Clarion Magazine

Clarion News

- » Clarion C6.3 Build 9051 Adds New Install Option
- » Clarion Third Party Best Worst Dressed List - March 30 2006 Update
- » d-Icons Licence More Developer-Friendly
- » Clarion From The Start Special Ends April 1
- » New Icon Set Available For $10
- » Dictionary Assistant 2.12
- » cpTracker Pro Source Code Bootstrap Sale
- » Version Control Beta Testers Needed

[More news]

Podcast

[Track lists, more podcasts]

Latest Free Content

- » Free FTP Client With Source - Update
- » Free FTP Client With Source

[More free articles]

Clarion Sites

Clarion Blogs

Latest Subscriber Content

Direct-To-USB Printing

Steve Parker concludes his direct-to-printer saga with a look printing directly to USB printers, using a freeware example by Trevor G. Leybourne.
Posted Friday, March 31, 2006

Hand Coding Export Files

Object-Oriented Programming (OOP) can be a wonderful thing. Unfortunately, writing a class can sometimes be easier than using it, especially if you need to export that class from a DLL. Harry Hickey explains the inner workings of the Clarion export file.
Posted Friday, March 31, 2006

Using DOS Files To Send Printer Codes

If your applications need to talk directly to printers on parallel ports, or to serial devices, you can use the Windows API. Or, as Olivier Cretey explains, you an simply use a Clarion DOS file.
Posted Friday, March 31, 2006

Using MS Visual Source Safe With Clarion

Most developers, at one time or another, lose programming changes due to overwritten files, or end up with numerous copies of an application on their machine in an attempt to save a trail of their work. Version control software solves the problem of lost code, and provides a nice tidy trail useful for tracking changes made to an app. In this article Marty Honea shows how to use Microsoft's Visual Source Safe with Clarion.
Posted Thursday, March 30, 2006

The Easiest Way To Write To A Printer Port

In the last less-than-action-packed installment, Steve Parker said that he was not going to create a template for the write-directly-to-port wrapper procedure discussed. There is a very good reason for not creating that template. It’s been done already. At least twice.
Posted Thursday, March 30, 2006

Clarion Challenge Results - Remove Links

The results for the March 2 "Remove Links" Clarion Challenge are in! And we have detailed analysis courtesy of CapeSoft's Profiler, plus a .NET surprise.
Posted Thursday, March 23, 2006

Free FTP Client With Source - Update (free article)

An update to the help file accompanying Veronica Chapman's FTP client is now available. This update covers some issues that may occur when using the client under Windows XP.
Posted Tuesday, March 21, 2006

Print Directly to Printer Made Easier

In his last adventure in API land, Steve Parker explored sending control codes directly to a printer port. In this installment he wraps that code up into a function call.
Posted Monday, March 20, 2006

[Last 10 articles] [Last 25 articles] [All content]

Printed Books & E-Books

From The Publisher

About Clarion Magazine

Clarion Magazine is your premier source for news about, and in-depth articles on Clarion software development. We publish articles by many of the leading developers in the Clarion community, covering subjects from everyday programming tasks to specialized techniques you won't learn anywhere else. Whether you're just getting started with Clarion, or are a seasoned veteran, Clarion Magazine has the information *you* need.

Subscriptions

While we do publish some free content, most Clarion Magazine articles are for subscribers only. Your subscription not only gets you premium content in the form of new articles, it also includes all the back issues. Our search engine lets you do simple or complex searches on both articles and news items. Subscribers can also post questions and comments directly to articles.

Satisfaction Guaranteed

For just pennies per day you can have this wealth of Clarion development information at your fingertips. Your Clarion magazine subscription will more than pay for itself - you have my personal guarantee.

Dave Harms

Clarion Magazine

Clarion News

Search the news archive

Clarion C6.3 Build 9051 Adds New Install Option

Clarion C6.3 build 9051 is now available for download from SoftVelocity, and email notices have gone. This release went out to third party vendors on March 23rd so they could prepare their products for release and conduct compatibility testing. As usual, a patch is available, but this time a second (and five times larger) download is also available, called a Service Pack install. This install will work with any prior version of Clarion 6. After completion, your C6 installation is the same no matter which route you take.
Posted Friday, March 31, 2006

Clarion Third Party Best Worst Dressed List - March 30 2006 Update

StrategyOnline has received a Clarion Best Dressed rating thanks to its JavaScript pull-down menus, product summary, download page, and RSS feed.
Posted Friday, March 31, 2006

d-Icons Licence More Developer-Friendly

Developers of tools or template files for other developers can now distribute d-icons with their third party tools for royalty free use by end user clients (i.e. other developers). Anyone who has already purchased any d-Icons sets may also now use them under these new conditions.
Posted Friday, March 31, 2006

Clarion From The Start Special Ends April 1

Just a reminder to all those who think about starting their own training on Clarion programming "From the Start" that you can save $200 if you buy before April 1.
Posted Tuesday, March 28, 2006

New Icon Set Available For $10

Dave Beggs has another icon set, this time with a "Vista" look and feel. While Windows Xp style icons look like bright light is shining on them, the newer Vista style icons have a slightly different color palette, and they look more like reflected light is shining on them. They also have a defined border, and the symbols are shaded rather than white. There are more than 620 icons in this pack. Each icon file includes 6 formats - 48*48, 32*32, and 16*16 in both True Color and 256 Color. There are more than 120 icon designs in 5 colors. PNG and GIF files are also provided, for more than 1500 files. Also, blank Icons are provided so you can make your own icons to match if you need to. These Icons were made with Axialis Icon Editor - a relatively straightforward procedure. You could do it yourself easily enough. But could you build a whole set for $10? The normal selling price is $US19.99 but Clarion developers are welcome to a 50% discount by using the coupon CPN2910722049. PayPal also available. The coupon can also be used for either Set 1 (Xp style) icons or the new Set 2 (Vista style icons) or both.
Posted Tuesday, March 28, 2006

Dictionary Assistant 2.12

Dictionary Assistant 2.12 is available free of charge to all registered customers of Dictionary Assistant 2. Similar to the recent update for Application Assistant, the significant enhancement in this release is the new Dictionary Assistant Dashboard. As always you may use the "Check for Updates" option within DA or download the update directly from the web site.
Posted Tuesday, March 28, 2006

cpTracker Pro Source Code Bootstrap Sale

cpTracker's Professional Vanilla Source Code is one sale for $47 through April 15, 2006 and this deal will never again be offered. All 3rd party products have been removed so you can add your own favorites. Also included: a free single-user copy of the full award winning cpTracker Professional or cpTracker Lite if you prefer task management only. You can use the application for yourself, for your business or to use as a starting point for creating a new product. There are no restrictions and no royalties. You can brand it with your own product name, logo and for your company.

Posted Tuesday, March 28, 2006

Version Control Beta Testers Needed

Data Equity is preparing to release a new commercial product (developed in Clarion) that integrates with various source control products. The first two integrations are MS Visual Source Safe followed by SubVersion.
Posted Tuesday, March 28, 2006

Epsilon Clarion Website Special Expires April 15

Until April 15, 2006 Epsilon Concepts is offering a Clarion Website Special at $695 (value over $2,000). This package consists of a professional web presence including two domain names, hosting, custom web design, consulting, copywriting, and Internet marketing. Hosting features include 250 megsbytes of space, 1 gigabyte of bandwidth, 500 email accounts, unlimited databases, FTP accounts, subdomains, and more. Design features include up to 12 hours design services, up to 12 pages, site map. Additional consultation on development, search engine optimization, search engine submission, and many other features.
Posted Tuesday, March 28, 2006

Data Equity Assistants and Moving From TPS to SQL

Data Equity LLC has a new spec sheet on how the Data Equity Assistants can improve your transition experience if you are migrating from the TPS file system to a client-server platform.
Posted Tuesday, March 28, 2006

SetupBuilder 5.4 Build 1445

SetupBuilder 5.4 Build 1445 is now available. As always you may use the "Check for Updates" option within SetupBuilder to auto-update your current SB5 version. You can also download the full SetupBuilder 5.4 install image.
Posted Tuesday, March 28, 2006

xToolTipPro 1.2

xToolTipPro 1.2 is now available. This class and set of templates allow you easily and quickly change standard tips into native Microsoft Tooltip Controls with Many additional

features for any Clarion Window. This is bugfix release for the Init and Kill methods.
Posted Tuesday, March 28, 2006

BG SoftFactory Virtual Training Center

BG Softfactory offers Clarion Classes through a web-based virtual training center. Clarion from the Start, offered on line by multimedia streaming, focuses exclusively on the Clarion language. 48 lessons, divided in 10 blocks, lead the programmer, beginner or experienced, toward writing a sophisticated Clarion program, using the Windows API, SQL and XML. Many other classes are planned.
Posted Tuesday, March 28, 2006

FullRecord Poll

Huenuleufu Development has a poll up for FullRecord users - vote on your preferred storage method for large data.
Posted Tuesday, March 28, 2006

EasyOpenOffice 1.02

EasyOpenOffice 1.02 adds the following new methods: SetBorders (EasyCalc) - draws a range borders in a Calc spreadsheet; SetBorders (EasyWriter) - draws the border lines of the text tables in the OpenOffice Writer document; FindFirst (EasyWriter) - finds a first entry of the text pattern in the OpenOffice.org Writer document; FindNext (EasyWriter) - finds the subsequent entries of the text pattern in the OpenOffice.org Writer document; ReplaceAll (EasyWriter) - replaces all entries of the text pattern in the OpenOffice.org Writer document. There are related new templates. Free update for all registered customers. EasyOpenOffice is a set of classes and templates allowing you to exchange data between Clarion applications and OpenOffice Calc and Writer. Price: $149.
Posted Monday, March 20, 2006

Microsoft releases Atlas AJAX framework

From the MS web site: "This new Web development technology from Microsoft integrates client script libraries with the ASP.NET 2.0 server-based development framework. In addition, 'Atlas' offers you the same type of development platform for client-based Web pages that ASP.NET offers for server-based pages. And because 'Atlas' is an extension of ASP.NET, it is fully integrated with server-based services. "Atlas" makes it possible to easily take advantage of AJAX techniques on the Web and enables you to create

ASP.NET pages with a rich, responsive UI and server communication. However, "Atlas" isn't just for ASP.NET. You can take advantage of the rich client framework to easily build client-centric Web applications that integrate with any backend data provider."
Posted Monday, March 20, 2006

xFunction Library 2.5 Freeware

The freeware xFunction Library version 2.5 is now available. This release adds a new function for getting the size of your program's client rectangle area (not including the
Posted Friday, March 17, 2006

DynaLib 4.0.4

DynaLib 4.0.4 is available for download. Some bugs were fixed, including a template bug in the MS Tooltip Library. A fixed version is available on Customer Download Area.
Posted Friday, March 17, 2006

Whitemarsh News

February was a short but action filled month. First, Whitemarsh attended an ANSI database standards meeting in St. Petersburg, FL in advance of an ISO international meeting in Kobe, Japan. Whitemarsh continued its outreach to DAMA by presenting before various chapters around the country. Whitemarsh also presented at the DebTech Metadata Conference in Orlando. Also this month: Whitemarsh created and presented a Data Management Capability Maturity presentation for the March DAMA New Jersey meeting; Whitemarsh revised the Data Interoperability course, workshops, and the one-day Data Interoperability Strategy seminar; Whitemarsh has begun initial steps to republish a number of its books in paperback in an on-demand format; Whitemarsh has updated and re-posted the Database Objects book to the website. It is now available for member download; Two Data Interoperability brochures have been posted, one for the Strategy Seminar and the other for the Workshop. Whitemarsh is delivering the Data Interoperability Strategy seminar to several audiences around the country in the next few months. Please check the website at http://www.wiscorp.com/upcoming_events.html. Last, and certainly not least is "The DAMA International Symposium & Wilshire Meta-Data Conference." This year it's in Denver, CO. Dates are: April 23-27, 2006. Whitemarsh has attended every year since 1998. This conference is truly, "Where the Data World Meets." There are always great lessons. Instructors are from all industry groups. This conference is truly first rate. Please visit the site at http://www.wilshireconferences.com/MD2006/index.htm.
Posted Friday, March 17, 2006

Clarion Third Party Profile Exchange Updated March 15 2006

The Clarion Third Party Profile Exchange consists primarily of profiles of third party add-on products and vendors. This includes freeware templates and tools as well. Online and Downloadable Profiles available. Online product profiles include Product Internet URL, Order URL, Currency code, Dated Price Quote, Grouped by Category, Clarion 6.x Compatible, Extended Description and Download Page Reference. Currently, there are 506 product profiles and 570 vendor profiles. You must have Product Scope 32 PRO Version 5.0 to view profiles with data files (downloadable profiles). 343 Clarion 6.x compatible products as of this release.
Posted Friday, March 17, 2006

Evolution Excel Import 1.01

Import Excel files into you Clarion Application. For Clarion 5, Clarion 5.5, Clarion 6. ABC Templates. $49
Posted Friday, March 17, 2006

CapeSoft On The Road

With the recent release of NetTalk 4 CapeSoft has had several requests for on-site training, not just for NetTalk 4 but also for other products, like Replicate, HotDates and Profiler. To this end CapeSoft staff will be doing a number of trips over the next few months to visit users in various parts of the world. The visit would take the form of a 1 or 2 day event, with the emphasis on training in the use of CapeSoft accessories. There will also be demonstrations of some of the Capesoft accessories, and special prices for the accessories will be on offer. A (hopefully small) fee for the day would be charged, to cover basic travel and accomodation costs. However you will receive a voucher to the full value of the fee, which you can use to offset any CapeSoft purchase at the event. The tour starts with a visit to Melbourne to attend the Australian Devcon 2006. So if you're in Australia make sure you sign up for that. Registrations are still open at http://www.fiscalservices.com.au/mb/. If you're interested in having a visit from Bruce, email him at bruce@capesoft.com.
Posted Friday, March 17, 2006

cwODBC Source At IceTips.com

Dan Pressnell has uploaded cwODBC source files for C5.5 and C6 to IceTips. The zips

contain the source, in app format, for the cwodbc object. You need to have the dp_class template registered before you can compile the apps. Instructions for using are in the app file. Also, there is a small demo app included.
Posted Friday, March 17, 2006

vuDeploy 3.0

vuDeploy version 3.0 is available for download. This release is a complete redesign. Changes include: New graphical user interface; Add, Edit, and Delete projects and display the projects in a list; Select files for inclusion, from anywhere and display them in a list; Automatically run multiple files (EXEs can be run either consecutively or concurrently); Change the order of the files (for consecutive execution); Automatically detect and add (via user confirmation) any additional files found in the extracted folder and roll them back up into the EXE; Automatically detect and delete from the EXE (via user confirmation) any files that were deleted while the files are extracted; Minimize all other programs during execution and automatically restore the file status on completion. vuDeploy 3.0 is offered as a free upgrade to all current users. The price will be going up at the end of the month.
Posted Friday, March 17, 2006

DevShout Looks at NetTalk

In this enthralling episode DevShout takes a quick look at Capesoft's NetTalk.
Posted Friday, March 17, 2006

Data Equity Application Assistant 2.10

Application Assistant 2.10 is now available. This release is free of charge to all registered customers of Application Assistant 2. The significant enhancement in this release is the new Application Assistant Dashboard. As always you may use the "Check for Updates" option within AA or download the update directly from the web site. Dashboard Views include: General Statistics Totals - Procedures, Templates, Tables, Windows, Reports, Procedures Categories; Control Types on Windows; Control Types on Reports; Template Usage; Procedure Maintenance Timeline.
Posted Friday, March 17, 2006

ImageEx 3.5.2

ImageEx 3.5.2 has been released. This is a bugfix release that fixes an issue with the

LoadFromResource method of the ImageExBitmapClass which would fail when trying to load icons. Thanks to Jack Patridge for pointing this out.
Posted Friday, March 17, 2006

NetTalk 4.03

NetTalk 4.03 has been built and uploaded. Example is running live at http://www.capesoft.com:7506/ - log in using "demo / demo". NetTalk users who purchased after 1 Feb 2005 will get a free upgrade, for the rest the upgrade price is $149.
Posted Friday, March 17, 2006

FullRecord 1.25

New in FullRecord 1.25: For Clarion 6.x only, support for storing the audited record in a BLOB field, which make it possible to use FIREBIRD as audit file database; Also for Clarion 6.x only, new example app using ODBC and Firebird as record storing audit file.
Posted Friday, March 17, 2006

CapeSoft's Clarion Version Support

CapeSoft's general policy on supported Clarion versions for our Accessories has always been more or less the last 3 major versions of Clarion. In other words, before Clarion 6, CapeSoft supported Clarion 4, Clarion 5, and Clarion 5.5. Since Clarion 6 releases, there have been many patch releases, and point releases, and many programmers have found the version that suits their programs best, and not necessarily upgraded to the latest version of C6. With these new versions, the runtime library has changed fairly frequently as well, requiring DLL based products to be recompiled for them. To make this process easier, CapeSoft now ships one install file containing builds for Clarion 5, Clarion 5.5, and the latest version of Clarion 6 (currently 6.3 9050). If you are using a previous version of Clarion 6 however, and using a DLL based product, then you simply download another patch install for the relevant build.
Posted Thursday, March 09, 2006

Clarion Magazine

Free FTP Client With Source

Published 2006-03-10

In 2003 Veronica Chapman wrote a number of highly informative articles for Clarion Magazine on the Windows API, including several on FTP transfers. Veronica's back with a complete FTP client package for Clarion, for all versions from Clarion for Windows 2.0 and up. The package is complete with a reference manual, client application, and source code. From the help file's background notes:

> This Package 'fell out of' my need to organise the various WebSites with which I am involved (in fighting the New World Order). I required a simple way of constructing WebPages (and other 'artefacts', such as images, setups, zips, etc,) and automatically uploading them to their defined places, on Servers, without having to worry I was choosing the right place each time. The reason I mention this is because the features and facilities afforded by this package enable me to:
>
> a. Scan any WebSite under my control, and create a full list of every Folder and File thereupon, where it is (i.e. in which Folder), the Size, Last Access Date/Time of each, and the Access Permissions. I can also scan my Local Platform 'mirror' and reconcile each to each.
> b. Arrange to readily discover and adjust any situation whereby my Local and Remote Copies ever become inconsistent.
>
> By this means I was recently able to move a WebSite from one WebHosting to another, firstly by automatically Downloading everything to a 'null' Local Platform 'mirror'. And then by automatically Uploading, from this 'mirror', to a new WebHosting. The entire process ran virtually automatically.

There are several ways to get the source code.

- In a setup file via Veronica's site. You can also get just the help file here. This is the recommended way to get the CW 2.x (and forward) compatible source.
- If you have problems downloading the file as above, I have a copy of the setup file (as of March 10, 2006) on Clarion Magazine. As time goes on this may or may not be the latest version.
- You can also get a C6 APP and sources from Clarion Magazine. I ported the 2.0 code via TXA, and made some adjustments based on new embed locations and standard equates. You will need the Clarion template chain registered to open this app. A TXA is included in the event you need to back port to an earlier version of Clarion - this may be easier than moving the CW 2.0 TXA forward.

If you're reading this at some later date, you can always check for changes by downloading the ClarionMag copy of the 2.0 code, and the latest version of the 2.0 code, and then porting those changes to the C6 version. I'm a big fan of Beyond Compare for this kind of work - it excels at comparing directories and files.

Help file update

An updated help file is available as of 20th March, 2006, with a number of corrections.

The original source release is unchanged.

If you have tried to use the Package, and had problems - particularly under WinXP - then the Errata should be implemented manually, according to the information given in the latest version of the Help Fil.

Fixes are defined in order to overcome the following problems:

1. Incorrect Help File statement in the definition of FTPFileTransferManager().
2. To correct a post-downloading problem.
3. To amend the CHMOD Access Permissions handling to accommodate Extended CHMOD.
4. Function to determine the Operating System under which the package is

running, and to create workarounds for Win XP issues.
5. Redundant Code in Function FTPCaptureCHMOD().

Download the updated help here.

Clarion Magazine

Direct-To-USB Printing

by Steven Parker

Published 2006-03-31

Using the API to write directly to printer, as outlined in Writing To A Printer Port: Sending Escape Codes, enables me to send text and/or control codes directly to parallel printers. Wrapping the API calls, as outlined in Print Directly to Printer Made Easier, makes it *easy* to send output directly to printers.

Setting up a Device Control Block, a non trivial exercise (see 16-bit Serial RS232 Communications), allows outputting to serial devices. Because it is so easy to mess up creating a DCB, I use Winevent (see Writing to a Printer: The Easy Way) to output to customer displays (pole displays like you see in many stores), read scales and, yes, print to serial printers.

The only thing I can’t do using the API or this template is print to USB printers.

USB need not apply

USB ports do not have convenient names like "LPT1" or "COM27" to pass to the `CreateFile` API to get the handle needed for a `WriteFile` call.

But.

But. The port is a property of a Windows printer and Clarion gives me access to a number of printer properties for the currently set printer. Check out `PROPRINT` in the on-line help.

One of the properties I can access is `PROPPRINT:Port`. The port! And, assuming the USB printer is installed and `PROPPRINT:Device` contains its name (as known to Windows):

```
    Printer{PROPPRINT:Port}
```

returns a port name that I can use in CreateFile (typically, it returns "USB001").

There is one wee problem: the handle returned by `CreateFile` is always `-1`, `INVALID_HANDLE`, for USB ports. Therefore `WriteFile` can't work to a USB port.

So much for building on what I already know.

Crawling around MSDN, I discovered why: USB is a bus, not a port. I don't happen to remember where I saw that. But that's why the port handle is invalid.

Further rooting in MSDN revealed this:

> The Microsoft® Windows® graphics device interface (GDI) enables applications to use graphics and formatted text on both the video display and the printer. Windows-based applications do not access the graphics hardware directly. Instead, GDI interacts with device drivers on behalf of applications.

So, it looks like outputting to USB devices is a GDI function. And, sure enough, the About Printing topic is indeed a subtopic of "Windows GDI."

You should follow the links on the "About Printing" page, attempting to decipher how to print to a Windows printer. You will then understand why I calmly, coolly and rationally threw my hands up in surrender.

It's not that I can't master the printing subsection of the GDI. It's that I simply don't have the ambition to master it. Nor, as it turns out, need I. It's been done for me.

DEVCON to the rescue

At the last DEVCON, I wandered into the Third Party Exhibition area. John Hickey

(POSitive Software) exclaimed "Parker! My goodness you look terrible!"

Now, I was freshly shaved and coiffed. I was wearing a freshly pressed suit, stiffly starched shirt and my tie was well knotted. I’d just had my first cup of coffee (okay, so it was mid-afternoon). How could I look terrible? Yes, I had been worrying over printing to USB devices and I don’t want to have to use a report structure nor force my users to install a USB receipt printer driver and set it as their default printer. But "look terrible?"

Joking aside (I don’t starch my shirts), John pointed me at a piece of freeware by Trevor G. Leybourne which allows programmatic control over printing to any installed printer. This .APP outputs directly to the printer but does so through the windows printer driver and spooler subsystem.

Since my interest is in receipt printers, I figured I could get away with the Generic/Text Only driver assigned to a USB "port." Since, Epson, for example, does not provide a driver disk with their receipt printers, using a built in Windows driver is appealing. In fact, Epson buries their drivers in a restricted-access web site. Once you find and download the driver, it turns out that it is a two step install: you have to select the printer model (one install covers a large number of printers) and the target port. One wrong mouse click and there’s no way to recover. Samsung requires two installs: one for the printer and one to move it to USB. My users couldn’t handle this (it later turned out that you can create an image of your Epson install that can be run from Setup Builder – but you must create an image for each O/S).

Trevor’s readme states, "You can only send Text and Control Codes (no graphics or true type fonts)." Just what I need. His pseudo-code proves it:

```
IF PRINTDRV::Initialise() THEN
  PRINTDRV::WriteText('This line one of text')
  PRINTDRV::WriteText('This line two of text')
  PRINTDRV::WriteText('This line three of text')
  PRINTDRV::WriteText('This line four of text')
  PRINTDRV::WriteText('This line five of text')
  PRINTDRV::Finalise()
END
```

Printing through the Windows’ spooler is a three step process, using Trevor’s wrapper for the Spooler API:

1. Initialize a printer

2. Write the text
3. Close the printer

(Leave it to Windows to obfuscate the obvious and complexify the simple.) From Trevor's documentation:

> **Initialise** (British spelling): The Spooler API requires a handle, just like CreateFile. But the Spooler uses a handle to a printer, not a file or port. The Initialise procedure gets that handle.

Initialise starts by checking whether a handle to the printer already exists. If so, it returns immediately. If not, it gets that handle and:

- opens the printer (OpenPrinter API)
- initializes some required structures
- starts a document (StartDocPrinte API)

and

- starts a page (StartPagePrinter API).

If you do not send Initialise a printer name, Trevor's code assumes the Windows default printer or the last printer assigned via `PRINTER{PROPPrint:Device}`. If you do send a name, it uses the passed printer.

> **WriteText**: Writes the text and/or control codes, up to 132 characters. This procedure uses the WritePrinter API and automatically appends a carriage return/line feed.
>
> **Finalise**: This is when the data is actually printed. As Trevor observes, this is a Windows thing (a "feature"). The Finalise procedure terminates the page (EndPagePrinter API), terminates the document (EndDocPrinter API) and closes the printer (ClosePrinter API). Yup, a Windows thing. Finally, it releases the handle (if it didn't do this, the printer would be locked).

A thought: the Clarion `PRINT` statement must wrap all these API calls.

Caveats

Printing one line at a time:

```
    IF PRINTDRV::Initialise()
        PRINTDRV::WriteText('This is my text')
        PRINTDRV::Finalise()
    END
```

will take forever. The spooler takes several seconds to flush and reset. Count on three to four seconds to `Finalise`.

If you have to do a single line or control code, well, you have to do what you have to do. But, saving as much as possible in a queue or IMDD file and dumping it in a loop is recommended.

Summary

I'm not one for reinventing the wheel. I doubt many Clarion developers are. I firmly believe that quality third party products are always cheaper, in the short run, than "rolling your own."

Mr. Leybourne's PrintDRV.APP (Clarion 5.5 -- a copy is available at my download center as well as a modification in C6 by Jim Gambon) solves direct-to-USB printing problems. I *can* use the Generic / Text Only driver or I can install a manufacturer's driver. I don't need a report procedure. My users get to use cool new thermal printers, and my sincerest thanks to Trevor.

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Reader Comments

Add a comment

Clarion Magazine

Hand Coding Export Files

by Harry Hickey

Published 2006-03-31

Object-Oriented Programming (OOP) can be a wonderful thing. Unfortunately, writing a class can sometimes be easier than using it, especially if you need to export that class from a DLL.

The simplest way to export a class in Clarion is to make the class part of the ABC library. Unfortunately, that ties the class directly to your Clarion ABC environment, usually exporting from the base "Dictionary" DLL in a DLL-based system. If this is not the desired goal, this is probably not the way to go.

There is a slightly more difficult method for exporting classes from any DLL, but it involves playing around with the global export list embed (for apps) or the EXP file (for projects). The DLL needs to export the Virtual Method Table (VMT), the `TYPE` for the class, and each individual method (procedure) in the class.

Understanding the nuts and bolts

It is useful to understand the way Clarion handles a class. The data elements, or properties, of a class are treated internally as a typed group structure. Therefore, when you create an instance (or object) of the class, the data portion is treated as a `GROUP`. For that reason, it is essential that the data elements in the class definition in the instantiating program precisely match the ones in the DLL. Clarion will make no attempt to line up your fields. It will simply pass a reference to the starting location of that group in memory.

When exporting a class from a DLL, you must export a `TYPE` for the class. For an example class called `MyClass`, there should be a `TYPE$MYCLASS` present in the export file.

The procedures and functions, or "methods", are generally exported by their "mangled" name. For those not familiar with name-mangling, a simple procedure `MyProc` which accepts no parameters would simply be exported as `MYPROC@F` (the `@F` defines this as a procedure/function). This helps enable procedure overloading. If you have another procedure, `MyProc(STRING)`, , it which will not be confused with `MyProc` since `MyProc(STRING)` will be exported as `MYPROC@Fsb`. The "sb" portion shows the first parameter to be a Clarion `STRING`.

In a class, the data portion is always passed as the first parameter automatically (which explains why, in class methods, checking for `OMITTED(1)` will never return `TRUE`.) This data portion is the `SELF` which represents the instance of the class. When calculating the mangled name for export, this instance data needs to be taken into account. Because your Class data is really a specially defined group type (like `TYPE$MYCLASS` above), Clarion requires that you preface the type name with the length of the type name so it knows how many letters are part of the name. So, for a method of the `MyClass` class that does not receive any additional parameters, the mangled name would be `WHATEVER@F7MYCLASS` (where `WHATEVER` is the *uppercase* name of the actual method and 7 is the number of letters in `MYCLASS`).

Any additional parameters follow; for instance if the Whatever method took a `BYTE` as a parameter, the mangled name would become `WHATEVER@F7MYCLASSUc` (Uc is a `BYTE` or unsigned char).

Return values from functions are ignored when mangling names. This gives the function `MyFuncton PROCEDURE` and `MyFunction PROCEDURE,BYTE` the same mangled names.

The only other piece of the puzzle is a reference to a Virtual Method Table (VMT). Without the VMT, derived methods in the instantiating program would not work with the class in the DLL. To export the VMT, Clarion needs to have a corresponding entry in the export list. For `MyClass`, it would look like `VMT$MYCLASS`.

Calculating the export list items

I will use the following class, `SampleClass`, as an example for building an export table. The class definition is as follows:

```
SampleClass CLASS, TYPE
MyString      STRING(10)
MyLong        LONG, PROTECTED
MyByte        BYTE, PRIVATE
Init          PROCEDURE(STRING,LONG,*BYTE,), BYTE
MyMethod      PROCEDURE,VIRTUAL
MyPrivate     PROCEDURE,PRIVATE
Kill          PROCEDURE
            END
```

In this example, there are three properties which establish the `SampleClass` data type. There are also four methods. In order to make the `MyPrivate` method truly `PRIVATE`, this method will not be exported (see Privacy Policy below).

The first step is calculating the mangled names for the three methods that will be exported. The `Kill` method is the easiest, since it takes no parameters. It would simply be written as `KILL@F11SAMPLECLASS` (remember, it is necessary to declare the 11 characters in `SAMPLECLASS` before using the class name itself).

The `MyMethod` method also takes no parameters. The `VIRTUAL` attribute does not affect the naming, although it does affect how the Virtual Method Table treats this method.

Init takes several parameters, and gets mangled as `INIT@F11SAMPLECLASSsblRUcOs`. If this looks confusing, see the list of mangled variable types at the end of this article.

Here is the example export list:

```
VMT@SAMPLECLASS              @1
TYPE@SAMPLECLASS             @2
INIT@F11SAMPLECLASSsblRUcOs  @3
MYMETHOD@F11SAMPLECLASS      @4
KILL@F11SAMPLECLASS          @5
```

In addition to exporting names, each exported item is expected to have a number, or ordinal, associated with it. In a project, it is simple enough to assign an ordinal to each exported item. If the exported items are being added to the export embed in an APP, it is safest to let Clarion handle the numbering itself by using `@?` instead of a specific number.

Hand-coded EXP files must include the line

```
    EXPORTS
```

prior to the export list itself.

Don't forget the basics

In order to use this DLL from another program, the LIB file must be included in the project, and the class declaration must also be present, with appropriate linking information. Changing the first line from

```
    SampleClass CLASS, TYPE
```

to

```
    SampleClass CLASS, TYPE, MODULE(‘MyDLL.DLL’)
    ,LINK(MyDLL.dll',_ABCDLLMode),DLL(_ABCDLLMode)
```

is all that is really needed here.

Privacy policy

The checking of `PRIVATE` and `PROTECTED` attributes is really a function of the compiler. Once the class is compiled, there really are no such attributes.

Because the entire data section is passed as a group, there really are no such things as `PRIVATE` or `PROTECTED` properties. Even if a property is marked as `PRIVATE` in the DLL declaration, simply removing the `PRIVATE` attribute from the property in the declaration of the instantiating program will make that property public.

As for the methods, any method that is exported can be accessed, even if it is `PRIVATE`, just by removing the `PRIVATE` attribute in the instantiating program. If a method must be truly private, it should not be exported. `PROTECTED` methods must be exported to be accessed from the instantiating program, so there’s no way to absolutely enforce the `PROTECTED` attribute.

When things go wrong

If the linker reports an error that a procedure is unresolved for export, check the mangled name in the export file. The export name must exactly match the name that Clarion will calculate.

Summary

All that is needed to export a class from a DLL is a properly defined export file. For each exported class, the export file must have an entry for a Virtual Method Table (`VMT$`), an entry for the class type (`TYPE$`), and an entry for each exported class method. Compiling the DLL will create a LIB file that can be used for linking to the DLL.

Resources

Some Parameter Types for Mangled Names

```
BYTE        = Uc
SHORT       = s
LONG/SIGNED = l
ASTRING     = sa
STRING      = sb
CSTRING     = sc
GROUP       = g
VIEW        = Bi
QUEUE       = Bq
KEY         = Bk
FILE        = Bf
REPORT      = Br
?           = u
```

Important Prefixes for Mangled Types

`O` for optional parameter

`R` for parameter passed by Reference (* parameters)

P for Optional Reference parameter

U for unsigned

Harry David Hickey has been programming in Clarion since 1990. He is currently a Senior Software Developer for Nebo Systems in Oakbrook Terrace, Illinois.

Reader Comments

Add a comment

- » Nice article, I been looking for a good explanation on...

Clarion Magazine

Using DOS Files To Send Printer Codes

by Olivier Cretey

Published 2006-03-31

Recently, Steve Parker wrote about using the API to write directly to parallel port devices. That's a useful technique, but there are other ways.

When introducing the Win32 model, Microsoft took some ideas from Unix, as it did with DOS. From these borrowings, Windows inherited some pipes and a console (remember the old `COPY > CON` syntax?). Microsoft also took the idea that a device can be managed just like a simple file. Because of that decision, the API calls Steve discussed in his article all reference basic file manipulations: opening a file, writing to that file, and closing it. Even the error checking is file oriented. So a device can be used like a file. Well, not all devices, but parallel and serial ports can be used this way.

This statement stayed in my mind for a long time until I had to write a little project with Clarion: I had to send some commands to a serial device, without the need to get an answer back. The simple answer was to use WinEvent or Clacom, two Clarion add-ons, which make serial port programming easier than calling the Windows API. But I called the API anyway, just like Steve did. A little later somebody asked me if I could write a small program to send ESC commands to a printer, so I did a cut and paste of my API code. My previous little project had required a file name of `'COM2:9600,N,8,1'`; I replaced this with `'LPT1:'`.

File name… a DOS file name… did I say DOS File ?

What a shame ! I'm using a tool to manipulate files all the day (and night sometimes) and I didn't realise that my favourite tool was offering the right file driver to accomplish what I was doing with these API calls! From a Clarion point of view, I needed to open a DOS File, write to its buffer, close it, and of course do some simple error checking. Nothing fancy, I doing this all the time when I'm writing ASCII log files. I ended up with a very simple Clarion project using the DOS file driver. Here's the code:

```
Program
  Map
  End
Prn  File, Driver('DOS'),Create,Name('LPT1:')
Record Record
Line   String(1024)
       End
     End
  Code
  Open(Prn)
  if Not ErrorCode()
     !          Bold on                       Bold off    Form Feed
     Prn.Line = '<27,60>' & 'Hello World !' & '<27,70>' & '<12>'
     Add(Prn)
     if ErrorCode()
        Message(ErrorCode() & ' ' & Error(), 'Writing Error !')
     End
     Message(Clip(Prn.Line) & ' Printed on ' & Clip(Prn{Prop:Name}),|
            'Printing Done !')
     Close(Prn)
    Else
     Message(ErrorCode() & ' ' & Error(), 'Opening Error !')
  End
```

As you can see, it *is* very simple, but there are a few lines worth explaining.

```
Prn File, Driver('DOS'),Create,Name('LPT1:')
```

I declare a standard DOS file and label it `Prn`, just because it is a short meaningful label. The only special thing is its name: `'LPT1:'` I could use `'PRN:'` for the default parallel port, or `'LPT2:'` or any valid (from a DOS point of view) device name. That means that I can use `'COM2:9600,N,8,1'` or any other COM port to send commands to a serial device. I use this technique to tell my modem to dial a number for me.

```
Line String(1024)
```

I'm using a 1024 byte buffer for this file. This value isn't required, it is only an old habit. I could use 2048 or 255, it does not matter.

After the Code statement I open the file:

```
Open(Prn)
```

The file must be open to enable communication from the program to the parallel port. When calling the API I need a handle to the file; using this technique, the Clarion DOS driver takes care of the handle. I just write to the file.

```
    Prn.Line = '<27,60>' & 'Hello World !' & '<27,70>' &
      '<12>'
```

Here we go! I'm filling the buffer with a mixed of ESC commands and text. 27 is the escape code, followed by its complement code; 60 (or "<" Bold on), 70 (or "F" Bold off) 12 is the form feed command. For more ESC commands, read your printer's documentation.

```
    Add(Prn)
```

The Add command sends the content of the buffer to the file; since the file is really the `LPT1:` device, the data goes right to the printer. I could break `Prn.Line` into shorter commands, or even use equates to define ESC codes to add clarity to the source. For example:

```
    ! Constant Equates
    BoldOn  Equate('<27,60>')
    BoldOff Equate('<27,70>')
    FormFeed        Equate('<12>')

    !Code
    Prn.Line = BoldOn & 'Hello World !' & BoldOff
    Add(Prn)
    Prn.Line = FormFeed
    Add(Prn)
    Close(Prn)
```

I've finished using the printer, so I close the file and therefore the connection to the printer.

USB printers

You should also be able to use a USB printer with another trick:

- Share the USB printer from the Windows printer control panel.
- From the command line (Windows Start Button, Execute, type `cmd.exe`), then type `net use LPT1 \\computer_name\printer_shared_name [enter]`

To verify that the USB printer is now using LPT1, type from the command line:

```
    net view \\computer_name [enter]
```

LPT1 should appear under "used as" column on the USB printer line.

Editor's note: Petar Subica in the SV newsgroups recommended adding

/persistent:yes to the end of the `NET USE` command.

Summary

Basic printing services and serial communications are not implemented by the Clarion language, but there are many places Clarion developers need them. There are Point Of Sale systems where ESC commands are needed to configure printers or open serial cash drawers.. Many applications need to do simple modem communications (the downloadable source includes an example). You can probably think of other examples. Using the simple technique I've described here, you can address many of these tasks without the complexity of a bunch of API calls or classes, and without the hassle of DLL updates and redistribution.

Download the source

After completing scientific studies, Olivier Cretey spent more than a decade as a professional sailor, skier, and golf instructor. During this time, at the age of 26, he discovered computers. One day he bought a Z80 computer (at 6:00 p.m.). He wrote his first program ("hello world!") some hours later (at 6:00 a.m.), first with the integrated 16K ROM BASIC, and some weeks after in ZEN assembler, stored on audio tapes. Olivier began taking night courses at Dijon University, and then progressed to full time studies at the Control Data Institute of Paris, France. One day he met a little thing named Clarion for DOS version 1.0...

Reader Comments

Add a comment

Clarion Magazine

Using MS Visual Source Safe With Clarion

by Marty Honea

Published 2006-03-30

Most developers, at one time or another, lose programming changes due to overwritten files, or end up with numerous copies of an application on their machine in an attempt to save a trail of their work. Version control software solves the problem of lost code and provides a nice tidy trail useful for tracking changes made to an app. In this article I'll talk about the software I use for version control, and I'll tell you how I use it.

Visual SourceSafe, also referred to as VSS, is a software solution to version control offered by Microsoft. Using VSS is quick and simple, but there are a few things that will end up ruining your day if you're not careful. I'll try to point out as many of those as I can in this outline of how I use VSS.

Installation of VSS is very simple. The only thing to consider is where you're going to install it. If you're using it by yourself, to keep revisions of your code, you'll be fine with installing it locally to your development machine. If you're using it to track development of multiple developers, preventing one developer from overwriting another's modifications, you'll want to install it in a shared directory on a server that is accessible to all the developers who will use it. The latter use is the most common use for version control, and is the one I'll focus on in this article. All my suggestions can apply to both uses, but will be worded to address multi-user environments.

In setting up VSS, you will want to consider the needs of your development staff and how they work. If you have a core set of applications, and multiple projects that use this core, you might want to create a project that contains this core set and then create VSS "shares" of that project for the other projects. This prevents duplication of your core pieces and keeps all the projects tied to a common set of files. If the core project changes, you'll get those changes the next time you check out your project.

If you have multiple projects that started with a similar set of applications, but will not stay synced up to the starting project, then you might set up a branch of the starting project.

In VSS, sharing a file means it exists equally in all the projects to which it is shared. Branching a file breaks the shared link, making the file in that project independent of all other projects. I try to share as many of the common files as possible. I hate trying to keep multiple dictionaries synced. I'm lazy and duplicating work is way down on my list of things I like to do.

The first time you log into VSS you will need to log in as Admin. You can use the Visual SourceSafe Administrator (SSADMIN.EXE) to add additional users as needed, giving them either read access, or read/write access. But for now I'll be working as the Admin to show you how to set up projects. When I log in for the first time, the screen looks like Figure 1.

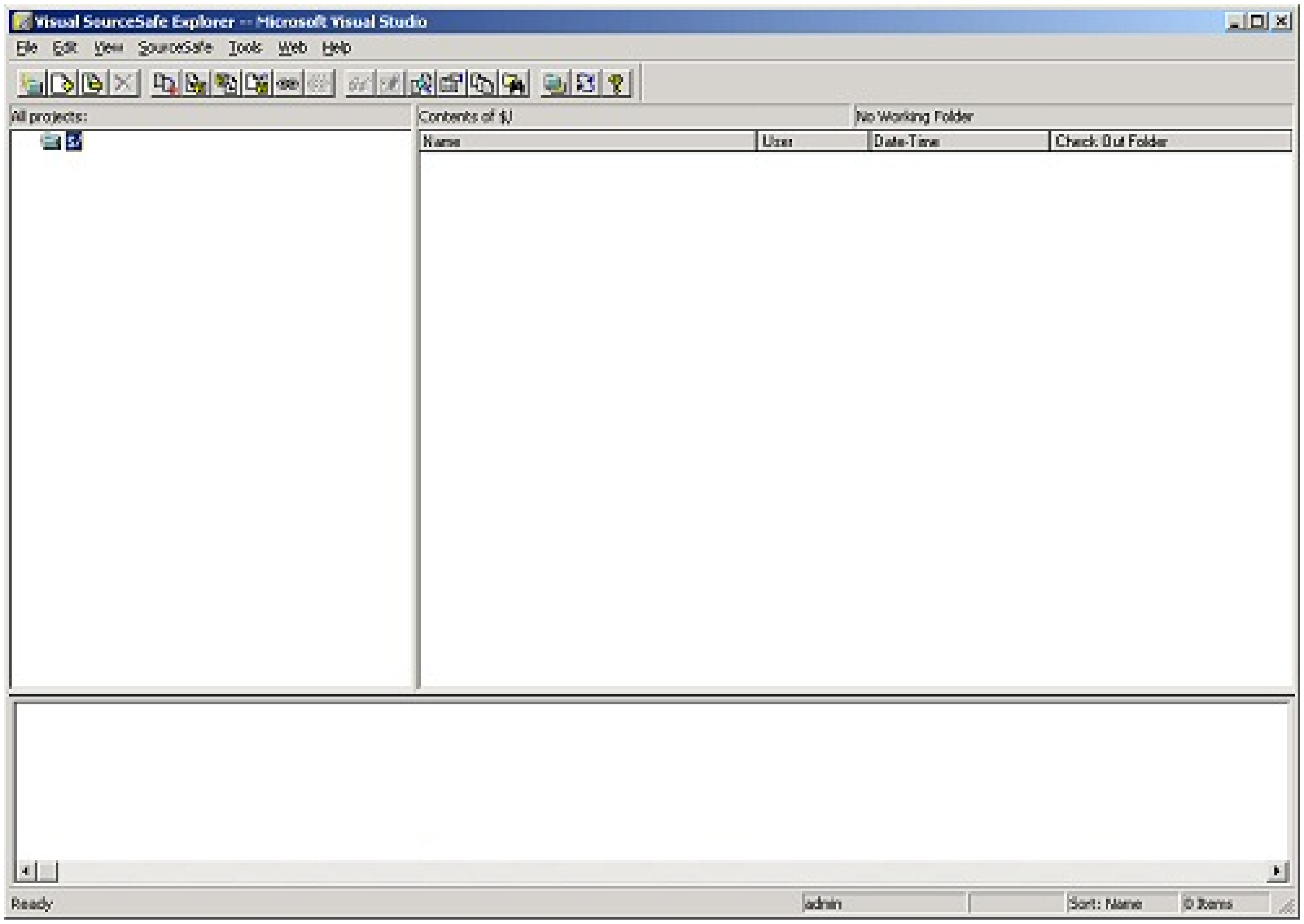
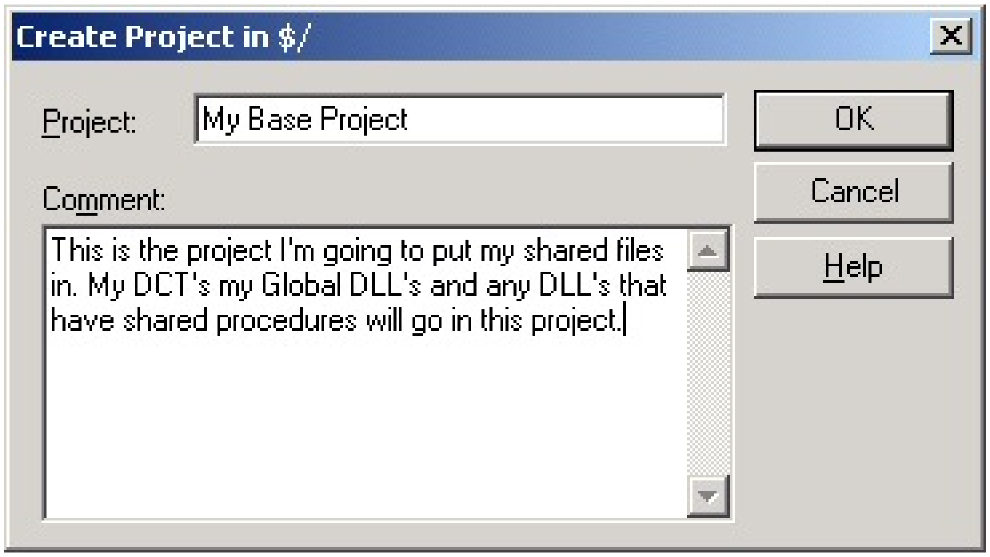


Figure 1. The VSS main window (view full size image)

On the left, a pane displaying All Projects is empty. And on the right, there is a pane that will display the files included in the projects. To create a project right click in the all projects pane and select Create project from the popup menu. The Create Project form will open, as shown in Figure 2.



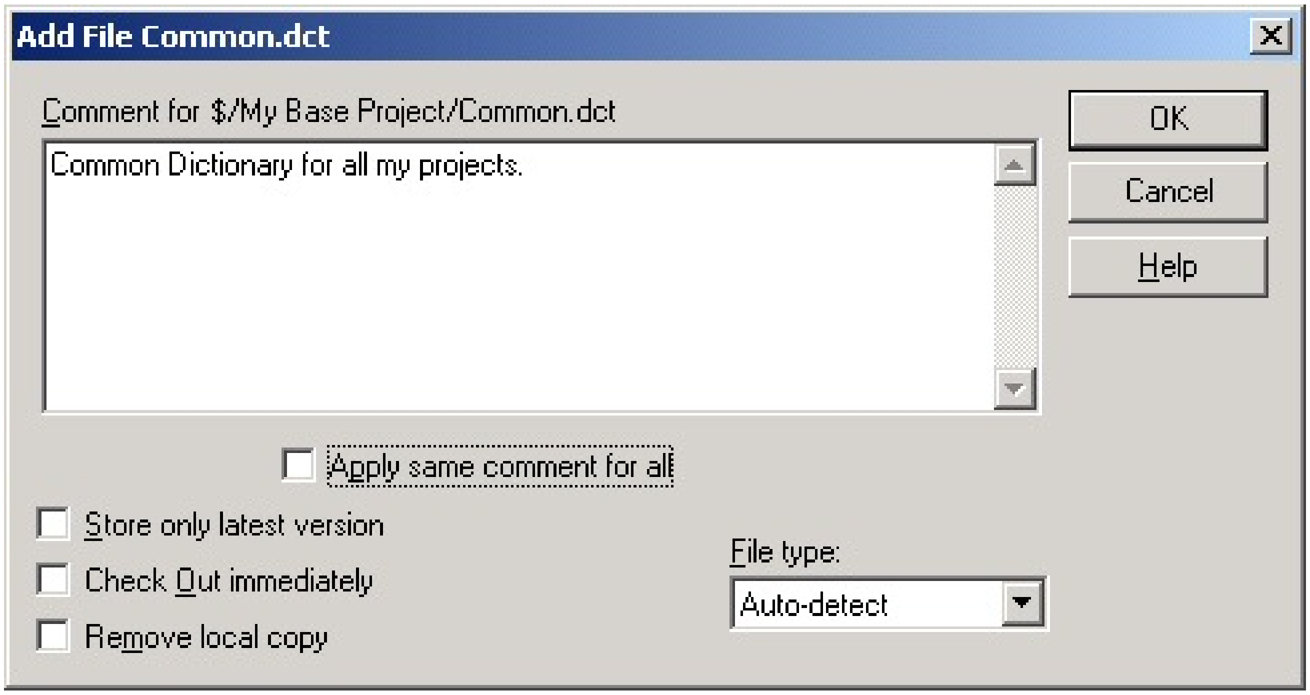
My Base Project

This is the project I'm going to put my shared files in. My DCT's my Global DLL's and any DLL's that have shared procedures will go in this project.

Figure 2. The Create Project form

Give the project a name, and a Comment that makes it easy for other developers to understand what’s contained in that project. Then click the OK button. The project is now listed on the left, and is ready to have files added to it.

To add files to a project, I can just drag the file to the pane on the right, and VSS will ask me if I want to set the folder I dragged the file from as the personal working directory for this project. I’ll go into more detail about personal working directories a little later, but since this directory is the one that I’ll do most of my work on this project in, I’m going to answer Yes. After selecting Yes, I’m presented a form to enter comments for this file. If I select the advanced button on the lower right hand side of this form, I get something that looks like Figure 3.



Comment for $/My Base Project/Common.dct

Common Dictionary for all my projects.

☐ Apply same comment for all

☐ Store only latest version

☐ Check Out immediately

☐ Remove local copy

File type: Auto-detect

Figure 3. File comments

The options here are pretty self explanatory. I’m big on commenting each file individually. It’s hard to remember everything you need to know about a file, when you haven’t looked at it in a year. This is the place to put that information. I uncheck the "Apply same comment to all" check box which is defaulted to checked.

You’ll notice there is a "Store only latest version" check box here too. To me, that defeats the purpose of having a version control system. But I’m sure that someone out there had that need, so here is where you would set it. The "Check Out immediately" option will store your file in VSS, and then immediately mark it as checked out so that no one else can do so before you get back to it. And the "Remove local copy" does exactly that. Once the file is copied to VSS, VSS removes the copy from your machine.

I never change the File type option from Auto-detect. It does a pretty good job of detecting if a file is text or binary on its own.

After checking in the core pieces, I create another project named "My Program". This will be the project that contains everything needed to compile this program. In it, I’ll share the core pieces and add the files unique to this project. To add the core pieces, I right click on "My Program" and select share from the popup menu.

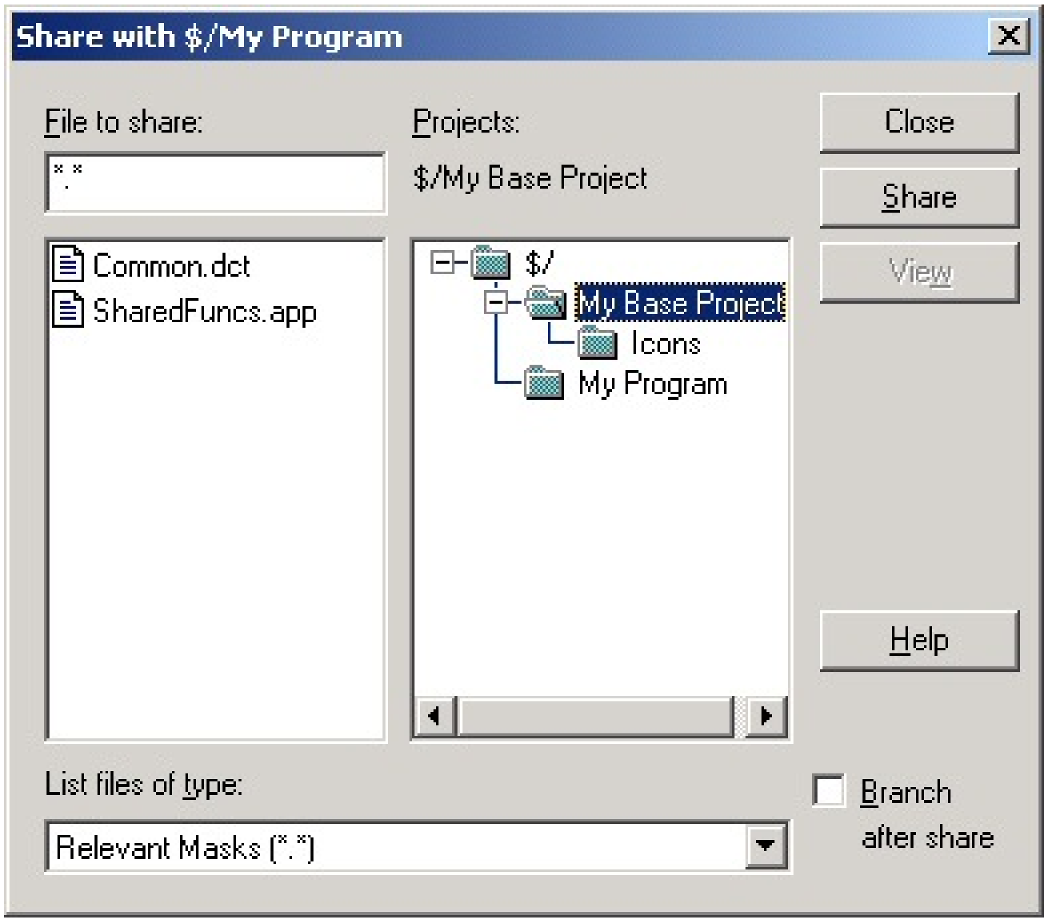


Figure 4. Creating a share

From this browse, I select the files I want to share. On the bottom right hand side of the browse is a check box labeled "Branch after share". If this is checked, VSS will make a separate copy of the files. This is useful if you're starting with an old project, and will be making changes to this new project that you don't want reflected in the old project.

After making sure all the common files are shared. I can drag and drop the unique files for this project into it in the same manner that I used creating the base project.

If you look at the icons used for the files in the "My Program" project, you'll notice the unique file uses a single page icon, and the shared files use an icon that looks like multiple pages. This makes it easy to identify shared files at a glance.

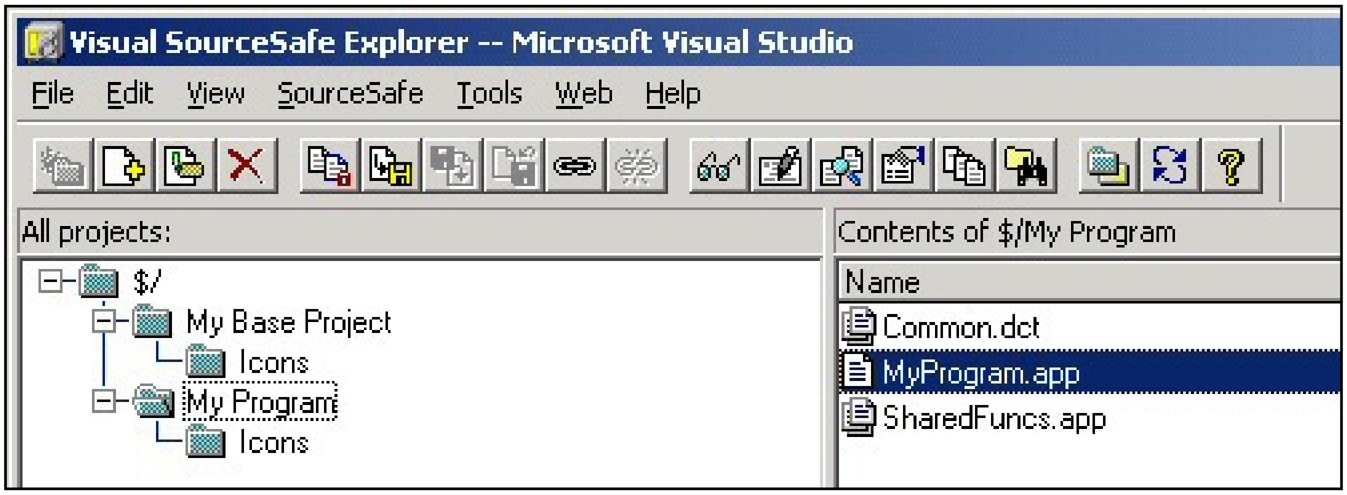


Figure 5. Specialized icons for shared files (view full size image)

To branch a project, I would follow the same steps as before but this time I would check the "Branch after share" option. This will copy the files from the other project to this new project. When working on files from this new "Stand alone project" I would not have to worry about compatibility with other projects.

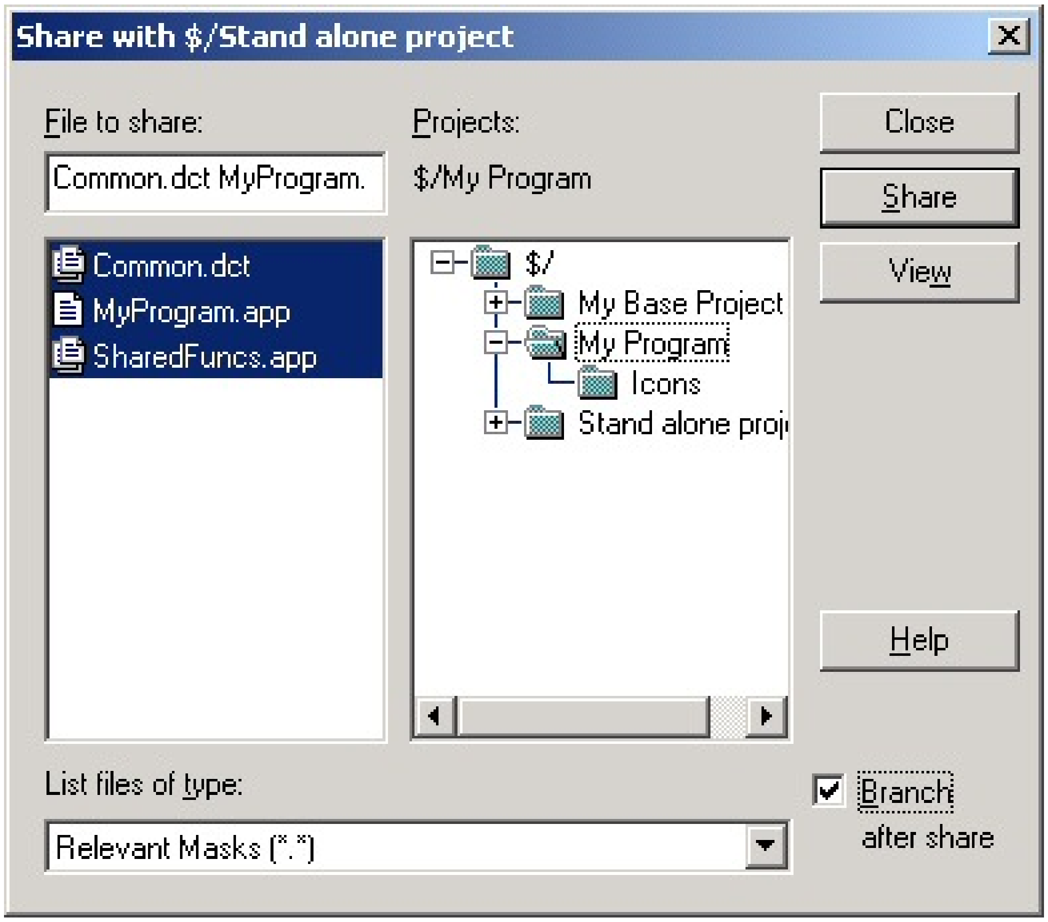
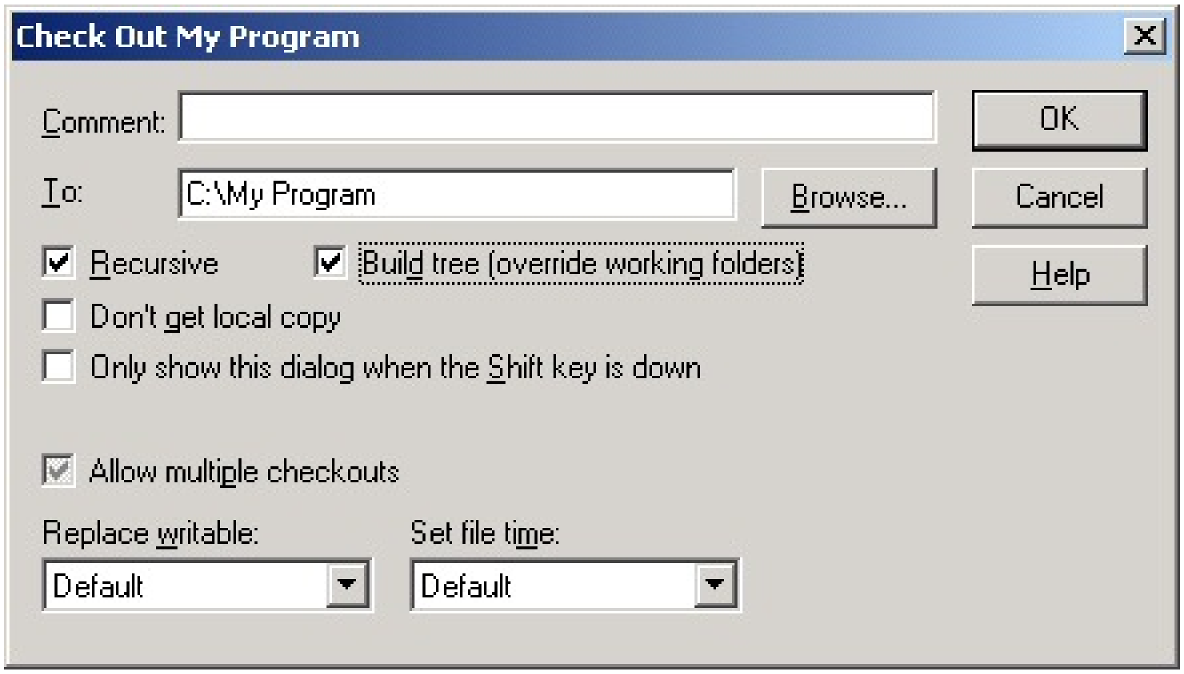


Figure 6. Using Branch after share

This is useful for working on different revisions of software. For instance, if you were moving an app from Clarion 5.5 to Clarion 6, you could start by making a branch of 5.5 as a starting point, and anything from that point forward would not affect the 5.5 version.

Now that my projects are set up, I'd like to check out a copy of the project. This is simple to do. I right click on the project and select the Check Out option from the popup menu.

If I select the advanced button on the resulting browse I see the window in Figure 7.



Comment:
To: C:\My Program
[✓] Recursive
[✓] Build tree (override working folders)
[ ] Don't get local copy
[ ] Only show this dialog when the Shift key is down
[✓] Allow multiple checkouts
Replace writable: Default
Set file time: Default

Figure 7. Advanced checkout options

Most of this is fairly self explanatory. The comment entry allows me to document why I checked out the project. The "To" entry lets me select a path to where I'm going to put the project. This will default to my personal working directory. Setting up a personal working directory that VSS will remember for each project saves me time when checking out, and helps keep me from checking out multiple copies of the project. The project is always located in the same place. I can check out a project, and then open it with Clarion's pick dialog. Like I said, I'm lazy and anything that saves me time makes me happy. If my timesaver helps me get into a routine that helps to minimize mistakes on my part, that's even better.

The Recursive check box checks out the project and all its subprojects. I use this for categorizing my files. For instance, I try to keep images in a separate folder. So under most of my projects there will be an Icons project. When I do a recursive checkout, all my icons are refreshed also.

The Build Tree checkbox is hidden until the Recursive checkbox is checked. It gives me the ability to create the working directory using the same directory structure defined in VSS.

The "Don't get local copy" check box allows me to check out a project, but leave my local files intact. This can be dangerous. Some developers will get the latest copy of a file without checking it out. They'll try different approaches to what they're trying to accomplish, and then when they get what they want, they'll check out the project without getting a local copy and check it back in. This opens up the possibility that someone else has checked out a copy of the file and checked it back in with changes since the developer started with their changes. It's a bad idea and defeats the purpose of Version Control. Just don't do it!

The "Allow multiple checkouts" check box allows multiple users to check out this file/project simultaneously. If another user already has this file checked out, the check box is selected and unavailable. If multiple check outs are disabled for this project, the check box is cleared and disabled. When a file or project is exclusively checked out, the icon for that item has a red box around it. Here again, Just don't do it! It defeats the purpose of Version control. Only let one developer makes change to a file at a time. If you keep this principle at all times, you won't have developers mad because someone else overwrote their changes. Nobody likes to have to make their changes twice.

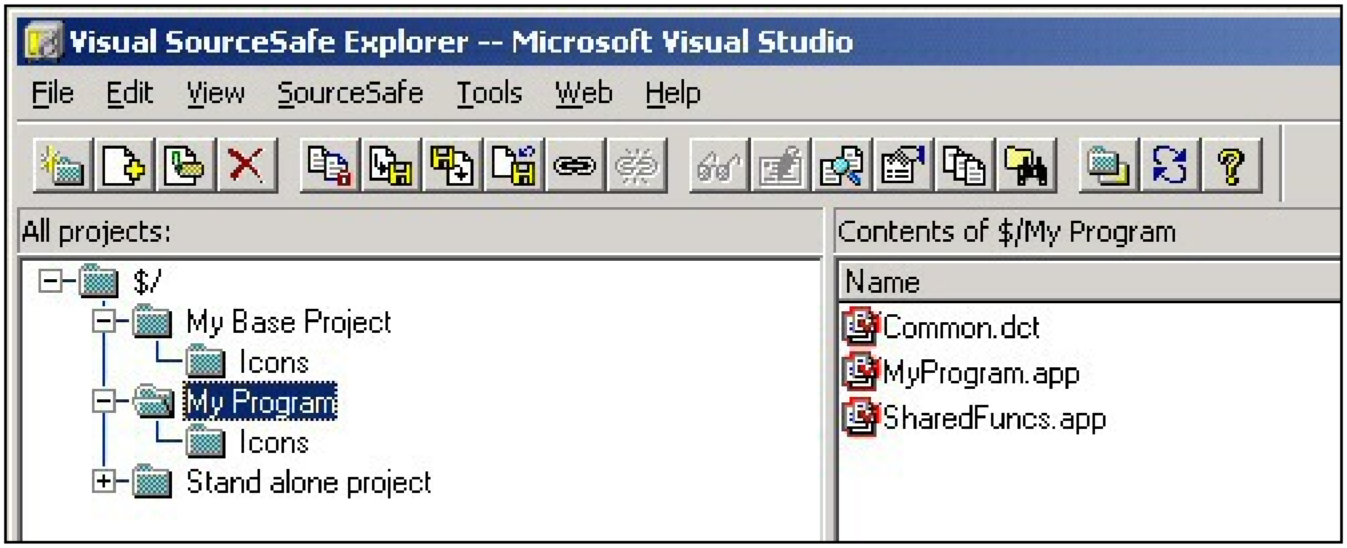


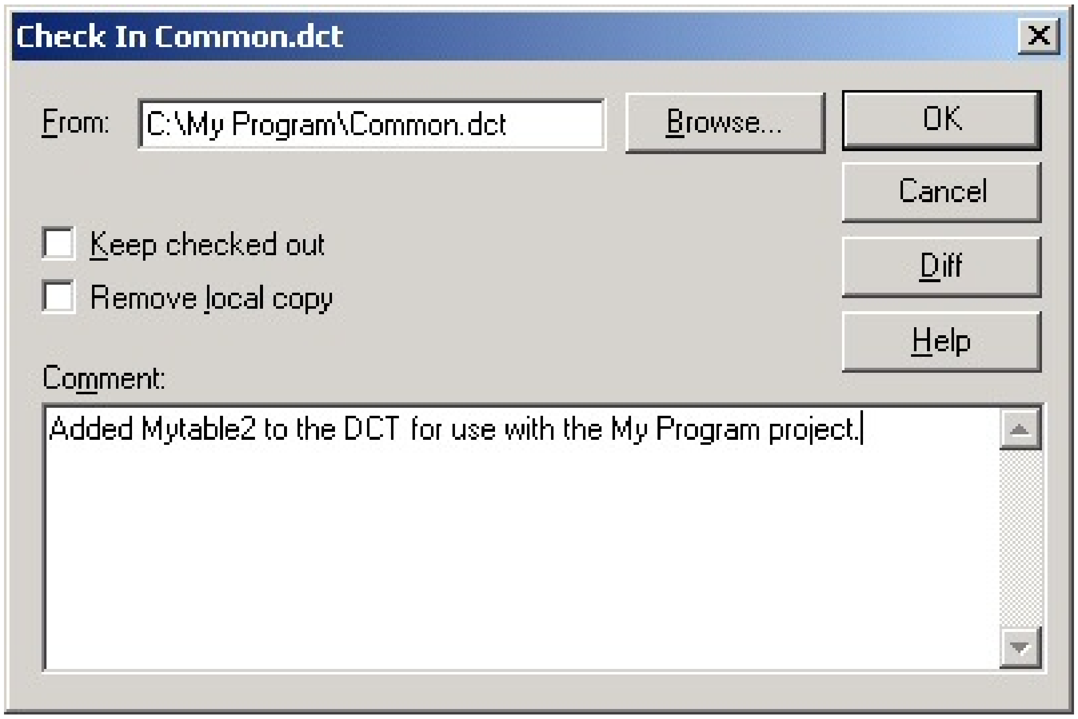
Figure 8. Icons for checked out files

When I check out the project all the files in the project will icon will have a red box around it (see Figure 8), and the details for that checkout will be displayed. Who checked out the file, when they checked it out, and which directory they checked it out to on their machine will be listed next to each file. VSS will put a writable copy of the file in that directory. Once I've checked out the needed file(s), I'm ready to work on the project in Clarion.

Developers can get the latest copy of a file or an entire project by right clicking and selecting Get Latest Version from the popup menu. They can do this at any time, even when a file is checked out. So it's a good idea to check in changes often. Getting the latest version copies a file to the local machine and sets the read only attribute.

Before checking in a file, always make sure you have closed the file in the Clarion IDE. Bad things happen when you don't. For instance, if you haven't closed your APP before checking it in, VSS will pick up the APP file, which doesn't necessarily have your latest changes stored in it yet. Meanwhile your changes are in the current working copy, which has the AP~ extension. This will give you the false impression that everything is backed up. When you check the file out the next time, VSS will overwrite your changes with the version you checked in. Saving the file would work, but closing the file leaves no doubt. Always, always, always, close your application before checking in.

Checking in a file is just as easy as checking out. I select the file/project I wish to check in and right click. This time I select the Check In option from the popup menu. The Check in window in Figure 9 appears.


Check In Common.dct

From: C:\My Program\Common.dct

Keep checked out [ ]

Remove local copy [ ]

Comment: Added Mytable2 to the DCT for use with the My Program project.

Figure 9. The Check In window

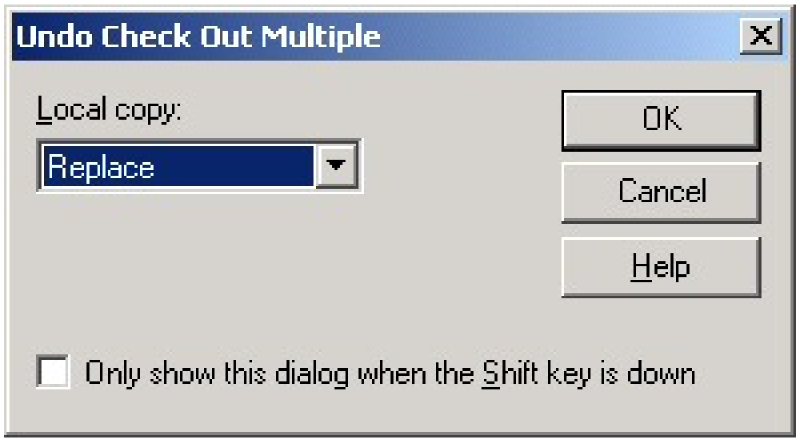
I try to add comments when I check in. It makes it easier to track versions if I have to go back a version for some reason.

The Keep checked out check box will put a copy of the file in the VSS database and will keep it checked out to me.

The Remove Local copy option will check the file in and remove it from my system.

I guess this is a good place to mention how VSS stores the files. VSS creates a set of subdirectories and makes duplicate copies of files that are checked in. It doesn't actually store everything in a database. This feature makes backing up VSS very simple. There's no worry about backing up a SQL database, just make sure the main directory for VSS is in the backup queue and you're good to go. This also means that the VSS directory will continue to grow as time goes on and more and more check ins are recorded. So keep this in mind when you decide where to put your directory.

From time to time I check out a project, and then I realize that for some reason or another I didn't really need to check it out. I might check out a file and then after going through the code find that the Procedure I need to modify is actually in another App, or I might realize that the problem I'm working on is a data issue and not a code issue. When this happens, I don't want to check the file back in, adding another unneeded copy to the VSS directories. Fortunately VSS includes an Undo checkout option on the popup menu that handles this very situation (Figure 10).



Local copy: Replace

Only show this dialog when the Shift key is down

Figure 10. Undoing a checkout

The drop down options for dealing with the local copy of the file are Replace, Leave, and Delete. They're pretty self explanatory.

Every window in VSS has an entry in the help file that can be accessed by pressing the help button. If there’s ever any doubt about which options to pick, click that button. The help is laid out in an easy to use manner, and should help prevent you doing something that you will regret later.

Summary

VSS isn’t the perfect solution to version control, but it’s one of the least painful routes I’ve found. VSS is fairly easy to set up, very simple to use, and if you keep in mind the few "gotcha’s" I’ve documented here it could save your bacon on more than one occasion. I know it has saved mine a time or two.

Marty Honea wrote his first program at a summer camp when he was 9, in BASIC on an Apple IIe. Among other jobs, he has worked as a draftsman, as an Emergency Medical Technician, and as an IT manager for three prisons. He picked up Clarion for Windows in 1998 and, despite having no formal training in computer programming, has managed to make a living with Clarion ever since.

Reader Comments

Add a comment

Clarion Magazine

The Easiest Way To Write To A Printer Port

by Steven Parker

Published 2006-03-30

In the last less-than-action-packed installment, I said that I was not going to create a template for the write-directly-to-port wrapper procedure discussed. There is a very good reason for not creating that template. It's been done already. At least twice that I know of. And it's been done very well.

The two third party products that I know about are Martin Allen's Nova templates and Capesoft's Winevent.

The Nova Templates

A US$20 shareware product, the Nova Templates provide four extensions. `LinePrint` is the one for which these templates are best known. In fact, this template set is often referred to as simply "the line print template."

> LinePrint Template adds a global function LinePrint() which allows bypassing the Windows Print Engine. With LinePrint you are able to print directly to the printer ports and files.

The documentation notes that `LinePrint` is designed for "barcode printers or label printers requiring certain control codes or plain text input to print correctly." The last mentioned use, sending control codes, is what started me on this series of articles,.

The `LinePrint` function is quite similar to the wrapper I created in the last article:

```
    LinePrint(StringToPrint, <DeviceName>, <CRLF>)
```

There are two differences between what I did in my wrapper procedure and what Marty does.

The first difference is that the port (second parameter) is optional. If omitted, the default printer is the target. If specified, the default printer is ignored. UNC names are supported.

The second is that carriage return, line feed is assumed. The template automatically appends a carriage return, line feed unless a non-zero value is passed in the third parameter. Also, if you examine the code in the TPL file, you will note that carriage return, line feed is automatically *not* appended if the `StringToPrint` is a form feed. Neat.

There is one documented "limitation" (it may or may not be a limitation but it is good that it is mentioned): "The maximum line (StringToPrint) length is 500 char which (in most cases) should be more than enough." This is immediately followed with a worthy piece of advice: "By customizing the template source you can increase the buffer size. Just set different size for the variable hpvBuffer."

Also note the comment: "The LinePrint uses API functions that are compatible with all the Windows versions since Windows 1.0." Looking at the template code, I do not find `CreateFile`, `WriteFile` or `CloseHandle`. These functions did not exist until 32 bit Windows.

The Nova Templates use `OpenFile`, `_lwrite` and `_lclose`, all 16 bit functions. All are included in Kernel32.lib so they continue to work.

I have tested the Nova templates with a serial printer (though not recently) and a parallel printer. WritePort_Easy2.APP (compiled locally and in the downloadable source) and WritePort_Nova.TXA show an implementation of the Nova Templates.

The template works and works well. It is easy to implement.

Winevent

Winevent offers a wide range of port communications functions. These include:

- `NewPort` – a wrapper for `CreateFile`
- `WritePort` – a wrapper for `WriteFile`
- `ClosePort` – a wrapper for `CloseHandle`

which are directly related to the task of sending control codes to printers. Winevent can also read from ports and, generally, service a substantial range of serial port settings and activities (not as many customized functions as CLAComm, but the basics are made much easier than using the API).

The three functions listed above are the three used to write to a port. WritePort_Easy.APP (compiled locally and in the downloadable source) and WritePort_Easy.TXA show an implementation using Winevent to write directly to a printer.

`NewPort`, however, is the most interesting function to me. It is interesting because it bailed me out of a deep hole a few years ago.

```
    NewPort (string pmode, <long pInb>,<long pOutb>,byte pUseEvents=0)
```

I support an app that has to write to customer displays. These are serial devices, often called pole displays. Serial ports require much more work than other kinds of ports. Devices on serial ports have attributes like baud, parity, stop bits, flow control, etc., etc.

In order to work with these devices, serial ports require a Device Control Block (the reader is referred to Paul Attryde's article Synchronous Serial Communications). The DCB contains the various settings required by the attached device. Several additional API calls are required to apply the DCB and initialize the port.

Creating a Device Control Blocks is very, very easy to get wrong. Very. My app failed when the Device Manager specs did not match the specs I was trying to set.

Using the information from Paul's article, using CLAComm, using… anything I could think of, three days later still no joy. I turned to Winevent and noticed that `NewPort`'s first parameter is a "string such as would be accepted by the DOS MODE command."

For example:

```
    PortId = NewPort('Com1:9600,n,8,1')
```

Well, I remember the MODE command and, in fact, had been getting around my problem by opening a DOS window and typing the Mode command for the device.

In other words, `NewPort` takes care of creating and applying the Device Control Block.

Eureka! It was working.

Summary

Either of these third party products does the job of sending control codes to printers. Given that they are available, there really is no need for me to make a template out of my wrapper procedure. In fact, given that they exist, there is no reason I should have bothered creating that wrapper at all. With quality third party products it is rarely the case that the cost/benefit ratio comes down on the side of "roll your own," doing it yourself.

The only thing neither of these templates support is printing to USB printers (on USB ports, not using serial or parallel converter cables).

Download the source

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Reader Comments

Add a comment

Clarion Magazine

Clarion Challenge Results - Remove Links

by Dave Harms

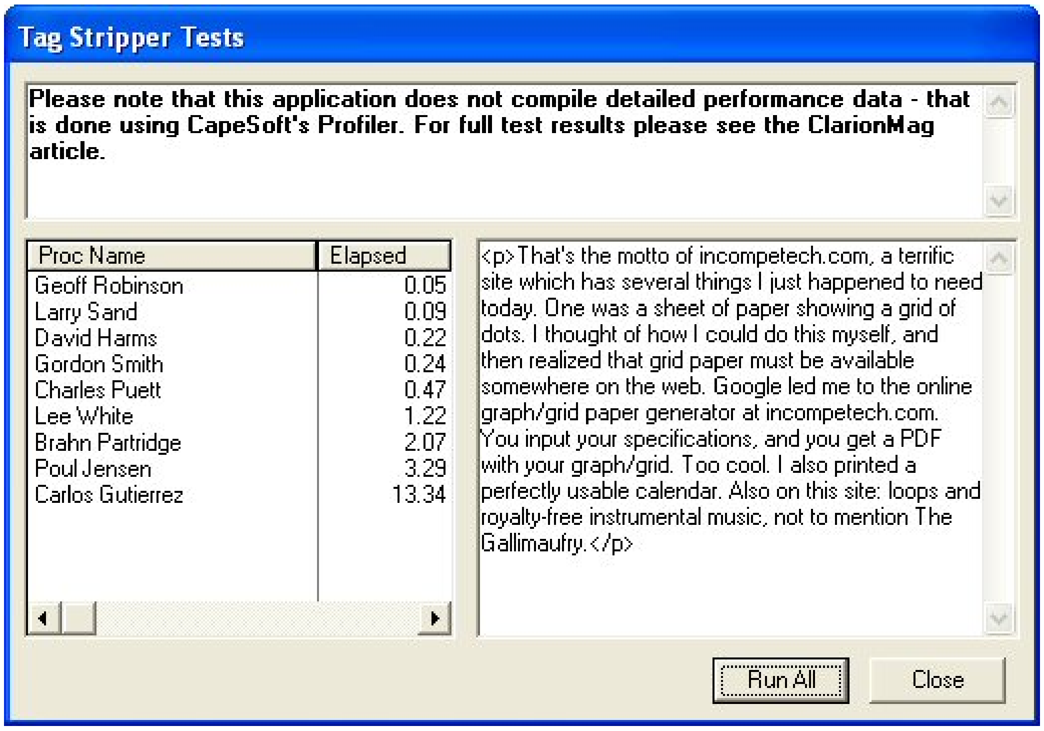
Published 2006-03-23

I received eight entries for the Clarion Challenge I posted on March 2. The task: remove all <a> (Anchor) tags from a block of HTML text, leaving behind everything else (including the text between the<a> and </a> tags).

The results: the fastest code was written by Geoff Robinson, with Larry Sand coming in second. Congratulations to Geoff and Larry! Both win a prize, and I'll explain why later.

But first, I learned a very important lesson, one I'd forgotten in the many months (years!) since the last Clarion Challenge, and that is to be as explicit as possible about the project requirements! Not only were there numerous questions about what it was I wanted the code to do, but I neglected to specify a test framework for the results. I received APPs, PRJs, and even text files. So it took me a while to get everything massaged into a test application.

I ran some very basic tests, with 5000 iterations of each call. The test machine is an Athlon 64 3200+, and the results are shown in Figure 1. Note that I included my own code (unofficially), which is what suggested the challenge to me in the first place, and I'm happy to see that I got my clock cleaned by the winners.



Please note that this application does not compile detailed performance data - that is done using CapeSoft's Profiler. For full test results please see the ClarionMag article.

Proc Name	Elapsed
Geoff Robinson	0.05
Larry Sand	0.09
David Harms	0.22
Gordon Smith	0.24
Charles Puett	0.47
Lee White	1.22
Brahn Partridge	2.07
Poul Jensen	3.29
Carlos Gutierrez	13.34

```
<p>That's the motto of incompetech.com, a terrific
site which has several things I just happened to need
today. One was a sheet of paper showing a grid of
dots. I thought of how I could do this myself, and
then realized that grid paper must be available
somewhere on the web. Google led me to the online
graph/grid paper generator at incompetech.com.
You input your specifications, and you get a PDF
with your graph/grid. Too cool. I also printed a
perfectly usable calendar. Also on this site: loops and
royalty-free instrumental music, not to mention The
Gallimaufry.</p>
```

Figure 1. Test results using clock()

After the rudimentary test done via code built into the application, I ran a set of tests under CapeSoft's Profiler, as shown in Figure 2. These confirmed the earlier results.

To get the timings in Figure 2 I only loaded the `Main()` procedure in Profiler - at this point I didn't want timing information on the content of each procedure call, just a comparison between procedure calls. Figure 2 shows the results (with some numbers abbreviated - Profiler shows figures with great precision, but all that data made the original table too wide).

Total Time	**Total Cycles**	**Longest Cycles**	**Shortest Cycles**	**Average Cycles**	**Source**
20.8	209056	2406.96	185	209	Geoff Robinson
37.2	374370	2546.69	367	374	Larry Sand
80.3	807825	11507.17	720	807	David Harms (unofficial)
92.0	925519	8491.31	886	925	Gordon Smith
145.9	1466973	7598.38	1,395	1466	Charles Puett
486.1	4885323	31867.91	4,724	4885	Lee White
826.4	8304779	34606.57	7,834	8304	Brahn Partridge
1329.9	13364747	44859.74	12,301	13364	Poul Jensen
5389.4	54159589	106861.07	51,635	54159	Carlos Gutierrez

Figure 2. Test results from Profiler

Both Profiler and the simple `clock()` test yielded the same overall test results. But `clock()` is a blunt tool, really only useful here after a high number of procedure calls, because its maximum resolution is a hundredth of a second. In a 32 bit application, you can use the `QueryPerformanceCounter` API call to get much finer resolution, and you can use Clarion's profiler hooks to get the time spent in any given procedure or routine, but I don't know any better way to evaluate Clarion code performance than CapeSoft Profiler.

To profile individual entries, I simply selected that procedure, and that procedure alone, from the list of procedures to profile. I then used the source view to examine individual line execution times, as the figures below demonstrate. I also reduced the number of iterations from 1000 to 50, as the extra profiling activity added a lot of overhead.

Here are the results in reverse order, from slowest to fastest.

Carlos Gutierrez

Carlos wrote one of the most compact and entries, and it's one I like a lot for its simplicity. Carlos looks for the string `<a` or `</a`, and, when found, sets the `InTag` flag to 1 until the closing > is found. Characters are only copied to the output string when a tag is not found. Carlos also does all the necessary array index checking to avoid running past the end of the input string.

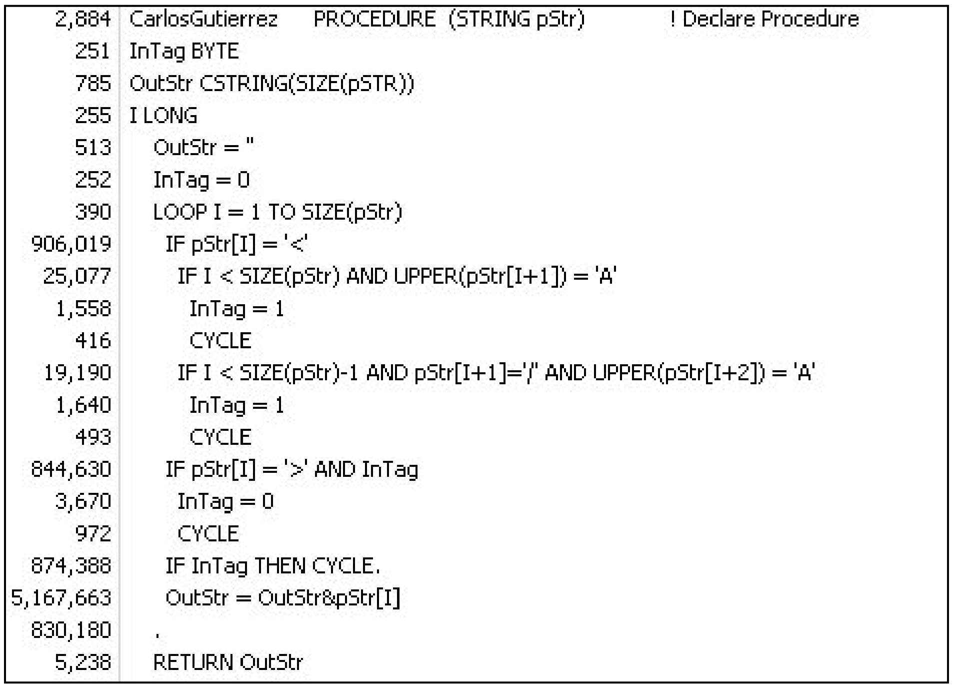
```
CarlosGutierrez      PROCEDURE  (STRING inStr)             ! Declare Procedure
InTag BYTE
OutStr CSTRING(SIZE(inSTR))
I LONG
pstr cstring(size(inStr)+1)
```

```
    CODE
      OutStr = ''
      InTag = 0
      LOOP I = 1 TO SIZE(pStr)
        IF pStr[I] = '<'
          IF I < SIZE(pStr) AND UPPER(pStr[I+1]) = 'A'
            InTag = 1
            CYCLE
          .
          IF I < SIZE(pStr)-1 AND pStr[I+1]='/' AND UPPER(pStr[I+2]) = 'A'
            InTag = 1
            CYCLE
          .
        .
        IF pStr[I] = '>' AND InTag
          InTag = 0
          CYCLE
        .
        IF InTag THEN CYCLE.
        OutStr = OutStr&pStr[I]
      .
      RETURN OutStr
```

There's only one problem with this code, and it's a total performance killer:

```
OutStr = OutStr & pStr[I]
```

Individual characters are appended to the output string using the & operator. This is very expensive, and results in very long execution times. Figure 3 shows Profiler's analysis, but I've abridged the screen shot to only show total execution time per line. You can see the detailed analysis here.



```
    2,884  CarlosGutierrez     PROCEDURE  (STRING pStr)          ! Declare Procedure
      251  InTag BYTE
      785  OutStr CSTRING(SIZE(pSTR))
      255  I LONG
      513    OutStr = ''
      252    InTag = 0
      390    LOOP I = 1 TO SIZE(pStr)
  906,019      IF pStr[I] = '<'
   25,077        IF I < SIZE(pStr) AND UPPER(pStr[I+1]) = 'A'
    1,558          InTag = 1
      416          CYCLE
   19,190        IF I < SIZE(pStr)-1 AND pStr[I+1]='/' AND UPPER(pStr[I+2]) = 'A'
    1,640          InTag = 1
      493          CYCLE
  844,630      IF pStr[I] = '>' AND InTag
    3,670        InTag = 0
      972        CYCLE
  874,388      IF InTag THEN CYCLE.
5,167,663      OutStr = OutStr&pStr[I]
  830,180    .
    5,238    RETURN OutStr
```

Figure 3. Cumulative execution times per line - CarlosGutierrez

Replacing the & with string slicing would move this procedure into somewhere around third or fourth place.

Editor's note: After publication, Carlos sent me two updated versions of his source, with string slicing. As expected, the improvement was dramatic. In testing, Gordon Smith's time was .23; Carlos' StripTag procedure turned in a time of .24, and StripATag came in at .25. Here's the source for those two procedures:

```
StripTag PROCEDURE(pStr)!,STRING
InTag BYTE
OutStr CSTRING(SIZE(pSTR))
I LONG
Pos LONG
    CODE
    OutStr = ''
    InTag = 0
    Pos = 1
    LOOP I = 1 TO SIZE(pStr)
      IF pStr[I] = '<'
        InTag = 1
        CYCLE
      .
      IF pStr[I] = '>'
        InTag = 0
        CYCLE
      .
      IF InTag THEN CYCLE.
      OutStr[Pos] = pStr[I]
      Pos += 1
    .
    OutStr[Pos] = '<0>'
    RETURN OutStr

StripATag PROCEDURE(pStr)!,STRING
InTag BYTE
OutStr CSTRING(SIZE(pSTR))
I LONG
Pos LONG
    CODE
    OutStr = ''
    InTag = 0
    Pos = 1
    LOOP I = 1 TO SIZE(pStr)
      IF pStr[I] = '<'
        IF I < SIZE(pStr) AND UPPER(pStr[I+1]) = 'A'
          InTag = 1
          CYCLE
        .
        IF I < SIZE(pStr)-1 AND pStr[I+1]='/' AND UPPER(pStr[I+2]) = 'A'
          InTag = 1
          CYCLE
        .
      .
      IF pStr[I] = '>' AND InTag
        InTag = 0
        CYCLE
      .
      IF InTag THEN CYCLE.
      OutStr[Pos] = pStr[I]
      Pos += 1
    .
    OutStr[Pos] = '<0>'
```

```
             RETURN OutStr
```

Poul Jensen

Poul wrote one of the longer entries, anticipating a number of potential problems I hadn't considered, such as an output string too short for the input value. Poul set up a `ScanMode ITEMIZE` structure to enumerate the possible states encountered while stepping through the input string:

```
 ITEMIZE(1)
Scan_Normal     EQUATE
Scan_Prefix     EQUATE
Scan_Strip      EQUATE
Scan_End        EQUATE
Scan_Fill       EQUATE
Scan_Close      EQUATE
!Scan_Finish    EQUATE
!Scan_Error     EQUATE
 END
ScanMode SHORT
```

Poul's code tests for the current scan mode, then uses the following `EXECUTE` statements to update the indexes and copy string data:

```
EXECUTE ScanMode   ! Update Target and Advance Position ...
  _TargetString[TargetPos : TargetPos ] =   _SourceString[Pos : Pos]   ! Keep 1
  Pos  =  Pos + 1    ! Ignore 1 before advancing  (2 bytes "<a")
  Pos  =  Pos + 0    ! Ignore 0 before advancing  (Ignoring/Stripping bytes)
  Pos  =  Pos + 0    ! Ignore 0 Before advancing            (remove the ">" byte)
  _TargetString[TargetPos : TargetPos ] =   _SourceString[Pos : Pos]   ! Keep 1
  Pos  =  Pos + 3    ! Ignore 3 before advancing            (remove the "</a>")
END!Execcute

POS       += 1       ! Advance 1 byte

EXECUTE ScanMode          ! Set Scanmode/Target position ...
  TargetPos += 1          !Still Normal
  ScanMode =  Scan_Strip  !prefix So Now Stripping
  ScanMode =  Scan_Strip  !Still Strippingl
  ScanMode =  Scan_Fill   !END so Now Filling
  TargetPos += 1          !Still Filling
  ScanMode =  Scan_Normal !Close so Normal
END!Execute
```

As Poul's code is quite lengthy I won't reproduce it all here. There were no single areas that looked eligible for improvement, other than my general observations, which you can read at the conclusion of this article.

Brahn Partridge

Brahn was the only contestant to write his code in the form of a class, called `HTMLCleanerClass`. The procedure code looks like this:

```
BrahnPartridge       PROCEDURE  (*cstring sampleText)
HTMLCleaner  HTMLCleanerClass
```

```
  CODE
  HTMLCleaner.StripTag(sampleText, HTML_TAG:Anchor)
  return
```

The prototype is in HTMLCleanerClass.inc:

```
_HTMLCleanerClass_ EQUATE(1)
! Generated by CapeSoft's Object Writer
HTML_TAG:ANCHOR EQUATE('a')

HTMLCleanerClass
Class(),Type,Module('HTMLCleanerClass.Clw'),LINK('HTMLCleanerClass.Clw',1)
StripTag               PROCEDURE (*CSTRING pCS, STRING pTag) ,LONG,PROC ,VIRTUAL
Replace                PROCEDURE (STRING pFind, STRING  pReplace, *CSTRING pInto)
,LONG,PROC ,VIRTUAL
                     END
```

Brahn is a big fan of Object Writer, one of CapeSoft's free utilities, which he used to create this class. Note the string equate used to specify `HTML_TAG:Anchor`.

The `StripTag` and `ReplaceTag` methods are as follows:

```
HTMLCleanerClass.StripTag PROCEDURE  (*CSTRING pCS, STRING pTag)
openingTagStart    LONG
openingTagEnd      LONG
count              LONG

  CODE
  ! Make everything lower case
  pTag = Lower(pTag)
  LOOP
    openingTagStart = InString('<60>' & pTag, Lower(pCS), 1, 1)
    IF openingTagStart = 0
      ! No more tags
      BREAK
    END
    openingTagEnd  = InString('>', Lower(pCS), 1, openingTagStart)
    IF openingTagEnd = 0
      ! This should never happen in properly formed HTML but just in case
      openingTagEnd = Len(pCS)
    END
    ! Replace all string matching the opening TAG with ''
    ! If we are lucky, we may even get more than one hit with this...
    !
    ! In fact, in the sample text there are 9 such luck hits
    count += SELF.Replace(pCS[openingTagStart : openingTagEnd], '', pCS)
  END

  ! Remove all the closing tags
  SELF.Replace('<60>/' & pTag & '>', '', pCS)

  RETURN count
HTMLCleanerClass.Replace PROCEDURE  (STRING pFind, |
                             STRING  pReplace, *CSTRING pInto)
locate             LONG,AUTO
lastlocate         LONG
lenFind            LONG,AUTO
```

```
lenReplace          LONG,AUTO
count               LONG
  CODE
  ! Pre-Compute some static values
  pFind      = Lower(pFind)
  lenFind    = Len(pFind)
  lenReplace = Len(pReplace)
  LOOP
    locate = InString(pFind, Lower(pInto), 1, lastlocate+1)
    IF ~locate
      RETURN count
    END
    count += 1
        ! so we dont end up having recursive replacement !
    lastLocate = locate + lenReplace-1
    pInto = Sub(pInto, 1, locate-1) & pReplace & |
          Sub(pInto, locate+lenFind, Len(pInto))
  END
```

The use of `INSTRING` and `SUB` appear to be the biggest performance bottlenecks in this code, as you can see by looking at the profile. One big advantage of this code is that it appears to be fully ready for multi-character tags such as `<div>`, `<table>` etc. I didn't test that capability, however.

Lee White

When I was first learning HTML, Lee was the guy I went to when I had questions. So it's no surprise that Lee's code looks for error conditions others might have missed! Here's his code:

```
LeeWhite             PROCEDURE  (*cstring tempdata)        ! Declare Procedure
!TempData            &CSTRING
hLen                LONG,AUTO
CharNum             LONG
CharNum2            LONG
Chars3              STRING(3)
  CODE
  hLen = LEN(TempData)
  LOOP CharNum = 1 TO hLen
    Chars3 = TempData[ CharNum : CharNum + 3 ]
    ! spaces are not allowed between leading characters so
        !  don't bother looking
    IF Chars3 = '<<a ' OR Chars3 = '<<A '
      LOOP CharNum2 = CharNum + 3 TO hLen
      ! ">" is not an allowable characters within an anchor
          !   so next ">" is end
        IF TempData[CharNum2] = '>'
          TempData = TempData[ 1 : CharNum-1 ] & TempData[ CharNum2+1 : hLen ]
          hLen    -= CharNum2-CharNum
          BREAK
        END
      END
    END
  END
  ! 2 loops remove worry about mismatched tags
  LOOP CharNum = 1 TO hLen
    Chars3 = TempData[ CharNum : CharNum + 3 ]
```

```
          ! spaces are not allowed between leading characters
          !   so don't bother looking
        IF Chars3 = '<</a' OR Chars3 = '<</A'
          LOOP CharNum2 = CharNum + 3 TO hLen
            ! a space CAN be found between "a" and ">"
            !   so account for the possibility
            IF TempData[CharNum2] = '>'
              TempData = TempData[ 1 : CharNum-1 ] & TempData[ CharNum2+1 : hLen ]
              hLen -= CharNum2-CharNum
              BREAK
            END
          END
        END
      END
      return
```

The second loop is basically a repeat of the first, except that it's looking for the closing tag. This two-loop approach means that the string is traversed twice, however, which significantly increases execution time. On the other hand, this code makes no assumption that `<a>` is followed by `</a>`, and will do less damage to badly-formatted HTML. See the complete profile.

Charles Puett

Charles' entry was the only one to use a recursive approach. His code is as follows:

```
    CharlesPuett         PROCEDURE  (String Astring)           ! Declare Procedure
    ATagText     STRING(LEN(Astring))
    StartPos     USHORT(0)
    ATagLen      USHORT(0)
    EndPos       USHORT(0)
    NewLen       ULONG
      CODE
      NewLen = Len(CLIP(Astring))
      !!! Find the beginning anchor (the <a )
      StartPos = INSTRING('<a ', LOWER(Astring),1)
      IF StartPos Then
         !!! If the anchor is found, determine where the end of it is at (the ">")
         ATagLen = INSTRING('>',LOWER(Astring),1,StartPos)
         IF ATagLen THEN
         !!! If the end is found then find the link delimiter (the </a>)
            EndPos   = INSTRING('</a>',LOWER(Astring),1,StartPos)
         END
      End
      IF EndPos Then
         !!!  Pull out the text between the ending ">" and the </a>"
         ATagText = Astring[ATagLen + 1 : EndPos - 1]
         !!!  Return the passed string from its beginning up to the beginning anchor
         !!!  plus the extracted text and recurse through the remainder of the passed
    string
         !!!  until no more complete delimiter sets are found.
         Return Astring[1 : StartPos - 1] & CLIP(ATagText) & CharlesPuett(Astring[EndPos
    + 4 : NewLen])
      End
      Return Astring
```

Each call to the function returns the passed string up to the first `<a>` tag, plus the code inside the tag, and then the function calls itself for the remaining portion of the string. Very few lines of code, and decent performance. As you're probably coming to expect, most of the performance hit is in the calls to `INSTRING`. See the complete profile.

Gordon Smith

Gordon used an approach I wondered if any would consider: instead of making a copy of the string, just rewrite the original string. After all, you're removing text, so you know that the new string will be shorter. And this is just what Gordon does, moving the characters one at a time as needed, and setting the string's last character to a `<0>` value. The function also returns the number of characters removed. Here's the code:

```
GordonSmith          PROCEDURE  (*cstring html)            ! Declare Procedure
ACharMap_           string('<0>{64}<1><0>{31}<1><0>{158}'), static
ACharMap            byte, dim(255), over(ACharMap_)
AlphaCharMap_       string('<0>{64}<1>{26}<0>{6}<1>{26}<0>{133}'), static
AlphaCharMap        byte, dim(255), over(AlphaCharMap_)
NotAlphaCharMap_    string('<1>{64}<0>{26}<1>{6}<0>{26}<1>{133}'), static
NotAlphaCharMap     byte, dim(255), over(NotAlphaCharMap_)

readPos unsigned(1)
writePos unsigned(1)
inATag bool(false)
  CODE
  loop while html[readPos] ~= '<0>'
    if not inATag
      if html[readPos] = '<'
        if (ACharMap[val(html[readPos + 1])] and |
                    NotAlphaCharMap[val(html[readPos + 2])])
          inATag = true
          readPos += 1 !  Minor optimization
        elsif (html[readPos + 1] = '/' and |
                    (ACharMap[val(html[readPos + 2])] and |
                         NotAlphaCharMap[val(html[readPos + 3])]))
          inATag = true
          readPos += 2 !  Minor optimization
        end
      end
    end

    if not inATag
      ! Most of the time it will be quicker to just
          ! copy the char rather than to test if we need
          ! to copy it...
      html[writePos] = html[readPos]
      writePos += 1
    elsif html[readPos] = '>'
      inATag = false
    end
    readPos += 1
  end
  html[writePos] = '<0>'
  return readPos - writePos
```

Had I simply been reading the source, I would've expected this code to come out the winner, based on its total reliance on string slicing and character testing via arrays. Performance was definitely good, though not the best, and the most expensive operations were copying the single characters within the string and incrementing counters. See the complete profile.

Larry Sand

Larry turned in the second fastest time by redeclaring the string as a byte array, and using numeric equivalents in place of string testing. Here's Larry's code:

```
LarrySand            PROCEDURE  (*String htmlToStrip)      ! Declare Procedure
b Byte,Dim(Size(htmlToStrip)),Over(htmlToStrip)

readPos              Long,Auto
writePos             Long,Auto
OpeningTagPos        Long,Auto

charsAsLong          Long,Auto
charsAsBytes         Byte,Dim(4),Over(charsAsLong)
charsAsWords         UShort,Dim(2),Over(charsAsLong)

readAheadPos         Long,Auto
j                    Long,Auto

SPACE_CHAR           Equate(20h)
LT_CHAR              Equate(3Ch)   ! '<'
GT_CHAR              Equate(3Eh)   ! '>'
BEGIN_LOWER_ATAG     Equate(3C61h) ! '<a'
BEGIN_UPPER_ATAG     Equate(3C41h) ! '<A'
CLOSE_LOWER_ATAG     Equate(3C2F613Eh) ! '</a>'
CLOSE_UPPER_ATAG     Equate(3C2F413Eh) ! '</A>'

  CODE
  writePos = 1

  Loop readPos = 1 to Size(htmlToStrip)

    If b[readPos] = LT_CHAR
      OpeningTagPos = readPos

      ! read the next four characters starting with the '<' into
      ! a long to allow testing for A tags using integer math
      charsAsLong = 0
      j = 4
      Loop readAheadPos = readPos to Size(htmlToStrip)
        charsAsBytes[j] = b[readAheadPos]
        j -=1
      Until j < 1

      If charsAsWords[2] = BEGIN_LOWER_ATAG Or charsAsWords[2]= BEGIN_UPPER_ATAG
        ! skip to the end of opening A tag
        Loop readPos = readAheadPos-1 to Size(htmlToStrip)
          If b[readPos] = LT_CHAR   !something's wrong with the A tag
            readPos = OpeningTagPos
            Break
          End
        Until b[readPos] = GT_CHAR

        If b[readPos] = GT_CHAR
          Cycle
        End
      ElsIf charsAsLong = CLOSE_LOWER_ATAG Or charsAsLong = CLOSE_UPPER_ATAG
        readPos = readAheadPos
        Cycle

      Else !it was not an A tag
        readPos = OpeningTagPos
      End
    End !Processing an opening tag character

    b[writePos] = b[readPos]
    writePos += 1
  End

  ! fill the remainder of the string with space characters
  Loop writePos = writePos to Size(htmlToStrip)
    b[writePos] = SPACE_CHAR
  End

  Return
```

Like Gordon, Larry works on a single string, but because it's a `STRING` not a `CSTRING` he has to pad it with spaces at the end.

I don't think there's a whole lot more you can squeeze out of pure Clarion code. See the complete profile.

And the winner is... Geoff Robinson!

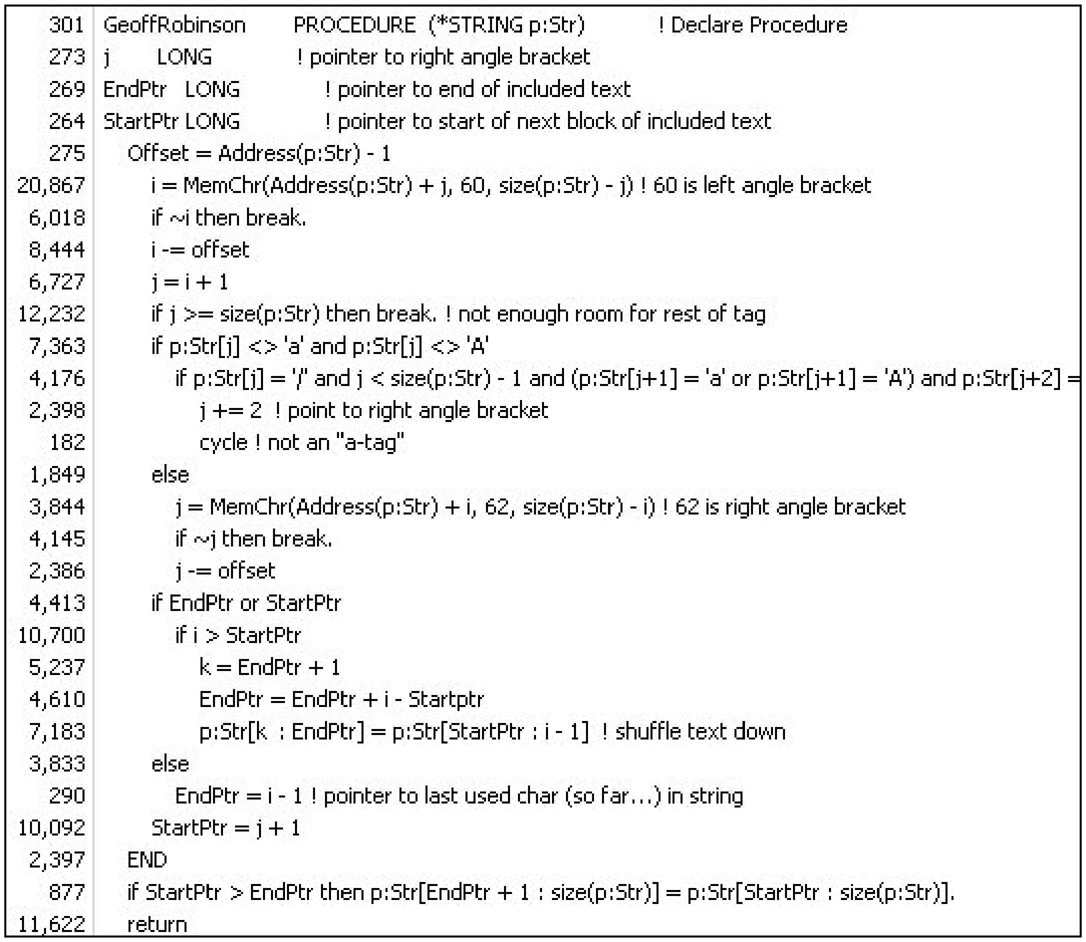
Geoff turned in the fastest time of all. His secret weapon? The memchr Windows API call, which offers very fast searching of a character buffer. Here's the code:

```
GeoffRobinson        PROCEDURE  (*STRING p:Str)
!  - uses memchr which should be faster than
!    instring, but requires:
!      map
!        MODULE('Standard C Library')
!          MemChr(ULONG,LONG,UNSIGNED),LONG,NAME('_memchr')
!        END
!      end
!
!--------------------------------------------------------------------------------------
------
i        LONG,AUTO,STATIC  ! pointer to left  angle bracket
j        LONG              ! pointer to right angle bracket
k        LONG,AUTO,STATIC  ! temp holding pointer
EndPtr   LONG              ! pointer to end of included text
StartPtr LONG              ! pointer to start of next block of
                           !  included text
Offset   LONG,AUTO,STATIC  ! used to convert absolute address
                           !  to clarion character pointer

  CODE
    Offset = Address(p:Str) - 1
    LOOP
            ! 60 is left angle bracket
        i = MemChr(Address(p:Str) + j, 60, size(p:Str) - j)
        if ~i then break.
```

```
            i -= offset
            j = i + 1
            if j >= size(p:Str) then break. ! not enough room for rest of tag
            if p:Str[j] <> 'a' and p:Str[j] <> 'A'
                if p:Str[j] = '/' and j < size(p:Str) - 1 and |
                              (p:Str[j+1] = 'a' or p:Str[j+1] = 'A') and p:Str[j+2] = '>'
                    j += 2  ! point to right angle bracket
                else
                    cycle ! not an "a-tag"
                end
            else
                    ! 62 is right angle bracket
                j = MemChr(Address(p:Str) + i, 62, size(p:Str) - i)
                if ~j then break.
                j -= offset
            end

            if EndPtr or StartPtr
                if i > StartPtr
                    k = EndPtr + 1
                    EndPtr = EndPtr + i - Startptr
                                    ! shuffle text down
                    p:Str[k  : EndPtr] = p:Str[StartPtr : i - 1]
                end
            else
                EndPtr = i - 1 ! pointer to last used char in string
            end
            StartPtr = j + 1
        END
        ! no more tags so shuffle down any text on the end
        if StartPtr > EndPtr then p:Str[EndPtr + 1 : size(p:Str)] = |
              p:Str[StartPtr : size(p:Str)].
        return
```

Once again, Geoff works directly on the passed string.

Figure 4 shows Profiler's analysis, but again only showing total clock cycles per line (Profiler also tracks longest, shortest, and average clock cycles).



```
   301  GeoffRobinson       PROCEDURE  (*STRING p:Str)          ! Declare Procedure
   273  j       LONG            ! pointer to right angle bracket
   269  EndPtr   LONG              ! pointer to end of included text
   264  StartPtr LONG              ! pointer to start of next block of included text
   275    Offset = Address(p:Str) - 1
20,867      i = MemChr(Address(p:Str) + j, 60, size(p:Str) - j) ! 60 is left angle bracket
 6,018      if ~i then break.
 8,444      i -= offset
 6,727      j = i + 1
12,232      if j >= size(p:Str) then break. ! not enough room for rest of tag
 7,363      if p:Str[j] <> 'a' and p:Str[j] <> 'A'
 4,176        if p:Str[j] = '/' and j < size(p:Str) - 1 and (p:Str[j+1] = 'a' or p:Str[j+1] = 'A') and p:Str[j+2] =
 2,398          j += 2  ! point to right angle bracket
   182          cycle ! not an "a-tag"
 1,849      else
 3,844        j = MemChr(Address(p:Str) + i, 62, size(p:Str) - i) ! 62 is right angle bracket
 4,145        if ~j then break.
 2,386        j -= offset
 4,413      if EndPtr or StartPtr
10,700        if i > StartPtr
 5,237          k = EndPtr + 1
 4,610          EndPtr = EndPtr + i - Startptr
 7,183          p:Str[k  : EndPtr] = p:Str[StartPtr : i - 1]  ! shuffle text down
 3,833      else
   290        EndPtr = i - 1 ! pointer to last used char (so far...) in string
10,092      StartPtr = j + 1
 2,397    END
   877    if StartPtr > EndPtr then p:Str[EndPtr + 1 : size(p:Str)] = p:Str[StartPtr : size(p:Str)].
11,622    return
```

Figure 4. Cumulative execution times per line - GeoffRobinson

For the complete profile of Geoff's code click here. Geoff wins a ClarionMag/Planet Clarion coffee mug of his choice.

I hadn't specifically said that entries had to be pure Clarion code, and using the Windows API is definitely a legitimate technique for speeding up Clarion apps, so Geoff's first place is fully deserved. But I'm also awarding a mug to Larry for the best "pure Clarion" entry.

Conclusions

After I had all the results in profiler, I did a bit of playing with the statistcal display, looking at which lines of code took the longest to execute once, and which took the most time, in total. As you'd expect, this revealed that string handling can be expensive if not done carefully.

`INSTRING`, `LEN`, and `SUB` are all quite slow functions, and `INSTRING` got used a lot in some of the entries. A single `INSTRING` call typically took from 75 to 200 clock cycles.

Moving string data round also takes time, a point made by Jim Gambon in his excellent two-parter on string handling. It's no accident that the fastest functions all operated directly on the original string, rather than make a copy.

If you've followed the links above, you'll already have seen the complete profiles of all the entries. It makes for interesting reading.

The Clarion.NET version

No, I don't have Clarion.NET in hand, but I think it's safe to say it will have standard .NET string handling, and full access to all of .NET. And that means that the solution to this problem, in Clarion.NET, will probably look like this:

```
    newString = System.Text.RegularExpressions.|
      Regex.Replace(origString,'</?(?i:a)(.|\n)*?>',     '')
```

That's right - one line of code, using regular expressions (RegEx). I found an example of this code in a comment by Jorge on this page. Clarion has some basic RegEx capability in the `MATCH` function, but nothing like this, as far as I know. I haven't checked performance, so this .NET code might not measure up to the entries above. But it's probably good enough for just about anything you want to do, and it does illustrate rather nicely the power and convenience awaiting Clarion developers in .NET.

Incidentally, you can easily expand the above code to remove multiple tags at once. Here's a version that removes `anchor`, `image`, `div`, and `script` tags (note the | separator character):

```
    newString = System.Text.RegularExpressions.Regex.|
      Replace(origString,'</?(?i:a|img|div|script)(.|\n)*?>',     '')
```

If you want to play with regular expressions in Clarion, be sure to read Carl Barnes' article on the subject. Among other things Carl notes the new `STRPOS` function, which handles regular expressions. None of the entries used it, and it might make for some flexible, easily maintainable code.

Download the results app

View the complete test results

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995), and has written or co-written several Java books. David produces the Planet Clarion podcast, which he co-hosts with Andrew Guidroz II.

Reader Comments

Add a comment

- » Carlos sent me an updated version using string slicing,...
- » When Adriaan Ieperen won the previous challenge he...
- » Not that it would change the results (removing the check...
- » Gordon, I meant to mention the string copying, but as...
- » My first thought was...if INSTRING() is such a dog, has...
- » David, Not that I know of. Would be nice to see...
- » Gordon, >> ...but most peoples code would fail on the...
- » Posted by: David Jung << My first thought was...if...

Clarion Magazine

Print Directly to Printer Made Easier

by Steven Parker

Published 2006-03-20

In my last adventure in API land, I explored sending text and control codes directly to a printer port.

Writing directly to the printer involves three API calls that must be prototyped in the Global Map:

```
    MODULE('win32.lib')
        CreateFileA(*CSTRING lpszName,      |
            ulong fdwAccess,                |
            ulong fdwShareMode,              |
            ulong SecurityDescriptor,       |
            ulong fewCreate,                |
            ulong fewAttrsAndFlags,         |
            ulong hTemplateFile),long,raw, pascal
        WriteFile(long hFile,
            *cstring buffer,                            |
            long nNumberOfButesToWrite,     |
            *long lpNumberOfBytesWritten,   |
            ulong lpOverlapped), short, raw, pascal
        CloseHandle( ulong hObject ), short, raw, pascal
```

Direct printing also requirs six variables:

```
    PortID              Long
    StringToPrint       String(40)
    PrintLen            LONG
    Written              LONG
    PortName             CSTRING(20)
    x                    LONG  ! receives various return values
```

And, finally, each and every time I want to write something directly to a port, I need to call five lines of code:

```
    PortName = 'LPT1:'
    PortID = CreateFileA( PortName, 040000000H, 0, 0, 3, 0, 0 )
    PrintLen = Len(Clip(StringToPrint))
    x = WriteFile(PortID,StringToPrint,PrintLen,Written,0)
    x = CloseHandle(PortID)
```

Repeating all the data and all the code each time I want to write directly is … not good form.

What I really want is a simple, single call whenever I want to print directly to the printer. What I want is something like:

```
    PrintLine( StringToPrint , PortToPrintToo )
```

Why pass the port each time? Yes, I could prime the `PortName` variable globally. Then I would not have to pass it to my new procedure. But that would mean also that I am restricted to one printer when running an application. While it is true that many, if not most applications run against a single printer, this is an artificial restriction that could negatively impact me. Easily.

Passing the port each time, however, also constrains me: I probably have to store the port or ports globally (in a data file, an INI file, a queue, memory file or the like).

Or I could use the `PROPPRINT:Port` property of `PRINTER{PROPPRINT:Device}`. Since I usually have to reset `PRINTER{PROPPRINT:Device}` to use another printer, `PROPPRINT:Port` should give the value needed for `PortName`. I leave this as an exercise for the reader (just remember to include `prnprop.clw` in the Global includes).

> **Digression**: Did you notice that I said that I "usually" reset `PRINTER{PROPPRINT:Device}`? If using a Clarion report, this variable will, indeed, need to be changed to print reports to different printers (to "redirect" output). In this case, `PROPRINT:Port` *is* sufficient to get the port name. However, the `WriteFile` API is completely unaware of the

> default printer or any other installed printer for that matter (it doesn’t know or care about what is installed in Windows). `WriteFile` can output to any printer whether or not Windows knows about it. Receipt printers, for example, are often not "installed" and, hence, are unknown to Windows. This does not stop `WriteFile` from printing to them. This is why getting the necessary value for `PortName` is a design decision.

In any case, the trade off seems to me to be worth it.

So, instead of the code shown, above, I want something like:

```
PortName = 'LPT1:'
PrintLine(StringToPrint,PortName)
```

where, in a real world application, `PortName` is primed programmatically.

Creating `PrintLine()` turns out to be quite easy. Create a Source procedure. Move all of the data required into it. But simply copying in the code will not work.

The port name and string to be written must be CStrings for the API calls. To avoid forcing myself to declare CStrings all over the place, the first thing `PrintLine()` must do is convert the incoming Strings to CStrings (if the incoming string is already a CString, no harm done):

```
StringToPrint = pLineToPrint
PortName = pPort
```

After that, the remaining code does the printing:

```
PortID = CreateFileA( PortName, 040000000H, 0, 0, 3, 0, 0 )
PrintLen = Len(Clip(StringToPrint))
x = WriteFile(PortID,StringToPrint,PrintLen,Written,0)
x = CloseHandle(PortID)
```

The sample app shows this and is available for download at the end of this article.

The obvious question is: Why not make a template out of this? The answer is that this has already been done and I’m not a big fan of wheel -reinventing. In fact, the templates written by others constitute what I called the "easy" way of writing directly to a printer. I intend to look at these easy ways next ….

Download the source

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Reader Comments

Add a comment

Clarion Magazine

The ClarionMag Blog

Get automatic notification of new items! RSS feeds are available for:

XML All blog entries
XML All new items, including blogs

Blog Categories

- »All Blog Entries
- »Clarion 7 Clarion.NET
- »Future Articles
- »News flashes
- »Nifty Stuff

Getting caught up

Direct link

Posted Tuesday, March 28, 2006 by Dave Harms

As you may have noticed, the February PDF went up on March 17, and included articles published through March 16. I have since backdated the February PDF to Feb 28.

This is going to be another busy week as we get our publishing schedule back on track after February's unavoidable interruption. In addition to the two articles already published, I have several weeks worth of articles going into final review tomorrow, making up the remainder of March's publication schedule. With a little luck everything will be up by Friday, and April should see ClarionMag back on its normal schedule (although the office will be closed for a week or so over the Easter holiday).

NetTalk 4 web server on a pen drive? (updated)

Direct link

Posted Monday, March 20, 2006 by Dave Harms

On Friday, in the comp.lang.clarion newsgroup at discuss.softvelocity.com, Ben Brady pointed out this nifty web server on a USB plug. Just plug it in to your USB port and presto - instant server. Ben wrote: "It would be so cool to be able to run NetTalk 4 webserver like this..."

The problem with putting a web server, or any other application, on a USB pen drive, is that Windows won't automatically execute any autorun.inf file it finds, which is the usual way to automatically run stuff on an added disc. But it appears that Ben's idea might be pretty easy to do anyway. A company called USB Software has a little utility called Autorun USB: "With Autorun USB you can autorun any application or document when you insert your USB Pen Drive." Autorun USB can also run automatically when Windows starts.

If you try this, let me know how it goes.

Update!

Glenn Rountree pointed me to another feeware utility, BusRunner by KMG Software, that lets you do an autorun from a USB flash drive. Glenn uses this tool to synch data to the archive folder on his desktop.

Clarion Challenge results on the way...

Direct link

Posted Friday, March 17, 2006 by Dave Harms

In answer to one contestant's question, yes, I am writing up the results of the Clarion Challenge. CapeSoft's Profiler is proving an excellent analysis tool, and I should have the results ready sometime next week.

The SV blog lives!

Direct link

Posted Friday, March 10, 2006 by Dave Harms

The SV blog lives! Bob Foreman has posted an item about further server-side autoinc improvements in 6.3 9051, soon to be released. Bob also hints at what else has been going on behind the company's mask of silence:

> We have got so much going on at the home office. It is exciting times for us all, and I hope that it will be the same for you soon.

You're not alone, Bob, you're not alone.

Origami unveiled

Direct link

Posted Thursday, March 09, 2006 by Dave Harms

So what's the deal with Origami anyway? Robert Scoble interviews the man behind the Origami class devices, Otto Berkes (who is also one of the original four Xbox team members).

Origami is meant to fill the large gap between laptops and other mobile devices. It's a touchscreen handheld, with a full Windows OS. You can run any Windows application on this device. Which means that you can develop Origami apps with Clarion, of course.

Check out the Haiku concept model at about 6:30 in the video. Now an Origami in that form factor would rock, but it's not practical just yet.

There are some hardware limits imposed by the size - this is a 1Ghz class device, with good (but not bleeding edge) graphics. This first device is running XP Tablet Edition, and has an integrated stylus with handwriting recognition. Two USB ports. Some models will be dockable. You can hook up an external monitor and keyboard. Native resolution is 800x480, 7" display, same as many portable DVD players. But higher resolutions can be displayed, with some image quality degradation.

Berkes did some hedging on the battery life - evidently the power consumption is still pretty high, and you probably can't expect much more than 2-3 hours for the initial product.

Features include WiFi, Bluetooth, network connectors, stereo speakers and a noise reduction microphone.You could plug in a GPS receiver, use a Bluetooth-enabled GPS receiver. You can also hook up to your phone, and the internet, via Bluetooth.

This is definitely a game-friendly machine, although without the bleeding edge graphics and major horsepower there are some Windows games you won't be able to run. Note that there is no DVD player, so if you want to watch movies (and that's a pretty obvious use) I expect you'll be encouraged to buy/download your media.

These devices have hard drives, and sizes vary from 30 to 120 GB. RAM? I'm not clear on that - again some hedging by Burkes on upgradability. Maybe you'll be able to buy additional RAM, maybe it'll be fixed. That will probably vary from model to model.

While the first models will ship with XP Tablet Edition, some prototypes are running early versions of Windows Vista.

Check out the Microsoft Ultra-Mobile PC page for more.

SV news server moves

Direct link

Posted Wednesday, March 08, 2006 by Dave Harms

The SV news server went onto a new T1 over the weekend, and the transition seems to have been pretty smooth. How soon you got discuss.softvelocity.com back depended on how fast your ISP propagated the DNS changes. If you're still having problems you can use the IP address directly, as noted in this post by Bob Zaunere. But as Bob also notes, that's not a great way to go because you'll probably forget about the IP changes and you'll get nailed the next time the address changes. And if you have been using the IP address, either in your hosts file or directly in your newsreader, now's a good time to change it back.

Revving up the blog machine

Direct link

Posted Wednesday, March 08, 2006 by Dave Harms

One blog item in a month? Is that me or SV? Okay, it's me. But that sound you hear in the back ground is the ClarionMag blog machine revving up. In fact, I've accumulated a fair blog backlog, and I need to start clearing some of this stuff out so I can see my desk(top) again.

Backup, backup, backup

Direct link

Posted Wednesday, March 01, 2006 by Dave Harms

In softvelocity.public.clarion6, on Feb 17, Ian Cook posted a tribute to iAlchemy's Cover Your Ass(ets). Ian thought he'd lost 12 hours of coding in an IDE crash, until he remembered he'd bought and installed CYA the month before. "I just selected the list of incremental backups and within a few minutes I was up and going again."

From the CYA web page:

> Cover Your Ass(ets) is implemented as a global extension as well as a utility that serves in the development process as an “automatic” generational backup mechanism at the application level. Simply dropping the ‘iAlchemy: Enable Cover Your Ass(ets)” global extension into any application will automatically create a standard Clarion TXA & TXD at compile time into a user specified folder.

This is a slick idea - CYA runs automatically, generating a TXA with a timestamped filename each time you compile.

Now, about the CYA *web page*...

Another backup program you should take note of is Jeff Slarve's In Back:

> Using a "Project System" in which you tell specifically which files belong to your project, all of those files can be quickly and safely backed up as frequently as desired. Any time that you are about do something to your work that means lots of changes or that could potentially "break" something, all you have to do is press a button and your work will be saved to that point in time. In addition, every time that you back up, a new backup file is created. You don't have to worry about accidentally overwriting your last "good” backup. When the number of backups for that project hits the max limit (that you specify), the oldest backup files are deleted from the disk.