unix Script to address PINE Configuration Error */
for i in `ls -l | grep ^d | cut -c 55-70`
do
cd $i
cp /auth/users/.pinerc .
mkdir mail
cp -r /auth/users/mail .
chown $i:pwc .pinerc
chown -R $i:pwc mail
cd ..
done
/* END Unix Script */
```

Once that problem was resolved I was able to successfully send emails from the UNIX command line from several different User Logins. I now had a good, reliable, configurable, and FREE program to send emails from a Unix command line.
Next I needed to find a way to use the program from within Progress that would be user friendly and efficient. I decided to build a basic email interface with Progress. This interface would be similar to most with a To: address line, From: address line, subject line, an address book, a line for attachments, and a message body (text). This interface would be familiar to most users and would require little or no training.

A problem quickly discovered in the Progress programs was that variables or parameters that were declared in the email interface program and shared with other programs would cause errors in running the programs even though the programs compiled correctly. The answer was to run the .p file and allow the programs to compile on the fly.

My main goals for the project were as follows:

1) Common include programs (incspool and incspl2) were already being used to help users choose the type of output desired – send to printer, send to file, or send to terminal (screen). These programs would need to be modified to include email as another option.

If the user chose email as an option then the program would produce the report and then present the user with the email interface with the report file already attached. I wanted to also include the ability to print or save AND email.

2) Users would also be able to bring up the email interface from a menu and email reports that had been saved to file from a previous run of the report program.
3) Users would be able to inspect and edit the subject, the body of the message, the attached files, and the TO: addresses (recipients) before sending the email.
4) Help would be provided to users for selecting report files and/or company email addresses.

1. Modifying the incspool and incspl2 include files:

The primary features of these files to look at for our purposes (running a report and emailing it) are the variables by which the user requests to email report and the code which calls the email interface to present to the user.
incspool and incspl2 are include files that are included in report programs to allow users choice of output - display, print, save to file, and/or email (our new option to be added).
incspool is a common program of include code that would be included at the beginning of a program. It would let users choose output direction (display, print, save to file, and/or email) and different options regarding their choice of output..
incspl2 is really a continuation of incspool which is placed at the end of the report program.. It is basically used to close output, perform actions based on the input from incspool, and clean up temp files.

Each report program includes an updateable variable (PRINTIT), which allows the user to choose the desired output type. The variable is not included in the include file because the variable's label and help may change depending on the report (e.g. certain users may not be allowed to save or email files, etc…). Then the incspool include file (and its incspl2 continuation file) would determine what to do based on the value of the variable.
For example:

```
/* sample code of report program - incomplete */
def var printit as char format "x"
  label "P)rint D)isplay F)ile E)mail" init "P" no-undo.
/* Notice the newly added Email option. If user chose another option the
ability to also email at the same time was added, as we will see later. */
Update
        PRINTIT
            with side-labels frame mgnmtrpt1 no-box.
hide frame mngmtrpt1.
{incspool} /* get user input for more options on selected type of output, open
and prepare output based on value of PRINTIT variable */
/* enter code to produce report here */
```

```
{incspl2} /* close output, perform action to produce desired output, cleanup
files, etc… */
/* end of sample code of report program - incomplete */

/* Begin incspool include file - incomplete sample code */
/* Would include all information regarding output that is determined by program
like unique filenames and allow users to update options like printer, page
size, number of copies, etc… */

/* New variable which user may use to email report */
def new global shared var ws-email as logical format "y/n" init false.

/* new variable which is automatically filled in with the report filename to
email. Large size of 500 characters allows for multiple filenames (including
full path) to be placed in variable */
DEFINE new shared VARIABLE ws-attach   AS CHARACTER FORMAT "X(500)"
                    view-as fill-in size 58 by 1
                    LABEL "Attachments" INITIAL "".

/* code for calculating unique filename for new report (wsv-loginpath) etc… */

/* if user chooses to Print to printer */
if printit = "P" and opsys = "unix" then do:
ws-email = false.
/* ask if user would also like to email as well print */
  update
/* New email variable allows user to choose to email report in addition to
printing it */
ws-email label "Email Y/N?"
  help "Enter Y to email this report"
  with side-labels frame frmincspl40
  title "Fast Print Management" centered row 10.
hide frame frmincspl40.

/* If email is chosen then preset page-size for text editor compatibility */
    if printit = "E" or ws-email = true then wsv-page-size = 56.
    output to  value(wsv-loginpath) page-size value(wsv-page-size).

/* Preload attachment filename for email so when email interface first displays
new report is already attached */
    ws-attach = wsv-loginpath.
end. /* printit = p */

/* if user chooses to save to file */
else if printit = "F"  then
fileblk:
do on error undo fileblk, retry fileblk:
       /* do similar code as above */
```

```
end. /* printit = f  */

/* if email is selected as main option */
else if printit = "E"  then
emailblk:
do on error undo emailblk, retry emailblk:
    ws-email = true.
    ws-pfile = "".
/* ask user what to name the file to email */
    update
    ws-pfile format "x(8)" label "Email Filename"
    help "Enter the name of the email file ."
    validate(ws-pfile <> "", "Email file cannot be blank *") skip
/* does user want to also save file for later reference */
    ws-save  label " Save Y/N?"
    help "Enter Y to save this report" skip
        with side-labels frame frmincspl400
    title "Fast Print Management" centered row 10.
/* set filename and put in users home directory */
    ws-pfile = os-getenv("HOME") + "/" + ws-pfile + ws-use-date.
    hide frame frmincspl400.

/* if file already exists then ask to overwrite */
    if search(ws-pfile) <> ? then do:
        BELL.
        message "Print file already exists.  Overwrite (y/n)?" Update ws-ans.
        if not ws-ans then undo emailblk, retry emailblk.
    end. /* file search */
    if printit = "E" or ws-email = true then wsv-page-size = 56.
    display "Creating email file ..." with frame frmincspl333.
    output to value(ws-pfile) paged.
/* preload filename in email interface */
    ws-attach = ws-pfile.
end. /* printit = e */
/* End incspool include file sample code */

/* incspl2 include file - final output routine for printing/saving/emailing a
report */
def shared var ws-save  as logical format "Y/N".
def shared var ws-email as logical format "Y/N".
def var wsv-print-save as char format "x(50)" init " " no-undo.

/* based on users selected output perform an action */
if printit = "P"   then do:
/* print file to printer */
  output close.
  wsv-print-save = " -c -s -d " + lower(wsv-quname)  .
  wsv-print-save = wsv-print-save.
```

```
  unix silent set value(wsv-print-save).
  unix silent  lp  value(wsv-print-save) value(wsv-loginpath).
end.
output close.

/* email report before deleting files if save = NO */
if ws-email then do:
/* if user chose to email report then run email program */
    run epbcnct.p . /* connects to EPB - Employee Phone Book - A Progress
database of all employees including valid company email addresses for each
employee */
    run coespool.p . /* presents user with a simple Progress CHUI email
interface preloaded with selected file attachment to email */
    run epbdcnt.p . /* disconnect user from EPB DB */
end.

if printit = "P" then do:
  if not(ws-save) then unix silent rm value(wsv-loginpath).
  ws-save = false.
end. /* printit = "P" */

if printit = "E" then do:
  if not(ws-save) then unix silent rm value(ws-pfile).
  ws-save = false.
end. /* printit = "E" */
/* End of incspl2 include file */
```

This code:

- Allows the user to choose to run and email a report
- Loads the filename of the report into a variable so the user does not have to find the report on his own
- Presents the user with an email interface (see below) with default email addresses and report filenames preloaded

2. Email reports that had been saved to file from a previous run of the report program:

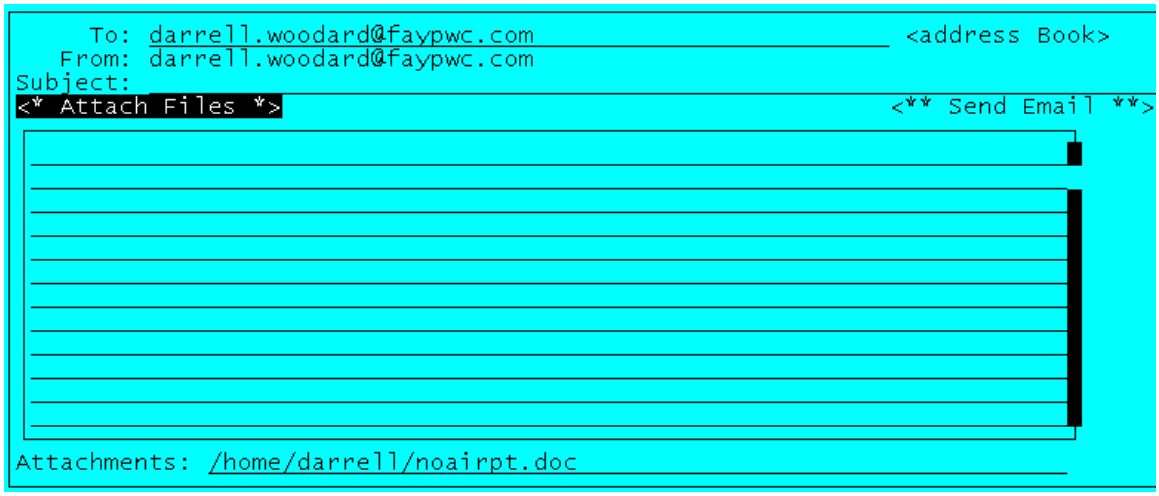
Next I had to actually build the program that would present the user with an email interface. This interface would be called from a menu to email previously saved reports using Pine.
The major obstacle I ran into when designing this application was making a stand-alone application (coemailpine.p) and also relying on values from other variables in the include files above. I finally decided the easiest and fastest solution was to use two separate but almost identical programs. The only difference was that one would use a shared variable while the stand-alone application would use the same variable name but define it as new. I am sure there are other means of accomplishing this feat but I was under a deadline and had only been programming in

Progress for a few months at the time. Currently the program has been working without maintenance for over a year and it seems ridiculous to “fix it if it ain’t broke”.

3. Users would be able to inspect and edit the subject, the body of the message, the attached files, and the TO: addresses (recipients):

coemailpine.p – Main program for creating basic email interface (From:, To:, Subject:, Message Text (body), attachments) within progress. The user may edit any of the fields except the From: field, which is determined by the system. The sister program coemailspool.p is used to send reports which are created on the fly. The only difference between the two programs is that the latter contains a shared variable (ws-attach) which is created in the incspool include file. This shared variable allows the program to preload the file attachment for the user. The user may also add (more) filenames (reports) to be attached that had previously been saved in the users home directory. The User may attach files by typing in the filename (including full path) or browsing home directory for files (see cofbrhlp.p below).

The email interface is the heart of the PINE with Progress emailing system. This is where the user actually sees the email and interacts with it USING PROGRESS before sending the email out. The user can do almost anything from this interface that can be done from a normal basic email interface.



CHUI PROGRESS Email Interface

When the user chooses the <** Send Email **> button the program performs the UNIX Command-Line Function to send the email using PINE.

Program Design:

```
/* Begin coempine.p - Email interface program */
/*************************************************************************
Program: coempine.p
```

```
By: Darrell Woodard
Descript: Create standard email interface and allow users to email existing
files in users home directory using Pine - also see sister program
coemailspool.p which preloads an attachment.
Because of variables shared with internal or external Procedures -
run  .p (do not compile) and allow programs to compile on the fly
*****************************************************************/
/* define variables to allow user maintainance of email and it's headers */
/* variable to maintain TO: addresses (recipients) */
def new shared var ws-emailaddr AS CHARACTER FORMAT "X(500)"
                   view-as fill-in size 50 by 1
                   LABEL "To"   INITIAL "toaddress@tocompany.com".
def            var ws-to       AS CHARACTER FORMAT "X(68)"
                   LABEL "To"   INITIAL "toaddress@tocompany.com".

/* variables to maintain FROM: address */
def            var ws-from     AS CHARACTER FORMAT "X(68)"
                   LABEL "From" INITIAL "myaddress@mycompany.com" NO-UNDO.
def            var ws-fromhold AS CHARACTER FORMAT "X(68)"
                   LABEL "From" INITIAL "myaddress@mycompany.com" NO-UNDO.
def            var ws-frompine AS CHARACTER FORMAT "X(68)"
                   LABEL "From" INITIAL "myaddress@mycompany.com" NO-UNDO.

/* variable to maintain SUBJECT: line */
def            var ws-subject  AS CHARACTER FORMAT "X(70)"
                   LABEL "Subject" INITIAL "" NO-UNDO.

/* variable to maintain body (text) of message */
def            var ws-body     AS CHARACTER
                   VIEW-AS EDITOR INNER-LINES 12 INNER-CHARS 70
                   SCROLLBAR-VERTICAL INITIAL "" NO-UNDO.

/* variable to maintain file attachments */
/* ws-attach is a shared (not NEW shared) variable in the coemailspool.p
program. This allows the filename to be loaded into the variable for the user.
By defining it as NEW in this program it allows the program to stand alone and
be called from a menu etc… */
def new shared var ws-attach   AS CHARACTER FORMAT "X(500)"
                   view-as fill-in size 58 by 1
                   LABEL "Attachments" INITIAL "".

/* buttons on email interface to perform actions */
def            BUTTON   btnaddr     Label "address Book".
def            BUTTON   btnattach   Label "* Attach Files *".
def            BUTTON   btnsend     LABEL "** Send Email **".

/* stream defined for getting senders email address */
def            STREAM   getname.
```

```
/* streams defined for actually sending email */
def            STREAM   outputsend.
def            STREAM   outputpine.

/* variables for setting senders email address */
def new shared var ws-fn       AS CHAR FORMAT "x(25)".
def new shared var ws-ln       AS CHAR FORMAT "x(25)".

def            var ws-sent     as log init no.

/* variables for deterining # of attachments */
def            var ws-ccnt as int.
def            var ws-fcnt as int.
def            var ws-numattach as char format "x(50)" extent 15 .

/* FORM definition - define email interface */
FORM
       ws-emailaddr    COLON 8
              HELP "Space between emails, PF2 for help, TAB to go to next
field."
           btnaddr
           SKIP
           ws-from        COLON 8
             HELP "Enter your email address,example:myaddress@mycompany.com"
           SKIP
           ws-subject format "x(68)"    COLON 8
                    help "Enter the subject of this email"
           skip
           btnattach
             help "Press Enter or PF1 to attach a file or Tab to go to next
field"
           space(41)
           btnsend
             help "Press Enter or PF1 to Send Email"
           SKIP
           ws-body
             help "Enter the text (body) of the email" NO-LABEL
           ws-attach colon 12
                    help "Enter full filename (including path)"
                           WITH FRAME sendmail SIDE-LABEL WIDTH 80 overlay.

hide all no-pause.
/* REDUNDANCY (searches for employee 4 different ways) CHECK FOR EMPLOYEE EMAIL
ADDRESSES */
/* while my company used a progress DB for maintaining valid email addresses
for employees, here other methods could be used such as checking exchange, etc…
*/
```

```
/* find users email address in employee phone book for IS Employee */
find first empphon where username = userid(ldbname(1)) no-error.
/* valid user has valid email address */
if available empphon and pwceaddr <> "" then
       ws-from = lc(pwceaddr) + "@faypwc.com". /* if 1 */
else
do: /* else 1 */
       /* find users email address in employee phone book for all other
employees */
             find first _user where _user._userid = userid(ldbname(1)) no-
error.
             if available _user then
                    find first empphon where
                           empphon.fn = entry(1, _user._user-name, " ") and
                           empphon.ln = entry(2, _user._user-name, " ") no-
error.
             /* valid user & has valid email address */
             if available empphon and pwceaddr <> "" then
             ws-from = lc(pwceaddr) + "@faypwc.com". /* if 2 */
             else
       do: /* else 2 */
             find first _user where _user._userid = userid(ldbname(1)) no-
error.
                    if available _user then /* if 3 */
             /* cant find user's valid company email address so use users first
             and last name */
                                  ws-from = lc(entry(1, _user._user-name, " "))
+ "." +
                                        lc(entry(2, _user._user-name, " ")) +
"@faypwc.com".
                    else
             do: /* else 3 */
                    /* use UNIX logon name as last resort */
                                  input stream getname thru whoami.
                                  import stream getname ws-from.
                                  input stream getname close.
                                  ws-from = lc(ws-from) + "@faypwc.com".
                    end. /* else 3 */
             end. /* else 2 */
end. /* else 1 */

/* set program to default to email report to self - this little piece of code
really pleased users. Most wanted to inspect the report, tweak it, or add to it
before sending to others */
ws-emailaddr = ws-from.

/* Triggers */
```

```
/* btnattach runs a program that allows user to browse files in home directory
and select
    multiple files to attach to email and auto load them into the ws-attach
variable
    - see below */
on F1,PF1,Enter of btnattach apply "choose" to btnAttach.

ON 'CHOOSE':U OF btnattach
DO:
       run cofbrhlp.p .
              ASSIGN ws-subject ws-emailaddr.
            DISPLAY ws-emailaddr ws-from ws-subject ws-body ws-attach
              WITH FRAME sendmail overlay.
              ENABLE ALL except /* btnattach */ ws-from btnsend
              WITH FRAME sendmail overlay.
            ENABLE /* btnattach */ btnsend WITH FRAME sendmail overlay.
END.

/* btnaddr runs a program that allows user to browse and select valid company
email
    addresses to be auto loaded into the TO: variable - see below */
on F1,PF1,Enter of btnaddr apply "choose" to btnAddr.

ON 'CHOOSE':U OF btnaddr
DO:
       run coeadhlp.p .
              DISPLAY ws-emailaddr
              WITH FRAME sendmail overlay.
              ENABLE ALL except /* btnattach */ ws-from btnsend
              WITH FRAME sendmail overlay.
              ENABLE /* btnattach */ btnsend WITH FRAME sendmail overlay.
END.

/* btnsend actually runs the procedures which send the email */
on F1,PF1,Enter of btnsend apply "choose" to btnsend.

ON 'CHOOSE':U OF btnsend
DO:
       ASSIGN ws-emailaddr ws-from ws-subject ws-body /* ws-attach */ .
              ws-fromhold = ws-from.
              ws-frompine = ws-from.
            /* if no attachment use procedure sendmail (not the UNIX command -
still send
           email using PINE) else use procedure Pinemail */
              if ws-attach = "" then
       do:
```

```
              /* send variable values to system file which will be input into
       PINE
                  command-line parameters */
              OUTPUT STREAM outputsend to value("/tmp/" + ws-fromhold).
              /* message "Sendmail" view-as alert-box. */
              /* ws-to = "To: " + ws-emailaddr.
                       ws-from = "From: " + ws-from.
                  ws-subject = "Subject: " + ws-subject. */
                     DISPLAY STREAM outputsend ws-body
                            with no-box no-labels frame frms.
                     OUTPUT  STREAM outputsend close.
                     RUN sendmail.
                     os-delete value("/tmp/" + ws-fromhold).
       end.
              else
       do:
              /* send variable values to system file which will be input into
       PINE
                  command-line parameters */
                     OUTPUT STREAM outputpine to value("/tmp/" + ws-frompine).
              /* message "Pinemail" view-as alert-box. */
              DISPLAY
                     STREAM outputpine
                     ws-body
                     skip
                     "Reports may need WIDE PAPER or printer set to condensed
                     format"
                                  skip
                     "attachments:"
                     skip
                                        with no-box no-labels frame frmp.
                     /* count # of attachments */
              repeat ws-ccnt = 0 to length(ws-attach).
                           if substring(ws-attach,ws-ccnt + 1,1) = " " then
                           ws-fcnt = ws-fcnt + 1.
              end.

              /* tell Pine how many attachments and what the attachments are */
              repeat ws-ccnt = 1 to ws-fcnt:
                        ws-numattach[ws-ccnt] = " -attach " + entry(ws-ccnt,ws-
attach," ").
              end.
                     repeat ws-ccnt = 1 to 15:
                     if ws-numattach[ws-ccnt] <> "" then
                                        DISPLAY
                                 STREAM outputpine
                                        ws-numattach[ws-ccnt]
```

```
                                                        with no-box no-labels
frame ws-ccnt.
              end.
                     OUTPUT  STREAM outputpine close.
              RUN pinemail.
              os-delete value("/tmp/" + ws-frompine).
            end.
            hide all no-pause.
            message "Email Sent" view-as alert-box.
END.

Main-Block:
DO ON ENDKEY UNDO, LEAVE:
            DISPLAY ws-emailaddr ws-from ws-subject ws-body
                    WITH FRAME sendmail overlay.
              message "If Your From: email address is incorrect then"
              skip
                     "You need to update the Employee Phone Book" .
       on value-changed of ws-emailaddr in frame sendmail
       do:
              ws-emailaddr = ws-emailaddr:screen-value.
       end.
       on value-changed of ws-attach in frame sendmail
       do:
              ws-attach = ws-attach:screen-value.
       end.

              ENABLE ALL except ws-from btnsend ws-attach
              WITH FRAME sendmail overlay.
            ENABLE  btnsend
              WITH FRAME sendmail overlay.
            WAIT-FOR WINDOW-CLOSE OF CURRENT-WINDOW or choose of btnsend.
       hide all no-pause.
END.
message "" . pause 0.
hide all no-pause.

PROCEDURE sendmail:
/* ws-subject = "\"" + ws-subject + "\"". */ /* this line worked on unix but
not win98 */
     ws-subject = '\"" + ws-subject + "\"'. /* this line worked on win98
appbuilder */
     ws-from = "-customized-hdrs=From:" + ws-from .

       /* the actual UNIX command-line PINE command */
       unix silent pine -feature-list=allow-changing-from
       -default-composer-hdrs=From
       value(ws-from)
```

```
       value(ws-emailaddr) value("</tmp/" + ws-fromhold)
       -I "^X",y -subject value(ws-subject) .
END.

PROCEDURE pinemail:
/* ws-subject = "\"" + ws-subject + "\"". */ /* this line worked on unix but
not win 98 */
     ws-subject = '\"" + ws-subject + "\"'. /* this line worked on win98
appbuilder */
     ws-from = "-customized-hdrs=From:" + ws-from .

       /* allow up to 15 attachments */
       /* the actual UNIX command-line PINE command */
       unix silent pine  -feature-list=allow-changing-from
           -default-composer-hdrs=From
           value(ws-from)
           value(ws-emailaddr) value("</tmp/" + ws-frompine)
              value(ws-numattach[1])
                    value(ws-numattach[2])
                    value(ws-numattach[3])
              value(ws-numattach[4])
                    value(ws-numattach[5])
              value(ws-numattach[6])
                    value(ws-numattach[7])
              value(ws-numattach[8])
                    value(ws-numattach[9])
              value(ws-numattach[10])
                    value(ws-numattach[11])
              value(ws-numattach[12])
                    value(ws-numattach[13])
              value(ws-numattach[14])
                    value(ws-numattach[15])
            -I "^X",y -subject value(ws-subject).
END.
/* End coempine.p - Email interface program */
```

coemailspool.p – same as coemailpine.p above except it uses a shared variable (ws-attach) with incspool to automatically load attachment filename into email interface when user selects type of output wanted from program (incspool).

The email interface was successful. It allowed users to email reports using PINE while being simple and user-friendly.
Next I would provide help for users in selecting attachments and email recipients.

4. Help would be provided to users for selecting report files and/or company email addresses:

Though this section is not directly related to the subject of using PINE to send emails it does relate to the email interface and making it user-friendly.
All report programs save reports to the users home directory by default. Therefore I wanted to offer users a way to see and choose reports in the users home directory.
cofbrhlp.p – program creates browser that allows user to browse files in home directory and select files (multiple selection enabled) to attach to email.

```
/* Begin cofbrhlp.p */
/*********************************************************************
Program: cofbrhlp.p
By: Darrell Woodard 01-20-2001
Descript: File browser for users home directory allows multiple selections
*********************************************************************/
def             var ws-file-size        as char FORMAT "X(10)".
def             var ws-date             as char.
def             var wsv-loginpath as char format "x(50)".
def             var ws-fn1cnt           as int.
def             var ws-fn2cnt           as int.
def             var ws-fscnt            as int.
def     shared  var ws-attach    as char FORMAT "X(500)"
                    LABEL "Attachments" INITIAL "".
def new shared  var ws-attach1 as char FORMAT "X(500)"
                    LABEL "Attachments" INITIAL "".
def             var ws-i                as int.
def             var v-i                 as int.
def             var v-ok                as log.
def             var ws-action           as char format "X" init "P" no-undo.
def             var ws-period           as int  init 1 no-undo.
def             var ws-slash            as char init "" no-undo.
def             var ws-position         as int  init 1 no-undo.
def             var ws-x                as int  init 1 no-undo.
def             var ws-length           as int  init 1 no-undo.
def             var ws-mnth             as char format "x(2)".
def             var ws-day              as char format "x(2)".
def             var ws-yr               as char format "x(2)".
def             var ws-file             as char format "x(20)" label "Report".
def             var ws-time             as char label "Time".
def             var ws-line             as char format "x(78)".
def             var ws-home             as char format "x(50)".
def             var ws-nomnth           as char format "x(3)".
def             var ws-noday            as char format "x(2)".
def             var ws-notime           as char label "Time".

def temp-table print-file
  field tt-mnth as char
```

```
  field tt-date as char
  field tt-file   as char format "x(40)" label "Name"
  field tt-file-size as char FORMAT "X(10)"
  field tt-time as char  label "Time"
  field tt-full-file as char format "x(60)"
  field slct as log init no
  index tt-full-file is unique tt-full-file
  index idx-tt-file tt-file tt-date tt-time.

def temp-table print-file2
    field tt-full-file2 as char format "x(80)"
    index tt-full-file2 is unique tt-full-file2 .

find first epb.company where epb.company.comp# = 1
                                                    no-lock no-error.
output to terminal.
ws-home = os-getenv("HOME") + "/".
wsv-loginpath = os-getenv("HOME") + "/*.__*".

def             query q-print-file for print-file scrolling.
def             browse b-brws query q-print-file
display
    tt-file
    tt-date label "Created"
    tt-time
    tt-file-size  label "Byte Size"
        with  12 down col 1 row 1
            title "Email Attachment" centered overlay multiple .

def frame f-test
 b-brws
 help "Use Spacebar to Select Files/PF1 when selection Complete"
    with overlay.
/* Create Browser for user to select files to Print/Delete */

message "PWC Email limits attachments to 1 mb (1,000,000 bytes)"
                                                 view-as alert-box.
/* delete any previous record */
for each print-file:
  delete print-file.
end.

/* Get files in Users home directory which contain *.doc string */
wsv-loginpath = os-getenv("HOME") + "/*.doc".
input through ls -l value(wsv-loginpath) | cut -c 33-100 no-echo.

repeat:
  set ws-file-size ws-nomnth ws-noday ws-notime ws-line
```

```
        with frame vg0000 no-box no-label no-error.
  if ws-file-size = "" then next.

  ws-length = length(ws-line).
  if ws-nomnth = "jan" then ws-mnth = "01".
  if ws-nomnth = "feb" then ws-mnth = "02".
  if ws-nomnth = "mar" then ws-mnth = "03".
  if ws-nomnth = "apr" then ws-mnth = "04".
  if ws-nomnth = "may" then ws-mnth = "05".
  if ws-nomnth = "jun" then ws-mnth = "06".
  if ws-nomnth = "jul" then ws-mnth = "07".
  if ws-nomnth = "aug" then ws-mnth = "08".
  if ws-nomnth = "sep" then ws-mnth = "09".
  if ws-nomnth = "oct" then ws-mnth = "10".
  if ws-nomnth = "nov" then ws-mnth = "11".
  if ws-nomnth = "dec" then ws-mnth = "12".
/* ws-mnth = "". substring(ws-line,(ws-length - 15),2). */
  ws-day  = ws-noday. /* substring(ws-line,(ws-length - 13),2). */
  ws-yr   = substring(string(year(today)),3,2).
    /* substring(ws-line,(ws-length - 11),2). */
  ws-time = ws-notime + ":00". /* substring(ws-line,(ws-length - 9),2) + ":" +
          substring(ws-line,(ws-length - 7),2) + ":" +
          substring(ws-line,(ws-length - 5),2). */
    repeat ws-fn1cnt  = 1 to length(ws-file).
        if substring(ws-file,ws-fn1cnt,1) = "/" then ws-fn2cnt = ws-fn1cnt.
    end.
    ws-fn2cnt = ws-fn2cnt + 1.
  /* get postion of "/"  */
  do ws-x =  1 to (ws-length - 1):
    ws-slash = substring(ws-line,(ws-length - ws-x),1).
    if ws-slash = "/" then  do:
      ws-position = ws-x.
      leave.
    end. /* ws-slash */
  end. /* do ws-x */
  /* locate position of the "." (period) */
  ws-slash = "".
  ws-period = ws-length.
  do ws-x =  1 to (ws-length - 1):
    ws-slash = substring(ws-line,(ws-length - ws-x),1).
    if ws-slash = "." then  do:
        ws-period = ws-x.
    end. /* ws-period */
    if ws-slash = "/" then
        leave.
  end. /* do ws-x */
  ws-file = substring(ws-line,(ws-length - ws-position + 1),
         (ws-position + 1) - (ws-period + 2)) .
```

```
  find first print-file where tt-full-file =
    substring(ws-line,(ws-length - ws-position + 1), (ws-position)) no-error.
  if not available print-file then do:
  create print-file no-error.
  assign  tt-date  = ws-mnth + "/" + ws-day + "/" + ws-yr
  tt-file = ws-file
  tt-time = ws-time
  /* get full name of file for unix os print/delete */
  tt-full-file = substring(ws-line,(ws-length - ws-position + 1),
    (ws-position))
  tt-file-size = ws-file-size no-error.
end.
end. /* repeat */

input close.

wsv-loginpath = os-getenv("HOME").

 on iteration-changed of b-brws in frame f-test do:
   ws-fscnt = 0.
   do v-i = 1 to self:num-selected-rows:
     v-ok = self:fetch-selected-row(v-i).
     print-file.slct = yes /* print-file */ .
   end.   /* v-i  = 1 */
    main-for-each2:
    for each print-file where slct = yes .
        if int(tt-file-size) < epb.company.esizelmt then do:
            if ws-fscnt + int(tt-file-size) > epb.company.esizelmt then do:
                message "Files selected exceed size limits for PWC Email"
                skip "File" tt-file " will not be attached".
                slct = no.
                leave main-for-each2.
            end.
        end.
        else do:
            message "File" tt-file "is too large to email" view-as alert-box.
            slct = no.
        end.
        if int(tt-file-size) < epb.company.esizelmt then
            ws-fscnt = ws-fscnt + int(tt-file-size).
    end. /* for each */
end. /* on iteration-changed */

 on "GO" of b-brws in frame f-test do:
   ws-fscnt = 0.
   do v-i = 1 to self:num-selected-rows:
     v-ok = self:fetch-selected-row(v-i).
     print-file.slct = yes /* print-file */ .
```

```
   end.   /* v-i  = 1 */
    ws-i = 0.
        ws-attach1 = ws-attach.
    main-for-each:
    for each print-file where slct = yes .
        if int(tt-file-size) < epb.company.esizelmt then do:
            if ws-fscnt + int(tt-file-size) > epb.company.esizelmt then do:
                message "Files selected exceed size limits for PWC Email"
                skip "File" tt-file "and remaining files will not be attached".
            leave main-for-each.
            end.
            ws-fscnt = ws-fscnt + int(tt-file-size).
            ws-i = ws-i + 1.
            if ws-i = 1 and ws-attach1 = "" then
                ws-attach1 = wsv-loginpath + "/" + print-file.tt-full-file .
            else
            ws-attach1 = ws-attach1 + " " + wsv-loginpath +
                "/" + print-file.tt-full-file .
        end.
        else message "File" tt-file "is too large to email" view-as alert-box.
    end. /* for each */
ws-attach = ws-attach1.
    apply "close" to this-procedure.
end. /* on "GO" */
main:
do:
hide all no-pause.
open query q-print-file for each print-file.
enable all with frame f-test overlay title "Select".
wait-for close of this-procedure.
end.
output close.
/* End cofbrhlp.p */
```

coeadhlp.p – help program that allows user to browse company email addresses stored in Employee Phone Book (EPB) DB. Allows multiple selection of email addresses for TO: box. A similar program could be created for searching an exchange server or other programs or databases.

```
/* Begin coeadhlp.p */
/*************************************************************************
Program: coeadhlp.p
By: Darrell Woodard 01-20-2001
Descript: Help file for employee email addresses from emp phone book DB
*************************************************************************/
```

```
def var v-cntr                      as      integer.
def var primaryid                   as      recid extent 24.
def                 var ws-recid        as      recid.
def        shared   var ws-emailaddr AS CHARACTER FORMAT "X(500)"
                    LABEL "To"   INITIAL "toaddress@tocompany.com".
def    new shared   var ws-emailaddr1 AS CHARACTER FORMAT "X(500)"
                    LABEL "To"   INITIAL "toaddress@tocompany.com".
def                 var ws-i as int.
def                 var v-i as int.
def                 var v-ok as logical.

def temp-table email-addr
  field email       like empphon.pwceaddr
  field email-fn    like empphon.fn
  field email-ln    like empphon.ln
  field slct as log init no
  index idx-email email-ln email-fn email.

output to terminal.
def             query q-email-addr for email-addr scrolling.
def             browse b-brws query q-email-addr
display
        email-addr.email-ln
        email-addr.email-fn
        email-addr.email
            with  10 down col 1 row 1
            title "Email Addresses" centered overlay multiple .
def frame f-test
 b-brws
 help "Spacebar to Select Email Recipients/PF1 when selection Complete"
    with overlay.
for each empphon where empphon.pwceaddr <> "":
    create email-addr.
        email-addr.email    = empphon.pwceaddr.
        email-addr.email-fn = empphon.fn.
        email-addr.email-ln = empphon.ln.
end.

 on any-printable
    of b-brws in frame f-test do:
    find first email-addr where email-ln begins keylabel(lastkey) no-error.
    if available email-addr then
        reposition q-email-addr to recid(recid(email-addr)).
    else message "No Last Name begins with " keylabel(lastkey).
 end.

 on "GO" of b-brws in frame f-test do:
   do v-i = 1 to self:num-selected-rows:
```

```
     v-ok = self:fetch-selected-row(v-i).
     email-addr.slct = yes /* email-addr */ .
   end.   /* v-i  = 1 */
    ws-i = 0.
    ws-emailaddr1 = ws-emailaddr.
    for each email-addr where slct = yes .
        ws-i = ws-i + 1.
        if ws-i = 1 and ws-emailaddr1 = "" then do:
        if email-addr.email <> "" then
            ws-emailaddr1 = lc(email-addr.email) + "@faypwc.com".
        else
            ws-emailaddr1 = lc(email-addr.email-fn) + "." +
                            lc(email-addr.email-ln) + "@faypwc.com".
        end.
        else do:
        if email-addr.email <> "" then
        ws-emailaddr1 = ws-emailaddr1 + " " + lc(email-addr.email)
                                                          + "@faypwc.com".
        else
        ws-emailaddr1 = ws-emailaddr1 + " " + lc(email-addr.email-fn) +
                       "." + lc(email-addr.email-ln) + "@faypwc.com".
        end.
    end. /* for each */
    ws-emailaddr = ws-emailaddr1.
/*    frame-value = frame-value + ws-emailaddr1. */
hide all no-pause.
    apply "close" to this-procedure.
end.
main:
do:
hide all no-pause.
message "Use Up/Down arrows or Page Up/Down to browse names".
message "Press first letter of last name to quick jump".
open query q-email-addr for each email-addr.
enable all with frame f-test overlay title "Select".
apply "iteration-changed" to b-brws.
wait-for close of this-procedure.
end.
hide all.
/* End coeadhlp.p */
```

About The Author:

Darrell Woodard was a career retail manager. In an effort to get out of the Retail Environment and make a career in another area, he joined the Army in 1994 at the ripe old age of 31. He had never been on a computer but knew what everyone was talking about and decided that the future of business was in the

technical fields. He spent all five years of his Army career with the 18th Personnel Service Battalion (PSB) at Fort Bragg, NC. as a programmer. He is self-taught in all his programming languages. He programmed in COBOL, Access, and Visual Basic as well as dealing with other software and hardware issues. In 1999, he left the Army and accepted a position with his current employer, Public Works Commission (PWC) of Fayetteville, NC as a Progress Programmer. He atended a 4-day getting started with Progress Class but is self-taught in CHUI Progress for the most part as well. He is responsible for the maintenance of approximately 35 existing Progress databases and Planning, design, and creation of new databases. He is also responsible for converting CHUI databases to GUI format. He LOVES his job and PROGRESS!

Administration Article: Making Progress a service on UNIX

Written by Scott Auge sauge@amduus.com

When one reboots a computer, it is proper for the database to be properly shut down and the application servers, if available, to be shutdown. With application servers and Webspeed, there is an order this should be done under. Also most system administrators are familiar with the "service" command, the "chkconfig" command, and "ntsysv" tool for starting and stopping services on a UNIX based computer. Making these changes will make your Progress application and database system easily understood by UNIX system administrators and administered via commonly known means.

This article focuses on how to write scripts for the /etc/rc.d/rc*.d directories and /etc/init.d directory on Linux. Linux allows both System V and BSD style start up and shutdown scripts for services and this article shows how both can be accomplished.

Using common UNIX administration commands on Progress:

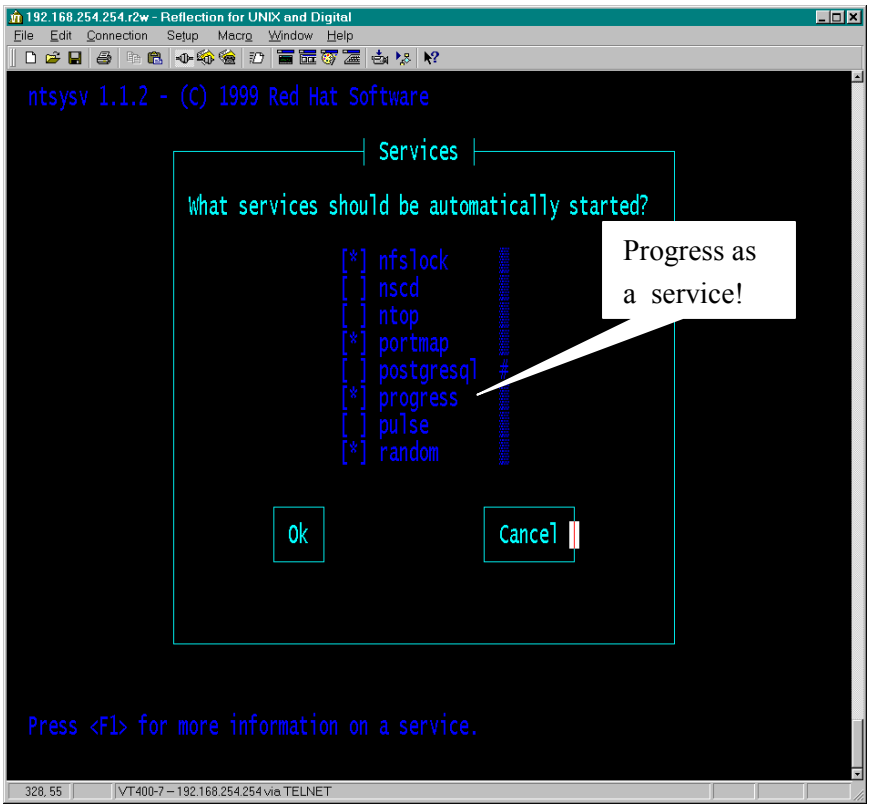
We wish to accomplish Progress databases and applications to be administered by system administrator's familiar with `service` *`servicename`* `option` commands similar to those below:

```
service progress start
service progress stop
service progress restart
```

Additional services can be made up like `prgmydb` and `prgmyappsvr` to split up the administration of different parts of the Progress application. Starting and stopping a database or

appserver will become as simple as starting and stopping a web server for the system administrator.

On Linux, you will see that Progress will appear in the ntsysv config tool:



as well is with the chkconfig tool:

```
# chkconfig --list progress
progress        0:off   1:off   2:off   3:on    4:off   5:off   6:off
```

The main service script

The main service script provides the start and stopping of services on the machine. This script is located in the init.d directory and is named as the service you wish to provide. Hence, if you wish to have a service named “progress” you should name the script “progress” – as well if you wanted a progress based application to act as a service to be called “portal” than you should call the script “portal.”

As in this example, it actually calls out to other scripts that are tailored to the application – hence when the administrator manipulates these scripts, it effects only the application under question. The benefits of modularity also plays a role in doing this.

```
#!/bin/sh
#
# Startup script for progress goodies
#
# chkconfig: 345 85 15
# description: Progress database servers



# Source function library.
. /etc/rc.d/init.d/functions

# Need some environmental varibles available for this to work

DLC=/usr/wsrt
export DLC

# See how we were called.
case "$1" in
  start)
       echo "Starting Progress: "
       echo "Starting Portal 0001 (Amduus Web Site) DB Server "
       /db/prod/clients/portal/0001/script/startdb.ksh
       echo "Starting Admin Server "
       $DLC/bin/proadsv -start
       echo "Starting Portal 0001 (Amduus Web Site) Broker "
       $DLC/bin/wtbman -i portal -start

       echo
       ;;
  stop)
       echo "Shutting down Progress: "
       echo "Stopping Portal 0001 (Amduus Web Site) Broker "
       $DLC/bin/wtbman -i portal -stop
       echo "Stopping Admin Server "
       $DLC/bin/proadsv -stop
       echo " Stopping Portal 0001 (Amduus Web Site) DB Server "
       /db/prod/clients/portal/0001/script/startdb.ksh
       echo
       ;;
```

```
  status)
      ps -ef | grep pro
      ;;
  restart)
      $0 stop
      $0 start
      ;;
  reload)
      ;;
  *)
      echo "Usage: $0 {start|stop|restart|reload|status}"
      exit 1
esac

exit 0
```

Note how the DLC environmental variable is set in the script so the Progress commands and attributes are available via common invocation.

The script is broken up into 5 main sections[1]: start, stop, restart, status, and the default. Under this implementation, the reload is not really needed because when one stops a progress session/server – it should be removed from memory automatically.

Note in the start section of the script, there is a specific order to the processes started. In the example given, there are three main progress components started: the database, the administration server, and a Webspeed broker. As you may understand, one should have the DB running before anything can attach to it. Next to start the Webspeed broker, one needs to have the administration server running. Finally we start the Webspeed broker.

The same goes for the stop section. The order of stopping processes related to Progress is done in the reverse order as in the start section.

Be sure to use absolute paths to your scripts because you are not guaranteed at start up or shutdown that the system will know how to find your scripts. Note in the DB start up and shutdown scripts below, that the sub-scripts and commands are called with full path names specified.

Restart is a simple recursive call to the script with the arguments stop and start.

[1] If you wish, you can add an additional piece of functionality by adding a new section to the case statement.

The status is a little more complicated. In many services, there is a way to call system of programs that will reflect what it is doing. In Progress, the best we can do is show the processes related to Progress DB to determine if the DB is running or not. One can call Webspeed brokers with the –q or –query in the status section to gain additional status information about the broker and admin server.

The DB startup script

This is a simple script that starts up a database related to an application. It calls a common script that defines the parameters to call the proserve command with. This common script (the environment script) allows one point of configuration for all the commands to call into to determine how they should behave.

```
#!/bin/ksh

. /home/db/amduus/script/setenv.ksh

$DLC/bin/proserve -pf $PFFILE
```

The DB stop script

```
#!/bin/ksh

. /home/db/amduus/script/setenv.ksh

$DLC/bin/proshut -by -pf $PFFILE
```

The environment script

```
#!/bin/ksh

export PFFILE=/db/amduus/parm/amduus.pf
export PROPATH=
export DLC=/usr/dlc
export CONFFILE=
```

Final configuration steps:

There are some final touches on a Linux and IRIX system to incorporate your script into the startup and shutdown process. The use of the chkconfig command does this work for you.

```
chkconfig –add progress
```

will add the service to your system.

```
chkconfig -level 345
```

will have your server started on run levels 3, 4, and 5.

You will need to restart your system to determine if it is implemented properly.

About the author: Scott Auge is the founder of Amduus Information Works, Inc. He has been programming in the Progress environment since 1994. His works have included E-Business initiatives and focuses on web applications on UNIX platforms.
sauge@amduus.com

Publishing Information:

Scott Auge publishes this document. I can be reached at sauge@amduus.com.

Currently there are over 800 subscribers and companies that receive this mailing! This mailing is not sent unsolicited, so it is not SPAM.

Amduus Information Works, Inc. assists in the publication of this document:

Amduus Information Works, Inc.
1818 Briarwood
Flint, MI 48507
http://www.amduus.com

Products Available From Amduus:

Blue Diamond – Use your E4GL code in this Webspeed alternative for the UNIX/Linux operating system (some compatibility with Windows)

Service Express – Web based work order/ticket system with work-flow capabilities.

Denkh – Create PDF documents/reports on your Linux/UNIX computer! Headers, Footers, bar charts, as well as tabular data results. Use for a web based reporting system.

Portal – Web content management system. Add web pages easily with linking and searching automatically accomplished.

Mail – Used to distribute this very E-Zine to 800+ readers. Allow your customers to know what you can accomplish for them with newsletters and ezines they can sign up for via the web.

We also lease these applications as well perform custom programming and Progress license sales.

Article Submission Information:

Please submit your article in Microsoft Word format or as text. Please include a little bit about yourself for the About the Author paragraph.

Looking for technical articles, *marketing Progress* articles, articles about books relevant to programming/software industry, white papers, etc.

Subscription Information

Send $20.00 for twelve issues to:

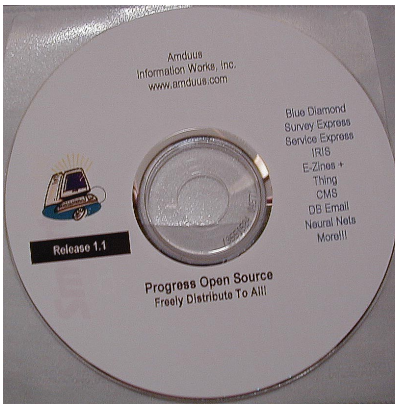
Amduus Information Works, Inc.
1818 Briarwood
Flint, MI 48507

Your Name: _____

Your Email Address: _____

Your Company Name: _____

Order Form for Progress Open Source CD-ROM
COUPON 001A



This is an offer for the CD-ROM at lower than list savings!

Mail this form to:

Amduus Information Works, Inc.
1818 Briarwood
Flint, MI 48507

Please send _____ copies of the Open Source CD-ROM at $15.00 per disk to:

Name _____
Company _____
Address _____
City _____
State _____
Zip _____

Please make your checks/money orders out to: Amduus Information Works, Inc. Cash works too!
This offer only valid in the United States of America.

The CD-ROM includes (all source code included):

- Blue Diamond/IRIS – Webspeed alternatives
- Survey Express – easily create text templates of surveys and then have the program generate the web pages automatically
- Service Express – Web based Help Desk.
- The Progress E-Zines, books on learning to program in Webspeed (PDF/Word/HTML)
- THING – simple tool to manipulate database records with
- CMS – a web content management system
- DB Email – Use pop3 to download emails into a Progress database
- Neural Networks – experiments in spam recognition and text message classification
- GenPDF – create PDF file reports for Webspeed/UNIX CHUI!
- More!