Clarion Magazine

SoftVelocity Sold On eBay

SoftVelocity, maker of the Clarion line of development tools, announced today that the company had accidentally been sold on eBay.

Posted Tuesday, April 01, 2003

Potholes On The Road To Open Source Database Nirvana

Once again a promising bend in the road turns into a jarring realization: this time it's the PostgreSQL ODBC driver that provides the unpleasant surprise.

Posted Thursday, April 03, 2003

A Naïve Look at The Mutex

Steve Parker finishes up his look at Clarion 6 synchronization objects with the mutex. Part 1 of 2

Posted Thursday, April 03, 2003

Mutexes: Serializing File Access

Steve Parker finishes up his look at Clarion 6 synchronization objects with the mutex. Part 2 of 2

Posted Friday, April 04, 2003

GPF Challenge Results

The response to the GPF challenge was more about quality than quantity. Try your hand at the solution to the lone entry.

Posted Friday, April 04, 2003

Weekly PDF For April 1-5, 2003

All ClarionMag articles for April 1-5, 2003 in PDF format.

Posted Monday, April 07, 2003

Clarion In eWeek

Peter Coffee writes about a recent comp.lang.clarion newsgroup discussion on SoftVelocity's marketing of Clarion.

Posted Tuesday, April 08, 2003

ASCIIing For More

Konrad Byers is back with a new and improved ASCII file

Clarion Magazine **subscriptions** (online only) are $49 for six months, $95 for one year and $170 for two years.

Subscribe **Renew**

News

Serial Communications Lib 1.1

Logic*Central 40% Off Sale

cpTracker Limited Time Sale/New Features

New Look For Gitano

Icetips Report Previewer

gReg Updated

Brazilian DevCon

Super QuickBooks Export Templates Enhanced

CWPlus 1.04

Beta Testers Needed For faxFUSE

RPM Upgrade Available

Clarion Jobs Page Updated

Ingasoft T-Shirts, Mugs And More

Easy3DStyle 1.04

EasyListPrint 1.01

TimeSavers Scheduler Beta

handing class that is interchangeable with the ABC FileManager, and handles multiple file layouts on the fly.

Posted Thursday, April 17, 2003

Who Calls Who - Keeping Track Of DLL Calling Order

Faced with a 31 DLL project and some build sequence errors, Steffen Rasmussen used a matrix and a simple set of rules to determine how to change the build order to resolve circular references.

Posted Thursday, April 17, 2003

Weekly PDF For April 13-19, 2003

All ClarionMag articles for April 13-19, 2003 in PDF format.

Posted Tuesday, April 22, 2003

Data Structures and Algorithms Part XVIII - Networks & Graphs

Clarion Magazine's All-Alison Neal week kicks off with an introduction to Networks and Graphs, two highly useful abstract data types.

Posted Wednesday, April 23, 2003

Data Structures and Algorithms Part XIX - Simple Graphs

All-Ali week continues with Alison Neal's implementation, in Clarion, of a Graph.

Posted Thursday, April 24, 2003

Data Structures and Algorithms Part XX - Topological Sort

Clarion Magazine's All-Ali week concludes with Alison Neal's discussion of a Graph Topological Sort.

Posted Friday, April 25, 2003

Clarion 6 EA4 Released

Clarion 6 Early Access Release version 4 is now available to EA program participants. This release includes new RTF support, business graphing, an XML parser interface, and report generators (Enterprise Edition only) for PDF, HTML, RTF and text. Various components and examples have also been updated.

Posted Tuesday, April 29, 2003

Win A Three Month Newsletter Service Subscription

Potentially Cheap MSDN Universal

Easy3DStyle 1.03

EMS PostgreSQL Manager 1.5 Released

RDRAW Diagramming Template Demo

File Manager 3 Beta 16

CapeSoft Mailer v1.9m

Replicate v1.0 Beta 14

Office Inside v1.0 Beta 4c

File Explorer v2.5d

NetTalk v2.73

CapeSoft Email Server 1.51

RInstall FAQ

EasyResizeAndSplit 1.07

ImageEx 2 Competitive Upgrade

PD Finance Library And Calculators Version 60-01

C55 PD Browse Button Lookup Updated

Clarion Training In South Africa

Join The Clarion World

Clarion Third Party Profile Exchange Updated

RADIntelliSense Trial Version

INN Bio For April 1, 2003

CWPlus 1.03

EasyHelper 1.00

cpTracker Gold 50% Off Sale

Search the news archive

Looking for more? Check out the **site index**, or **search the back issues**. This site now contains more than 700 articles and a total of over a million words of Clarion-related information.

Clarion Magazine

SoftVelocity Sold On eBay

Published 2003-04-01

SoftVelocity, maker of the Clarion line of development tools, announced today that the company had accidentally been sold on eBay. Details have been hard to come by, but according to an employee, who goes by the code name “Bob,” someone at the company’s Pompano Beach, FL office attempted to post a Clarion car stereo for sale on the popular Internet auction site.

“Near as we can figure out, he did a cut and paste from a Solodex app, which he’d adapted as a sort of personal inventory application. There was already an internal company memo in the copy buffer, and we think the Solodex app mixed the words together,” said Bob, who did not want his real name used. “We’ve been trying all day to reproduce the bug.”

The description on eBay read: “Slightly used Clarion software company, open architecture for expandability, voice-activated control with 200 word vocabulary and vehicle mileage log. Each support team member has a detachable face, and the built in GPS means your code will never go missing again. Comes with Service Pack 1 CD. With 140 watts of power you’ll be 100 times as productive as before.”

The company was purchased by an anonymous Texan. Contacted through eBay, the buyer complained that he still hadn’t received his stereo. When told by Clarion Magazine that he had bought a software development firm and not an audio product, he was heard to call to someone in the room, “Hey, Belinda, call VISA and have those [yahoos] cancel that [dang] Clarion stereo I bought on eBay! It ain’t comin’. And why is that horse in the livin' room again?”

Ironically, the second last bid for the company was by a former employee, Andy Ireland, who took ownership the following day. “I really had no idea,” said Ireland. “I thought I was buying a rather nice bit of car stereo equipment. But I think it’ll work out fine. I just got the assets this morning and noticed that the COM support needed to be rewritten again. Fortunately I had a few minutes to spare and it’s all running smoothly now. Release date? No idea.”

For more information on this breaking story, click here.

Reader Comments

Add a comment

I primarily buy from Ubid.com , they would have had a...
OMG, what a riot:)
I kept bidding on it to drive the price up. I think Andy...
Damn. Spend the day in customer service training and see...

Clarion Magazine

Topics > Databases > MySQL

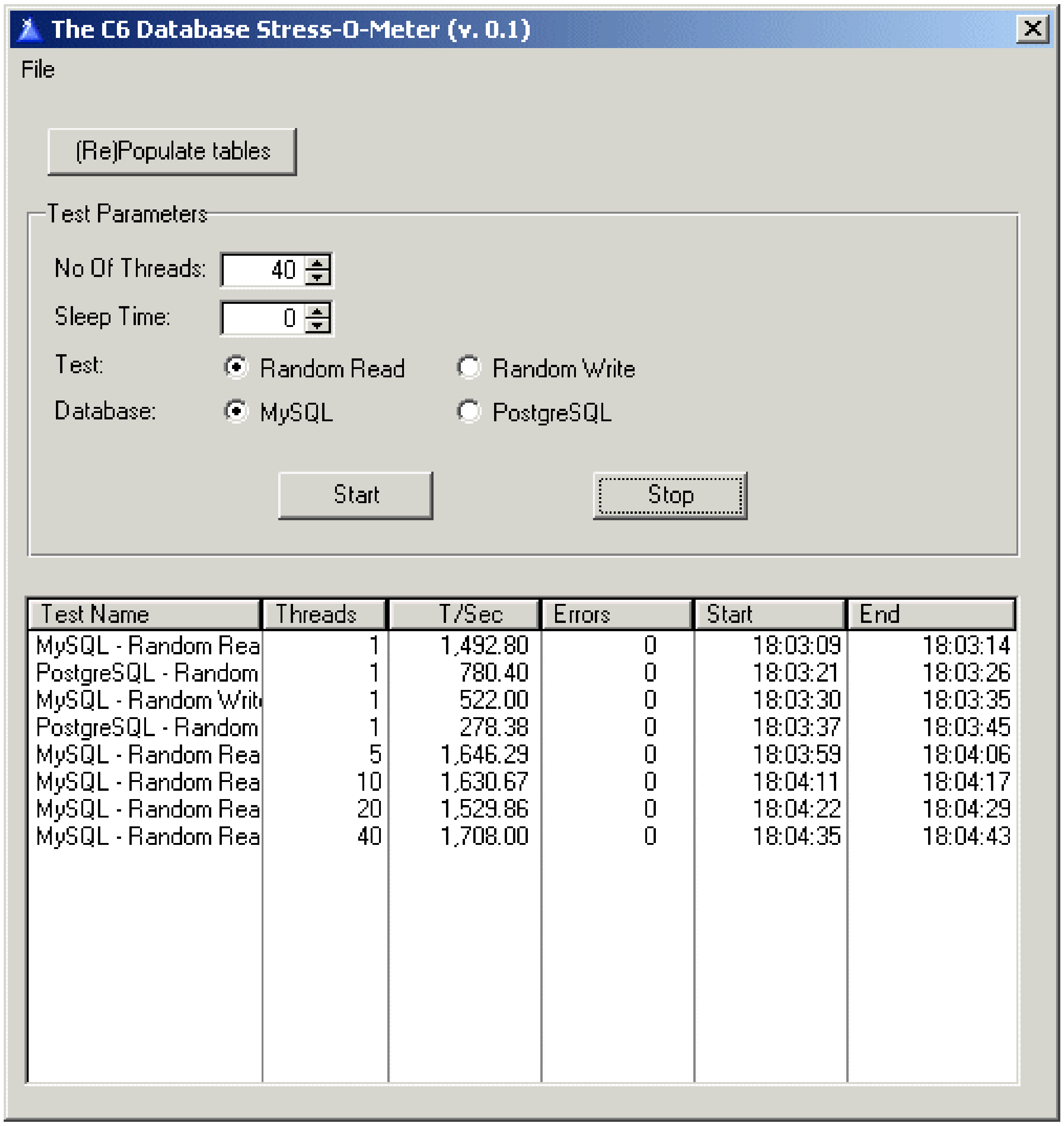
Potholes On The Road To Open Source Database Nirvana

by David Harms

Published 2003-04-03

I've begun some comparative testing between MySQL and PostgreSQL, and while the testing isn't far enough along that I'm ready to post results, I have discovered one very unsettling problem with the PostgreSQL psqlodbc driver: It appears the current version is not thread safe, and that could mean problems with Clarion 6 apps.

I discovered the problem while running a Clarion 6 database stress tester I cobbled together. I decided I would take advantage of C6's true threading capabilities and create an application that could spawn a specified number of threads, each hammering on the database in a particular way. The stress tester is still in its infancy and I'm not ready to release any code yet, but some of the early results were interesting enough that I feel they are worth posting. Figure 1 shows a screen shot of an early version of the tester.



Test Parameters

No Of Threads: 40

Sleep Time: 0

Test: [✓ Random Read] Random Write

Database: [✓ MySQL] PostgreSQL

Test Name	Threads	T/Sec	Errors	Start	End
MySQL - Random Rea	1	1,492.80	0	18:03:09	18:03:14
PostgreSQL - Random	1	780.40	0	18:03:21	18:03:26
MySQL - Random Writ	1	522.00	0	18:03:30	18:03:35
PostgreSQL - Random	1	278.38	0	18:03:37	18:03:45
MySQL - Random Rea	5	1,646.29	0	18:03:59	18:04:06
MySQL - Random Rea	10	1,630.67	0	18:04:11	18:04:17
MySQL - Random Rea	20	1,529.86	0	18:04:22	18:04:29
MySQL - Random Rea	40	1,708.00	0	18:04:35	18:04:43

Figure 1. The stress tester

So far I've tested the psqlodbc driver on PostgreSQL 7.3.2, and MyODBC 3.51.06 on MySQL 4.0. The very simple tests I ran were randomly reading or writing records in a table with 5000 records. Here are the create scripts I used for the two databases:

```
# Create the MySQL table

CREATE TABLE demodata (
ID int(11) NOT NULL,
Name varchar(50) NOT NULL default 'test',
Value decimal(9,2) default 0,
PRIMARY KEY (ID),
KEY demodata_index_name (name),
KEY demodata_index_value (value),
KEY demodata_index_namevalue (name,value),
KEY demodata_index_valuename (value,name)
) TYPE=MyISAM;

# Create the PostgreSQL table

CREATE TABLE demodata (
ID serial NOT NULL primary key,
Name varchar(50) NOT NULL default 'test',
Value decimal(9,2) default 0
);
CREATE INDEX demodata_index_name on demodata(name);
CREATE INDEX demodata_index_value on demodata (value);
CREATE INDEX demodata_index_namevalue on demodata (name,value);
CREATE INDEX demodata_index_valuename on demodata (value,name);
```

I ran the test application on an Athlon 1800 Windows 2000 workstation, connected on a 100Mb network to a Linux box running RedHat 8.0.

Note that the first tests, in which I compared PostgreSQL and MySQL times, were restricted to one thread each. That's because attempting to run more than one PostgreSQL thread resulted in a "Connection already in use" error from the driver. Scott Ferret has reviewed the logs and is of the opinion that this is indeed a back end driver problem. Vernon Godwin did a lot of digging on this issue and unearthed information indicating that a thread safe version of the pgsqlodbc driver is in fact in development, and he even found some more recent betas, but not even the latest drivers fix this problem.

I did a bit of experimenting with a C6 application using a plain old browse and form - with two browses up, and with two forms open at once, I was unable to generate the error, so for normal everyday use you may not have a problem. Vernon also reports that the pgsqlodbc driver works very well with Clarion 5.5. It may be that the thread safety problem will only manifest itself when you have two threads pounding on the same table at *exactly* the same time.

What about the test results themselves? For starters, in tests with very simple tables, it appears that MySQL does hold a considerable edge over PostgreSQL in raw speed. And my test data is a bit deceptive: it shows a two times speed difference, but if you are running PostgreSQL in its standard configuration the difference will be more like four times! By default, PostgreSQL writes every single UPDATE statement out to disk immediately, which results in a *lot* of disk thrashing. You can turn this off by setting fsync=false in postgresql.conf and restarting the postmaster. But you probably only want to do this in a situation where you have a *lot* of faith in your RAID storage, or where you really don't care if some data goes missing in a power failure.

I'm encouraged by the indications that a thread safe postgresql driver is at least in the works, but the failure of the present driver in the test scenario certainly puts a damper on my PostgreSQL migration plans. You may recall that I ran into a related concern with the MySQL driver last year. Happily that driver, at least in its latest incarnation, handles multiple threads quite nicely.

Vernon Godwin did point me to a commercial PostgreSQL driver by OpenLink - I'm testing this driver now, and although I've run into some problems that so far have prevented me from running multi-threaded tests the driver is supposed to be thread safe, and I'm in contact with OpenLink's tech support people.

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). His most recent book is JSP, Servlets, and MySQL, published by HungryMinds Inc. (2001).

Reader Comments

Add a comment

Clarion Magazine

Topics > CLARION 6 > C6 Threading

A Naïve Look at The Mutex

by Steven Parker

Published 2003-04-03

Last time, I waded into some real eye-crossing issues in pre-emptive threads. Mostly, I struggled with the new design criteria and vocabulary of threading. Typical of Microsoft, none of the terms needed to understand threading mean what they appear to mean.

A Ray of Sunshine

Amidst all this new information, these new concepts, practices and terms, I found a synchronization object that, stylistically, appeals to me: the mutex. Moreover, it is one I can easily get a handle on.

Better, "mutex," while sounding like something that needs to be cleaned up, is not a typical Microsoftism, it is not a *meaningless* neologism. According to Webopedia, "mutex" is

> (1) Short for ***mut****ual* ***ex****clusion object*. In computer programming, a mutex is a program object that allows multiple program threads to share the same resource, such as file access, but not simultaneously. When a program is started, a mutex is created with a unique name. After this stage, any thread that needs the resource must lock the mutex from other threads while it is using the resource. The mutex is set to unlock when the data is no longer needed or the routine is finished.

At last, a synchronization object that says what it does – it excludes others and stops them from mucking about with my data.

Microsoft’s documentation (MSDN, Mutex Class) defines a mutex as follows:

> When two or more threads need to access a shared resource at the same time, the system needs a synchronization mechanism to ensure that only one thread at a time uses the resource. **Mutex** is a synchronization primitive that grants exclusive access to the shared resource to only one thread. If a thread acquires a mutex, the second thread that wants to acquire that mutex is suspended until the first thread releases the mutex….
>
> You can use a Mutex object to synchronize between threads and across processes. Although **Mutex** doesn't have all of the wait and pulse functionality of the **Monitor** class, it does offer the creation of named mutexes that can be used between processes.

And, according to SoftVelocity's Multi-Threaded Programming:

> An IMutex is used when you want to allow only one thread to access a resource. Just like an ICriticalSection. However, IMutexes have the added features of being able to not only synchronize threads, but also synchronize different processes. Thus, if you have a resource that can only have one process accessing it at one time (e.g. a registration file that controls access to multiple programs) then you will need to use an IMutex that is created by calling NewMutex(*Name*). *Name* must be the same for all processes that use it to access the same set of shared resources.

SoftVelocity's Scott Ferrett informs me that inter-process synchronization (an application, even a multithreaded application, is a "process") is the most appropriate use of a mutex. He also notes that mutexes are a little less efficient than Critical Sections.

Despite that, I am struck by …

The Appeal of the Mutex

Please take a moment and re-read the last two citations. There are three points that stand out, at least to me.

First, a mutex can be used to synchronize threads within an application (even if that is not its best or most efficient use). It can also "synchronize different processes." That is it can mediate multiple applications.

Second, mutexes are named. That is, when I create a mutex, I give it an name, a unique identifier. Thus, a datum exists for which I can test.

A unique identifier for a mutex is analogous to an operating system-wide flag. If the mutex exists, its name is available from any application running on the computer. In a sense, a mutex’s name is the ultimate global variable.

Third, digging further into the documentation, and this is the part that really appeals to me, mutexes are "wait-able." And, as it turns out, "wait-ability" includes the ability to set a time period for the wait; if I can get the mutex within the time period, take it. If not, I can take an action.

Most importantly, "wait-ability" means I have the ability to test for the mutex, to test whether it is locked or not. Consequently, I have the ability to act if the mutex is locked by another process or thread. In other words, I decide what to do when I have to stand in line too long waiting for a resource.

This, as far as I am concerned, is *very* good.

Consider the code sample in the SoftVelocity documentation:

```
PROGRAM
INCLUDE('CWSYNCHM.INC'),ONCE
MAP
END
Limiter &IMutex,AUTO
Result SIGNED,AUTO
CODE
Limiter &= NewMutex('MyApplication')
IF Limiter &= NULL
  MESSAGE ('ERROR: Mutex can not be created')
ELSE
  Result = Limiter.TryWait(50)
  IF Result <= WAIT:OK
    !Do Everything
  ELSIF Result = WAIT:TIMEOUT
    MESSAGE('Timeout')
  ELSE
  MESSAGE('Waiting is failed')
END
Limiter.Kill()
END
```

The `TryWait` method’s parameter is the number of milliseconds to stand in line ("wait") for control of the mutex. The code above waits for 50 milliseconds (.05 seconds, five one-hundredths of a second – pretty quick). If the wait "succeeds" (i.e., I didn’t have to wait), processing continues. If it does not, in the sample above, the end user gets a message.

What I like about this is that I have the choice of waiting until the mutex is released *or* giving the user a message *or* bailing out *or* …. I have the choice to terminate the wait. I have the choice to tell `TryWait` to wait forever.

Mutexing

Like many examples, however, SoftVelocity's mutex example is "hand coded." How do I implement a mutex in an .APP? How do I adapt this code to the way I use Clarion?

The answer is: "It depends."

It depends on how I intend to use the mutex.

If I will be using the mutex at the app level, as I would for inter-process synchronization, everything is done in the global embeds.

In **Global Data**, I embed the necessary prototypes and data:

```
INCLUDE('CWSYNCHM.INC'),ONCE
Limiter &IMutex,AUTO
Result  SIGNED,AUTO
```

In **Program End**, I release the mutex:

```
Limiter.Kill()
```

And, in **Program Setup**, I embed the code to test whether I can lock the mutex or have to stand in line:

```
Limiter &= NewMutex('MyApplication')
IF Limiter &= NULL
  MESSAGE ('ERROR: Mutex can not be created')
ELSE
  Result = Limiter.TryWait(50)
  IF Result <= WAIT:OK
    !Mutex is available, no one else "has" it
         !do stuff
  ELSIF Result = WAIT:TIMEOUT
    MESSAGE('Timeout')
  ELSE
    MESSAGE('Waiting is failed')
  End
END
```

However, if I want to use it at a lower level, say a procedure, two changes are necessary.

First, the code to test whether the mutex is available or already locked has to be moved into the procedure's `INIT` method. If, for example, the mutex will be used to synchronize access to a file, it must be tested before attempting to open that file. Similarly, the code to release the mutex (along with a quick check that the procedure successfully locked it) has to be moved into the procedure' `Kill` method:

```
If SELF.FilesOpened
  Limiter.Kill()
End
```

(`SELF.FilesOpened` can only be true if the procedure successfully opened all files.)

Second,

```
Limiter &IMutex,AUTO
```

must be changed to:

```
Limiter &IMutex,AUTO,Thread
```

As Scott Ferrett pointed out to me, when I hadn't `Thread`ed the reference:

> If you are going to create and destroy it on every thread, then it needs to be threaded. Your code does this:
>
> - Create Mutex
> - Create Mutex !Leaking old Mutex
> - Delete Mutex
> - Delete Mutex !Bang

The two "Create Mutex" notations come from the two starts of the procedure trying to get the mutex. The app I tried this with was GPFing at program end.

You may verify this for yourself by modifying the FileOnce.APP example. Remove the `Thread` attribute from the `Limiter` variable (in **Global Data**). Run the app and try to open one of the browses a second time. When you quit the app, you will GPF. (I have included two example apps, one with a global mutex and one with a procedure-level mutex. They are C6. So, as soon as you get your copy, check the implementations out.)

Ugly stuff. Play it straight and thread the reference variable.

Uses for Mutexes

Perhaps the first thing that struck me about the description of mutexes is its ability to set a truly global flag. Thus, a mutex provides a way to tell if an application is running.

A mutex sets a global flag and it allows me to test for it. If I can lock the mutex, the application was not already running. If I get queued up, the application is already running.

If I decide not to continue waiting, I can use a mutex to ensure that an application is running only once on a machine.

The ability to detect another running instance of an application and act on it becomes increasingly important as operating systems become more "advanced."

I have been using a DDE-based method for detecting another instance of an application. I'm using a technique first suggested by Richard Taylor in 1999 in the TopSpeed news groups. Many Clarion developers are, I suspect, using this method or a variant of it.

The last year or so, there have been claims that Microsoft has been intentionally slowing DDE in XP ("crippling" it). There are other claims that it just never was very stable. And there are further claims that it always has been poorly implemented (new, fast machines really highlight this).

I have reports of developers on XP starting Clarion and then starting my batch compiler, GTL. If they don't wait a while after starting Clarion, GTL reports Clarion is not running. Clarion registers itself as a DDE server and GTL checks for that. Clearly the DDE messages aren't passing quickly enough.

By rapidly double clicking, I can start two instances of an app with this DDE "start once" code in place.

With C6, I can use a mutex to ensure only a single instance is running as follows:

```
Limiter &= NewMutex('thisApp')
If Limiter &= NULL
  !very serious problem
  Message('Error: Mutex cannot be created')
  Return
Else
  !try for half second
  Result = Limiter.TryWait(500)
  If Result <= WAIT:Ok
    !program not already running, continue
    ProgramStarted = True
  Elsif Result = WAIT:TIMEOUT
    !program is already running, warn user
    Message('Another instance of this program is running. ' &|
            'Please switch to that instance.','Don''t Do ' &|
            'That!',ICON:Hand)
    !and terminate
    Return
  Else
    !another  very serious problem
    Message('Wait failed')
    Return
  End
End
```

Note how I set `ProgramStarted` when the mutex is locked, i.e. the program starts. It is used in the **Program End** global to release the mutex when the app closes:

```
If ProgramStarted
  Limiter.Kill()
End
```

And, if experience shows that someone can start a second instance quickly enough, I just increase the `TryWait` period.

To see this in action (even if you don’t have C6, I have provided the required DLLs), run the demo app, StartOnce. Start it a second time.

All that is missing is the ability to "switch" to the active instance. To cover this, I’d just incorporate the "old" technique into the mutex (since timing and speed are no longer relevant).

Something like the following ought to do (though I haven’t tested it):

```
Limiter &= NewMutex('thisApp')
If Limiter &= NULL
  Message('Error: Mutex cannot be created')
  Return
Else
  Result = Limiter.TryWait(500)
  If Result <= WAIT:Ok
    ProgramStarted = True
    Channel = DDEServer('thisApp')
  Elsif Result = WAIT:TIMEOUT
    Message('Another instance of this program is running. ' &|
            'Please switch to that instance.','Don''t Do ' &|
            'That!',ICON:Hand)
    IF Channel <> 0
      DDEEXECUTE(Channel,'INFRONT')
      DDECLOSE(Channel)
    END
    Return
  Else
    Message('Wait failed')
    Return
  End
End
```

Summary

A mutex may not be the most efficient synchronization object. It may not be the "best" synchronization object for intra-process synchronization (as the example app shows, it is a real gem for inter-process work).

But, it is straight-forward and easy to understand.

Next time: mutexing access to a file (just as described by Softvelocity).

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

Topics > CLARION 6 > C6 Threading

Mutexes: Serializing File Access

by Steven Parker

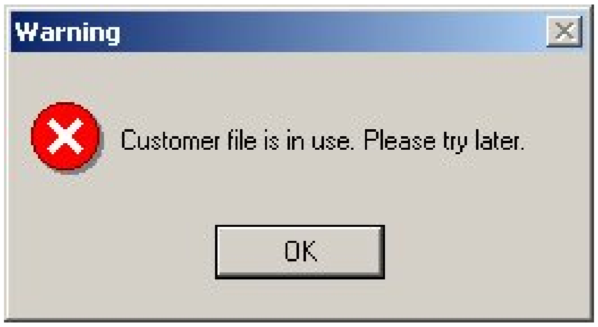
Published 2003-04-04

Read Part 1

Attached to this article you should find a Clarion 6 .APP, along with the EXE and all relevant DLLs (so everyone can run the demo and follow along), called FileOne. Start up FileOne.EXE.

In the Files menu, you will see "Names (1/10 sec. wait)." Click on that menu item. You will get a browse. Leaving the browse open, select "Names (1/10 sec. wait)" again.

You will not get another copy of the Names browse, even though the browse procedure *is* threaded (if you don't have C6, you'll have to take my word on that). Instead, you will get a message.


Warning

Customer file is in use. Please try later.

Figure 1. Second attempt to open Names browse

Close any open browses.

Now click on the "Names (10 sec Wait)" menu item. You will get the an almost identical browse on the Names file. Click the menu item again.

Close the browse. It doesn’t go away. But, if you close it again, it does.

If you open this browse twice yet again but wait for 10 seconds (duh!), you’ll get the message again.

Close the app and repeat the calls to the "Names (10 sec Wait)" item. This time, pay careful attention to the string display above the browse box.

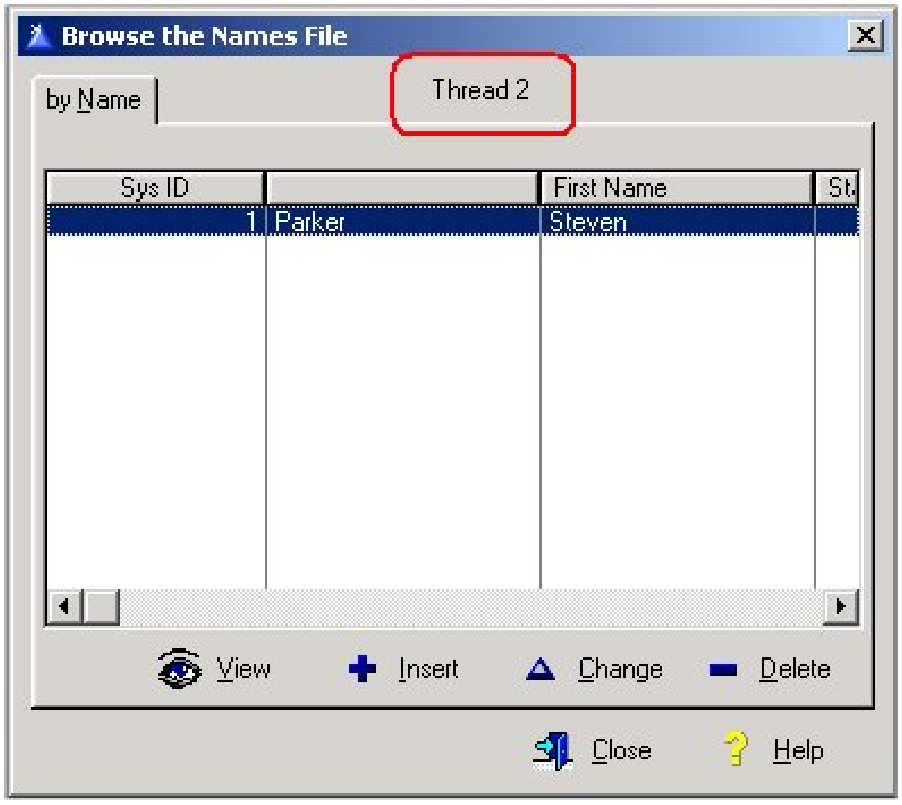
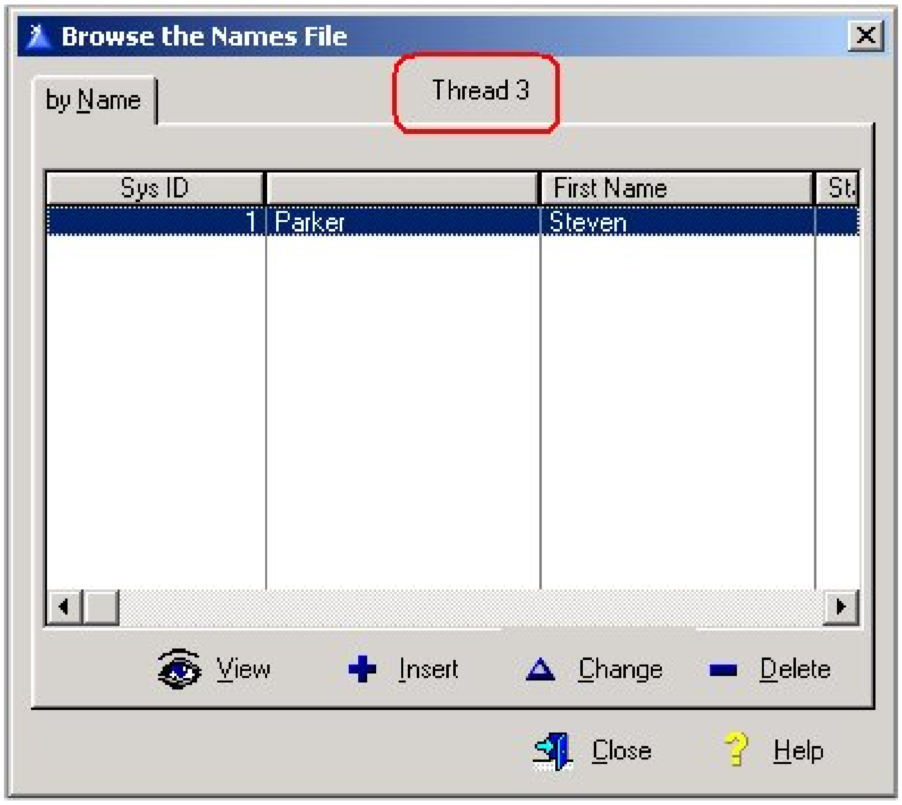


Figure 2. Names browse when first opened



Sys ID		First Name	St.
1	Parker	Steven	

Figure 3. Names browse after closing it once

Note that the thread number displayed is different. In fact, the browse did close but another copy was waiting (see A Naïve Look At Pre-Emption for an explanation of the real meaning of "wait") to open. When the first instance closed, within the wait period I had defined, the second instance opened.

For the final experiment this morning/afternoon/evening (pick the one appropriate to your time zone), start FileOnce.EXE again and click on the "Names (1/10 sec. wait)" menu item.

Start a second instance of FileOnce.EXE and again click on the "Names (1/10 sec. wait)" menu item. Figure 1, above, shows the result of this experiment.

Pretty cool, huh? But, truth is, this is just another a simple use of a mutex.

Why?

What earthly purpose is there in limiting access to a file to a single procedure?-Especially with a technique that locks a resource as tightly as a mutex?

Reading a configuration file containing the next unique number for <supply the name of any file you use unique identifiers for> immediately springs to mind.

Even the built-in autonumbering (for non-SQL files) can be beaten. While the frequency with which duplicate numbers are created is small in absolute terms, it happens often enough to be an issue for end users.

Many Clarion developers have been using control files to hold "next numbers" since CDD days. But even the built in restrictions, `Lock` and `Hold`, have not proven reliable enough for many.

A mutex is a lock that ***locks***. And, as I have stated before, it is easy to test whether the lock is in place and I like the sense of control that gives me.

Implementing mutexes to restrict file access

Implementing a mutex is fairly straightforward, as I described last time.

Add the following to the application’s Global Data embed:

```
 Include('cwsynchm.inc'),once
Limiter &IMutex,auto,Thread
Result Signed,auto
```

This code includes the interface into the supported synchronization objects and declares the required variables. (I suppose it is also possible to embed this locally but I haven’t tested that.)

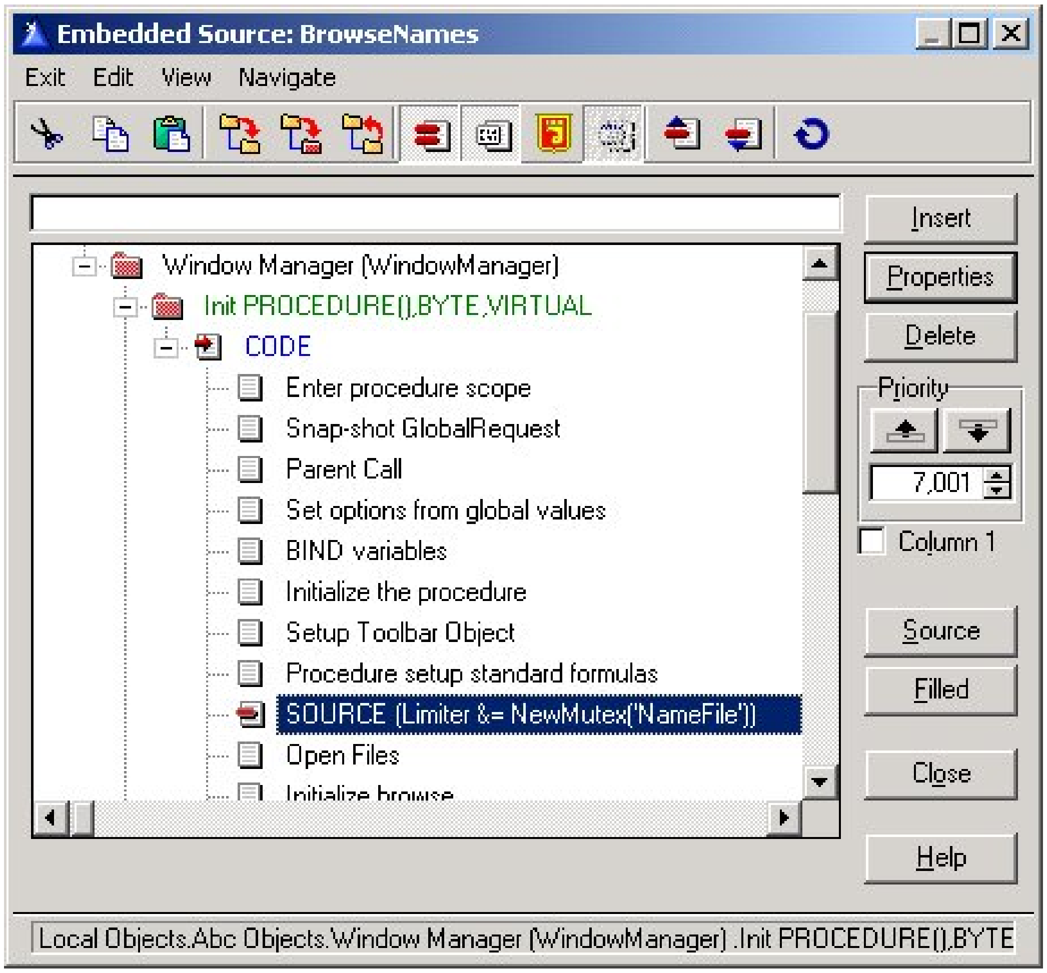
Note that the reference to a mutex (`Limiter`) is threaded. This allows creating a mutex from a threaded procedure without risking leaking mutexes (much like leaking memory but messier).

In the procedure where exclusive access to the file is required, add the following code to the **Before opening files** embed:

```
Limiter &= NewMutex('NameFile')
 If Limiter &= NULL
   Message('Major problem','Uh Oh!',ICON:Exclamation)
   Return Level:Fatal
 Else
   Result = Limiter.TryWait(100)
   If Result <= Wait:OK
     !ok to open file
   Elsif Result = Wait:Timeout
     Message('Customer file is in use. Please try ¿
             later.','Warning',ICON:Hand)
     Return Level:Fatal
   Else
     Message('Unable to get near the customer file.','Go ¿
              Away',ICON:Hand)
   End
 End
```

(Please don't duplicate my somewhat less-than-professional messages.)

The location for this is important; it must be before attempting to open the file(s):



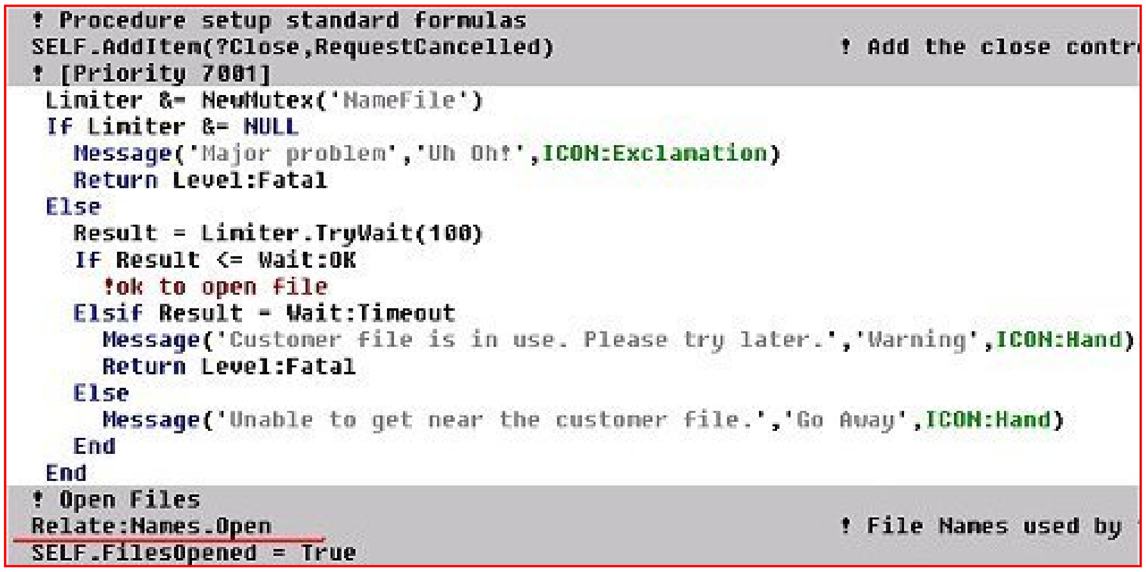
Window Manager (WindowManager)
Init PROCEDURE(),BYTE,VIRTUAL
CODE
Enter procedure scope
Snap-shot GlobalRequest
Parent Call
Set options from global values
BIND variables
Initialize the procedure
Setup Toolbar Object
Procedure setup standard formulas
SOURCE (Limiter &= NewMutex('NameFile'))
Open Files
Initialize browse

Local Objects.Abc Objects.Window Manager (WindowManager) .Init PROCEDURE(),BYTE

Figure 4. Embed location for a file access mutex

There are two reasons for the code location. First, it would be extremely dangerous to try to open the file if another process or procedure has the file open. This would also defeat the entire purpose of restricting access to one or more files if it succeeded.

Second, the templates, at this point, provide a built in flag that tells me whether or not the file open was successful:



```
! Procedure setup standard formulas
SELF.AddItem(?Close,RequestCancelled)                        ! Add the close contr
! [Priority 7001]
 Limiter &= NewMutex('NameFile')
 If Limiter &= NULL
   Message('Major problem','Uh Oh!',ICON:Exclamation)
   Return Level:Fatal
 Else
   Result = Limiter.TryWait(100)
   If Result <= Wait:OK
     !ok to open file
   Elsif Result = Wait:Timeout
     Message('Customer file is in use. Please try later.','Warning',ICON:Hand)
     Return Level:Fatal
   Else
     Message('Unable to get near the customer file.','Go Away',ICON:Hand)
   End
 End
! Open Files
Relate:Names.Open                                            ! File Names used by
SELF.FilesOpened = True
```

Figure 5. Code generated with built in flag

The FilesOpened flag is useful when leaving the procedure – you can test it to know whether or not the mutex was locked and, therefore, needs to be released:

```
If SELF.FilesOpened
  Limiter.Kill()
End
```

(If the file is not needed for the entire scope of the procedure, this code should be placed at the earliest moment it is safe to do so.)

Summary

Synchronization objects, necessary to safely use pre-emptive threads, are new and complex.

The mutex, at least, is one that is easy to understand and straightforward to implement. I hope this provides a foundation on which you can understand the more intricate techniques with which we are all about to become more familiar.

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

I could not get the thread number to display above the...
I could not get the thread number to display above the...
I've posted an updated zip - refresh the page to make sure...
In the zipfile c60ascx.dll is missing

Clarion Magazine

GPF Challenge Results

by David Harms

Published 2003-04-04

Last month Clarion Magazine asked you for your nastiest GPFs. Russ Eggen had just written an article post mortem debugging, and was ready to take on all challengers. The response to the challenge was, well, underwhelming. In fact only Gordon Smith entered, with an example that demonstrates Clarion's "virtual list box" capabilities.

Gordon's entry is fairly short (you can download both the source and the PRJ at the end of this article, but it certainly demonstrates a sneaky kind of GPF. I confess that I was unable to locate the problem using the debugger, although I might have had better luck had I been able to read assembly language.

Now Russ was supposed to have written up his findings already, but he's been hunkered down doing some real work (no, really - I believe him, of course I do) and hasn't been able to find the time. Rather than wait any longer, and as the winner is, well, fairly obvious, here's Gordon's entry:

```
  PROGRAM

  MAP
  END

StripedListQ        QUEUE,TYPE
S                     STRING(20)
                    END

StripedList         CLASS,TYPE
Init                  PROCEDURE(WINDOW w, SIGNED feq, StripedListQ Q)
VLBproc               PROCEDURE(LONG row, SHORT column),STRING,PRIVATE
Q                     &StripedListQ,PRIVATE
ochanges              LONG,PRIVATE
                    END
```

```
window WINDOW('Caption'),AT(,,153,103),SYSTEM,GRAY
       LIST,AT(33,12,80,80),USE(?List1),FORMAT('20L*')
     END

Q      QUEUE(StripedListQ)
       END

SL     StripedList
i      SIGNED

  CODE
  LOOP i = 1 TO 20
    Q.s = 'Line ' & i
    ADD(Q)
  END
  OPEN(window)
  SL.Init(window, ?list1, Q)
  ACCEPT
  END

StripedList.Init PROCEDURE(WINDOW w, SIGNED feq, StripedListQ Q)

  CODE
  SELF.Q &= Q
  SELF.ochanges = CHANGES(Q)
  w $ feq{PROP:VLBval} = ADDRESS(SELF)
  w $ feq{PROP:VLBproc} = ADDRESS(SELF.VLBproc)

StripedList.VLBproc PROCEDURE(LONG row, SHORT col)

nchanges LONG

  CODE
  CASE row
  OF -1
    RETURN RECORDS(SELF.Q)
  OF -2
    RETURN 5
  OF -3
    nchanges = CHANGES(SELF.Q)
    IF nchanges <> SELF.ochanges THEN
      SELF.ochanges = nchanges
      RETURN 1
    ELSE
      RETURN 0
    END
  ELSE
    GET(SELF.Q, row)
    CASE col
    OF 1
      RETURN WHAT(SELF.Q,1)
    OF 3
      RETURN CHOOSE(BAND(row,1), COLOR:none, 0c00000H)
    END
  END
```

No prizes for sorting this one out (although I'll give bonus points to anyone describing how they used the debugger to find the bug), but feel free to post your discovery as a reader comment. In deference to those who may come to the article after you, please do *not* include the solution in the first sentence - it will show up in the comment summaries and be visible to anyone downloading the source.

Have fun!

Download the source

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). His most recent book is JSP, Servlets, and MySQL, published by HungryMinds Inc. (2001).

Reader Comments

Add a comment

The first time I ran the program I got Duplicate symbol on...
Another way to debug this is to compare it to the source...
Notes: The VLB Proc is just a smokescreen for the...
Interestingly, I just tried it in C6. When it gets to the...

Clarion Magazine

Topics > Tips/Techniques > Tips & Techniques

ASCIIing For More

by Konrad Byers

Published 2003-04-17

My last article was A Class for the ASCIIing, and described a class for conveniently managing ASCII file processing. Apparently people were asking for something like this because it has developed quite a following. There was a follow-up article by Dave Harms and Steve Parker on using the field pairs class to automatically do field/record assignments on a record by record basis, and another article by Steve Parker on using it to string parse a file.

My favorite comment was a suggestion by Bruce Johnson of CapeSoft Technologies to make the class compatible with the ABC `FileManager` class. Also, as the class will often be used for reading files that need formatting or string parsing, I got to thinking about all the poor lonely BASIC files that would be left in the data dictionary after the ASCII file definitions are implemented using this class (instead of data dictionary-described `FileManager`s).

I know I said I would talk about a buffering version of the `AnyAsciiFileClass` class next, but now there are some important things to cover first. In this article I'll describe a modified version of the `AnyAsciiFileClass` class that is interchangeable with the ABC FileManager and supports BASIC/CSV files (in addition to just the ASCII format of the previous version).This revised class ties a lot of text file parsing jobs together, namely: record layout support; multiple record layouts within a single file; error handling and reporting; both ASCII and BASIC/CSV file format processing and record filtering (that can also be used for non-record/ascii text line filtering). As an added benefit this class is implemented to be interchangeable with the `FileManager` class objects generated from data dictionary declared files. There are also a few enhancements above and beyond that of the `FileManager` class for overriding default behavior.

As the class is now compatible with the ABC `FileManager` class you should be familiar with the Clarion help topic ‘Dual Approach to Database Operations.’ This topic describes the difference between, and the nature of, having two of most class methods, one prefixed with `Try`. Keep in mind that this new `AnyAsciiFileClass` uses the same error handling and reporting as the ABC `FileManager` class. In fact, except as I am about to describe (for the most part), the Clarion online help documentation for the `FileManager` methods is equally applicable to the `AnyAsciiFileClass`.

Overriding properties

It has always been one of my peeves to have to change a member property of a class prior to calling a method instead of just passing in the necessary parameters. I understand why the `FileManager` methods rely on properties such as the `Create` and `OpenMode` properties and the `SetName` method to establish default values/behavior, but often I wish to override the current settings of a class. Instead of being required to save the current state, make the method call and then worry about having to set it back properly (so that code elsewhere still works as before), I find it better to allow the class to handle overrides specifically.

In addition to implementing the `FileManager` class interface, the `AnyAsciiFileClass` class addresses the configuration problem I just described by implementing a superset of the `FileManager` class methods. You can see in the include file (AnyAscii.inc) that methods in common with the `FileManager` accept additional parameters. The methods capable of opening a file, `OpenExisting` or `CreateFile` for example, accept `FileName` and `OpenMode` argument parameters. These parameters will be obeyed but do not change the default behavior of the class when such a method is called without these overriding parameters.

Additionally, extra methods exist for determining/indicating if a file should be created. For example, suppose that the `Create` attribute of a `FileManager` is supposed to be `False` so that, by default, calling `Open` or `TryOpen` will not create an empty file. Under certain conditions, however, or at a particular point in a program it may be necessary to create the file. Using a `FileManager` object in this instance will require code similar to the following snippet to implement properly. (I’m forgetting about lazy open behavior here, for the sake of simplicity, which would make this even more of a nuisance.)

```
OrigCreate = ACCESS:SomeFileLabel.Create
```

```
ACCESS:SomeFileLabel.Create = True
ACCESS:SomeFileLabel.Open()
ACCESS:SomeFileLabel.Create = OrigCreate
```

When using the `AnyAsciiFileClass` you may simply use the `OpenOrCreate` method, regardless of the default behavior of the class. Likewise, if you know that you do not want to create an empty file even if there will be no file otherwise, you may call the `AnyAsciiFileClass.OpenExisting` method.

If you want to explicitly create an empty file then you may call the `CreateFile` method, which is a method that the `FileManager` class does not provide at all. The preceding default behavior override examples take care of overriding the `Create` property of the class. The `OpenMode` property and `FileName` property (set with the `SetName` method) can be temporarily overridden as well. All methods for the purpose of opening/creating a file receive optional (omittable) file name and open mode value overriding parameters, making it easy to temporarily override the create property.

Some examples

Your record processing loop will look similar to the following (when using a singular record format). Notice the explicit specification of a file name, and the create behavior using the `OpenExisting` (and in the next example, `CreateFile`) method.

```
Record GROUP,PRE(IN)
Field1 STRING(1)
Field2 STRING(2)
 END
 CODE
 Input.OpenExisting('Input.txt')
 Input.SetRecordLayout(IN:Record)
 LOOP WHILE Input.TryNext() = LEVEL:Benign
    ! Record processing code
 END
 Input.Close()
```

Here’s another example:

```
Record GROUP,PRE(OUT)
Field1 STRING(1)
Field2 STRING(2)
 END
 CODE
 Output.CreateFile('Output.txt')
 Input.SetRecordLayout(OUT:Record)
```

```
LOOP WHILE Output.TryNext() = LEVEL:Benign
   ! Record population code such as
   OUT:Field1 = '1'
   OUT:Field2 = '2'
   Output.Insert()
END
Output.Close()
```

A `FileManager` object for an ASCII/BASIC file is replaceable with an `AnyAsciiFileClass` object. After having called the `SetRecordLayout` and the `SetName` methods (and optionally setting the `Create` and `OpenMode` properties if you want to) an `AnyAsciiFileClass` object will function just as a `FileManager` object does. You may use the Clarion `FileManager` method online help topics as a quick reference (especially when you are using the class to implement record formats).

I mentioned using the class to handle BASIC and/or CSV (comma separated files) as easily as ASCII files, and it really is very easy. When you use the `SetRecordLayout` method to specify the record layout/format to use to process records, you can additionally specify two more parameters in addition to the group structure specifying the record format.

The first optional parameter is a `FieldSeparator` argument that, when present, indicates a BASIC, CSV or other delimited value field format. Some common delimiter characters are a comma (that's a "," character as in comma separated values or CSV format), and a TAB character. Both of those formats are commonly used for exchanging information, with Microsoft Excel for example.

When the `FieldSeparator` argument is used, a `FieldEnclosure` character may additionally be used to enclose field values. This would never be necessary except that a field value itself can actually contain a field separator character. For example, when using a comma delimited format and with a field containing Last Name, First Name (e.g. Byers, Konrad) as its actual value or contents, the field will be interpreted as two separate values when it is read. The solution is using an additionally field value enclosing character. Using the previous Last Name, First Name field value example and a " field enclosure character the field value of Byers, Konrad will become "Byers, Konrad".

Enclosing field values is almost mandatory for reliable operation, but nonetheless formats without field enclosing characters are still supported. The most common field enclosing character is almost undoubtedly the double quote (a " character). This makes `SetRecordLayout(SomeGroup, ',', '"')` and `SetRecordLayout(SomeGroup, TabKey, '"')` the most popular separated value record formats, supporting comma and tab delimited quoted values respectively.

Just to throw a monkey wrench into things, a field delimiter character (in addition to a field separator character) can also appear in a field's value. The separated value format is a widely used standard for information exchange. Any field-enclosing characters must be doubled up prior to writing the value. Of course the code that reads the file must be able to properly interpret these special case values. If the separator value is a single quote character, for example, (just as Clarion itself uses to enclose string values), then a string value that includes a single quote character itself, as part of its value, will be expressed just as it would be in Clarion (by the technique of doubling up the single quote character). The specifics of how this behavior is implemented is beyond the scope of this article (especially as the purpose of such a class is to free you from these considerations so that you do not have to care). In general, however, the protected methods `PopulateRecord` and `UnpopulateRecord` handle all the string parsing necessary to implement record formats, and will be of interest to many readers. A `SetRecordLayout(`*`SomeGroup`*`)` call (that's without specifying and `FieldDelimiter` or `FieldSeparator` values) establishes a fixed length record/field format. This is the format that is generally accomplished using an ASCII driver file, while a BASIC file usually implements the field delimited format just described.

A common use of both ASCII and BASIC or CSV (comma separated values) is exchanging information with mainframes. Although the use of multiple record layouts/formats within the same file is unusual in the PC world, it *is* quite common in the mainframe world. A typical technique is to use the first byte of a record to indicate the record format.

The class supports multiple record formats using the previously described method named, appropriately, `SetRecordLayout`. This method receives a `GROUP` structure as its first parameter that describes the desired record layout. You can call this method mid-stream to change the record format currently in use. Any record read with the `Next` or `TryNext` methods (when no overriding parameter is specified to read the data into instead) will result in the data read being processed into the individual fields of this pre-established group or record structure. It is important to note, however, that even the current record (the last record read) will also be (re)processed into the newly specified record format whenever the `SetRecordLayout` method is called. This allows not only for the normal pre-established single record format processing described and shown previously, but also makes multiple record layout support a breeze.

You can use code similar to the snippet below to determine the specific record layout that should be used (based on the first character of a record in this example, as is a common practice) and set or reset the current record format to use appropriately, on-the-fly. (Notice that each of the record formats used even have very different total lengths.)

```
RecordLayoutA GROUP,PRE(RECA)
RecordType STRING(1)
Field1 STRING(3)
END
RecordLayoutB GROUP,PRE(RECB)
RecordType STRING(1)
Field1 STRING(13)
Field2 STRING(70)
END
RecordLayoutC GROUP,PRE(RECC)
RecordType STRING(1)
Field1 STRING(12)
Field2 STRING(23)
Field3 STRING(12)
END
Input AnyAsciiFileClass
AsciiLine CSTRING(ANYASCII_IO_SIZE+1)
 CODE
 Input.OpenExisting('Input.txt')
 LOOP WHILE Input.TryNext(AsciiLine) = LEVEL:Benign
    CASE AsciiLine[1]
    OF 'A'
       Input.SetRecordLayout(RecordLayoutA)
       ! RecordLayoutA field processing
       RECA:Field1 = 'A'
    OF 'B'
       Input.SetRecordLayout(RecordLayoutB)
       ! RecordLayoutB field processing
       RECB:Field2 = 'B'
    OF 'C'
       Input.SetRecordLayout(RecordLayoutC)
       ! RecordLayoutC field processing
       RECC:Field3 = 'C'
    END
 END
 Input.Close()
```

There are a couple of "bells and whistles" methods that do come in very handy that I should briefly describe, as they can have a large impact on your code. The `SetCtrlZ` method is used to indicate if a `Ctrl-Z` character should be interpreted as an end of file indicator. The default is `ON` simply because this is the default for the ASCII database driver. The SetAutoFlush method is used to determine if the file should be closed after each disk write to force the file to be flushed to disk (described in AClass For The ASCIIng). The default is off, but if you are using the class to write a debugging log, for example, then it may be crucial to turn this on, or the last few disk writes may be lost (not actually flushed to disk) if the program crashes.

The `SetClipFields` method sets field value clipping behavior when "writing records" and can be very useful. Clipping fixed length records or delimited value records fields ensures all values written are string values even if the record layout contains some non-string values (although using non-string values is generally not recommended). Additionally, clipping delimited field value records will change the values written to the file so that no white space appears in the value. For example a, field value of `" some value "` will become `"some value"` (with no leading or trailing spaces).

As record filtering is such a large part of record processing, you can use the `SetRecordFilter` method to alleviate some of the headache. This method is an easy way of defining a particular file result set of interest. Most Clarion programmers are very familiar with record processing procedures using the associated ABC template so this will be a familiar concept.

The class also implements what is sure to be another very familiar concept in record filtering, the `ValidateRecord` method. This works the same way as the `ViewManager.ValidateRecord` method. The `SetRecordFilter` method and a user-overridden `ValidateRecord` method serve the same purpose, with the `SetRecordFilter` being very easy to use but not as capable of complex evaluation as an overridden `ValidateRecord` method. (Note that either of these techniques is equally appropriate for non-record layout/ASCII text line filtering as well.)

Following is an example of using an overridden `ValidateRecord` method. Once the `ValidateRecord` method filtering has been established, the remainder of any record processing implementation will be the same, but using the example filter below, "*some condition*" records will be excluded and will not be available just as if they were not present in the file.

```
AsciiFile CLASS(AnyAsciiFileClass)
ValidateRecord PROCEDURE(), BYTE, VIRTUAL, DERIVED
  END

AsciiFile.ValidateRecord PROCEDURE()
 CODE
 IF "SOME CONDITION" THEN RETURN Record:Filtered.
 RETURN PARENT.ValidateRecord()
```

Simply calling the `SetRecordFilter("`*`some condition`*`")` can exclude the same records for you, but an overridden `ValidateRecord` method is capable of infinitely more complicated filtering. The `SetRecordFilter` expression must be evaluated using a call to Clarion's `EVALUATE` and as such must meet the rules of a valid filter (in particular with regards to `BIND`ing the expression). See the Clarion online help documentation on `EVALUATE` and `PROP:Filter` for more information on these considerations. (Some of you in the know may now be thinking that using a `VIEW` can be a more efficient means of record filtering, and I will no doubt be talking about this in a future article.)

An additional dilemma (or at least a consideration) in processing ASCII/BASIC files is agreement in record format. The absence or presence of a header record that describes the record format that the rest of the file will use can have a large impact and provides an opportunity to not have to cast record layouts in stone. Having a header record that establishes field names means not having to rely on field position or field order within the record. Another approach is to use field order or position explicitly and not have to rely on field names, the number of fields within a record or their definition.

`AnyAsciiFileClass` can be used to read and/or write ASCII text files (as described in AClass For The ASCIIng), and also implements record formats when the `SetReclordLayout` method has established a record layout. You can use either a fixed length (ASCII) or variable length / delimited value (BASCI, CSV) record format; the record processing methods will adhere to 'dual-approach to database operations' standard described in the Clarion online help for performing error handling. In fact, once the record format has been established (using the `SetRecordLayout` method) record processing may be performed in completely identical fashion to using a data dictionary declared file and its associated `FileManager` class.

As `AnyAsciiFileClass` is not dependant on a particular record layout, multiple record formats do not present a problem, and you can change the record format at will while processing the file. The `SetRecordFilter` and `ValidateRecord` methods make file/record sub-set filtering both easy to use and capable of complex evaluation.

Download the source

Konrad Byers is originally from Nova Scotia, Canada and has been living in Florida since 1995. He starting programming in Clarion with 2.1 for DOS, and is still an avid user of the Clarion C/C++ compiler. He is *currently available for hire. A beta version of Konrad's MAPI-enabled eSoftAnywhere DSP & More digital signal processing software for shortwave and CB users and Ham Radio operators is now available for download.*

Reader Comments

Add a comment

Next up… Now that all record handling (for ASCII or BASIC...
Perhaps an interim step... You mention going from the...
I would not pursue an API file class for speed...

Clarion Magazine

online sales and delivery
for your applications & tools
DeveloperPLUS

Topics > News > ClarionMag News

Clarion News

Serial Communications Lib 1.1

This object library will allow you to develop 32-bit Clarion serial communication applications. It's compatible with C5 and C55.

Posted Saturday, April 19, 2003

Logic*Central 40% Off Sale

Logic*Central is having a 40% off end-of-the-month sale. IFT: HTTP Client less than $60; ShapeMaker:SMX less than $60; ShapeMaker:Windows less than $45; more savings at the web site.

Posted Saturday, April 19, 2003

cpTracker Limited Time Sale/New Features

From now through Saturday, April 26, 2003, you can get your copy of cpTracker for $89.00. Version 1.05c of cpTracker features a new gReg interface. gReg is a product from Gitano Software that allows Clarion, Visual Basic and Delphi developers to protect themselves against software pirating via state-of-the-art encryption that locks out thieves permanently. gReg has a customer file and a product registration file. cpTracker also allows you to store this information. If you only need to use gReg for generating product registration codes you can use cpTracker for maintaining customers, prospects, product sales and registration details.

Posted Saturday, April 19, 2003

New Look For Gitano

The Gitano Software web site is about to get a new look. The logo has been updated and the site has a brand new look. Send your comments to: jfmoreno@gitanosoftware.com

Posted Saturday, April 19, 2003

Icetips Report Previewer

The Icetips Previewer is a fully customizable report previewer. No DLLs, no black boxes, no fixed look and feel. The Icetips Previewer is created as a window procedure in your application, giving you full control over everything it does and how it looks. Use the icons you want, translate it, use other third party products with it, add functionality to it, remove functionality from it if you don't want it. A simple template wizard generates the Previewer procedure into your application using the procedure name you want. All the embedded code to make it work is in there ready for you to look at and modify if you want to. Apply a simple extension template to your reports to call the Icetips Previewer procedure that you want to call and you're done. Compatible with Clarion 4, 5, 5.5 and 6, ABC, Clarion and CPCS report templates.

Posted Saturday, April 19, 2003

gReg Updated

A new update for gReg is available for download. This update fixes a bug that was introduced in the latest build and it only applies to local compiles. It is available for v4.5 for Clarion 5 and 5.5H and v4.51 for Clarion 5, 5.5D and 5.5H

Posted Saturday, April 19, 2003

Brazilian DevCon

This year's Brazilian DevCon will be in the Lizon Hotel in Curitiba from June 19th to June 22nd, 2003. Speakers include Juan Domingo Herrera, Sebastian Tallamoni, Marco Antonio Machado, Matias Flores and others. Among the topics to be covered are: Clarion, Clarion ASP, Clarion PHP, Clarion with Fenix.net, Templates, SQL, API, Data Replication and PostgreSQL. The conference will be held in Curitiba, one of the beatiful cities in Brazil

Posted Saturday, April 19, 2003

Super QuickBooks Export Templates Enhanced

Super QuickBooks-Export creates IIF files that can be imported into virtually all versions of QuickBooks. The templates let you specify were your data is coming from and where it goes, automatically creating the views, ranges, filters and access logic to fetch the data. For more information, download the templates from Mitten Software and run the installation. You can install the documentation without a password. Super QuickBooks-Export is normally $299 US. If you act by April 30, you pay $249 US. The Mitten Software 90 day money-back guarantee applies.

Posted Saturday, April 19, 2003

CWPlus 1.04

CWPlus 1.04 is now available. New features include: CWPlus is now on the Clarion IDE editor toolbar; Ability to format source code with various indent and terminator options; Fixes for known bugs. C6 upgrade is free to registered users of C5/C55 version.

Posted Saturday, April 19, 2003

Beta Testers Needed For faxFUSE

ThinkData Inc. needs additional beta testers for their faxFUSE product to test the WinFax, Microsoft Fax, and RightFax interfaces. faxFUSE is another product in ThinkData's FUSE product line providing native COM automation.

Posted Thursday, April 17, 2003

RPM Upgrade Available

An upgrade for Report and Presentation Manager is now available. Once Clarion 6 goes "gold" all new functionality will be migrated into a new 5.5 release of RPM. Once C6 goes "gold" the price will increase. The upgrade includes both the C6 version as well as the upcoming C5.5 version.

Posted Thursday, April 17, 2003

Clarion Jobs Page Updated

The Clarion Jobs page has just been updated. No new jobs but some positions are still available.

Posted Thursday, April 17, 2003

Ingasoft T-Shirts, Mugs And More

Now you can buy Ingasoft-branded apparel, housewares, hats, auto goods, cards, bags, prints and more.

Posted Thursday, April 17, 2003

Easy3DStyle 1.04

Changes in Easy3DStyle 1.04: Added new Image Check control; New window events and embed point for Image Check control; Support for Select button; Fixed bug with built-in cursors or cursors in variables.

Posted Thursday, April 17, 2003

EasyListPrint 1.01

Changes in EasyListPrint 1.01: Added code page for RTF report; Added support for colors both for Standard and RTF report; Added localization support - you can translate or write your own title, prompts, text and messages for dialog and progress windows. EasyListPrint (ELP) class and template is a set of classes and templates for automatic creation of tabulated reports, using queues or lists (for example, automatic printing of BrowseBox contents). Creates both standard WMF reports, and editable RTF reports. You may customize Title, Header, Footer, Total sections (fonts, alignments etc), number of pages. Widths of columns will be automatically calculated during printing in case they obviously are not defined in the settings. Requires Clarion 5.0b or Clarion 5.5, ABC or Legacy, 32 bit.

Posted Thursday, April 17, 2003

TimeSavers Scheduler Beta

A new TimeSavers scheduler beta has been posted. The templates have been designed in a more logical manner, the help has been updated to HTML help, and there are several new features, including the ability to group schedule grids onto one window. Each is connected so you can drag & drop appointments from one day to another. Each displays a separate day of the week. Also, you can now break out a recurring appointment, so that if you have lunch scheduled for noon every day of the week, and decide on Wednesday to schedule lunch for 1:00, only the one Wednesday appointment is affected. This release only has a C5.5 ABC demo included. The final release will have a demo for Clarion 5 and C5.5 (and maybe even 6.0), both ABC and Legacy. The Help is 95% done.

Posted Thursday, April 17, 2003

Win A Three Month Newsletter Service Subscription

Castle Computer is having a contest for a free three month subscription to their newsletter service. The next winner will be picked on Friday, April 18. Only one winner per weekly drawing.

Posted Thursday, April 17, 2003

Potentially Cheap MSDN Universal

Mark Riffey points out the return of a possibly cheaper way to get MSDN Universal.

Posted Thursday, April 17, 2003

Easy3DStyle 1.03

Easy3DStyle ver 1.03 is now available. Changes in this release: Added highlighting of the active (selected) Tab by text color and styles and background color; Added assigning cursor for button controls; Fixed GPF with Controls on MODAL window.

Posted Thursday, April 17, 2003

EMS PostgreSQL Manager 1.5 Released

EMS HiTech has released PostgreSQL Manager 1.5 - this version, like the current EMS MySQL Manager release, sports an MDI interface (the old SDI interface is still available).

Posted Thursday, April 17, 2003

RDRAW Diagramming Template Demo

A new Demo of the RDRAW template set for creating Visio- or Smartdraw-style flowcharts in Clarion has been uploaded. This demo displays the drawing capabilities of the templates (manually in the demo- they can also be done under program control or from data in a database).

Posted Thursday, April 17, 2003

File Manager 3 Beta 16

File Manager 3 continues moving through the beta stages. Currently supporting the Microsoft SQL, Oracle, and ODBC drivers (ODBC limited to Microsoft SQL, Oracle and MySQL backends). File Manager 3 manages and upgrades the file structures of many of Clarion's supported drivers including SQL drivers. Price is $149.

Posted Thursday, April 17, 2003

CapeSoft Mailer v1.9m

CS Mailer is CapeSoft's bulk email distribution tool, which has undergone improvements and tweaking over the past months.

Posted Thursday, April 17, 2003

Replicate v1.0 Beta 14

Replicate provides an automatic, driver independent, file-version independent, mechanism for replicating the data in two or more databases. Basically, Replicate logs your changes, adds and deletes and then using a transport manager of your choice, exports the changes to the other sites, where the changes, adds and deletes are imported to that data set. This all done completely automatically. Replicate supports both offline and online environments. Currently still in beta, Replicate is moving towards Gold status. Price is $349.

Posted Thursday, April 17, 2003

Office Inside v1.0 Beta 4c

Office Inside wraps the MS Office COM objects, offering a stable approach to integrating and automating MS Office functionality from within your applications. Beta 4 has a fairly extensive library wrapping MS Word, and has the beginnings of Excel, PowerPoint and Outlook are taking shape. Price is $199.

Posted Thursday, April 17, 2003

File Explorer v2.5d

File Explorer is currently back in beta pending a version 3.0 release. The WebBrowser object and underlying engine have been rewritten from the ground up, opening up new functional possibilities including moving the Html Editing control away from the Clarion OLE dependencies to a point where it now uses only pure (stable) COM, implementing the Plugware engine. Lots of new methods, new examples. Price is $99.

Posted Thursday, April 17, 2003

NetTalk v2.73

NetTalk features include: SMTP / POP3 (Email), NNTP (News), HTTP (Web) and FTP. Dial-Up-Networking. NetTalk Protocol, for Clarion-to-Clarion communications. 150+ pages of documentation, 10 Jump-Starts and 20 Examples. Price is $299. Changes in version 2.73 include: NetSimple objects now support asynchronous opens and automatic timeouts; NetRefresh - improved refresh algorithm; FTP speed improvements; NetDIP object tweaked.

Posted Thursday, April 17, 2003

CapeSoft Email Server 1.51

CapeSoft Email Server, which has been built using CapeSoft's NetTalk accessory, has the following functionality: A fully working SMTP server (email sending) and POP3 server (email collecting); Dial-up (modem) features including a schedule; Direct MX SMTP (Email Sending) for permanent Internet connections; An easy to use interface and a configuration wizard which makes setting up an Email Server easy; One price for as many mailboxes (users) as you like; Free product upgrades (buy once, but upgrade as often as you like); Works with all email clients (e.g. Outlook, Outlook Express, Netscape email client, Eudora). Price is $30, 60 day trial available.

Posted Thursday, April 17, 2003

RInstall FAQ

A FAQ for the RInstall template set from Riebens Systems is now available.

Posted Thursday, April 17, 2003

EasyResizeAndSplit 1.07

EasyResizeAndSplit 1.07 is now available. This release adds complete support (for C55) of RTFControl Template.

Posted Thursday, April 17, 2003

ImageEx 2 Competitive Upgrade

Upgrade any third party graphics or imaging toolkit to ImageEx 2.0 at a reduced

price of US$ 139. This includes Clarion products (for example, Capesoft Draw, Image-XChange, Imaging Templates, ...) as well as other DLL or ActiveX products (LeadTools, ImageEn, etc.). Proof of purchase is required.
Posted Thursday, April 17, 2003

PD Finance Library And Calculators Version 60-01

ProDomus has released PD Finance and Financial Calculators Version 60-01. The Financial Library, originally released in 1994, is now included as an all source code class library. Six new customizable calculators have been added: Financial Calculator (PV, FV, Rate, Payment, Periods); Amortization Calculator (Year or Period displays with/without balloon payments); Days Duration Calculator (360 Day Calendar Year); Discounted Cash Flow Calculator (IRR and NPV for variable cash flows); Interest Calculator (APR, Simple, Compound, and Continuous Interest); Simple (Add, subtract, divide … usable with numeric keypad). Calculator windows are declared in a translation file that may be easily customized. A translation method makes run time translation easy using the ABC or ProDomus translator libraries. Available for C5, C55, and C6. Supports both ABC and Clarion template chains. Price: $149. Upgrade from DLL version pdFin for $100 (reflects the release of source code).
Posted Monday, April 07, 2003

C55 PD Browse Button Lookup Updated

A upgrade of C55 PD Browse Button Lookup (Version 55-07) has been posted to the ProDomus web site. This has minor template changes to better handle differences between legacy and ABC applications.
Posted Monday, April 07, 2003

Clarion Training In South Africa

Incasu (Pty) Ltd will start presenting Clarion-related courses from May 2003 onwards in the Pretoria area. You can also call 012 644 0400 for more information.
Posted Monday, April 07, 2003

Join The Clarion World

Ron Schofield has set up a web page that allows you to put yourself on the map. When you join, another dot is added to the map.
Posted Monday, April 07, 2003

Clarion Third Party Profile Exchange Updated

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, Dated Price Quote, Grouped by Category, Extended Description and Download Page Reference. Currently, there are 439 product profiles and 363 vendor profiles. You must have Product Scope 32 PRO Version 4.5 or 4.5a to view profiles with data files (downloadable profiles).

Posted Monday, April 07, 2003

RADIntelliSense Trial Version

A trial version of RADIntelliSense (RADIS) is now available. This version has two limitations: It will stop working after two weeks; It automatically stops after 30 minutes per session. RADIntelliSense (RADIS) gives you rapid access to the Clarion commands, dictionary, local and global data, classes and windows control while your writing your code. RADIntelliSense is not restricted to the Clarion editor but can be used from any text editing program of your choosing. Restructure your code with by pressing the hotkey of your choice, and add mousewheel functionality to the Clarion editor. Record macros, learns new words and easily insert them into your code. And freely configure your RADIntelliSense sources, which you link to a specific application. The RADIntelliSense works on Windows NT 4.0, Windows 2000 and Windows XP and it isn't limited to any version of Clarion for Windows.

Posted Monday, April 07, 2003

INN Bio For April 1, 2003

A father of four human children and one new software child, this Clarionite is a busy daddy. Also honest - he admits that his kids drive him crazy (but would that be the human kids or software ones?;) He's lived on both coasts and places between, been paid to be a parrot head, and was the author of CompuTax (which makes him a tax man, doesn't it?). Don't miss the photo at the end, it's a stunning shot.

Posted Monday, April 07, 2003

CWPlus 1.03

CWPlus ver 1.03 is now available. Fixed: Working in the "Property List" (F12) in the "Window formatter" and "Report formatter". Error "variable not exist" after deleting of the Global template. Changed: Working with the pools of the data is improved - in case of concurrence of file prefixes and other data they are carried in different pools of PopBox, it is possible to choose a pool by Alt+LeftKey or Alt+RightKey or by mouse. Added: Tab "General" - a new setting "Max. record for PopBox" - it is possible now specify an amount of records which will be displayed in PopBox; Tab "Extra" - a new setting "Don't

process specified files" to exclude files from processing by CWPlus.
Posted Monday, April 07, 2003

EasyHelper 1.00
EasyHelper is a class and templates which let you add context help to your application. Features include: Add What's this? button to the upper-right corner of the window and right-click popup context menu for calling context help to any control; Implement standard MS Windows context help called from WinHelp - context help topics stored in .HLP file, HTMLHelp - context help topics stored in .CHM file, or LazyHelp - you don't need any external help file at all; Use your favorite HAT (Help Authoring Tools) to create context help file or automatically generate help projects and receive help file without any extra HAT; Call context help from the application help file or use different help files for calling different types of help (use .CHM as application's main help file and .HLP - as a context help file, for example); Change the font and colour of context help popup windows in runtime; Add What's this? context menu to the standard browse-box menus
Posted Monday, April 07, 2003

cpTracker Gold 50% Off Sale
cpTracker Gold is now available and includes complete F1 context sensitive help. Sale price of $65 is good until midnight, April 6 2003.
Posted Tuesday, April 01, 2003

Reader Comments

Add a comment

Serial Communications Lib 1.1 has new URL now! The new...

Clarion Magazine

Who Calls Who - Keeping Track Of DLL Calling Order

by Steffen Rasmussen

Published 2003-04-17

Recently I took on a rather large Clarion project with an application consisting of 31 DLLs. At first it seemed manageable, but since I have never worked with DLLs before I had quite a bit to catch up on. After getting through the initial phase of understanding this new way of programming, I felt confident enough to start the new assignment. As with most other projects I have seen and worked on, this one lacked documentation, and if there was some it was most probably not up to date. So only thing to do was to start from scratch, make my own notes and figure out the overall structure of this project.

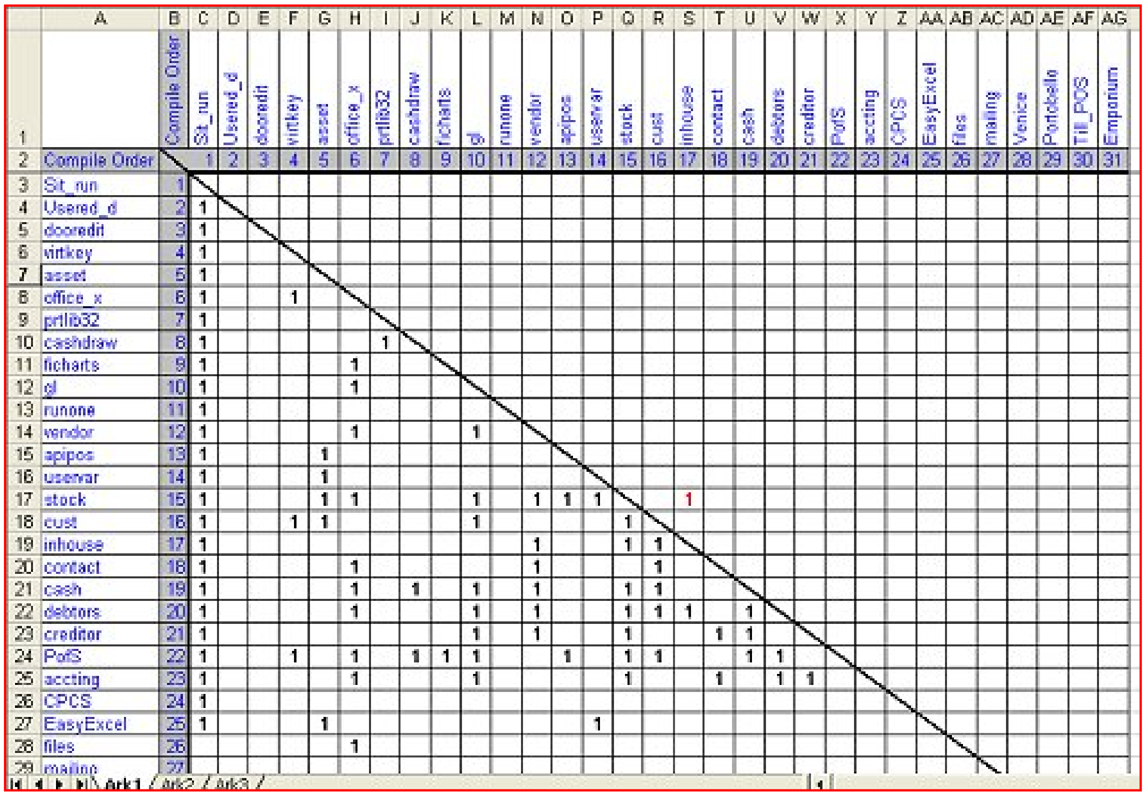
Most of my work usually starts by scribbling notes down on a piece of paper as I progress through the code. From the documentation I had received I knew the name of each DLL, its purpose and the compile order.

I started out by making a clean compile of all the applications that generated the programs DLLs. By clean compile I mean that the directory in which the application resides consist of nothing else but the .APP files. Half way through compiling all the applications I got an error: a DLL was not found. Oh well, that just meant the compile order was not correct. I corrected it and tried a new clean compile, but now there was another DLL missing! Something was wrong, and finding the error by trial and error would take an eternity. Just compiling all the 31 applications that the program consisted of took nearly 3 hours!

I needed to know who was calling who among the DLLs. This is really boring work, going through hundreds of procedures and scribbling down which applications they call. For the same reason this is also very prone to errors; you might easily miss a procedure or two.

With 31 applications a data flow diagram was out of the question since there was no way they could all fit on one piece of paper, and grouping them into logical units where each unit could fit one peace of paper didn’t seem possible in this case.

Creating a matrix was a much better solution. For this I use an excel spreadsheet where the first column has all the applications listed in compile order, and likewise for the first row. The second row and the second column contain the compile order, 1 to 31 in this case. From the cell B2 I draw a diagonal line going through C3 to AG33, which is the last column and row of the last compiled application. Each cell has a 1 if the Y axis DLL calls the X axis DLL. As long as the DLL calls stay under the line, the compile will proceed without problems.



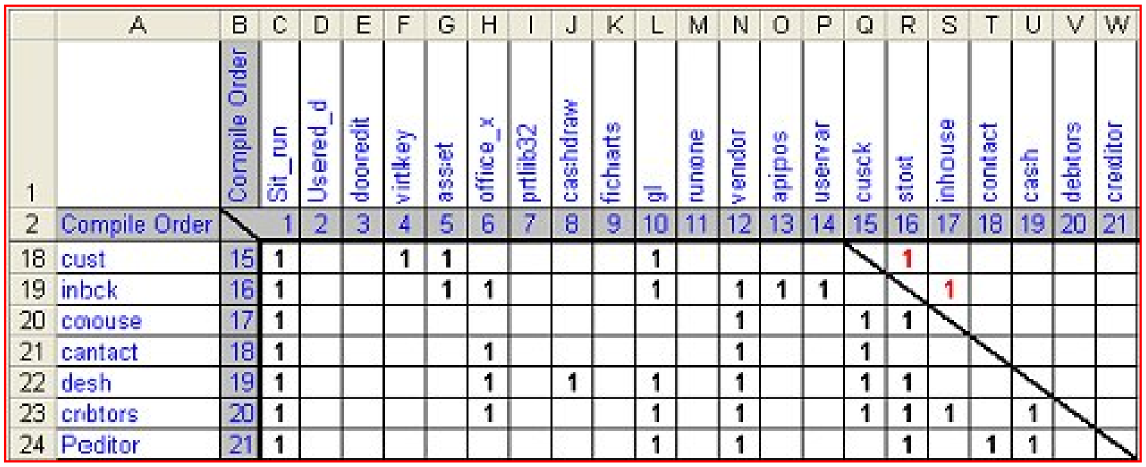
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
1		Compile Order	Sit_run	Usered_d	dooredit	virtkey	asset	office_x	prtlib32	cashdraw	ficharts	gl	runone	vendor	apipos	uservar	stock	cust	inhouse	contact	cash	debtors	creditor	PofS	accting	CPCS	EasyExcel	files	mailing	Venice	Portobello	Till_POS	Emporium
2	Compile Order		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
3	Sit_run	1																															
4	Usered_d	2	1																														
5	dooredit	3	1																														
6	virtkey	4	1																														
7	asset	5	1																														
8	office_x	6	1			1																											
9	prtlib32	7	1																														
10	cashdraw	8	1						1																								
11	ficharts	9	1					1																									
12	gl	10	1					1																									
13	runone	11	1																														
14	vendor	12	1					1				1																					
15	apipos	13	1				1																										
16	uservar	14	1				1																										
17	stock	15	1				1	1				1		1	1	1			1														
18	cust	16	1			1	1					1					1																
19	inhouse	17	1											1			1	1															
20	contact	18	1					1						1				1															
21	cash	19	1					1		1		1		1			1	1															
22	debtors	20	1					1				1		1			1	1	1		1												
23	creditor	21	1									1		1			1			1	1												
24	PofS	22	1			1		1		1	1	1			1		1	1			1	1											
25	accting	23	1					1				1					1			1		1	1										
26	CPCS	24	1																														
27	EasyExcel	25	1				1									1																	
28	files	26						1																									
29	mailing	27																															

Ark1 / Ark2 / Ark3 /

Figure 1. DLL Matrix (click here for the full sized image)

You can see a compile error in cell S17. Here the stock application is being compiled as number 15 and among the applications it calls is the inhouse application that is first compiled as number 17.

I know that in order to call another application that application has to be compiled prior to the one being compiled. If I let the stock applications swap compile order with the cust application I’ll just make things even worse, because now I have problem with both the cust and stock application (see Figure 2).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		Compile Order	Sit_run	Usered_d	dooredit	virtkey	asset	office_x	prtlib32	cashdraw	fichiarts	gl	runone	vendor	apipos	uservar	cusck	stost	inhouse	contact	cash	debitors	creditor
2	Compile Order		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
18	cust	15	1			1	1					1						1					
19	inbck	16	1				1	1				1		1	1	1			1				
20	conouse	17	1											1			1	1					
21	cantact	18	1					1						1			1						
22	desh	19	1					1		1		1		1			1	1					
23	cnbtors	20	1					1				1		1			1	1	1		1		
24	Peditor	21	1									1		1				1		1	1		

Figure 2. Swapping compile order

In a matrix layout as described above you can see how and if it is possible to change the compile order of your DLLs without physically moving them around by just following three basic rules.

Rule Number One: DLL calls:

> *The DLL's row number and column number sets the boundaries for calls to that DLL.*

If you look at the Vendor.DLL in the matrix, its boundaries are from C14 to N14, and within these boundaries it makes all its different DLL calls. Rule number one is graphically illustrated in the diagonal line in figure 1 which splits the spreadsheet into two sections. Section 1 is the primary sections and any marking within this section illustrates from where the application calls has been made (left column) and what application is being called (first row). It also tells you that every call within the first section is a legal application call since the call stays within the border as stated in rule one. Now the second section is exactly the same as the first section except for one thing and that is the application call. If an application call falls into the second section it is illegal because here you are outside rule number one's borders. Remember, when you compile the application any call to another application is linked into the application being compiled. If the application calls a DLL, which has not been compiled, it will come up with an error because the linker can't find the DLL's LIB, which has not been created - yet.

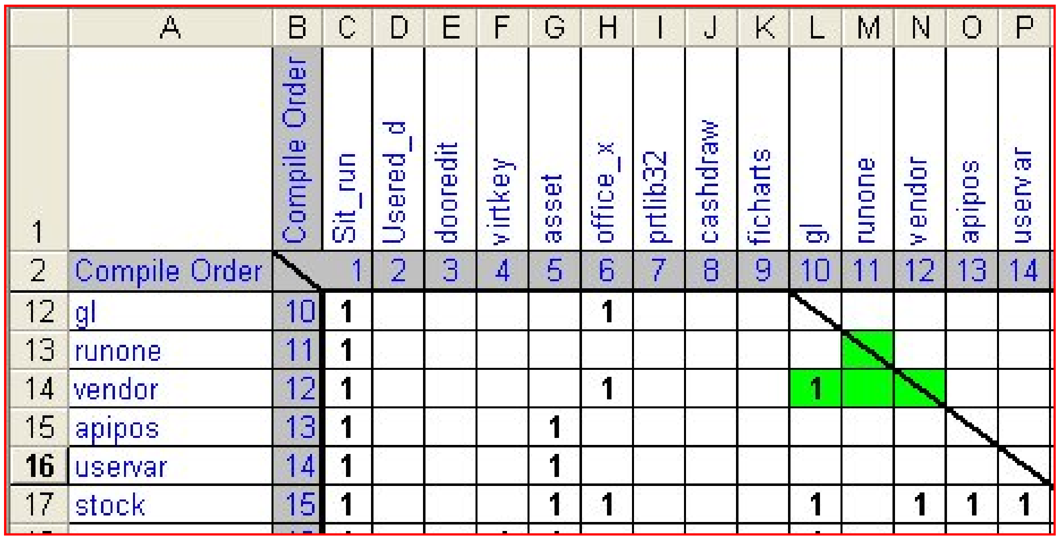
If you take a look at the problem which I ran into (see Figure 1) you will notice that within the stock application a call is made to the InHouse.DLL. The ladder, marked with red, falls outside the boundary set by rule number one. When the boundary is exceeded by the application there are only two possibilities to solve the conflict:

1. Change the compile order by moving the applications compile order down.
2. Redesign the procedure calls within the application so that they don’t call the application that violates rule number one.

Rule Number Two: Moving the DLL compile order up:

The number of free cells prior to the DLL, in the same row, determines how many rows up the matrix the DLL can be moved.

For example the Vendor.DLL in N14 has one free cell prior to it (in M14) because L14 is occupied. This means that the Vendor.DLL can be moved up one row to M13 since it does not call the RunOne.DLL.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		Compile Order	Sit_run	Usered_d	dooredit	virtkey	asset	office_x	prtlib32	cashdraw	ficharts	gl	runone	vendor	apipos	uservar
2	Compile Order		1	2	3	4	5	6	7	8	9	10	11	12	13	14
12	gl	10	1					1								
13	runone	11	1										🟩			
14	vendor	12	1					1				1	🟩	🟩		
15	apipos	13	1				1									
16	uservar	14	1				1									
17	stock	15	1				1	1				1		1	1	1

Figure 3. Moving compile order up

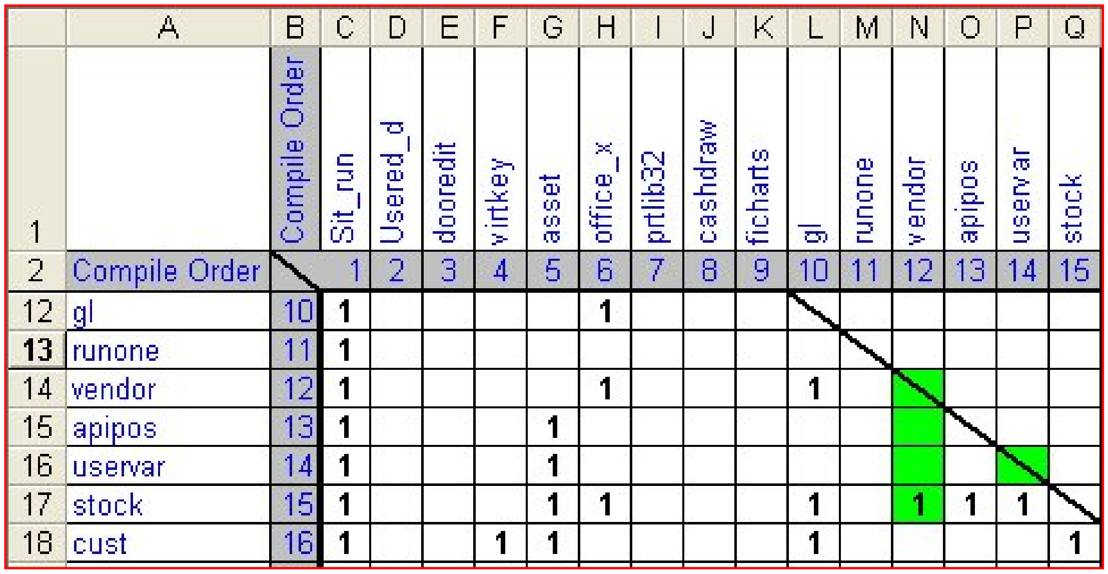
If you look at my particularly problem with the Stock.DLL in Q17 you can see that P17 is occupied and therefore it cannot be moved up. In this case moving up is not an option anyway since the Stock.DLL violates rule number one, I want to move the compile order down, and hence we come to rule number three.

Rule Number Three: Moving the DLL compile order down:

The number of free cells underneath in the same column determines how many rows down the matrix the DLL can be moved.

As you can see rule number three is just the opposite of rule number two. For example, if you take the Vendor.DLL again in N14, it has two free cells underneath it in N15 and N16. N17 is occupied which means that the

Vendor.DLL can only be moved down two rows to P16.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Compile Order	Sit_run	Usered_d	dooredit	virtkey	asset	office_x	prtlib32	cashdraw	ficharts	gl	runone	vendor	apipos	uservar	stock
2	Compile Order		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	gl	10	1					1									
13	runone	11	1														
14	vendor	12	1					1				1		🟩			
15	apipos	13	1				1							🟩			
16	uservar	14	1				1							🟩		🟩	
17	stock	15	1				1	1				1		1	1	1	
18	cust	16	1			1	1					1					1

Figure 4. Moving compile order down

If you go back to my problem in the Stock.DLL in Q17, you will notice that Q18 is occupied, which violates rule number three. If both rule number one and rule number three are violated the only solution to the problem is to redesign the procedure call's within the application so that they don't call the application that violates rule number one

As you can see it is a simple solution to a problem which can be rather complicated and time consuming to solve. The downside of this solution is that the matrix has to be kept updated. Both you and I know that this is unlikely so the perfect solution would be to create a program that could auto generate the matrix at any given time should the need arise.

Steffen S. Rasmussen has graduated in Computer Science from Copenhagen Business College. Since then he has worked as a programmer, system technician and network administrator, and is currently IT manager. Clarion is a quite a new language to Steffen since his only been working with it since January 2000. But what better way to learn it than by trying to teach others! Steffen has also set up a web site to collect as many examples of different user interfaces as possible to inspire Clarion developers.

Reader Comments

Add a comment

In an ideal world there are no circular calls between...
Build your libs from the exp files first, then the compile...
This came up before publication - while it's true that you...
I agree with you Jim that circular calls are not a...

Clarion Magazine

Data Structures and Algorithms Part XVIII - Networks & Graphs

by Alison Neal

Published 2003-04-23

In the next group of articles I will be discussing some new and exciting Abstract Data Types (ADTs), namely Graphs and Networks. The thing that is so exciting about Graphs and Networks is that they open the door to arbitrary relationships, and provide simple representations and solutions to some very complex problems. First things being first, this initial article will cover the theory of what Networks and Graphs are, what they are used for, and some of the associated terminology. After that I will describe Network and Graph implementations and a few of the extremely useful algorithms that have been developed for them.

Networks and Graphs can be used to solve a myriad of complex problems. This includes several amusing puzzles that have been solved quite simply using graphical representations, e.g. the Seven Konigsberg Bridges. The town of Konigsberg (now Kaliningrad in Russia) lies on both sides of the river Pregel, at a point where there are two islands. There are several bridges as shown in Figure 1. (N and S are north and south of the river, I and J are the islands). The problem is to cross each bridge exactly once and return to the starting point, starting wherever you like.

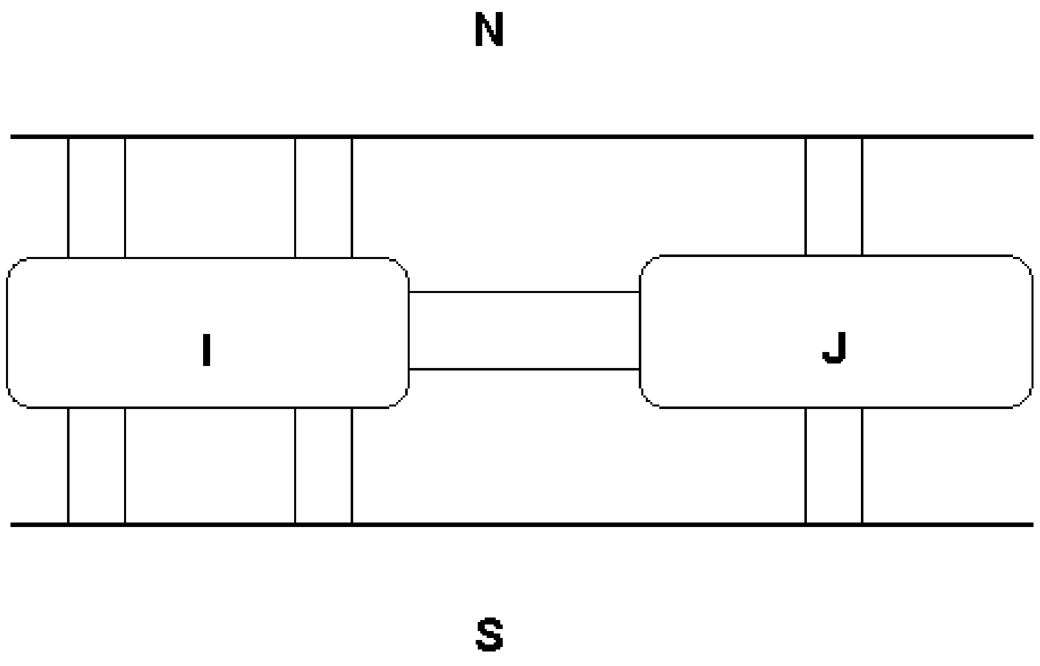


Figure 1. The Seven Konigsberg Bridges

How would you go about solving the problem? Where would you start programming for a problem like this? You'd use a Network or a Graph, of course.

Other more commercial problems that a Graph or Network can be applied to include:

- Project Planning – developing critical paths, timelines, budgets etc.
- Communications networks – identifying critical network points, routing etc.
- Transport – calculating distances, routes, costs, durations, planning etc.
- Road Planning – identifying critical network points, routing etc.

The potential uses for Networks and Graphs are many, but before you can use such a powerful tool, you need to understand it.

A Graph is defined as having a set of vertices, sometimes called points or nodes, and a set of edges, sometimes called arcs, which are lines which join one vertex (node) to another. Figure 2 provides an illustration of a simple Graph.

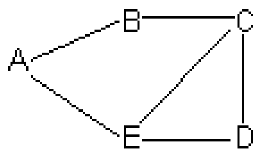


Figure 2. A simple Graph

The edges of a simple Graph are not directed; they can be traversed in either direction. A Digraph (Figure 3) is a variation on a Graph – its edges can only be traversed in the direction of the arrow.

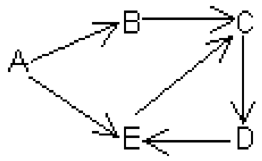


Figure 3. A Digraph

Each edge or arc can also have a *weight* associated with it, for example if each node represents a city on a map then the weight of the edge may represent the distance, the cost, or the duration of the trip. Another example is a construction project, where each node represents a task to be performed such as laying the foundations, or doing the wiring, or digging trenches; the weight of the edges could represent the time it takes for each task to be completed.

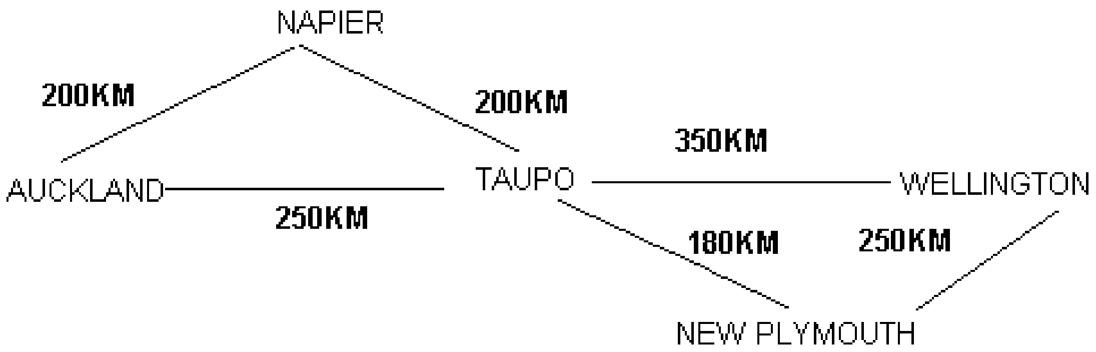


Figure 4. A Weighted Graph

Figure 4 represents a Weighted Graph. If Figure 4 were directionally oriented (had arrows) then it would be a Weighted Digraph. If all the edges have a weight then the structure also becomes a *Network.*

The one thing that you may be starting to realize about graphs is their lack of structure. There is no definitive starting point, and the relationships between nodes are arbitrary. Lists, Queues, Stacks all have linear relationships. You may remember from my previous articles that one of the key characteristics of the Tree is its hierarchical structure; every node in a tree has one parent (except for the root node). The relationships in a tree are also substantially different from those of a graph, as they are expressed by reference from parent to child; in a Graph or Network, this type of one-way relationship is often not sufficient.

Consider club and association memberships, where one person can belong to more than one association or club and each association has more than one member. This is an example of a many to many relationship. This type of relationship is best served by a data structure that allows a non-hierarchical bi-directional relationship between the associations and the members. If it were to be done using trees you would need to use two to represent both relationships.

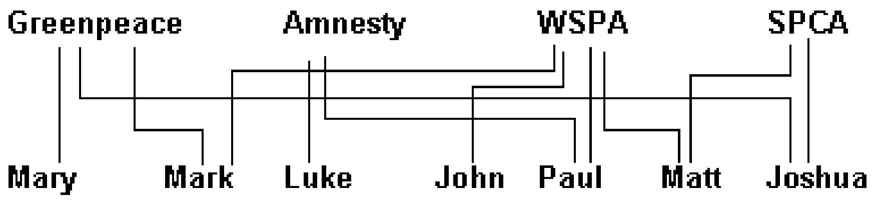


Figure 5. A many to many relationship

In Figure 5 the graph nodes are the people and the associations, and the graph edges are the lines linking each node. The definition of a graph does not rule out the possibility of an edge connecting two associations or two people together.

The nodes that are linked by an edge are known as the *endpoints* of the edge. I would say that the *endpoints* are *directly connected* by the edge. The two end points do not have to be distinct nodes. Where both endpoints of an edge are the same node, it is called a *loop* (refer Figure 6).

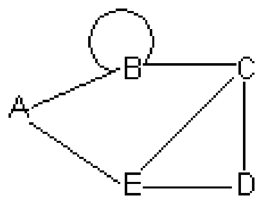


Figure 6. A Graph with a loop edge

Note the loop edge that traverses from end point B to B, or edge BB. The edge that runs from A to B, is edge AB. If Figure 6 was directionally oriented (Digraph), with an arrow from node A to node B then edge AB would be called an *arc* and it would be notated as A->B.

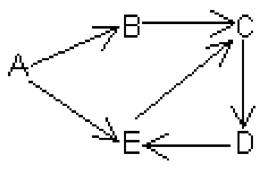


Figure 7. A Digraph

In Figure 7, node A is *directly connected* to B but *indirectly connected* to C via the arcs A->B, B->C, and it has a path of length of 2.

A Digraph is known as *strongly connected* if for any nodes I and J there is a directed path from I->J and J->I. The Digraph is called *weakly connected* if there is a directed path from I->J or J->I.

Outdegree is the number of edges issuing from any node, so in Figure 7, the outdegree of A is 2 and the outdegree of E is 1. The *Indegree* is the opposite, so in Figure 7, A has an indegree of zero, and E has an indegree of 2. The indegree and outdegree of any given node can be useful for indicating the importance of particular nodes. A Node with an outdegree of zero is called a *sink node,* whereas a node with an indegree of zero is known as a *source node*.

A *cycle* is a directed path of length one or greater that originates and terminates at the same node in the graph. The example in Figure 7 is C->D,D->E,E->C, which has a path length of 3. An *Acyclic Graph* has no cycles in it.

With simple Graphs or non-directed Graphs the indegree and outdegree cannot be applied, but the *degree* of a node provides the number of direct connections, so in Figure 6, A has a degree of 2 and B has a degree of 4, as the loop constitutes 2 connections.

You can even use this terminology to describe other, more common structures. For instance, a Linked List is a Digraph with one source, one sink, and all the other nodes having an indegree and an outdegree of 1.

Summary

Hopefully you will already be able to see the potential power of the Graph for solving complex problems. With its non-hierarchical, bi-directional, arbitrary structure, the Graph has the potential to represent an endless variety of data, and solve an infinite number of problems.

When I studied computer science I found Graphs to be the most exciting subject covered on the entire course, but of course it was also the most complex. In my next article I'll write some Graph code that shouldn't be too difficult to follow.

Alison Neal has been using Clarion since 2000, whilst working for Asset Information Systems (AIS) in Auckland, New Zealand. Some years ago (at the tender age of 19) Alison graduated from the Central Institute of Technology in Wellington, New Zealand with a major in Cobol. She also has a BA in English literature and has studied Computer Science, Philosophy and Information Systems. AIS is an independent division of Asset Forestry Ltd, and has a team of five programmers developing almost exclusively in Clarion. AIS also offers web (ClarioNET) and email services for the customer who needs everything. The company has many and varied customers bridging across a wide range of industries including Telecommunications, Forestry & Agriculture, Manufacturers, Military & Government, Legal & Financial, and Retail.

Reader Comments

Add a comment

Data Structures and Algorithms Part XIX - Simple Graphs

by Alison Neal

Published 2003-04-24

In my previous article I discussed the theory behind Graphs. You may have the impression from that article that I think the Graph Abstract Data Type (ADT) is a better idea than sliced bread. I can slice my own bread with a knife, but if you give me a large map and tell me to memorize it and calculate distances, I need a computer and a Graph.

In this article I'm going to introduce a very simple implementation of a Graph. You may remember from last time that a Graph is defined as having a set of vertices, sometimes called points or nodes, and a set of edges, sometimes called arcs, which are lines which join one vertex (node) to another. Figure 1 provides an illustration of a simple Graph.

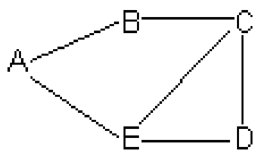


Figure 1. A simple Graph

Any definition of a graph needs definitions of edges and nodes:

```
edgeNode     CLASS,TYPE
eData        LONG  !Secondary Node number
eWeight      LONG
nextE        &edgeNode
        END
graphNode       CLASS,TYPE
```

```
xName       STRING(500)
headE       &edgeNode
CurrE       &edgeNode
tailE       &edgeNode
nextN       &graphNode
        END
```

The `Edge` structure I am using contains an `eData` variable to contain the `Node` number of the second point on the edge, and a weight. Not all graphs need to be weighted; it depends on what you are using the graph for. The weight of an edge can represent any number of things, such as a duration, a distance, a monetary value, or the number of red lights you run going from point A to point B. What you may also note about the `Edge` structure I have coded is that it is fundamentally a Linked List, since `edgeNode` can contain a reference (`nextE`) to another `edgeNode`, which can contain a reference to another `edgeNode`, and so on and so on. This does *not* mean, however, that a this linked list represents a path through the Graph – rather it's a way of keeping a list of edges associated with a particular node. And speaking of nodes…

The `graphNode` structure has three edge references, one for the first edge in the list, one for the last, and one for the current item. These references make navigation through the edge list possible. A list is a good option where you are not sure how many edges a given node may have connecting to it.

The Graph Nodes are also a linked list, which means that the `Graph` class (which I'll get to in a moment) can traverse the list of nodes in exactly the same way as each `graphNode` can traverse its list of edges.. Each Graph Node also has a name, which can be anything, a place name, a task name, a persons name etc. etc.

The methods I have included in my `Graph` class are:

`Init`	Initialize the Graph
`Kill`	Clean up
`AddNode`	Add a Vector, point or node to the graph
`AddEdge`	Add an Edge to the Graph (between two specified nodes)
`NumNode`	Return the number of nodes in the Graph

`NodeNum`	Return the node number for a given node name
`NodeName`	Return the node name for a given node number
`First Node`	Make the first node in the graph the current node
`Next Node`	Make the next node in the graph the current node
`First Edge`	Make the first edge for a given node the current edge for that node.
`Next Edge`	Make the next edge for a given node the current edge for that node

The following is the test file that I have used for the example application, which is downloadable from the bottom of this page (note that these are place names and the numbers are arbitrary).

Node1	**Node2**	**Weight**
Wellington	Auckland	500
Auckland	Napier	350
Napier	Gisborne	100
Gisborne	Whakatane	150
Gisborne	Taupo	200
Napier	Taupo	100
Rotorua	Taupo	50
Rotorua	Napier	50
Wellington	Taupo	300
Taupo	Auckland	200

Each record represents an Edge, so for each unique place name I want to create a Node, and for each record I want to create an Edge joining the Nodes.

In my class I've used two init methods:

```
graph.Init                  PROCEDURE()
  CODE
  SELF.HeadN &= NULL
  SELF.CurrN &= NULL
  SELF.TailN &= NULL
  SELF.NumNodes = 0

graph.Init      PROCEDURE(*STRING Node1, *STRING Node2, LONG Edge)
i   LONG(0)
j   LONG(0)
  CODE

  i = SELF.nodeNum(Node1)
  IF i < 0 THEN i = SELF.addNode(Node1).
  j = SELF.nodeNum(Node2)
  IF j < 0 THEN  j = SELF.addNode(Node2).
  SELF.addEdge(i,j,Edge)
```

The first `Init` method is as it should be, initialising the reference variables to `NULL` and the node count to 0. The second one just allows me to insert the data in accord with my import file (my quirk). If you really wanted you could pass the filename and have your `init` method do the lot, reading the entire file and inserting the data. My second `init` method is more of an `Add` method, but I already had several of those! The use of duplicate method names can be confusing at times, and the naming convention used is entirely your choice.

Here's the code I use to load up the graph:

```
SET(ImportFile)
  LOOP
    NEXT(ImportFile)
    IF ERRORCODE() THEN BREAK.
    G.Init(IMP:Node1,IMP:Node2,IMP:Edge)
  END
```

Take a closer look at the second init method. Given the test file above, the first call will pass `Node1` as Wellington, `Node2` as Auckland, and an `Edge` value of 500.

A call is then made to the `nodeNum()` method:

```
graph.nodeNum                  PROCEDURE(*STRING n)
i   LONG(0)
t   &graphNode
  CODE
  t &= SELF.HeadN
  LOOP WHILE ~t &= NULL
    i += 1
    IF t.xname = n THEN RETURN i.
    t &= t.nextN
  END
  RETURN -1
```

The `nodeNum()` method should return the number that the passed node name relates to. In this case Wellington is the first entry so it will return –1. As I is therefore less than 0, a new node will be added for Wellington:

```
graph.addNode                  PROCEDURE(*STRING S)
n   &GraphNode
  CODE

  n &= SELF.HeadN

  LOOP WHILE (~n &= NULL)
    IF SELF.HeadN.xName = S
      MESSAGE('Duplicate Entry')
      RETURN -1
    END
    n &= n.nextN
  END
  SELF.CurrN &= NEW(GraphNode)
? ASSERT(~SELF.CurrN &= NULL)
  SELF.CurrN.nextN &= NULL
  SELF.CurrN.headE &= NULL
  SELF.CurrN.currE &= NULL
  SELF.CurrN.tailE &= NULL
  SELF.CurrN.xName = S
  SELF.numNodes += 1
  N &= SELF.CurrN
  IF SELF.HeadN &= NULL
     SELF.HeadN &= N
     SELF.TailN &= SELF.HeadN
  ELSE
     SELF.TailN.nextN &= N
     SELF.TailN &= SELF.TailN.nextN
  END
  SELF.CurrN &= SELF.tailN
  RETURN SELF.numNodes
```

The first thing that the `addNode()` method does is check (again) to see that the node doesn't already exist. If not, it then creates a new `Node`, sets the node's reference variables to `NULL`, assigns the node name (Wellington) and adds the new node to the tail of the `Node` list. If there is nothing else currently in the list then the new node also becomes the head. The `addNode()` method then returns the node number; in this example Wellington is node number 1.

The `init()` method then follows the same process for Auckland, adding the node to the tail of the node list and keeping the node number. Then it adds the `Edge`:

```
graph.addEdge                  PROCEDURE(LONG i, LONG j,LONG W)
  CODE

  SELF.makeCurrent(i)
  SELF.addEdge(j,W)
```

The first call to the `addEdge(LONG i, LONG j,LONG W)` method passes `i` as 1, `j` as 2 (representing the two new node numbers), and `W` as 500 (the weight of the edge). A call is then made to the `makecurrent()` method to ensure that the first node on the edge is the current node.

```
graph.makeCurrent                  PROCEDURE(LONG p)
  CODE

  SELF.currN &= SELF.HeadN
  LOOP WHILE (~SELF.CurrN &= NULL) AND (p > 1)
    p-=1
    SELF.CurrN &= SELF.CurrN.nextN
  END
? ASSERT(~SELF.CurrN &= NULL)
```

Once the correct `Node` is located then the `addEdge(LONG i, LONG j,LONG W)` method calls the `addEdge(LONG j,LONG W)` method so that the new `Edge` can be added to the `Node` held by the Graph's `SELF.CurrN` reference:

```
graph.addEdge                  PROCEDURE(LONG j,LONG W)
  CODE
  SELF.CurrEdge &= NEW(EdgeNode)
? ASSERT(~SELF.CurrEdge &= NULL)
  SELF.CurrEdge.nextE &= NULL
  SELF.CurrEdge.EData = j
  SELF.CurrEdge.eWeight = W
  IF SELF.CurrN.headE &= NULL
    SELF.CurrN.headE &= SELF.CurrEdge
    SELF.CurrN.tailE &= SELF.CurrEdge
  ELSE
    SELF.CurrN.TailE.NextE &= SELF.CurrEdge
    SELF.CurrN.TailE &= SELF.CurrEdge
  END
  SELF.CurrN.CurrE &= SELF.CurrN.TailE
```

In the `addEdge()` method, a new `Edge` is created and assigned its second node number and weight. The new `Edge` is then assigned to the tail of the current node’s edge list. In the parameters to the `addEdge(LONG i, LONG j,LONG W)` call, the first node number is the node which contains the reference to the edge, and the edge contains the number of the second node. The edge could be made to hold a reference to the node rather than the number, it would speed things up considerably for a larger graph, but holding a reference requires a greater amount of memory so the trade-off between speed and memory needs to be thought out before implementing.

In this manner each and every record is added, and the code creates any new nodes as necessary (but only one per node name).

Summary

Those are the basics of implementing a Graph. It's a very handy tool, and in my next article I will start to discuss some of the algorithms that use the graph, which include Topological Sorts, Critical Path Analysis, General Search Strategies, and (numerous) Path Algorithms.

Download the source

Alison Neal has been using Clarion since 2000, whilst working for Asset Information Systems (AIS) in Auckland, New Zealand. Some years ago (at the tender age of 19) Alison graduated from the Central Institute of Technology in Wellington, New Zealand with a major in Cobol. She also has a BA in English literature and has studied Computer Science, Philosophy and Information Systems. AIS is an independent division of Asset Forestry Ltd, and has a team of five programmers developing almost exclusively in Clarion. AIS also offers web (ClarioNET) and email services for the customer who needs everything. The company has many and varied customers bridging across a wide range of industries including Telecommunications, Forestry & Agriculture, Manufacturers, Military & Government, Legal & Financial, and Retail.

Reader Comments

Add a comment

Clarion Magazine

Data Structures and Algorithms Part XX - Topological Sort

by Alison Neal

Published 2003-04-25

In my last article I discussed how to implement a basic Graph Abstract Data Type (ADT); in this article I'm going to discuss how to do a Topological Sort. The Topological Sort has two purposes: One, to detect whether the Graph has any cycles, and; Two, to provide a list of nodes in topological order, indicating the relationships between nodes.

Figure 1 represents the input file for a development project for which I want to build a graph:

Concept Document	Client Quote
Research Requirements	Concept Document
Client Quote	Requirements Document
Requirements Document	Design Document
Develop Dictionary	Develop Windows
Develop Dictionary	Develop Reports
Design Document	Develop Dictionary
Develop Reports	Test
Develop Windows	Test

Figure 1. Input file for a development project

Assuming that I want a directed graph and that each task (node) in the project relies on the other being completed first, I can traverse the graph and provide an ordered list of which tasks come first. The first task on the list should have an in-degree of zero, which means that there is nothing that needs doing first; this task is not dependent on the completion of any other task. The input file once loaded into the graph would look a little like Figure 2:

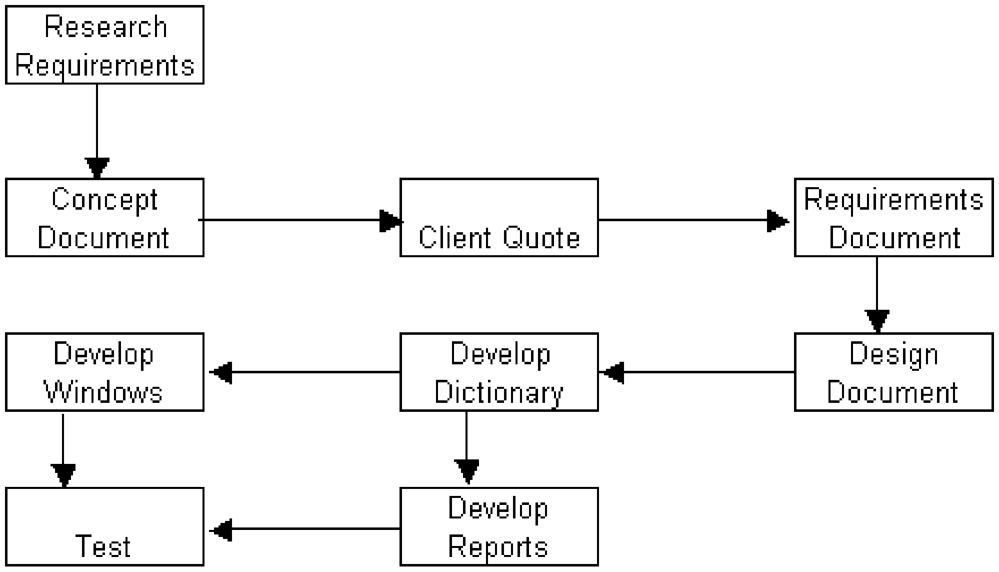


Figure 2. A task flow chart

The list that the Topological sort would then generate would start with *Research Requirements*, as it is the only task with an in-degree of zero, and finish with *Test*. The order *Develop Windows* and *Develop Reports* come out in could vary; however, as they both occur after *Develop Dictionary* and before *test*, the order doesn't really matter. If it did, the graph nodes could be allocated a priority, or a secondary order principle could be applied for the highest out-degree (has the most tasks dependent on its completion).

The other purpose for the Topological Sort is to identify that there are no cycles in the graph. Suppose for example that I added a new edge to the graph depicted in Figure 1 as an arrow from *Test* to *Research Requirements*. This would indicate that the Requirements couldn't be researched until the testing had been completed, and the testing could not start until the Requirements had been researched. Such a cycle would mean there was no starting point for the project, and the project was in jeopardy before it began. It's the sort of thing that project managers have nightmares about.

Project Managers are also very interested in Critical Paths, and the duration of the tasks. As soon as a graph gets task durations it becomes a *Weighted Directed*

Acyclic Graph, or a *Weighted DAG* if you're Australian. I will discuss Critical Path Analysis or Critical Path Method in my next article.

Topological Sort

The quick and dirty way of performing a topological sort, based on the graph implementation presented in my last article, is as follows:

```
TopolSort:Routine           ROUTINE
  DATA
idegree     LONG,DIM(numNodes)
i           LONG(0)
j           LONG(0)
E           LONG(0)
Q           QUEUE
NodeNum     LONG
            END
  CODE
  !Initialize array
  LOOP i = 1 TO numNodes
    idegree[i] = 0
  END

  !Calculate the in-degree value of each node
  LOOP i = 1 TO numNodes
    E = G.firstEdge(i)
    LOOP WHILE E > 0
      idegree[E] += 1
      E = G.nextEdge(i)
    END
  END
  !Add those nodes with an idegree of zero to the queue
  LOOP i = 1 TO numNodes
    IF idegree[i] = 0
      Q.NodeNum = i
      ADD(Q)
?     ASSERT(~ERRORCODE())
    END
  END
  !Add the remaining as the idegree lessens
  LOOP i = 1 TO numNodes
    !Check we don't have cycle
    IF ~RECORDS(Q)
      MESSAGE('Error: Graph contains a cycle'|
        ,'Sytem Error',ICON:Exclamation)
      EXIT
    END
    !Get the first record from our holding Q and delete
    GET(Q,1)
    j = Q.NodeNum
    DELETE(Q)
    !Add the name of that node to the display Q
    DspQ.xName = G.NodeName(j)
    ADD(DspQ)
```

```
?   ASSERT(~ERRORCODE())
    !LOOP through the edges for the current node
    !and decrement the indegree accordingly
    E = G.firstEdge(j)
    LOOP WHILE E > 0
      idegree[E] -= 1
      IF idegree[E] <= 0
        Q.NodeNum = E
        ADD(Q)
?       ASSERT(~ERRORCODE())
      END
      E = G.nextEdge(j)
    END
  END
```

The first thing that the algorithm does is initialize an array to the size of the number of nodes in the graph. Once it has done that, it loops through all the graph node edges and calculates the in-degree value of each node. All nodes (tasks) with an in-degree of zero, having no dependencies, are added to a holding Queue, which I've aptly named `Q`.

Again I loop through the nodes, and check to see if at any stage `Q` is empty (there are no nodes at this point with an in-degree of zero); if so, then there must be a cycle in graph and processing stops. If `Q` is not empty I add the first node in the Queue to my display Queue (`dspQ`), which is where I keep the final result.

As I've already accounted for a particular node, I need to recalculate the in-degree of all the other nodes that that one node is linked to. So for each edge belonging to the node for which I've completed processing, the in-degree of the edges' second node is decremented by one.

If any of these edge nodes now have an in-degree of zero or less I can also add them to my holding Queue. The end result for the example given in figure 1 is:

```
 1. Research Requirements
 2. Concept Document
 3. Client Quote
 4. Requirements Document
 5. Design Document
 6. Develop Dictionary
 7. Develop Windows
 8. Develop Reports
 9. Test
```

Note the order of *Develop Windows* and *Develop Reports* is the order in which the nodes were inserted into the Graph.

Summary

Project managers are not only interested in task orders and dependencies; they require a lot of other information. In my next article I will discuss Critical Path Analysis. Continuing with the development project example, I want to be able to show the project manager the earliest and latest dates that a task will be started and finished, so that contractors can be hired without having the cost of them sitting idle while other tasks are being completed.

Download the source

Alison Neal has been using Clarion since 2000, whilst working for Asset Information Systems (AIS) in Auckland, New Zealand. Some years ago (at the tender age of 19) Alison graduated from the Central Institute of Technology in Wellington, New Zealand with a major in Cobol. She also has a BA in English literature and has studied Computer Science, Philosophy and Information Systems. AIS is an independent division of Asset Forestry Ltd, and has a team of five programmers developing almost exclusively in Clarion. AIS also offers web (ClarioNET) and email services for the customer who needs everything. The company has many and varied customers bridging across a wide range of industries including Telecommunications, Forestry & Agriculture, Manufacturers, Military & Government, Legal & Financial, and Retail.

Reader Comments

Add a comment