The Progress Electronic Magazine

In this issue:

Did you sign up to receive this E-Zine? Send email to sauge@amduus.com to subscribe or fill out the forms at http://www.amduus.com/online/dev/ezine/EZineHome.html ! It's free! (Though donations are certainly welcome – whatever you feel is fair!)

Publisher's Statement:

Once again we have a reader supplied article! Here are additional tricks one can use to make vi that much more powerful to quickly manipulate source code. The cutting and pasting section let's one set up multiple "buffers" that can be shared between users of vi (aren't multi-user systems neat?) for code sharing, as well as indentation techniques.

Other exciting news is that this ezine is going out to over 800 subscribers! Holy Moly! I remember when I thought it was a milestone to reach 200 people! As you read this issue, know that over 800 programmers are reading along with you, in over 15 countries, in 500+ Progress using organizations. So if you ever feel like you are the lone Progress programmer – know that you are not! And feel free to pass this around! We want people to know how to use Progress – it is in all of our best interests to see valuable and useful programs in Progress reach the marketplace.

Please keep Amduus Information Works, Inc. in mind when you have a project coming up! There is more than just one guy associated with this corporation! Amduus is already influencing the programming practices of hundreds of programmers in hundreds of companies – can't be all that bad!

I always try to post useful code in this E-Zine and I can't think of anything more useful than a menu system. There are plenty of menu organizations out there, and this code gives the ol' college try at a tree based menu structure. In part one, we look at the tables used, a few pictures of what we are attempting to do, and the code to render the menu. In part two, we will look at the maintenance program used to conveniently edit the menu structure.

The code works in both Webspeed and Blue Diamond. It should be cross platform compatible working on not only UNIX, but also Windows based operating systems.

As with a lot of code developed at Amduus, this code is being open sourced under the Berkley License which means -- long enough you give credit to Amduus Information Works, Inc. and Scott Auge – you are free to use it any way you wish. (But you must also include the don't sue Amduus legalese.)

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

Coding Article: Using vi features to increase programmer productivity part II

Written by Sam Paakki

This is a follow up article to Scott Auge's last coding article that introduced the *ex abbreviate* command, which can help programmers save on key strokes when coding.

Let's get things straight from the get go. Vi rocks! As you see, I'm a huge fan of this editor, despite initially hating being force to learn it by my first employer. Side note: I was also forced into to learning how to type at the same time. Learning both at the same time wasn't an easy charter, but it was well worth the effort. These two skills allow me to get an idea into a file super-fast!

It can be hard to learn how truly powerful vi is by simply reading the man pages, as you really have to read the man pages for *vi, ex, ed, regexp*, etc; and we all know how hard reading man pages can be. I was fortunate enough to be around other vi power-users who constantly amazed me with their abilities to enter and modify code. There are many good books on vi out there. Start with a small (i.e. thin) one and persist.

In this article I will describe some of the editing features and functions that I simply can't do without and how I perform them with vi.

Function 1: Copy and Paste

Your telnet emulator may support copy and paste functionality, but this is a the LCD (lowest common denominator) method I use that works everywhere I go.

Firstly you'll need to add two *ex* commands to your *editing environment defaults*. All this means is you vi your *.exrc* file in your home directory to add a few lines. This only needs to be done once.

First add a key mapping using the *ex map* command for our *Yank* (copy) function by adding the following line to the end of your *$HOME/.exrc* file.

```
map ^Y :'a,.w!/tmp/
```

To enter the ^*Y* character above, you'll need to press `<Ctrl>-<v>`, which will display a '^' and then you press `<Ctrl>-<y>` to replace the '^' with the single '^Y' character. You can see that

this is in fact a single character by moving the cursor over it and you'll notice (if you've done it correctly) that the cursor will not go on top of the '^' any longer.

In vi you can *escape* any keyboard character input by pressing <Ctrl>-<v>, when in INSERT MODE, followed by the key you want to escape. This is very handy for editing protermcap files, should the need arise.

Next we need to add a key mapping for our *Paste* function by adding the following line to the end of your *$HOME/.exrc* file.

```
map ^P :r /tmp/
```

N.B., *editing environment defaults* are read on start-up of vi, so whenever you modify your *$HOME/.exrc* file, you'll need to save it and then restart any vi sessions you are already running in order to access the changes.

Now, when you're in vi, in command mode, you'll be able to use the new yank and paste functions by pressing <Ctrl>-<y> and <Ctrl>-<p> respectively.

To yank, position the cursor to the first line that you want to copy down from and then press the <m> key, and then press the <a>. This marks the current line as line *a* and it can be address with *'a*. Now move the cursor **down** to the last line that you want to copy down to. Press <Ctrl>-<y> and you will see ':'a,.w!/tmp/' on the bottom line with the cursor flashing after the trailing forward slash character. Now enter a filename of your choice followed by <Enter>. I use the file 's0' out of habit and if I need more than one file at a time I simply increment the trailing integer (e.g., s1, s2, etc). This creates a file in /tmp with the name that you entered containing the selected lines of text.

Now you can move to another line or even vi another file, before you paste the yanked text back in. Once you've repositioned the cursor to the line before the point where you want to paste the text back into, press <Ctrl>-<p> and you will see ':r /tmp/' on the bottom line with the cursor flashing after the trailing forward slash character. Now, type the filename of a file that you've already yanked to and then press <Enter>. The yanked text should now have been placed back into your current file.

All that these 2 mappings for <Ctrl>-<y> and <Ctrl>-<p> do is save typing a few keys for the *ex write* and *read* commands. You can just type the ':'a,.w!/tmp/' or ':r /tmp/' manually if you really want to.

Function 2: Move block indentation to right/left

Some times I find it necessary to indent or un-indent a block of code when I add or remove blocking statements. In vi you can move the current line one *shift-width* to the right by pressing the greater-than key twice (ie, '>>') and conversely to the left by pressing the less-than key twice (ie, '<<').

It is also possible to use the *ex* commands > and < which shift groups of lines to the right and left. Once again, I set up a couple of mappings to automate this. Add the following two lines to the end of your *editing environment defaults* file (ie, *$HOME/.exrc*).

```
map ^[OP :'a,.<^M
map ^[OQ :'a,.>^M
```

In this case, the ^[OP above is the sequence of keys my telnet emulator sends when I press my <F1> key and the ^[OQ above is a <F2>. N.B., my emulator is emulating a vt220. Also the ^M above is entered by pressing <Ctrl>-<v> followed by <Enter>.

Therefore when you enter these two lines you should press <Ctrl>-<v> followed by <F1> or <F2> as appropriate so that you escape the key sequence.

Now in vi you can put your cursor on the first line of the group of lines that you want to shift and then press <m> followed by <a>. Do you see a pattern emerging? I always mark the top line as *a* and then I move **down** to the last line of the group and perform the function by addressing this group of lines with ':'a,.'. Think of this as saying *from line a to the current line do ...* as the *comma* means *to* and the *period* means the *current line*.

This makes it possible to mark your first line, then move to your last line of the group and press either <F1> or <F2> repeatedly to shift the block right or left. NOTE: if you change your terminal type, you will need to change the mapping to look for the new sequence being sent for these function keys. However, I find that once I've done the initial set up of these mappings, I rarely change my terminal type or emulator.

Function 3: Comment and Un-comment

These two functions are quite complex, but very useful. I won't try to explain them fully, but you should give them a try.

Again, you need to add two mappings to your .exrc file. We will use <F3> to comment out code and <F4> to remove comments (sort of). This is how they appear in my .exrc file: -

```
map ^[OR mbo^[i */^[:'a^MO^[i/*^M *^M*^[j:.,'bs/^/ * /^M:'a-2^M
map ^[OS :'a,.s/^.../^M
```

Here are the keystrokes that you need to enter for the first line: -

```
map<Space><Ctrl-v><F3><Space>mbo<Ctrl-v><Esc>i<Space>*/
<Ctrl-v><Esc>:'a<Ctrl-v><Enter><Shift-O><Ctrl-v><Esc>
i/*<Ctrl-v><Enter><Space>*<Ctrl-v><Enter>*<Ctrl-v><Esc>
j:.,'bs/^/<Space>*<Space>/<Ctrl-v><Enter>:'a-2<Ctrl-v><Enter>
```

And here are the keystrokes for the second line: -

```
map<Space><Ctrl-v><F4><Space>:'a,.s/^.../<Ctrl-v><Enter>
```

This commands removes the first 3 characters (whatever they are) from the beginning of the addressed lines. You have to manually delete the comment start and end marks yourself.

Give them a try by using the common theme of mark *a* on the first line, and the move to the last line, followed by your newly mapped function key. If you have any problems, you can email me directly at smp@varacom.com.au with your questions.

Once you understand regular expressions, you can really do some amazing search and replaces with vi (or sed). I'll save my tricks and tips in this area for another day. Happy coding.

About the author: Sam Paakki is the founder of Varacom Pty Ltd, which is based in Brisbane Australia. He has been programming in the Progress environment since 1990. His works have included the extension of Progress-based core systems and their integration to 3rd party systems for SME businesses, along with performance tuning and disaster recovery planning for these systems. He is also the president of QPUG (Qld Progress User Group). smp@varacom.com.au

Coding Article: Web based menuing system Part I

Written by Scott Auge sauge@amduus.com

Often when one is creating web applications, a web based menu system comes in handy to help the user navigate the site. This article discusses such a menu with the code available. One can certainly dress up the menu with graphics, etc. The menu system is a tree of folders and pages that one can set up with a simple maintenance program.

What we are trying to achieve:

Here we have a picture of the menu showing a level of the branches available on the menu. The menu system is totally database driven with the data entered on a menu maintenance screen (discussed later.)

Upon clicking the Tickets link, we wish the sub-links of the menu to appear. An example of this is shown to the right.

There are indentions to help the user determine how far into the tree they have clicked through.

The menu has the option of placing “targets” in the hyperlink to allow the opening of a new window, or the page the link refers to into some pre-defined frame.

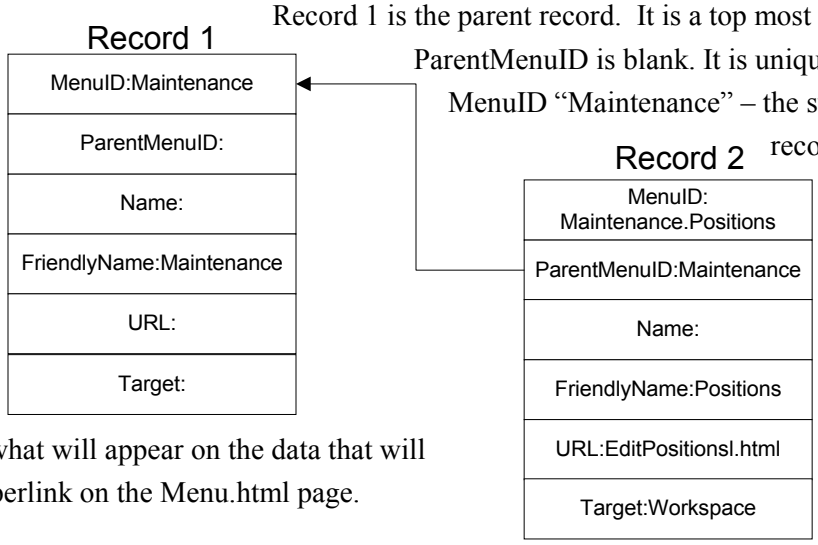
The table structure

The menu information is stored within a table called Menu. The table has the following fields:

Field	Purpose
MenuID	Unique identifier for the menu item
ParentMenuID	Used to identify the menu item this menu item is found under. The top most have a blank value in this field.

Name	Not Used
URL	The URL of the page that when the item is clicked should appear in the Target area. Leave this blank to have the item be a "folder" for a collection of other menu entries.
FriendlyName	What appears on the hyperlink viewed by the user.
Target	Target window/frame the page should appear in.

Here is a visual relation between two records, one which is the parent and the other is a child. The data is represented in each box, with the field name to the left of the colon , and the sample data to the right of the colon.



Record 1 is the parent record. It is a top most record because it's ParentMenuID is blank. It is uniquely identified by the MenuID "Maintenance" – the sub-menu items of this record will have "Maintenance" as their ParentMenuID value. The FriendlyName of Record 1 is "Maintenance" This is what will appear on the data that will appear as the hyperlink on the Menu.html page.

Record 2 has a MenuID of "Maintenance.Positions." I tend to use a dot notation of the path the menu item should be found under as it's unique identifer. When I look at data dumps, it helps organize the information a little better for comprehension. Notice that Record 2's ParentMenuID is the MenuID of Record 1. This will place it below the parent on the screen as a sub-item on the menu. The FriendlyName will appear as Positions on the Menu.html screen for the user to click on. When the user clicks on this item, the page specified in URL will appear, in this case it is EditPositions.html. The EditPositions.html page will appear in the Target called "Workspace." This is either the name of a <FRAME> in a <FRAMESET> or a window that exists (if a window does not exist, it will become a pop-up window.)

The Menu.html program

The following is the source code to the menu screen we see above. It is written in the E4GL style.

```
<!--WSS

DEF VAR RCSVersion AS CHARACTER INIT "$Header:
/home/appl/opensrc/menu/src/RCS/Menu.html,v 1.2 2002/04/22 21:59:10 sauge Exp
sauge $" NO-UNDO.

DEF VAR cMenuTree AS CHARACTER NO-UNDO.
DEF VAR cMenuID AS CHARACTER NO-UNDO.

ASSIGN cMenuID = GET-VALUE("MenuID").

RUN GenTree.p (input cMenuID, "Menu.html", output cMenuTree ).

-->
<html>
<head>
<title>Menu</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#CCCCFF">
<p><font face="Arial Narrow" size="+1" color="#000066"><b>Company
Name</b></font></p>
<a href="Menu.html">Top</a><br>
`cMenuTree`

</body>
</html>
```

It is a straight forward program that identifies the last menuid clicked on the tree of links provided by the GenTree.p program. It then renders the menu hierarchy on a HTML page.

The GenTree.p Program

The GenTree.p program uses FindDepth.p to identify how far into the menu the item the user clicked on is. It then begins rendering the HTML needed for the hyperlinks using the GenTreeHTML.p program.

It returns a string with the value of the HTML code needed to render the menu on the page. In the Menu.html

program, this is represented by the variable `cMenuTree`.

```
DEF VAR RCSVersion AS CHARACTER INIT "$Header:
/home/appl/opensrc/menu/src/RCS/GenTree.p,v 1.1 2002/04/22 21:59:17 sauge Exp
sauge $" NO-UNDO.

DEF INPUT PARAMETER cMenuID AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER cHTMLPageName AS CHARACTER NO-UNDO.
DEF OUTPUT PARAMETER cMenuHTML AS CHARACTER NO-UNDO.

DEF VAR iMaxDepth AS INTEGER NO-UNDO.
DEF VAR cHTML AS CHARACTER NO-UNDO.

FIND FIRST Menu NO-LOCK
WHERE Menu.ParentMenuID = cMenuID
NO-ERROR.

IF NOT AVAILABLE Menu THEN RETURN.

RUN FindDepth.p
(Menu.MenuID,
 0,
 OUTPUT iMaxDepth
).

RUN GenTreeHTML.p
(INPUT cHTML,
 INPUT cMenuID,
 INPUT cMenuID,
 INPUT iMaxDepth - 1,
 INPUT cHTMLPageName,
 OUTPUT cMenuHTML
).
```

Notice that there is an cHTMLPageName as an argument. This is to effectively use the routines in pages named other than Menu.html. One may want to create a page that starts with a specific menu and this would allow that to happen. For other programs, these routines will have to act as templates (the maintenance program has some variation on these routines.)

The FindDepth.p program

This is a very simple recursive program that when given a menu item, walks up the tree to the top finding out how many deep the item is located in the menu structure. It returns this value as output.

```
DEF VAR RCSVersion AS CHARACTER INIT "$Header:
/home/appl/opensrc/menu/src/RCS/FindDepth.p,v 1.1 2002/04/22 21:59:17 sauge Exp
sauge $" NO-UNDO.

DEF INPUT PARAMETER cParentMenuID  AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER cCount         AS INTEGER NO-UNDO.
DEF OUTPUT PARAMETER cDepthCount   AS INTEGER NO-UNDO.

FIND FIRST Menu NO-LOCK
WHERE Menu.MenuID = cParentMenuID
```

```
NO-ERROR.

IF AVAILABLE Menu THEN ASSIGN cCount = cCount + 1.
IF Menu.ParentMenuID <> "" THEN RUN FindDepth.p (Menu.ParentMenuID,
                                                 cCount,
                                                 OUTPUT cDepthCount).
ELSE cDepthCount = cCount.
```

The GenTreeHTML.p program

The GenTreeHTML.p program does the most grunt work for actually rendering the important part of the page – the menu hierarchy.

It does this by receiving a MenuID as input, as well as the depth the menu ID is at. Remember the depth is computed in the FindDepth.p program and then passed along to this program. This program calls it's self recursively, and decrements the depth on each call as it moves up the tree of menu items. This depth acts as a multiplier to SUBSTRING () a line of –'s appropriate to the depth of the current menuid examined. It then returns the string, as well as the <A> hyperlink for the given MenuID.

```
DEF VAR RCSVersion AS CHARACTER INIT "$Header:
/home/appl/opensrc/menu/src/RCS/GenTreeHTML.p,v 1.3 2002/04/22 21:59:17 sauge
Exp sauge $" NO-UNDO.

DEF INPUT PARAMETER cHTML AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER cMenuID AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER cPopoutMenuID AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER iMaxDepth AS INTEGER NO-UNDO.
DEF INPUT PARAMETER cHTMLPageName AS CHARACTER NO-UNDO.
DEF OUTPUT PARAMETER cOutHTML AS CHARACTER NO-UNDO.

DEF BUFFER BufMenu FOR Menu.

DEF VAR cString AS CHARACTER NO-UNDO.

ASSIGN cString = "---------------------------------".

FIND Menu NO-LOCK
WHERE Menu.MenuID = cMenuID
NO-ERROR.

IF AVAILABLE Menu THEN RUN GenTreeHTML.p (INPUT cOutHTML,
                                          INPUT Menu.ParentMenuID,
                                          INPUT Menu.MenuID,
                                          INPUT iMaxDepth - 1,
                                          INPUT cHTMLPageName,
                                          OUTPUT cOutHTML).
```

```
FOR EACH Menu NO-LOCK
WHERE Menu.ParentMenuID = cMenuID:

  ASSIGN cOutHtml = cOutHTML
                  + SUBSTRING (cString, 1, iMaxDepth * 3).

  IF Menu.URL = "" THEN
  ASSIGN cOutHtml = cOutHTML
                  + "<a href=~"" + cHTMLPageName + "?MenuID=" + Menu.MenuID .
  ELSE
  ASSIGN cOutHtml = cOutHTML
                  + "<a href=~"" + Menu.URL .

  ASSIGN cOutHtml = cOutHTML
                  + "~"" .

  ASSIGN cOutHTML = cOutHTML
                  + IF Menu.Target = "" THEN ">"
                    ELSE " target=~"" + Menu.Target + "~">".

  ASSIGN cOutHTML = cOutHTML
                  + Menu.FriendlyName + "</a>"
                  + "<br>~n".

  IF Menu.MenuID = cPopoutMenuID THEN LEAVE.

END.
```

These hyperlinks are then concatenated together and returned to previous calls of the routine until finally it comes back to the GenTree.p routine with all the strings concatenated together.

One may ask why not use internal procedures or functions for these recursive routines. The answer is that buffer scoping does not work to an advantage in recursive calls to these kinds of routines. The last buffer found is used as the recursion winds upwards, and this messes up some of the values that are concatenated together based on what is in the database record found for that recursion.

That pretty much handles it for the Menu presentation routines. In the next E-Zine, I will present the maintenance program to conveniently edit this structure.

License

```
/*
 * Written by Scott Auge scott_auge@yahoo.com sauge@amduus.com
 * Copyright (c) 2002 Amduus Information Works, Inc.  www.amduus.com
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by Amduus Information Works
 *      Inc. and its contributors.
 * 4. Neither the name of Amduus Information Works, Inc. nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY AMDUUS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AMDUUS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */
```

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

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.

I am looking for work, if you have any knowledge of potential work, I would appreciate hearing from you!

http://www.amduus.com/Resumes/ScottAuge.html

Thanks!

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.
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
- More!