The Progress Electronic Magazine

In this issue:

Did you sign up to receive this E-Zine? Send email to sauge@amduus.com to subscribe! It's free!

Publisher's Statement:

Yet another issue of the E-Zine out! This one continues on the theme of handling mail from a mail server. In particular are the routines for handling the memptr manipulations needed to handle socket based interactions. One will be able to see some code that swaps character data in and out of raw memory. Yes, the answer of how to get char based variable data out that socket!

The second more management oriented article involves valuing data in your organization. As we pack away gigabytes and gigabytes of data the questions of what should be archived and what should be trashed comes to mind. Database people are the worst pack rats when it comes to data! Also, these ideas may help you further define the importance of your IT department or position as it tries to directly related data to money.

The next exciting piece of news (or advertising as some of the more cynical of our readership) is that Amduus Information Works has incorporated! So it's title is now Amduus Information Works, Inc. We are also in the process of becoming a Progress re-seller – you will soon be seeing some applications we can sell as complete packages, from just the application, to the application with Progress, to the application, Progress, and hardware! Also there will be collections open source available and additional information at http://www.amduus.com .

Finally, I have returned to the MS Word format. It is to hard to cut and paste code out of PDF files. Add to that, I do not agree with Adobe nailing a Russian programmer with the Digital Millennium Copyright Act. There is a piece of legislation that needs some re-writing. Xerox was lucky they made the copy machine 40 years ago!

OK – lets get on to the good stuff!

To your success,

Scott Auge
Founder, Amduus Information Works, Inc.

Coding Article: Useful memory functions

Written by Scott Auge sauge@amduus.com

The previous issue was composed of the guts of getting email from a POP3 server. That code relied on some code that worked with raw allocated memory. When using sockets, one must also work with MEMPTR structures to send out data and read in data from the socket.

This code basically handles allocating the memory, setting the memory to an initialized value, setting it to the data stored in a char variable, and getting the data from the raw memory into a char variable. They primarily based off of similar functions in ANSI C library.

Any program using the code shown in the last issue needs the pop3.i include file. This file defines a memory pointer and allocates roughly 22 kilobytes of memory to it. It is through this space that we send and receive data to and from the socket.

```
/* pop3.i
 * Written by Scott Auge
 * sauge@amduus.com
 *
 * Released into public domain.
 *
 */

DEF VAR RCSVersionpop3i AS CHARACTER INIT "$Header:
/home/sdk/mail/source/RCS/pop3.i,v 1.1 2001/01/11 15:35:06 root Exp $"
NO-UNDO.

&GLOBAL-DEFINE NOERROR  0
&GLOBAL-DEFINE NOCONNECT 1

DEF VAR mData             AS MEMPTR    NO-UNDO.

SET-SIZE (mData) = 22000.
```

The following procedure sets a memory space to a certain value. One sends in a variable of type MEMPTR that has memory allocated to it, note that the routine does not allocate memory for the user. Also a character is input that will be copied over to the memory area pointed to by the memory pointer. The character sent over cannot be a string, else the routine will bomb out.

Since memory pointers do not actually hold the data it's self, only information about where to find the data in memory, it does not need to be an OUTPUT PARAMETER because the value that tells the language where to find the memory is done in the SET-SIZE() function.

```
PROCEDURE lmemset:

  DEF INPUT PARAMETER m AS MEMPTR NO-UNDO.
  DEF INPUT PARAMETER c AS CHARACTER NO-UNDO.

  DEF VAR i AS INTEGER INIT 1 NO-UNDO.

  DO while i < GET-SIZE(m):
    PUT-BYTE (mData, i) = ASC (c).
    ASSIGN i = i + 1.
  END.

END.
```

The lmemput procedure allows one to take a character string from a Progress variable and place it into the memory space allocated. A MEMPTR variable that has been through the SET-SIZE() function is sent in as an input parameter. A string is also sent along which will have it's contents copied over to the memory space. When doing so, one usually will want to set the memory space to a NULL character before calling this routine if one is using the string.h functions in C/C++. One needs to be very careful because it is easy to walk into outer space with these kinds of raw memory operations.

```
PROCEDURE lmemput:

  DEF INPUT PARAMETER m AS MEMPTR NO-UNDO.
  DEF INPUT PARAMETER s AS CHARACTER NO-UNDO.

  DEF VAR i AS INTEGER INIT 1 NO-UNDO.

  DO WHILE i < GET-SIZE(m):
    IF i < LENGTH (s) + 1 THEN
      PUT-BYTE (mData, i) = ASC(SUBSTRING (s, i, 1)).
    ELSE
      PUT-BYTE (mData, i) = 0.
    ASSIGN i = i + 1.
  END.

END.
```

The lmemget procedure to pull data out of a MEMPTR into a Progress character string. An input parameter of type MEMPTR is sent to the function. This variable should have been set with the SET-SIZE() function and hopefully, previously manipulated by a C routine or some actions with the 4GL before use. Since a Progress 4GL variable is the reference for a piece of data instead of a pointer, we need to OUTPUT PARAMETER a string. Note that this string will contain ALL of the characters found in the memory space. So that means, if you have allocated twenty bytes to the space, and have used C to put in a string of ten bytes, you might think you will be getting the ten bytes. But this routine does not use the 0x00 character as a delimiter of a string, so the Progress string will also have the entire allocated space.

In use with the POP3 routines, be aware that your 4GL string will be 22,000 bytes long. By trimming the string you can work around getting it to the proper size of the data one wants in it.

```
PROCEDURE lmemget:

  DEF INPUT PARAMETER  m AS memptr NO-UNDO.
  DEF OUTPUT PARAMETER c AS CHARACTER NO-UNDO.

  c = GET-BYTES (mData, 1, GET-SIZE(mData)).

END.
```

This code is available at the http://www.amduus.com website under the Progress Community page.

About the author: Scott Auge is the founder of Amduus Information Works. 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

Management Article: Valuing your data

Written by Scott Auge sauge@amduus.com

I recently had an interesting meeting in the company I am working for. In this meeting, I discovered the company was spending literally $1,000,000 a month on an MVS mainframe to keep an eight gigabytes database online. This had the company paying $125,000 per month per gigabyte and put my jaw on the floor.

This got me asking the question "just how valuable is one's data?" and I thought I would share some of my thinking and learning on this subject. It should be helpful for anyone who is attempting to compute ROI or the value of their information systems. The goal is to translate data items into dollars – these are best understood in a business management context when negotiations about IT come up.

Data could be viewed in these sets: Operational, Analytical., Infrastructure, and Connections.

<u>Operational</u> data represents that data currently being used by the company to perform some service or create some product for the customer that the customer has or will pay for. A short way of valuing this data is to look at the accounts receivable for the company at the moment. Since companies are so dependent on their information systems these days, loss of this data basically means the company will be unable to provide the product or service meant to be presented to the customer. One can break this up into various accounts receivable for each product or service the data is meant to be processed for, or per customer order in process.

<u>Infrastructure</u> data represents data and information that forms the basis of processing for transactions within the system. Examples of infrastructure data are customer lists, engineering BOMS, and electronic documents that are used to master distributed documents (user manuals, etc.) A way to value these is to examine the cost of replacement for these items. Another way to examine the value of data, is the cost of replacement for that data. Some examples might be, how much would it cost to replace the customer list? How much would it cost to find the contacts on the customer list? How much would it cost to replace the engineering BOMS for a product? How much would it cost to replace electronic documents?

<u>Connections</u> data would be the communications infrastructure. Such items as IP address assignment, email management, voicemail management, etc. Questions about replacement value might be: how much would it cost to find someone to do this? How long would it take to replace or re-work the network configuration data?

Analytical data represents that data which has been used by the company to complete some transaction of product or service. It represents data that has been used in the past to extract money from the market place. This data is pretty much "dead" in terms of current work in process. However, it may be valuable for future transactions in terms of pattern analysis or servicing the account. The amount of revenue generated by referencing this information may be something to track.

I would be curious about how other people and companies examine the value of their data. Please send me some email on this at sauge@amduus.com.

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

Contractors/Consulting Companies Available:

If you do work in the Progress world – let me know and I will be able to include you! Price is $10.00 per listing per issue.

Analysts Express Inc.

The Reliable Source for I.T. Consulting and Recruiting
Contact: James Arnold @ 888/889-9091

02-1

Amduus Information Works, Inc.
4506 Carlyle Court Suite 712
Santa Clara, CA 95054
408-980-8447
sauge@amduus.com

Creation of modules and products for re-sale. Internet/Intranet programming for E-Business in the marketing/manufacturing/service/law enforcement industries.

Community Announcements:

A place to announce Progress User Groups, Open Source Exchanges, etc. No charge!

Wonder where to find those Progress marketing articles?

http://www.progress.com/analyst/
http://www.progress.com/profiles/index.htm
http://www.progress.com/success/index.htm

Publishing Information:

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

Currently there are over 300 subscribers and companies that receive this mailing!

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

Amduus Information Works, Inc.
4506 Carlyle Court Suite 712
Santa Clara, CA 95054
sauge@amduus.com

Amduus Information Works, Inc. programmers specialize in the following operating systems: AIX, UNIXWare, Solaris, HP-UX, and Linux.

Rates are:

Closed Source/On Site	$900.00 per person per day + expenses
Closed Source/Off Site	$700.00 per person per day
Open Source/On Site	$600.00 per person per day + expenses
Open Source/Off Site	$500.00 per person per day

Amduus Information Works, Inc. is a sanctioned Progress Reseller. It works mostly with web applications in the service/manufacturing/law enforcement industries.

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.