How Software Really Gets Written

Adventures in J/JAR programming

John D. Baker

bakerjd99@hotmail.com

June 27, 2003

Real JARs proved their worth aeons ago!

On August 18, 1997 I gave a presentation at the APL97 conference in Toronto Ontario titled: "J Automation a Brain Transplant for Access." I spent a reasonable amount of time preparing the talk and it went fairly well. Still I was disappointed by the response. Technical presentations take sometime to sink in. It's a rare audience that:

1. Really cares about the topic.
2. Is able to quickly comprehend what the the speaker is doing.
3. Is polite enough to give the speaker some useful feedback.

I had four of five people come up after my talk and a few congratulated me over beers afterwards. This was better than the response metered out to many speakers but frankly anything less than an auditorium full of wide eyed fans chanting, "We are unworthy." is not good enough for *moi*.

Despite my minor disappointment I drew solace from the not insignificant number of attendees that upon reading my name tag remarked that they really enjoyed my Internet postings. A large number of APL'ers either subscribe to the `APL-L` list server or monitor the `comp.lang.apl` usenet group. The server and the usenet group trade messages so everyone sees all the messages.

Like most Internet groups `comp.lang.apl` is dominated by a small number of vocal individuals that feel their unique irreplacable views deserve immediate global distribution. I am not a member of this loud minority. I belong to the "keep your mouth shut unless you have something useful to say" faction. On the Internet silence has a higher market value than gold!

Occasionally, when I come across something really interesting, or when someone else posts a message that in Wolfgang Pauli's immortal words, "Isn't even wrong!" I will rev up the word processor and post. I try to write coherent, definitive messages that converge on the point I'm trying to make. I usually post illustrative programs that support my case. The programs are always appreciated.

Apparently more people follow my net diatribes than I expected. Learning this is both good and bad. It's good to find you have an audience but it also cranks up the pressure. A disappointed audience can quickly turn on you. I'll have to really watch my p's and q's and make sure that what I post in the future is worth reading.

All of which brings me to *The ReJoinder*. For sometime I have been toying with the idea of setting up a web site devoted to the J language and my take on it. The ReJoinder would present to the world *edited* literate J code of real systems. The first system will be JDict 97. A throughly updated J dictionary system that is written entirely in J.

I have a feeling that if some people are following my usenet postings I will draw a much large audience with the ReJoinder. Not enough to make any money directly but more than enough to make money indirectly by clearly demonstrating to a concept starved world that I actually have a few ideas.

This log will track the development of first JDict 97 (I'll probably rename it) and then the ReJoinder.

Sep 7 1997

1. **Bitch:** Jdict 97 has survived a moral crisis. All of last week was spent dithering over whether I should continue using ODBC for this project. ODBC or, *(Odious Damm Bucket of Crap)*, as I refer to it has been a constant source of disappointment to me ever since I played with my first ODBC driver three or four years ago.

   To this day ODBC is still more of an idea than an actuality which is to damm bad because the world really needs an ultra-high-performance 100% standard, platform independent, relational database access mechanism. When ODBC was announced to the world I thought it might become such a mechanism.

   Well I was naive, idealistic, filled with optimism. Of course an excellent implementation of ODBC would kill off all other database systems, including Microsoft's, but so what, we must never mourn the demise of an obsolete technology. Alas Microsoft, indeed all software vendors, are more interested in making money than software. ODBC, viewed in this cynical light was never more than a marketing ploy. It was a way of touting database independence but never actually supplying it.

   Compared to native access methods OBDC is:

   (a) Slow

   (b) Buggy

   (c) Idiosyncratic

   (d) Incomplete

   (e) Infuriating

   ODBC may well be the most non-standard standard around. Every driver implements a different subset of SQL. Some drivers just pass the through, others make obscure modifications to some SQL constructs but not others. Finally, no driver hides the "features" of the underlying database. The ACCESS driver will not provide you with lower case in your keys because ACCESS will not turn such a trick.

   The end result is if you want the one benefit ODBC offers, the ability to switch backends, provided you do all the "standardization," you are

going to pay a very high price. I wasn't sure, indeed I still have doubts, whether this benefit is worth the price.

On 166MHZ machines I can tolerate ODBC. The hardware is so fast it overpowers the sluggish, indeed pathetic OBDC driver environment. Machines are only going to get faster and given monsters like ODBC it's a good thing.

2. **Plan:** I've decided to press on using ODBC/ACCESS until I have system that's almost ready for prime time. At that point I will get at least one other ODBC database system, probably something like Sybase Anywhere, or the new RBASE implementation and make whatever changes have to made to insure the client software provides the same services regardless of the backend.

   I suspect I will have to provide a client for every database system I support but that's OK because it reveals to the world the truth about ODBC. It isn't a technical standard, only a marketing one.

3. The J client software will provide:

   (a) `calls` words calling, groups in et cetera

   (b) `copy` copy words, tests, from one dictionary to another

   (c) `del` delete words, tests, et cetera

   (d) `did` dictionary identification

   (e) `doc` create and search documentation

   (f) `dpnl` dictionary name lists from patterns

   (g) `get` get words, tests from dictionary

   (h) `grp` group words, tests

   (i) `refs` word and test name references

   (j) `help` dictionary help

   (k) `make` generate J scripts, test scripts, latex, html, dumps

   (l) `new` create new dictionary

   (m) `notput` checks whether locale words, test, are in dictionary

   (n) `pick` open and close dictionaries

   (o) `put` put words, tests into dictionary

(p) `reg` register/unregister a dictionary

(q) `uses` words used by word, tests

The exact call syntax of these verbs has not been completely worked out yet but I am going to spend. a lot more time making things consistent and symmetric than I did for the first JDict.

Sep 14 1997

1. A week has passed and I have been working at a steady unspectacular pace on JDict 97. I have `put` almost done and I'm hoping to finish the easier `get` by tonight or tomorrow.

2. My "not impressed" opinion of ODBC is still firmly in place and I'm still looking around for either better drivers or something altogether better. In the *better* driver vein I downloaded a trial version of a new MIRCORIM product called Oterro. I had read a very positive review of this product in PCINFO WHATEVER that claimed this product was intended to be used as a replacement for the MS JET engine and was primarily a file server vs. client server engine. But what caught my eye was the assertion that this engine was built from the ground up as an ODBC database. Even MicroSoft, god of ODBC, hasn't taken this step.

   After a few days of Internet searching I found Oterro, (funny that I should forget the name ehhh), read through the online propaganda and liked the warm fuzzies it invoked. *This thing looks promising.* Basically Oterro is the old respected R:BASE database re-engineered as a 32-bit ODBC driver.

   I was just about ready to reach for my wallet when I found, (after a hell of lot of web wandering), what the Oterro folks think their product is worth. **$3,200 US in god we trust dollars!** Oterro may indeed be better than MS Jet but for that price it has fall into the miraculous software category not the merely well done category.

   I still downloaded the trial version and a sample application and played around a bit. Otterro looks pretty good. The demo is certainly fast and unlike Jet Oterro supports some key SQL commands like `CREATE SCHEMA`.

Oterro can do one thing that MS Jet cannot, namely create a fucking database. You can access, sort of, data in extant databases with the MS ODBC drivers but you cannot (with the insignificant exception of the text driver) create a database with any of them. You want to create ACCESS databases, well buddy you have to have the entire bloated disk-hogging ACCESS program. Of course if we could create databases the incentive to buy the disk hogging full system might fall below the, it's more of a pain than it's worth, threshold thereby depriving Bill of some dollars. We can't have that can we. Let's face it industries dominated by a few powerful neo-monopolies prefer to tell the customer what to do rather do what the customers actually want them to do.

Sep 23 1997

1. It's been awhile since my last entry. I spent most of last weekend playing with the demonstration copy of Oterro. I found:

2. Oterro's BLOB support is better than ACCESS via ODBC. I was able to load the binary image of a 1000 by 1000 array into Oterro with no trouble. Oterro loads binary images far slower than J. But it can do it. Score 1 for Oterro.

3. After defining a copy of my Jdict 97 database. I wrote a few verbs tailored to Oterro and loaded up the words table with 1,300 words. Oterro seems to over allocate disk space for memo items. The Oterro database with one table filled was over 5 megabytes. The ACCESS database with complete copies of all eight tables was 4 megabytes. And I thought ACCESS was a disk hog. Score 1 for ACCESS.

4. Character table searches using SQL wild cards on indexed fields is dissappointly slow in Oterro. The time required to match all items on the Oterro table was roughly 5 times slower than ACCESS for the same ODBC query. Oterro may find single keys in large tables faster than ACCESS but for this text pattern intensive application ACCESS is faster. Score 1 for ACCESS.

5. I managed to corrupt my test Oterro database by over INSERT'ing. Try running a few thousand insert's one on top of each other. Most ODBC database's take forever or choke. Doesn't give one the warm fuzzies does it.

6. ACCESS's table attachment is far superior to Oterro and for this application it's the reason I'm sticking with ACCESS. I cleverly bypass the mass INSERT problem with my ACCESS database by having J format a complete tab-delimited table that is attached to the ACCESS database. Having J write the table directly is 10 to 20 times faster than ODBC and FAR MORE RELIABLE. This neat hack cannot be used in the Oterro environment. If they extend the table attachments to tab or comma delimited types I will look at Oterro again. For now I cannot justify forking over $300 US (they came to their senses price wise-pretty quickly). I would go with Oterro if it sold at $150. At that price it would be cheaper than ACCESS and I could live with a mild slow down for my peculiar application.

7. All-in-all this wasn't a waste. I resolved some cross-SQL issues with these tests. One of my ACCESS table names clashed with an Oterro reserved word. So I changed all my database table and field names to begin with the "jj" prefix. If this clashes with any SQL reserved word on earth to hell with that implementation.

8. Also made me take a step back and redo the `put` verb. It was getting to be a mess as I hurried to get something done for APL97. I'm now going to go back and make sure that only one verb writes the swap file and that all cases of put go through this verb. **Modularity is GOD in software.**

Oct 10 1997

1. New Software! Since my last entry I have installed new versions of J (3.05) and PcTex (3.3). Both packages are going to "do" something for me.

   The new J provides an embedded regular expression processor and support for infinite precision rational numbers. The rationals are nice and clearly put J somewhere between the worlds of general purpose array oriented programming languages and computer algebra systems. However, the regular expression processor is, for *moi* a godsend.

   As part of my JDict 97 project I am going to improve my J test script macro processor and write a LaTeX to `HTML` converter for my program documentation. Both of these tasks will be greatly simplified by J's

regular expression processor. The latter task may not have been feasible in J without it. Now J is on par with any text processing tool and given it's extraordinary powers in other arenas is clearly *the programming language* for mainly manly men.

As for good old PcTex. I've been using this package since my days at the Queen's Epidemiology department where I was harshly critized for spending department money on it. I was impressed with PcTeX back then and it's gotten better, (not bloater), over the years.

The heart of PcTeX is Knuth's work of genius TEX. Knuth has frozen development on TEX and given TEX's high level of perfection and stability this may not be a bad thing. We don't upgrade our Shakespeare's or Mozart's why should we upgrade our Knuth's. Because TEX is stable improvements in processor power shows up in TEX performance. On my 166 PcTeX32 RIPS! This sharply contrasts with MS Word that roughly doubles in size for each doubling of processor performance. Word 97 is, as far as I can see, no faster on a Pentium 166 than Word 2 was a 66 486 four years ago.

The main new PcTeX goody is a significantly improved `*.tfm` generator. I was able to apply it without trouble to the *APL* True-Type fonts on my system. It neatly generated the TEX font metric file and a nice character map. The other major goody is dynamic font management. This is a big plus. You no longer have to install the 180 or so TEX fonts in Windows. TEX fonts use a different encoding than Windows fonts and are not very useful for other Windows programs. Now PcTeX loads and unloads fonts on demand. This frees up a lot of resources and doesn't clutter up your system fonts. Finally, PcTeX has greatly increased the size of the files the built in editor can handle and they've embedded a spelling checker that ducks around the embedded TEX commands. I won't use Word except when people pay me to!

Oct 11 1997

1. A few brief notes: **Tool Bug:** J 3.05's debugging verbs do not set stops on explict adverbs. Bummer!

2. Have rewritten most of my original `put` verb to enforce a higher level of modularity. This weekend I hope to finish it up and write a few

cover verbs to allow JDict 97 to load the dump scripts generated by my FoxPro JDict.

Pat on the Back: the idea of dumping JDict as a J script has proven to be one of the best ideas that can out of the Antwerp Edition of JDict. Plain old ASCII is for the ages. Binary formats come and go and are shamelessly dicked with by software manufacturers. I will also supply J script dumps for JDict 97

Oct 12 1997

1. Worked on `put` and `grp`. `put` is almost done which is good because until I get into `make` this is the most difficult verb in the new client. I have to get `grp` working before I can complete `put` since I need some groups and test groups in the dictionary for testing. Once I'm finished with `put` it's onto `get`

2. **Problem:** there is a problem with J/ACCESS ODBC connections. If you reattach a database a number of times from J ACCESS sometimes reports that it's LINK SPECIFICATION FILE is missing. The Link specification files define the layout of the TAB delimited text files for ACCESS. Checking in ACCESS I see the spec files. Any ODBC operation that references these files gives the link missing error. By closing and opening J I can make this go away which makes me think something in J's ODBC support is not quite right. I'll review this with Eric Iverson when I have better diagnostics.

Oct 13 1997

1. The link specification problem bit me again. It seems to be related to reconnecting to the same ODBC datasource. This is very likely a J problem. The `dddis` verb doesn't seem to completely disconnect the datasource.

2. Worked on `grp` tonight. Ended up writing more code than expected for the word group case. The test group case is identical with name changes so I'm going to revise what I've done and make the two cases share a common fairly complex subverb.

   The size of the ODBC JDict 97 client is going to be substantially bigger than the old DDE client. The reason for this is than a lot of the

logic in the DDE system was embedded in the FoxPro server. A fair chunk of this logic will end up in the new client. The `grp` verb is good example. The new client has to handle all the missing words, empty vs. nonexistent groups that the DDE server handled. The new J client won't be anywhere near the 6,000 lines of code in the FoxPro server but I expect it will be around three times the size of the DDE client.

3. *You can move complexity around but it's hard to make it go away.*

Oct 14 1997

1. **Idea:** ever since the incorporation of a regular expression processor in J I've known my J test script macroprocessor is cumbersome and obsolete. Today it occurred to me that a natural and powerful generalization of the processor is to redefine the `NB.#locdef` directive as `NB.#locrex` Currently the `NB.#locdef` replaces valid J tokens with subsequent strings. The `NB.#locrex` command would replace any text matching a regular expression with either following text or if the following text is a J noun name any value assigned to the name. A user defined prefix character like `@` would indicate a "constant" regular expression. Some J char list defined in designated locale.

```
NB. replaces all J names with 666
NB.#prefix   @
NB.#locrex   @Jname   666

NB. substrings matching rex are replaced with
NB. the value of jnoun (if there is a value)
NB. otherwise the string 'jnoun'
NB.#locrex  ' ..rex..  '   jnoun
```

   With this single change the versatility and power of the macro processor would be greatly enhanced.

2. **Implemented:** Finished the `grp` verb. and almost finished the put locales case of the put verb. I have to make some changes to my SQL to insure that changed timestamps are always updated when something changes. Added a new field to the `JSWAP1` table. A default timestamp.

One of the advantages of putting to swap tables is that you can set a default timestamp field for the tables and the database will automatically generate a date whenever a record is inserted. As Forest Gump might say "Just one less thing to worry about."

Oct 15 1997

1. The Cassini spacecraft was launched today. If all goes well in the year 2004 it will be orbiting Saturn for an extended survey of that fascinating planet. Maybe we'll even have good ODBC drivers by then.

2. **Idea:** I realized today that a problem I was having coming up with some way to specify `z` locale interface words was due to faulty thinking on my part.

   One of the important attributes of good J code is that it can be relocated. It shouldn't care what locale it's in. In the case of locale intefaces and locale implmentations you have to tie the interface code to the implementation but you do not have to specify were the interface itself is loaded.

   For example the following script loads the `foo` locale with the implemenation that is then referenced by the interface words.

```
NB. load foo locale with code
'foo' load 'implementation'

NB. define external interface to foo locale
fooguy =: foox_foo_
foogal =: fooy_foo_
```

   This script can be loaded like any J script

```
'anylocale' load '...'  NB. load interface into anylocale
'z' load '...'          NB. load interface into z locale
'' load '...'           NB. load into base locale
```

Not only is this easiar to program but it confers greater flexibility. Win Win!

Oct 18 1997

1. **Bitch:** I've made some headway on writing a set of words to load the dump scripts generated by the FoxPro system (`JDict A`) for Antwerp edition into JDict 97. The load script is almost a worst case scenario for Jet Database file space allocation. **The Jet doesn't seem to make ANY EFFORT TO REUSE file space.** If you delete or overwrite an existing item it seems to copy everything to a new location leaving the old stuff hanging around. **This is so lame as to border on unbelievable. Sort of like the NO LOWERCASE IN KEYS idiocy.** The Jet has some serious problems that seriously hampler it's over all utility.

   After running my load script into JDict 97 the database had balloned to 9 megabtyes. If you run the compact routine (which seems to be inaccessible from ODBC) the `*.mdb` drops back to 1 meg. This is pretty sad. It's safe to say that the Jet would be almost useless for large databases that experience massive inserts and deletes. Fortunately my database is small (less than 20,000 words).

2. **Idea:** While working on code to load old dump scripts I noticed that there is an omission in my client word design. I can put macro scripts into the dictionary but I have no way to associate the macros with words, groups, test groups et cetera. I could extend the `uses` design but instead I'm opting for a new word `macro` that will associate macros. *You have to design, and personally use your design, to see if it's any good.*

Oct 19 1997

1. `put` is almost done. I only have to implement the group locale interface procedures and I'm done. `put` is far more complex in JDict 97 than it was in JDict A. This is because it subsumes most of the functions of `stuff` and also can work on entire lists and tables of data. JDict A `put` could only put one word.

   The new `put`, as I said earliar, is slower than the old once until you start putting large numbers, (greater than 50), words. Then the mass put catches up and overtakes n DDE commands.

The same is true of `get`. Getting one word is slower in JDict 97, getting 100 words is faster.

If ACCESS ever generates indexes to `*.csv` tables I can reduce `put` time. However it's fast enough on 166MHZ machines. Typical operations are all done in a less than a second. The only time this isn't true is when Windows has paged out JDict or ODBC out and has to reload. Then I may wait two or three seconds.

2. Old dump file processing is all done except for reloading the uses cross reference file. Being able to load old dumps into the new structure is useful for testing the new words and generating realistic test data. The reload process clearly reveals ACCESS's pathetic file space recyling. Staring from an empty database. The "empty" `*.mdb` is about 400K. The dump load will cause the `*.mdb` to swell to over 12megs. Compacting reduces this to 1.3 which compares very favorably to the total size of the old JDict A FoxPro files.

   By putting larger numbers of objects at a time I could reduce file `*.mdb` bloat and significantly speed up the dump load process. When I write the ASCII dump verbs for JDict 97 I will keep this in mind.

Nov 1 1997

1. Very little progress since my last entry. Last weekend I flew home for the last time. Come November 26, 1997 I'm outta here. Head'in er home. Returning to vast frozen wastes of the north. Where I will begin work on a new year 2000 contract at Reuters in Toronto. This is pure contract work that will gross some $7,000 per month. The most I've ever been paid. This APL gig is going to be pretty lucrative in the dying days of the 20th century.

   This last visit was particularly pleasant since my parents met me at the Toronto airport. I rode with them to Kingston and they stayed until I flew home. I played some golf with my dad and Jacob. Watched Helen skate, played Monopoly, still one of the all time great board games, with my Mother, Helen and Jacob. I hardly got a moment with Ruth. She was on call and keeping up her skating and precision team schedule.

2. As for JDict I had a **Revelation:**. For sometime, years I'm afraid to admit, I have been struggling with the concept of the *locale interface*.

The basic idea is that since you can bury entire J subsystems in locales that are created at load time and refer to the objects in these locales through interface words. It makes sense to "define" the locale interface in some way. This is mainly for cross referencing. Defined interfaces insure there are no undefined references. Avoiding undefined references is a number one design goal of this entire undertaking.

I first thought that all you had to do was denote a certain subset of group words as the locale interface. This works, provided all the words are in the dictionary, and the words load into only one locale. This is restrictive but I was prepared to live with it. Then came the notion of the *alien* script.

One of the defiencies of JDict A is that it has no facility for hiding code. A prime example is J's component file system. The component file system is nothing more than a collection of J words that get loaded into a `jfiles` locale. The load process defines an interface to this locale. It's through these interface words that you refer to component file functions. Hardly rocket science just clean component design.

The only way I could store the jfiles component in JDict A was to load all the words into the dictionary, define a group and then hard code the interface in the jfile's group `DO` field code. This is fine except it exposes to much code. I learned this the hard way. After doing this I started using some of the non-interface words in the file system. Then the next release of the file system totally changed the internals of the file system. The non-interface words I was using disappeared. Of course I deserved my punishment. Interface violators should be flayed with dull knives, beheaded and then sued!

After this grim experience I vowed that the next JDict would have a facility for storing entire unanalyzed scripts. JDict 97 has this facility and it has even generalized it to non-J scripts. The problem is that arbitrary scripts complicate locale interface definition.

The arbitrary script can do anything, like take over the world, at load time. It can create multiple locales, destroy existing subsystems, spawn new tasks, whatever. Defining a clean interface for such beasts requires a complete and intimate understanding of what the script does at load time. This contradicts the goal of component design which strives to hide the gruesome details of component internals from component

users.

From contradiction comes revelation. You can quote me on that!

3. **Idea:** I have decided to abandon the definition of locale interfaces and change the way cross referencing works. Cross referencing tools attempt to dig all the way to the bottom of call trees. This is a mistake. Now I will cut off the search when I encounter words with explicit locales. For example if `jfile_files_` is called by `jfile` the new cross reference tool will consider `jfile_files_` to be like a system primitive and quit searching. This simple trick will allow me to define interfaces to black box scripts that create one or more locales.

   Suppose an arbitrary script is loaded that creates and populates three locales. To define an interface all I have to do is insert words in the dictionary like:

```
boo =: whatever_locale1_
hoo =: LikeIcare_locale2_
foo =: fooness_isgoodness_
```

   And then add these words to the group that stores the black box script. The `make` verb will generate a group script that runs the black box first and then defines the interface. I will still require a *load locale* for the black box script but where it loads and what it does are two different things. It can load in any locale and still take over the world.

   Note: that the interface locale is not specified and can be chosen at load time. The above interface could be loaded into `z` locale with:

```
'z' load 'interface'
```

   Of course I will have to know what locales the black box script creates which requires a little deeper understanding of internals than I would like but the payoff is that this scheme allows a far greater variety of single and multilocale components to be defined.

Nov 2 1997

1. **Implemented:**. I finished the last "feature" of `put`. I can now define group interfaces. I still define the interface but make no attempt to associate it with locales. The locale field is now used as a load load locale. The `make` verb will have to distinguish between cover words and legitimate word definitions and generate group and load scripts appropriately. If you can't solve a problem defer it.

2. **Bug Fix:** I wasn't testing for the presence of carriage returns in put text. It's important to exclude carriage returns since it is used as a line end delimiter in the `*.csv` files linked to the ACCESS database. If bogus CR's are in the text the structure of the `*.csv` file is screwed up leading to very misleading error messages from the Jet database.

3. **To Do:** I should update the documentation fields in my ever growing JDict 97 group before I forget what all this stuff does.

4. **To Do:** complete `get` coding. The overriding goal is — if you can `put` it you can `get` it. Symmetry, symmetry, symmetry ...

Nov 3 1997

1. **Idea:** One of the goals of JDict 97 is the ability to run multiple instances of the system. Ie, load a number of JDict locales, open a number of dictionaries and run commands against each dictionary. It occurred to me that a clever way to run a *large number of instances* is to load the system code into the `z` locale and then load each instance locale with only the locale globals that distinguish each instance and some cover functions that will "tell" the `z` locale code what instance is executing: just setting the `z` globals would suffice.

2. Spent sometime documenting code. This is a good thing to do when you don't have the energy to code but still want to produce something.

3. Went over my design for `get` and insured that it matches `put`. Remember if you can put it you can get it. `get` will be almost trivial to code compared to `put` because SQL is good at extracting data — it sucks, big long donkey dongs, when it comes to storing it.

4. **Changed:** I changed the order of option codes in the left argument of `dnl`. This change makes the first two codes of `dnl` like other words. The

search qualifier is now an optional third code. The default search for all cases is *prefix string search*. This is not a good default for document or script fields where I expect the typical search will be *contains string*. For these cases you'll have to spell it out pardner.

5. I bought the game **Riven**, mostly for myself and another **Magic School Bus** adventure for the kids. I'm curious to see what all the fuss is about. I only hope this game isn't to absorbing or JDict 97 will turn into JDict 99.

   Riven like all good computer games pushes the PC power envelope. This sucker comes on 5 CD-ROM's and requires at least a 100MHZ Pentium, 16Megs and 75 megs of free disk space. One game sucking up the resources of Megabloat Office. What the hell. Computer resources are cheap and getting cheaper as we speak.

Nov 4 1997

1. I'm not sure if this is the appropriate place to comment about **Riven**. Perhaps as an example of a productivity sapping distraction that gets in the way of a programmer and his delusions of coding grandeur.

   First, this is a great game! I can see it easily sucking up a fair fraction of my free time for the next month, maybe longer. The premise is so unlikely that when people describe Riven, and Myst before it, you wonder what's all the fuss about. "Well you wander around in this strange beautiful world solving puzzles." Sounds a lot like the ancient and venerable computer game Adventure. In a sense Riven is an adventure-style game. In the early seventies computers were simply unable to run a Riven style game. Graphics took forever to render, and the result was not high resolution, mass storage was far from massive, computer sound consisted of a few terminal beeps. Text was supreme. Adventure made superb use of text and created one of the first computer game worlds that sucked people in.

   Riven is like Adventure in that you wander in a strange compelling world but it takes advantage of all the advances in PC computer technology to render that world without words. This is no mean feat. Think of all the great books that have been made into horrible movies. Aside from a few pages in a diary, beautifully displayed as a handwritten

book, and a few brief live action sequences, there is no "English" in Riven. This is completely appropriate and will give the game a universal appeal. The Riven world is laid out before you as a series of fabulous interconnected images. You move in Riven by clicking where you want to go. There are so many images and they are so well stitched together and matched with background music that the illusion of moving in the Riven world works. Boy does it work!

After installing the game I thought I'd give it a try before getting on with a some serious real world money making programming. Well the Riven's opening sequence, easily the most impressive thing I've seen in computer games, pulled me in. After eight straight, can't leave my computer to pee hours, I realized it was 3PM. I had wasted an entire night on a game. Oh my god it's got me. I'm finished. My social life, my pet projects, my astronomy habit, and personal hygiene are all going in the dumper until the Riven spell is broken. Ain't obsession grand?

March 20 1998

1. I was right about Riven distracting me and pusing JDict97 into 1998. But I was wrong about my ever changing attitude about this project. About a month ago I simply snapped. I'd had enough pissing around with trying to get ODBC to work the way I want. My conclusion about ODBC and SQL is that they are fine for getting data out of databases but suck when it comes to putting it in. ODBC is particularily bad at handling large binary objects which, oddly enough is the preferred data object for J arrays. I'm not the only one with this opinion. Microsoft has also "discovered" this problem during their efforts to bolt an OLAP bag onto SQL server. OLAP needs large binaries to work at all so Microsoft will have to come up with something credible. The fact that OLAP is also oriented toward high rank arrays means it's likely that the MS OLAP server will turn out to be a decent place to store traditional J numeric data. Given this sea change in system software I decide to scap non-renumerated ODBC work for the time being.

   The trouble is I need a new dictionary system to deal with all the changes in J especially the somewhat radical changes in release 4. All of which bring me to JDict2.

2. **JDict2 The Next Generation**. After some testing of J's `i.` verb I decided that it's plenty fast enough to handle dictionaries of up to 20,000 words on my current machine. Actually for dictionaries in this size range it is **FASTER than Access ODBC** and even beats the FoxPro DDE search mechanism. Roger Hui continues to achieve amazing results with J optimization.

   Given that my master dictionary, after four years of entries is around 1300 words, I decided that a pure J solution, no ODBC, no DDE, no external databases would be perfectly acceptable for single user systems. Given the rate at which PC hardware is improving by next year this same system will easily scale to 100,000 or more words. This is more than big enough to accomadate a lifetime of J coding. So it's onto a pure J JDict2.

Apr 11 1998

1. I have started coding JDict2. Last Friday I produced my first significant procedure `jwordscreate` which creates the main words file `jwords.ijf`. Tonight I'm going to work on a new `put` for the words case only.

2. I've started a Microsoft Word document to describe and document the new JDict2 client verbs. I'm not particularily fond of Word, though I must admit it is growing on me but this time around I want people to actually read the documents associated with this system. LaTeXis great system but it is not widely used outside of academic book publishing. I will continue to use LaTeXfor documenting words within the dictionary because:

   (a) It is a pure `ASCII` format.

   (b) It is still the best mathematics markup language.

   (c) It is stable. I'll be able to read my document decades from now. I'm not so sure about Word or even `HTML` that is mutating into some gaudy multimedia monster.

Apr 14 1998

1. I've built the `put` shell by salvaging code from my ODBC system. I spent time paper and pen programming on the train the last two days and I'm hoping to at least get the append case of `putwords` working tonight.

2. **Changed:** on the train I realized that some of of the data in my inverted directory fields could not be reconstructed from data in the word nouns. This is a problem because it's only a matter of time until I blow away my directory. I spent sometime thinking about inserting duplicate data in the word nouns but abandoned this option because it increases the complexity of the `put` process and penalizes the user (me) for virtually every simple addition of data to the dictionary. Directory crashes will be rare. I've decided to add a `chkp` verb to the system that will back up the directory components in each data file. By storing successfully appended component numbers this strategy will allow a quick recovery to a previous directory state.

Apr 19 1998

1. Reasonable progress. I have most of the `put` word case built. I can now append words to a new `jwords.ijf` component file. The append process for a single word is slower that the DDE system. On the order of 170 milliseconds vs. 20 to 40 milliseconds. For larger numbers of words the `jfiles` based system is faster. It takes roughly 22 seconds to put 1300 words in the DDE system and about 8 seconds in the the new system.

   Retrieval time, or `get` speed compares nicely to the DDE system. Simple single words get tests indicate the two methods are about the same speed for 1200 word dictionaries. I expect `jfiles` will easily outperform the DDE system for gets of many words. This is roughly the same profile observed for the ODBC system. Putting a single word was slower but putting many was faster. However with the ODBC system I could never approach the speed of the FoxPro/DDE system for single word gets which is a very frequent and hence important case to optimize. The `jfiles` system will come close or match DDE for single word gets and clobber the old system for large numbers of words.

2. **Idea:** while planning `rplcwords` it occurred to me that backing up each word everytime it is replaced will more than double the amount

of file reading and writing a typical replace operation will take. I've now decided to add a third argument to all put cases that will essentially request that a back up be made for this `put`. The default option will be no back up. I like to live life on the edge!

```
NB. Example

NB. back up old word definitions before replacing.
0 7 1 put 'thisword'

NB. back up old test, if any, before storing new
1 7 1 put 'testname';'test text ....'
```

Apr 23 1998

1. **Note:** I observed the other day that my current set of commands does not include any means for setting and clearing the readonly bit of objects. I've decided to add new object codes that will handle this case. To set or clear a readonly bit on an object requires that that object exist. In put's case it will look like:

```
0 13 put 'wordname'   NB. make word readonly
0 14 put 'wordname'   NB. turn off readonly

5 13 put 'project'    NB. make project readonly
5 14 put 'project'    NB. turn off readonly
```

Apr 28 1998

1. Had a great nights sleep last night: eleven uninterrupted hours. Sleep is an essential programmer maintenance activity; right up there with backing up your hard drive. To underscore this fact upon waking I had a good **Idea:**. To simplify many dictionary file operations it makes sense for each dictionary file to have identical directory layouts whenever possible. So I revised all my `*.txt` layout files. During this time I

also realized that it made sense to split the 10 reserved components at the end of the directory section and devote 5 components to the directory section and 5 more to it's backup. Hence I can add up to five new inverted lists without changing another file byte. This rationalization will make file creation and the directory checkpointing trivial which is a sign of progress in software design.

May 2 1998

1. I'm almost done with the first verion of `newd`. The changes I made to the file directories made it possible to tweak a clone of `jwordscreate` to make all dictionary files except the words and uses files. I spent most of tonight generalizing the standard directory creation facilities of `newd`. One of the limitations of the original JDict system was it's inability to create anything but toplevel directories.

```
c:\toplevel   ok
c:\toplevel\deeper\deeper\  not ok
```

   The JDict2 creation method will build anything the underlying host OS will allow.

2. **Tests:** to try

   (a) Create path with names longer than host allows

   (b) Create path with to many levels

   (c) Create path with non-alpha characters

   (d) Create empty sub paths

May 24 1998

1. `newd` is almost finished. I have to add code to update the timestamp, the master directory backup and test the size of nouns but all this can wait as I make progress on other commands. I had to build a fair amount of the dictionary infrastructure just to implement `newd` and `od`. I'm a few hours away from finishing `od` alas I will not be able to get in this weekend.

My coding time on this project is severly limited due to the lack of a Win95/NT machine at work. I cannot sneak a few lines of code during the slow hours at work and I cannot work after hours when I'm in Toronto. As for uninterrupted coding at home. I only get it after all the kids are in bed or on my "working days at home" At the rate I'm going it will be months before I'm finished with the interface verbs. I'm thinking about leasing a portable to use in Toronto, especially during the month I'm with Helen while she attends skating school. This would boast my coding hours but it would cost me and I'm cheap.

Speaking of cheap. The latest version of J-Pro is selling for $895.00 US. This is about $1300.00 CDN far more than I wanted to pay. The Pro version includes all the manuals, a CD-ROM, and free updates for a year. I know there are typically three or four bug fix versions during a year so I'd end up spending the same if I bought a series of standard versions. This high price is worrying. It either means the J folk think they can get away with it or they are going broke and this is a bid, probably ill-advised, to raise revenue.

I went ahead and bought the package. God Knows I've gotten about three or four free versions in exchange for beta testing and I will be able to claim it as a legitimate business expense. Still it's $1300 more dollars I didn't want to spend. God I still have to save up for my bloody next years taxes.

Don't get me started about taxes.

May 27 1998

1. Fascinating Toronto APLSIG meeting last night. Two brothers Barry and Brian Silverman talked about how they have resurrected classic early computer programs by simulating the machine instructions and OS commands (in Java/J++) They demonstrated SpaceWar. The very first computer game written way back in the early 60's at MIT. As it was an APLSIG they also showed an early version of `APL\360`. Running on a simulated 360 with enough OS and channel support to run APL. The APL code they were running originated from an old tape found in a friends basement. Endangered bits as they called it.

   The room was filled with some of the people that worked on this old software. Ken Iverson, Ian Sharp, Roger Moore (the programmer not

the actor), Eric Iverson, Bob Bernecky and it was scary how much these guys could remember about these old assembly language programs. Moore could still remember program comments in 30 year old source and pointed out where some floating point tricks took place. The people that worked on this stuff we're pretty much first stringers: all extremely bright, some certifiable geniuses and some just plain certifieable. Very different from today's crop of commodity programmers. Many of whom have never seem an assembly language intruction and wouldn't recognize a channel if they fell in one.

You might ask why? Bringing back 30 year old software hardly strikes one as a rational activity. The Silverman's justified their impressive work as follows:

(a) It's a lot of fun.

(b) The research introduced them to a large number of fascinating computer pioneers.

(c) Computer programs are major artifacts of our age. Preserving them in their original form has historical value. As time goes by this will be more widely appreciated. Remember Van Gogh couldn't give away his paintings. Maybe in the next century old card decks containing original programs may be worth millions.

(d) Finally, people that love old trains often build models of old trains. The Silverman brothers described themselves as the first software model train builders.

This topic was so intrinsically weird that I loved it.

2. On other fronts I had a brief chat with Eric Iverson about his recent J work. Eric and company are finally implementing memory mapped files for J arrays. He talked about this a few years ago but now has decided the underlying technology is stable enough to actually pull it off. (Others, particularily Arthur Whitney has already done this for K).

   Memory mapped files will be an enormous boost to the utility of J. It will make it possible to hold gigantic arrays in memory and operate on them just like any other J array. Eric's test data for this adventure is a CD-ROM containing all north american stock exchange closes for every

stock traded on the continent at any exchange for an entire month. It all amounts to about 300 megabytes of integer data. He plans to store all this data in a small number (maybe one) J array memory mapped files.

J tasks will attach to these files and viola you will have one honking massive object in memory that can be manipulated, searched and modified just exactly like any other J array. Apparently more than one task can see these objects at the same time. Something you cannot do with current J arrays.

This facility will, among other things, will render my dictionary partically done JDict2 system obsolete. It would be silly to use `jfiles` when you can use mapped files to store the entire contents of such files in more conveniantly accessed arrays. As I build my dictionary I'm now going to build a model assumming everything is in memory. Because soon it all will be!

May 30 1998

1. The last day of May. Three weeks to the shortest day of the year. Not much to report on the JDict2 front. I've finished: at least as much as I'm going to do now with a fairly definitive Word document describing each interface verb. I now know pretty much exactly what I I want and it's just a matter of a few hundred programmer hours before I get there.

2. I spent a lot of time this week pondering whether I should change course again and design primarily for the upcoming memory mapped files. I've decided not to for the following reasons:

   (a) Memory mapped arrays are not out yet. There may be limiting show stopper gotchas. For example it occurred to me that maintaining a memory mapped nested array may be very difficult. I'm sure that simple arrays of any type and rank can be trivially mapped as they consist of a header followed by a large single list of data. Nested arrays are not this simple. They consist of pointers and data at the leaves. The data is not contigous and can be scattered all over memory. Maybe the mapping facility can easily handle this but right now I just don't know. Even if

mapped arrays are simple this still will be a very useful facility for processing very large chunks of data however simple arrays will not do for JDict2

(b) I want to get this done. Changing your underlying database every few months is a way to postpone finishing forever.

3. **Changed:** one of the planned facilities of JDict2 is a facility that is similar to project and make files in other systems. To avoid confusing this facility with the new project manager in J I changed the name of the `proj` verb to `bund`. I also changed all references to projects to bundles. A bundle, an arbitrary collection of possibly, but not necessarily related dictionary objects, actually comes closer to describing this facilty. A bundle can be a project but it doesn't have to be.

June 17 1998

1. It's been awhile seen my last log entry. Believe it or not this system is taking shape. You spend ages planning, writing documenation, fiddling with database designs, coding and then going back to plan some more, document some more, and fiddle some more. At some point you reach a critical mass and "the system" takes off. I believe I've reached that point.

   I decided to work on many verbs at once. Coding the important cases, laying out the overall structure of the verb and then coming back to fill in details. Using this approach I have *nearly complete* working versions of:

   (a) `put`

   (b) `chkp`

   (c) `did`

   (d) `dnl`

   (e) `od`

   (f) `newd`

   (g) `grab` - a `get` prototype

June 18 1998

1. I met with Eric last night and he showed me what he's planning for memory mapped files. My intuitions have been proven correct. The first release of mapped files will be simple open arrays. Infact they will be fairly static open arrays. It will be easy to replace parts of arrays if you do not change their shape but adding new items will require some explicit remapping calls on behalf of the programmer. Eric said it would be possible to do all the resizing in J but for the first release they will take this approach. As they learn more mapped arrays may come to behave more like ordinary arrays. Even with these restrictions mapped arrays will be very useful. It will the fastest way to get at large chunks of data.

2. **Note:** to self — stay out of bookstores. Yesterday while cruising the path — downtown Toronto's wonderful network of underground tunnels — I popped into one of the bookstores I frequent and quickly spotted an interesting title "AntiPatterns." For sometime now I've heard rumblings about Patterns. Like many computer buzzwords it's hard to determine if the word signifies a useful new idea or simply a rephrasing off old ones.

   In my opinion Patterns are basically a rehash of a very old idea. A Pattern is basically a description of a design that has proven successful in the past. The whole pattern movement is essentially a cataloging exercise. Let's make a big honking list of successful software designs — give them nice catchy names and write buzzword dense books about them. Patterns are not entirely useless but it's not an exaggeration to say that the phrase — learning from experience — encapsulates, (notice my OOP speak), 90% of the value here.

   The AntiPattern authors of course do not share this view. To them Patterns are a revolutionary tool. I find it rather ironic that very early on in AntiPatterns they quote research that shows:

   > New software techniques have not had any meaningful impact on the success of large software projects. OOP projects are just has likely to fail, be late and over budget as non-OOP projects.

   They don't really pursue this embarrassing result but the rest of the book exudes the righteous thunder that if only these people got "it"

— it being Patterns, Object Oriented Design – blah, blah — et cetera, things would so be much better in software land. I get the impression that authors don't really think there are any excellent non-OOP programs. This contempt of history is very typical of the OOP crowd, as it was of the STRUCTURED crowd, as it was of the HIGHER level language crowd, as it was of the MACRO assembler crowd, as it was of the VACUUM tube crowd ... ignorance and arrogance is a serial AntiPattern.

Despite the books pontificating tone the list of AntiPatterns — basically industry standard fuck-ups — is useful. The first AntiPattern, THE BLOB, was directly relevant to my JDict2 project. A BLOB is single large class that subsumes most if not all of the operations of a system. In the author's view BLOB's are often created by old fart procedural programmers when they first use OOP programming languages. The BLOB's symptoms are a large number of methods in one class, use of auxiliary objects to store mostly data, and a lack of overall design.

3. In JDict2's case the problem is not lack of design. I have spent a lot of time thinking things through. My working JDict2 interface document is the best design I've ever worked from. I know exactly what I want and more importantly why I want it.

   As for the code, I've known for sometime that I've been putting way to much stuff in one system class. I've been spawning directory objects but then directly modifying their contents, (something you can trivially do in J but is harder in standard OOP languages). Early on I decided that a directory object would be useful but more or less didn't bother providing a standard J OOP implementation. Mostly because I was to lazy to absorb the J OOP style but also because I couldn't see any great advantages in directory objects. Unless there's big time payback forget it.

   After thinking about the BLOB AnitPattern and my particular problem it occurred to that:

   (a) I was right about the relative pointlessness of directory objects. The directory is a database artifact. It shouldn't even be visible outside the STORAGE class. I'm forced to deal with directories

because I'm forced to build my own database system to overcome some highly annoying limitations of ODBC.

(b) It would be valuable to break the system into classes that facilitate ripping out underlying subsystems without traumatizing the rest of the code. One goal would be to isolate the system from the underlying data storage mechanism. I'm planning on providing and SQL-Server based system and possibly an XML web page type store so I might as well build a proper STORAGE class that makes it possible to replace the database without dicking with the rest of the system.

(c) Other parts of this system — the script generators, the GUI database browser and so on would also benefit from being programmed as classes in their own right.

So I guess it's sort of back to the drawing board. I won't have to chuck a lot of code just rearrange things and tweak. The underlying database design doesn't have to change at all.

June 26 1998

1. I have rearranged my JDict2 code and defined an object class hierarchy. I haven't actually *run* my code yet but the mere fact that I was able to easily reorganize it into a few class objects bodes well. The classes are:

   (a) `JDict` root class — defines the user command interface and contains dictionary wide utilities and constants. All other dictionary classes will be an extension of this class. I though about further breaking up this class into a command and utilities classes but decided that, *for now*, it makes sense to keep them together.

   (b) `JDstore` storage class — hides the underlying database or file system. This will be a fairly large class for this version of JDict2. It has the potential of being replaced by a very small class if I can ever find a J/ODBC/database combination that supports SQL-stored procedures.

   (c) `JDdob` — dictionary object class. The `JDstore` class will create and destroy dictionary objects as it runs. The dictionary objects

exist only to serve the storage class and will not necessarily exist in other storage classes.

(d) `JDmake` — make root class. Will contain all general make utilities. I expect each type of script generated by `make` will have it's own class `JDlatex, JDhtml, JDjscript`.

(e) `JDmaster` — master file class. The master file is basically a user dictionary registry. All the dictionaries a user can open are listed in this file. I expect the master file will retain it's `jfile` nature even if I produce other storage classes in the future.

(f) `JDbr` — dictionary browser root class. The J GUI grid dictionary browser will grow here.

At runtime most of these classes will have only one instance with the exception of the directory class.

2. Next Monday I begin a month in Toronto with my daughter. She's in town for an intense figure skating session at the Toronto Cricket and Curling club in North York. I'll be staying with her. During this month I'm going to bring TEX, the computer I bought in Dallas, to town and spend my evenings — after Helen has gone to bed implementing the hair-brained scheme outlined above.

July 1, 1998

1. We are now in the apartment. Helen's skating is underway and I have managed to start working on JDict2 in Toronto. I think it will work out reasonably well provided I can endure the schedule. I have to get up about 5:45 every morning of the week to get Helen to the rink at 7:15. Helen works hard skating, dancing and swimming the whole day leaving zonked by her bedtime of 9:30. If I can stay awake past 9:30 I'll get a few hours every night to work on this.

2. **Changed:** merged the `JDmaster` class into the root `JDict` class. This simplifies the binding of references at the expense of some modularity.

3. Managed to create objects from `JDstore` and `JDdob` tonight. The storage and directory objects are extensions of the root dictionary object. Tomorrow I reform the verbs in the directory object and improve the storage object creation verb.

July 2, 1998

1. It's been very slow at work for the last two weeks. My role at Reuters is to test *SHARP APL* software for "y2k compliance." Up until a few weeks ago I was very busy but I've been waiting for the next set of test plans to arrive. In the meanwhile I'm sneaking sometime to work on JDict2.

2. Today is my 45th birthday. In five more years I will be officially old!

3. Wrote a `jdstart.ijs` script that creates a storage object. This object needs to "know" it's object locale to properly create a `z` locale interface.

July 8, 1998

1. My 20th wedding aniversary—dear lord. I doubt I'll make another 20 years with the same woman.

2. JDict2 work is progressing. I have almost recovered all the functionality of my pre-object-oriented system. It's been a bit of learning curve to figure out how J's class object `colib.ijs` system operates but I can see it will be worth it in the long run.

   I'm very impressed with this little system—and it is little: a few dozen lines of code. Eric has managed to boil OOP done to it's bare essentials. I've been amazed at how quickly you can take advantage of OOP goodies. For example inheritence is a no brainer you literally don't have to do squat. This contrasts with Delphi, (my last OOPey experience), where you had to do a fair bit of up front work to do the same thing.

   I've also been impressed by the performance of J OOP. The path search mechanism doesn't seem to impose a lot of overhead. It's hard to tell the difference between code that is executing in one locale vs. code divided up into many locales (objects). Apparently the underlying mechanism in this release of J is *naive* and Roger has already worked out a much faster object-creation-destruction and path searching system.

3. **Changed:** I have dropped object level read-only settings. The ability to open an entire dictionary read-only confers most of the advantages of object level settings without the additional headaches and overhead of maintaining yet another inverted list.

July 9, 1998

1. **Bug Fix:** at the end of yesterday's nightly coding session I notices that my `od` verb was misbehaving. I was able to open a dictionary read-only and then, without closing it, open it again read-write. It's not supposed to work like that. I thought I was missing an open list check but it turned out to be far more serious almost boarding on a *design error*.

   The open behavior was a symptom of a master file and individual dictionary inconsistency. My master file is corrupted—a remnant of working of `newd`. One of my `newd` tests failed leaving a partially updated master. Because I haven't converted my master file creation routine to the new J OOP style I couldn't regenerate my master file. I left it and went on to other things. One of the lessons of my first JDict effort is that it's advantageous to work with bad data during system development. It forces you to deal with errors you may miss otherwise. Now I am checking the `didnum` in the master file table against the `didnum` in individual dictionaries when they are opened. These numbers should always match. If they don't we have a problem.

   The only fix for this type of inconsistency in the future is:

   (a) Try restoring a copy of the master file table with `dpset` and see if the inconsistency is fixed.

   (b) Go to your backups.

2. **Idea:** It wouldn't be all that hard to write a little utility that would rebuild the main master file table from a list of dictionary root directories: e.g.

   ```
   fixmastertable '\here';'\and\here';'\and\here\to';'\ehh'
   ```

3. **Bug:** after the nuances of the previous error the next problem is no-brainer. As in no-brain was operating when the error was made. My directory updating code in `rplcwords` was not properly modifying inverted lists. I obviously copied part of the code from `appwords` because instead of changing items I was appending new ones on—DUH! This of course screwed up the lengths of inverted data lists with respect to

the word index list. I didn't notice this until I tried to use an inverted list (name class) in `dnlsearch`.

4. **Bug:** `delwords` is not updating inverted data lists. My deletion algorithm is basically like `appwords` in that I have to modify all inverted lists. I considered maintaining a deletion boolean. This would speed up deletions but slow done gets, searches and updates. Deletion is a relatively infrequent operation,, based on my use of JDictA,, so it's a good verb to suck cycles on.

5. **Problem:** I may have to generalize the planned arguments of `globs` to allows lists of words and suites otherwise there will be excessive inverted list updating, (slow and risky), for common requests like:

```
0 globs&> }. grp 'somegroup'
```

July 10, 1998

1. **Idea:** while editing error message text to increase message consistency and brevity it occurred to me that a debugger version of `jderr` would be very handy for development. Basically look at the stack inside `jderr` and return the name of the word calling `jderr` and the line triggering the error.

2. While looking at directory updating code in `delwords` it occurred to me that if any `jread` failed the directory would be corrupted. There is a simple but tedious fix. Check every read before replacing data. This introduces a lot of code and slows things down. I've opted to check the first read. If there is anything wrong with the file this will catch the vast majority of errors. Subsequent checks will not dramatically increase safety.

July 13, 1998

1. Confession time. I bought the official Myst hints and strategy guide. My rational for "cheating" is that I don't want to waste excessive amounts of time on this game. I spent three weeks, playing eight to ten hours a day on Riven before I succumbed to the temptation of looking

for hints on the Internet. A few hints were all I needed to finish the game. I want to avoid such obsessive play with Myst and spend no more than a few days playing.

I have read the guide up to where I was stuck. My theory about the spaceship was right but I must confess to missing the change of time on the clock tower when you diddle with the controls. As for the link between the planetarium, the stone ship, and the humming icons I doubt I would have ever seen that!

Now I have a pretty good idea how to visit three ages. I'll spend the next few days exploring before refering to the book again.

July 14, 1998

1. I'm exhausted. These days I have to get up at 5:45am to get Helen to the rink. Such early mornings are not compatible with playing computer games until 1:30am.

   I visited the Stone Ship, Channelwood and Machine Ages of Myst. I used a book hint to enter the Stone Ship Age but the other two I figured out on my own.

   Currently I'm stuck in the Machine Age. To return to Myst Island I have to find a linking book. To find the book I'm pretty sure I have to solve the machine control puzzle. After spending hours and hours with Riven and Myst I am beginning to see how the puzzles are designed.

2. Myst-Riven puzzles generally follow this pattern:

   (a) Simple manipulator puzzles. To solve these puzzles you have to fiddle with various controls, knobs and doo–dads and observe what happens. The steam pipes powering the walkways in Riven and the Windmill pipes in Channelwood are examples. The only trick in most of these puzzles is getting things in the right order. I've never had any problem with these puzzles. Even fairly complex variations like how to drive the submarine in Riven.

   (b) Can't get there from here puzzles. In this class of puzzle you have to set up the proper conditions in one place to advance. The log cabin and giant tree puzzle of Myst is a perfect example. Until you get that boiler going you're never going to get in the tree.

The tank puzzle on Riven's crater island is another example. You have to fiddle with a series of controls to open up the drain pipe. Crawling through the pipe was the only way to get to the Frog Cave and ultimately the laboratory. I've generally been able to solve these puzzles whenever I can find them.

(c) Hidden clue puzzles. Hidden clue puzzles test your powers of observation. Once you see them you've solved them. I've had a lot trouble with hidden clues. My two biggest and most embarrassing failures were:

- Failing to look behind the Frog Cave doors in Riven.
- Not pressing the little lamp knob near the Forest Idol in Riven.

In both cases vast, easily explored, parts of the game opened up.

(d) Association puzzles. An association puzzle relates information from a number of places in the game. The D'ni number, and animal code in Riven was superb example as was the book, observatory, humming switch and stoneship puzzle in Myst. All of these puzzles build up a chain of associations. In Riven you had to play with the class room toy to figure out D'Ni numerals less than 10. Then you had to notice the numbers on the back of the wooden balls, then you had to associate the balls with animals. Finally you had to find all the balls to find what animals needed to be pushed to, in the numbered order, to open a linking book gate.

These puzzles are the hardest I've encountered in Myst and Riven and my performance has been less than stellar. Infact until I started thinking about classifying the puzzles I had failed to recognize the essential feature of chains of associations.

One thing experience has taught me is that combinatorial attack fails. I could have pushed humming icons on Myst or dropped colored balls into the dome grid on Riven for years and never solved the puzzle. Infact the presence of a combinatorial puzzle so far is pretty good indication that an association puzzle needs to be solved.

I'm looking a big fat combinatorial puzzle in the Machine age.

July 16, 1998

1. I am out of the Machine Age and back on Myst Island. The combination lock was easily solved once you figured out how to get to the Fortress rotation controls. It all keyed on the elevator delay switch which let you get out of the elevator on the top floor and lower the controls.

   I have to get into the Selentic or Rocket-Ship Age. I know how this age is opened but I cannot set the damm controls. To set the controls you have to match audible tones on a piano and sound bars. I just cannot hear the differences. My daughter had a crack at it and she couldn't set it either. Maybe my PC's sound system isn't up to snatch or, more likely, I'm tone deaf. Ruth has always claimed I'm tone deaf—maybe she's right.

   Anyway I have a photocopy, from the Myst cheat book, of the correct tone bar settings. I may not be able to hear but I can still see—when I put my glasses on.

   I doubt I'm the only person that figured out how the lock worked but couldn't open it because of questionable hearing. In Riven there are no sound matching locks. I doubt this is an accident because you see every other type of Myst puzzle in Riven.

   Tonight I hope to explore the Selentic Age retrieve the remaing pages and end this little diversion. Then it's back to JDict2.

2. Met with Morten Kromberg at Sutton Place this morning to discuss "employment opportunities." It sounds like Morten's company is making great use of modern APL's. Maybe this APL gig will work in the 21th century.

July 17, 1998

1. I finally managed to set the tone lock to the Selentic age. Even with a photocopy of the puzzle's solution in hand it took me about twenty minutes to find the right setting. This is a poorly designed puzzle. As I said earlier there is nothing like it in Riven.

   Now I'm waundering around in the Selentic age and I must confess that the spell is wearing off. I am beginning to tire of the routine. Wander around, solve puzzles, look at stuff—yawn! This is a problem with many computer games; they wear out quickly. I experienced the same

fatique playing Riven. Eventually you just want it over and you don't care if someone tells you how it all comes out.

The major thrill of a game is the initial sense of mystery—that feeling of being lost in strange world where everything is new. As you play the game you quickly learn about your new world and in no time it's like driving to work.

July 23, 1998

1. I'm tired of Myst for now. It's back to JDict2 coding when I'm done with this skating episode. The last four weeks have proven to be exhausting. Getting up every morning at a quarter to six morning after morning after morning to drive Helen to the rink just grinds me down. Looking after Helen until she gets to bed at 9:30 has been just as draining. I managed to get some JDict2 work done and once I'm back on my normal schedule I'll pick up the pace.

July 23, 1998

1. **Idea:** Sometime ago I had the silly notion of an XML internet based dictionary. I wanted to be able to pull words right off the internet into executing J locales. I was attracted to XML because it makes it possible to bypass browser specific bullshit. Hell it makes it possible to bypass the browser all together. The last thing you want is a multi-megabyte-monster coming between you and a few snippets of text. I still view this as a good idea but this morning I had a better one.

   The APL/J/K universe needs an XML standard for representing general array data on the web. The notion is simple and powerful. Any array could be sent to a web page in XML format. By providing an XML export and import facility it would be possible for array languages to easily trade data among themselves and other tools. This is such a good idea that I'm not going to keep it to myself. I feel a usenet posting coming on.

August 12, 1998

1. I'm in a down-cycle for things binary. My enthusiasm for this project has been way down of late. I know it will pick up again but right now it's hard to argue with the little voices that are constantly nagging:

"You're wasting you time on J. You should be learning Java like everyone else."

"The market for this product is limited—very, very limited."

"Get a real job!"

If I was in the habit of stampeding with the herd I would be alarmed but *just because society deems an activity a waste of time doesn't mean it is a waste of time.*

August 15 1998

1. I spent a few hours today implementing word groups. The amount of code it takes to check arguments, update files and verify the changes is quite annoying. If I gave up error checking there would be a lot let less code. I know that my grouping code could be smaller but not dramatically so. Needless to say I am very suspicous of all these `comp.lang.apl` messages that claim to implement relational database updates in a few lines of K or APL. You can express the essence of the update in a few lines but by the time you add in all the error handling you've got more than a few lines of code.

2. After I implement `put`'s for tests I'll go back to my word group code and modify it to handle both groups and suites. Both cases are structurally identical so it makes sense to use the same code if possible.

3. Given the rate I'm working on this system I've decided to code a set of file checker verbs. The checkers will not be part of the system but having them around should speed up the rate at which I find errors in my database files. I'll write one checker for each file and exhaustively test the directory and the contents pointed to by the directory. I've been planning to do this for sometime but now that I am updating a many files at once the need is greater.

August 18, 1998

1. On the weekend I finally downloaded *klite* from `www.kx.com`. K is Arthur Whitney's baby. Arthur is on my very short list of gifted programmers. When I first met Arthur back in Edmonton in the early 80's he had just left I.P. Sharp after an extensive collaboration with Ken

Iverson. They built a model of an ideal Sharp APL. Many of the best ideas in Sharp APL, and later J, can be traced to this effort. Infact some of the very best ideas can be traced to Arthur.

When I knew Arthur he was busy programming in Nial and talking about what an ideal programming language would be like. I'm pleased to see that many of the ideas he was kicking about have materialized in K.

K has been a mystery despite the rather large amount of `comp.lang.apl` traffic about the language. Everything reported about the language has been very favorable. K's performance has been described as awesome — far beyond what you see in APL and J systems. Arthur had a knack for super fast code when he was at Sharp. In one famous case he optimized a database routine that ran about three orders of magnitude faster than the version it replaced. With K he's apparently carried on the tradition.

Even better K makes extensive use of memory mapped files meaning a K database can simply appear as a gigantic array in memory. This idea is so natural that Eric Iverson and Roger Hui are now adding memory mapped arrays to J. Despite all these pluses and postive rumors most of us have never had a chance to learn K for two simple reasons:

(a) . K is very expensive. Basically if you're not a trading floor or an international bank you can't afford it.

(b) There was no easily available language describing the language.

With the advent of *klite* it's now possible to read a K manual and play with a toy demo system. I'm going to study the system and play a bit. More later.

August 19, 1998

1. **Implemented:** group content queries. A few days ago I added in group content queries to JDict. Now you can issue commands like:

```
grp 'group'   NB. list group members
```

and the system will return a unique sorted list of group members. This is the first data retrieve operation that works properly with paths. It basically returns the contents of the first group found on the path. My next bit of work will be to extend `get` to work with paths. Once `get` is running the dictionary will actually be fairly useful despite being about one third complete.

August 20 1998

1. I started work on `get`. I should finish the word case by this weekend. After `get` it's back to `put` I need to code `put` cases for explain and document text, group scripts and test cases. We're getting there. Just don't hold your breath.

August 22 1998

1. **Implemented:** `get` is mostly done.

2. While coding `get` I realized that `grp, del, chkp` need some work. When a group is formed `grp` has to add the words in the group to a master words in uses list. `del` refers to this list to prevent deletions of objects in use. Currently it happily deletes objects in use. Similarly `chkp` has to empty groups as well as reset the main references file. Otherwise it would be possible to introduce names in groups that are not on the path.

   All of this is to preserve the referential integrity of the database. I thought about screwing integrity and allowing deletes and spurious groupings — there are definite performance advantages — but then I realized I would never be able to implement the dictionary with a relational database as a store. Such systems enforce integrity and getting around it is a counter productive pain.

   However, in line with my new "defeaturing" strategy I have decided to maintain only necessary references. It would be nice to do thing like list all the groups a word belongs to. (The FoxPro `calls` verb does this) but maintaining the references imposes to much overhead. Also, it's a trivial matter to write a little utility program using the result of `grp` to achieve this end.

   So before I fill in more `put` cases I will make these adjustments.

August 27, 1998

1. Yesterday I decided to change the way I was updating the inverted data in my directories. Until introducing `invdelete, invappend, invreplace` a number of verbs were writing to these regions. From now on all updates will go through these three routines. This marginally slows things done but greatly increases safety. The three routines have rudimentary directory corruption detection. While coding these verbs I decided that I would make another change directory design change.

   Basically when inverted data and prime directories are updated the entire operation should be treated as a single transaction. If the process bombs part way through there is an excellent chance the directory will be corrupted. **This is not good!** I've decide to introduce basic transaction processing for directories and inverted data by devoting two components to each list. During an update one set of components will be updated. If all writes succeed an offset bit in the directory mark (component 0) will be flipped. This marks the last update as good. If the process fails for any reason this bit will not be set and the directory will use the previous set of good components. By alternating between component sets I will insure the previous set is always good. The directory bit will be used to select which component set gets read into memory.

   This redundant component technique will double the size of directories but file space is the last thing you should be worrying about these days. A thousand word database will probably come in around 4 megs based on past experience of which a few hundred K will be consumed in redundant directory data. Given the gigabytes of free space on most hard drives these days this is a reasonable sacrifice.

2. I took the afternoon off and saw *Saving Private Ryan* yesterday. It certainly put the trivial software world in perspective. This is easily one of Spielberg's best films. I found it even more harrowing than *Shindler's List*. I'm still disturbed and mulling it over. Despite it's grim portrayal of combat this is not an antiwar film. I'd classify it as a *think damm and hard and make fucking well sure that you have to do this shit* before marching off to the nearest war. I've been considering under what circumstances would I allow my son and daughter to go off to war. It's absurd to say that under no circumstances would I permit

this but one thing is clear. It won't be some cynical policy war like Vietnam or some campaign to make the world cheap for automobiles which, when all the bullshit is shovelled aside, is what the Gulf War was all about.

September 2 1998

1. **Implemented:** I've mostly finished `del` for words, tests, groups, suites, macros and bundles. I was able to generalize the word deletion routine to handle all these cases with an insignificant increase in overhead. I have yet to finish the *in use* protection and I'm not sure if I'm going to implement reference deletion.

   Reference deletion proved very useful in the FoxPro based dictionary. It was mainly used to remove words from word calls word lists so it could be easily deleted without rebuilding the reference lists of every single word that ever called the creature. I may still have to provide such a feature in JDict2 but I'm leaning toward coding it as a utility verb that calls the more primitive dictionary verbs.

2. Defeaturing strikes again. I have removed `rdrop` and `chkp` from the dictionary. My redundant component writes will make it possible to provide a simple one operation undo facility. I'm considering.

```
del ;:'trash these words ehh'

undo 0  NB.  undo last word  directory change
        NB.  will restore deleted words

grp ;: 'thisshit is a group'  NB. make <thisshit> group

undo 2  NB. maybe not!
```

September 4 1998

1. **changed:** I've removed reference deleting from `del`. In a pathed environment this feature would cause more trouble than it's worth. If I need it in the future I will write some utilities using dictionary primitives to remove objects from all groups et cetera. Defeature!

2. `chkp` is back in. After thinking about the undo I realized that the only safe way to provide such a facility would be to keep track on the last operation performed. Way to tedious. Defeaturing says when in doubt throw it out. `rdrop` stays out and I've removed `calls` as well. A few of the planned functions of `calls` have proven useful but, frankly, the underlying file structure of this system is not suited to this operation and I don't want to start updating more indexes and cross reference structures. The few usual functions I planned for `calls` will be provided with utilities.

3. My next task is to implement `put`'s for test cases and `grp`'s for suites. I also need a rudimentary group `make` facility. Once this is in the system will be quite useful.

4. I am delaying the introduction of redudant directory and inverted components. Now that these operations are covered with verbs I can change it later on.

5. I will also **enhance:** the `did` dyad to display the dictionary's reference path. A crucial bit of information that will be hammered into the user's brain.

September 5 1998

1. **Implemented:** path reporting for dyadic `did`.

2. More quandaries about how to handle objects in use. Maintaining in use information in the put dictionary is not sufficient. Consider this: A group is created in a put dictionary that has a sub dictionary on the path. All the grouping information is stored in the put dictionary. All the grouped objects are in the sub-dictionary. Now the put dictionary is closed and objects in the sub-dictionary group are deleted. They will not be detected as in use. Now reopen the original put dictionary and try to get the group. Some objects will be missing.

   There is no way to circumvent this without maintaining a central in use file (which would be very vulnerable to file system screw ups as every task would have to write to it) or by maintaining client dictionary in use information in sub-dictionaries which again runs into multiple write problems. I've decided to live with this.

When two or more dictionaries are on the path preventing in use deletions for put objects will suffice to prevent most screw ups. Programmers will just have to understand that sub dictionaries are like low level code libraries or classes. Very few of us would except a C program to compile and run if we started off by willy nilly trashing the code in stdio.

September 11 1998

1. I've decided to drop in use checking and permit unrestricted object deletions. This can break any aggregates, word calls, groups, suites and bundles but it eliminates the need for special code to handle references.

   This is not ideal but given that I would have to track all references in all dictionaries to "really" prevent in use deletion I have decided that the cure is worse than the disesese. As I noted before this speeds things up and results in less file pounding.

2. I'm now working with the first beta of J4.02.

September 12 1998

1. My decision to drop in use checking will have consequences down the road. Using a relational database to store dictionaries will be a little more difficult as care will have to be taken to insure that word names in a group table are not related to word names in the word table. I may backtrack and reverse this decision if I ever implement this system using a relational store.

2. My K studies are continuing. K is a competitor of J despite having a different focus. I sometimes feel like I'm cheating on my J lover by sneaking around reading K manuals and writing toy functions. There are many thing I like about K but it clearly has it's drawbacks. From my studies so far K has:

   (a) Dangerous arithmetic. When two large integers are multiplied in K you don't get an implicit conversion to an approximate floating point result like virtually every other language. You get what the *chip throws* which is equivalent to multiplication modulo (usually

32). Hence two large positive numbers can result in a negative result. This behavior is very fast but has obvious drawbacks.

K enthusiasts gloss over this by claiming that it will make them better numerical analysts. Yup! I spent three years studying numerical analysis in university. It's one of the most difficult and demanding subspecialties in computer science. Most programmers have a very limited understanding of numerical analysis and continually make incredible blunders as a result. Knowing that K offers essentially no protection for a common class of arithmetic errors I would be forced to view many K calculations the same way I view claims about UFO's.

(b) No intrinsic complex numbers. Unless I've missed something you have to roll your own complex numbers in K. Complex arithmetic is important, particularly when it comes to interfacing with well established numerical libraries like LAPACK, IMSL et cetera. Admittly complex numbers have a limited use in *current* financial calculations.

(c) How do you define operators (APL2 terminology) in K? I haven't figured this out yet. I would dearly hope this is possible. Virtually all other array languages, APL2, Nial, J et cetera provide this key facility.

September 17 1998

1. I started work on `put`'ing tests last night. The test append case `apptests` worked on my first test. This verb is derived from `appwords` and is quite similar. There are enough differences to justify two routines but this will not be the case for the suites in `grp`. Word groups and test suites are structurally identical. I will morph the `putgroups` verb into a new verb `putgs` that will handle groups and suites.

September 20 1998

1. **Implemented:** `put`'s and `get`'s for test cases. I've also added the word `get`'s variation that does not define words in a locale. The test and non-defining word gets are very similar. The same code can be made to get macros as well. Currently I have two verbs but I will generalize and reduce it to one.

2. Yesterday Jack Licther dropped off a SUN ULTRA for me to play with. It's an older (about 3 years) Enterprise series machine with a 12GB disk drive, 340 megs of ram, tape cartidge backup, CDROM, floppy drive et cetera. I don't have a SUN monitor and keyboard and I'm not sure what the documentation is like. Considering I know very like about SUN machines this is a great opportunity to play and learn. The thing has an older version of SUN's unix OS Solarius and there is even a version of the ORACLE database on it. Altogether about $100,000 dollars worth of soft and hardware just sitting on the floor of my basement.

3. How *suite* it is! **Implemented:** suite formation mechanism. I now have to do macros and bundles and all the basic objects can be stored and retrieved from JDict dictionaries. This thing is finally coming together. I am beginning to see the light at the end of tunnel.

September 21 1998

1. **Idea:** I have started marking all public object verbs with the comment string `NB.->` on their first line. This mark will make it a simple matter to write a little procedure that picks out all these verbs and inserts them into the file header.

2. I have decided to add a new user verb to the dictionary `dnc` — dictionary name class.

3. I am begining to gravitate toward adding group postscripts. I already have group and suite *prescripts*. The group postscript would make it easy to define objects (usually nouns) that crucially depend on the definitions of others. For example:

```
PARM1 =: 1            NB. any changes to 1 and 2
PARM2 =: 2            NB. are reflected in 3
PARM3 =: PARM1,PARM2
```

September 22 1998

1. The shame! I discovered a **Bug:** in `didstats`. The verb breaks on an index error when it tries to build a named path for dictionaries that have more than one dictionary on their reference path and not all of them are open. I was getting to clever here. I will have to refer to the master file. To make sure I have a complete list. Infact even that will not do the trick because it's possible to unregister a dictionary that appears on the reference path of other dictionaries. I will therefore: resolve the names I can and then just refer to any missing dictionaries with there complete `DIDNUM` which is always available.

September 23 1998

1. **Bug Fix:** I corrected the `didstats` path bug I discovered yesterday.

2. **Implemented:** I've extended `get` to fetch the name class, macro type, objects size, creation and put dates. Essentially all the inverted data that is stored in the directories. If you store data you might as well be able to easily use it. This feature added a few new codes and essentially one verb `invfetch` to the system.

3. This `get` extension eliminates the need for a new `dnc`. No need - no code.

4. **To Do:** modify `JDict2.doc` to reflect changes to `get`.

September 25 1998

1. **Bug:** I found another error in `dnl` the other day. I had forgotten[1] that I had implemented name class filtering with `dnl` for example:

```
0 1 1 dnl ''    NB. list all adverbs
0 2 2 dnl 'eq'  NB. conjunctions with names containing "eq"
```

   Unfortunately this breaks when you attempt to filter on dictionary objects that do not have name class or type.

[1] When you work on something a bit here and bit there over months it's hard to keep all the code threads in mind.

```
1 1 1 dnl ''     NB. type 1 tests (should yield option error)
2 2 21 dnl 'eq'  NB. tyoe 21 groups (should yield option error)
```

This bug is indicative of suspected problems in a number of verbs. I have always planned to construct a set of test scripts that test every single valid option and a host of invalid options. I cannot test all the valid options until the dictionary is complete but I can certainly build some extensive invalid option tests. I'll do this after I implement documentation `put`'s and `get`'s

September 30 1998

1. Last Sunday I wrote and debugged test cases for `dnl` in the process I discovered and corrected a number of errors in `dnl` including the one reported above. I also found some inconsistencies in my documentation and code. The horror!

2. Testing is like money. You can never get enough of it.

3. Many long months ago, before I decided to abandon ODBC, I briefly considered building a dictionary based on flat character files. These files would be easily indexed with J's primitive facilities. The store would be a simple set of flat files where the row length would be powers of 2:

   ```
   32 64 128 245 ....... 16K (with overflow)
   ```

   To find an object you need to now its length and position number. Once this is known simply go to the appropriate file and read one line. The powers of 2 would insure that disk space was not squandered on blanks. Data would be slotted into a file with a row length long enough. Very large items would be written to a directory in simple binary form. This simple design would be fast but I realized at the time the whole thing is a direct result of the nature of J file indexing so I gave it up.

   Yesterday I realized the same design is ideal for the first iteration of memory mapped files in J. Memory mapped arrays will be large open arrays that can only be accessed by indexing and changed by replacing data at existing positions. To change the size of a mapped array

will require a remapping so they will not be fluid and dynamic like in memory arrays.

So dear people I have decided to implement a second storage class `JDstore` that employs this skeme. I am going to continue with the `jfiles` based store as well. The second storage class will force me to clearly define the object interface. I've also decided to generalize the interface `(z) locale words to support any number of different stores at the same time. Hence I will be able to have a jfiles based dictionary, a mapped dictionary, an SQL dictionary, all open and usuable at the same time. Oddly this is not a lot of work once you have the store class. You simple generate an object of the class and then keep track of what type of store a dictionary is to use the correct object.`

October 12 1998

1. **Note:** today I observed a possible problem with grouping when the path has been hacked. I really need to finish `regd`. I'm beginning to use the system to store important code. Everything I screw up a dictionary during testing, followed by a deletion and rebuild of test dictionaries I have to patch in my `jdict2` dictionary. The first time I did this I observed that `grp` failed because the path did not match. This is correct — the path didn't match.

October 19 1998

1. The first J beta with memory mapping was made available over the weekend. I quickly found two problems that that Eric and Roger have already fixed. Memory mapping offers some interesting possibilities but the current J implemenation has some serious limitations. Currently:

   (a) There is no support for nested arrays.

   (b) I have yet to find a fast way to search large arrays. Using `i.` on a 1,000,000 row name arrays takes 12 to 15 seconds to find one name. This is way to slow to be useful.

   (c) I need to conduct more experiments to determine if operations like `+/` can be performed — I rather doubt it as the first thing J is likely to do is make a copy of the data.

Still, even in it's limited form mapping is the fastest way to get at and modify large data sets. I've already tried it on 100 meg arrays and reading the 10,000,000 row of a table is just as fast as reading the first.

2. For the last few days I've been distracted by Stephen Ambrose's *Undaunted Courage*. This is a fascinating account of the journey of Lewis and Clark. I was up all last night following the expedition to the head waters of the Missouri river which is near Three Forks — just down the road from Livingston Montana where my grandparents lived during my childhood.

   For someone born in Montana Lewis and Clark has always been more than dull history. Livingston Montana has a city park named Sacagewa, after Lewis and Clark's native guide. The Yellowstone, traversed by Clark on the way home, cuts through the town and winds its way past the country home of my maternal grandparents. Two hundred years ago the expedition passed within a stones through of where I spent the happiest summers of my life. Montana is still a great place but reading this book makes you realize how incredily luckly Lewis and Clark were to see the West before it was savaged by our environment destroying civilization. Replacing vast herds of elk, deer, buffalo, wolves, bears, free flowing rivers, open paries, with McDonalds's, gas stations, shopping malls, landfills and Bingo palors seems like shitting on an altar. One AMAZON.COM reviewer of *Undaunted Courage* confessed he was "intensely jealous" Lewis and Clark for what they were priveleged to witness. I have to agree.

October 22 1998

1. I finished *Undaunted Courgage* that's one distraction out of the way.

2. Oh, I've decided that the current version of J mapped arrays do not offer compelling advantages over J component files for `JDict` I'm going to stick with my component design until I finish the dictionary.

December 7 1998

1. I am finally finished with my Reuter's y2k contract. It was very very good from a money point of view and I'm glad I did it. As Michael Douglas said in Wall Street, "Greed is good." Unfortunately y2k work

is well dull, dull, dull did I mention dull. Thank god the year 2000 crisis will soon be behind us. The entire mind numbing y2k mess has been a good learning experience for the software industry. The lesson is simple:

$$shortcuts = fuckups$$

. We really saved a lot of time and money with those two digit dates. From now on whenever you take an obvious kludgey shortcut to save a few cycles or avoid dealing with complexity you cannot be bothered with, like the Gregorian Calendar, keep y2k in mind.

2. I will now have a fair amount of time over the next two months to focus on this system. There is still a lot of work to do. I will not release `JDict2` on an innocent word until:

   (a) The system is complete and I have used it for at least a month.

   (b) A few beta testers - if I can find some - have played with the system and rendered their verdicts.

   (c) All necessary documentation is finished and available in a format that can be widely read on the Internet.

   (d) My J word library has been converted to `JDict2`.

   (e) The `JDict2` code is stored in a `JDict2` dictionary.

   (f) There is a good test suite that can be distributed with the system. The test suite will be stored and generated from a `JDict2` dictionary.

December 13 1998

1. So much for my free time over the next three months. On Friday I took the train into Toronto primarily to visit Eric Iverson. I was expecting a social visit but it turned into a contract. I have agreed to replace the native ODBC support in J with direct calls to the Windows ODBC API. For this work I get $2500.00 at the end of February or March. This is not a lot of money. Especially when I factor in the expenses. I am going to be ordering an NT capable 400 MHZ, 196 MEG, 6 GIG machine either tomorrow or the next day — plus install a peer-to-peer LAN to tie together the ever increasing PC population in this house.

And, we're not done yet, get ahold of the workstation edition of SQL-server and use it for my ODBC test development test bed.

I am spending the money now so I can claim the NT machine on this tax year. I need every legitimate deduction I can muster. I have been planning to get into NT and SQL-server for months now. This opportunity just made me do it now rather than later. If I do a good job my work will eventually show up on the J distribution disk and find it's way to every J user on the planet: fame and glory! More importantly, I'll be albe to give Eric as reference. His opinion counts for a lot in the APL world and probably beyond as well.

With Eric's contract and the work I have at Revenue Canada and the DHC plus my last check from Reuters plus my Reuters bonus I have revenues of $12,000 for the first two or three months of next year. I can probably scare up another $2,000 to $5,000 from the DHC. I will continue looking for a solid three to six month contract in the Eastern Ontario region so I can borrow money and make a huge RRSP contribution to keep as much tax money out of the Governments filthy hands as possible.

2. My J/ODBC/API will have a benificial work on `JDict`. An SQL-server based store for this system has been on my mind for ages. I have not undertaken it because of the limitations of J/ODBC, the lack of an SQL-server system to test on and lack of time. Soon I will have all the materials and the time to tackle this. Stay tuned.

3. **Bug Fix:** I fixed a problem I introduced when I reorganized the dictionary to consist of a main dictionary object that contains a storage and maker object. The directory object references did not always point to the put dictionary. `put` is now, once again, only modifying the put dictionary.

4. Another thing that came out of my visit to Eric's was the verb `freedisk` that makes a Windows API call to determine the total free space on a volume. I need this number to determine if:

   (a) I have enough space to create a new dictionary.

   (b) I have enough space to perform a backup `packd` operation.

January 13 1998

1. It's been awhile since my last entry and a lot has happened. I have three consulting projects on the go with two more solid possiblities in the pipeline. I'm working on:

   (a) Eric's ODBC project

   (b) A DHC Ambulance statistics project

   (c) A software fraud investigation for Revenue Canada.

2. The ODBC project is the most work by far. So much work that I am shutting this entry down to get back to it. I am hoping to meet a self imposed deadline by the end of this week.

January 22 1999

1. I have been using my dictionary for a number of projects I am working on and I have *discerned an irritant*. My procedure of setting a bit in the master file when I open a dictionary readonly to prevent other tasks from mucking with the files when one is using them is certainly a good idea from the all important **DON'T FUCK UP YOUR DATA perspective** but unless I explicity close the dictionary before ending the task — something I always forget. The next time I load and try and open the dictionary I get the other task opened message. A simple reset fixes this but it's annoying. It's also annoying to constantly open your main dictionary. Hence I've decided to add a profile script to the dictionary. The profile will be an arbitrary J script that is executed after the dictionary code loads. I can put resets, default opens and other commands in the profile. This will eliminate the irritants and greatly increase the flexibility of the system,

Feburary 8 1999

1. I am almost done with my ODBC replacecement script a few more hours and it will be finished.

March 3 1999

1. I have finally managed to get some dictionary related work done. Yesterday I got feed up with my various work projects and devoted the whole day to dictionary work. I have implemented part of `globs`. I

also fixed a problem with my name test. It was failing on indirect locale references like `BOO__HOO` because when you evaluate the name class of such a word a noun `HOO` must exist in the current scope. This is an unavoidable side effect of the way indirect locales work. You don't know the name class until you evaluate the noun and find the word. It's classic late binding.

Anyway I know have a regular expression based method that works with out evaluating name class. I still have to implement the most common indirect assignments e.g.

```
'these names are local' =. y.
'and we are global' =: x.
```

These types of assignments are very common in J code and my current method fails to find them because the names occur in quotes. I have typically "declared" such names with my `(*)=` convention but this is a pain, especially when dealing with other peoples code. Handling these names should be diffcult just look at the token that preceeds `=.` or `=:` and extract the names it it's a quoted name list.

2. I have finally come up with a better name than `JDict2`. From now on this system will be known as **JAR**. You know like cookie `JAR` or **J AR**`chive`.

March 6 1999

1. I got another assessment notice from those goddam thieves at Revenue Canada yesterday. They want another $1,300.00. That's two assessments for 1997. I have a feeling the tax people basically hate the self-employed because we have some options. It's actually possbile to keep some of the money we earn espcecially if we earn it out of country. This little bit of tax sodomy is not as bad as I first thought. Ruth gets an $800.00 refund but we're still down $500.00. God I'm tired of paying for lesbians on welfare and all the rest of the policitcally correct money wasting shit that the gutless Canadian government finances.

2. I have been making progress on `JAR`. The first `globs` case is down. I've solved the quoted assignment problems and implicit `for.` locals. My

new identifier analaysis is a bit slower than the old method but it has the nice property of producing the correct results. This is probably the only reliable J name analysis tool on the planet. I've read messages about other people attempts but they don't handle all the exceptions and complexities. God I'm good.

March 7 1999

1. Sometimes your problems get solved by other people. I my case it doesn't happen very often but this last weekend I learned about some J code that Kirk Iverson and Oleg Kobchenko wrote that goes a long way toward solving a problem that has been sitting on my back burner for sometime.

   I have longed for a means to take J programs and convert them into structured web documents. Basically combine the documentation, cross references and other scripts into one literate program product.

   As you know I am a big fan of literate programming as pioneered by Knuth but I am also aware of it's problems. The main problem is that computer programs are not read like novels or short stories and rendering them as books or articles has not proven very successful. Fortunately a new style of reading computer programs has slowly creeped on the scene that is much better suited to the material.

   When we read a program we want to quickly find out what's relevant to the parts we're interested in. For large programs, (hundreds of thousands of lines), noboby is going to sit down and read everyline. Still we have to deal with these beasts which means designing our software so it can be inspected in small comphrensible chunks.

   When it comes time to read the chunks two problems crop up over and over. Relationships to other parts of the program (considered relevant by the authors) is often not stated or lost and the sheer mechanical drudgery of searching source files for the relevant bits. If you structure your documentation as a series of linked web pages you (maintain relationships) and simulatanously provide a quick way to jump around in all those lines of code.

   Of course if it's a pain to structure source code in this fashion it won't get done. That's were the source to HTML systems like Oleg's and Kirks come in.

More later.

June 7 1999

1. I'm back! I know you didn't miss me. For the last two months I have been in Dallas Texas working in APL2. About a year and a half ago I wrote a complex APL2 system for Blue Cross of Texas. The fellow I worked for decided it was easiar to bring me back than attempt to change the system himself or find someone with my guru level APL skills. I appreciated the work. It was an easy $6,000 US and it gave Ruth and I a break from each other. This system is probably my finest work in APL. It's a very reliable program that hasn't crashed in two years of production work. It does an important complex job for an large actuarial department. I was brought back to change a few things that people wanted done differently but everyone was very happy with my work.

   While I was away I was offline, except for email, and I was away from J programming and JAR. During my absence beta tests for J4.03, which include my ODBC interface, were conducted. It was the first round of J beta testing that I haven't been involved in for five years. I missed it but I am glad to see that others are picking up the slack and testing the system in ways I would never try.

   My ODBC interface script had two little problems others found. I made a typo in one of my type conversion verbs and my database connection verb did not properly handle passwords for Oracle. It's hard to test on systems you don't have! Chris Burke, very expertly, found and fixed both problems. Otherwise no complaints and a lot of compliments. In Chris's words the ODBC script is great.

2. I've set up the JAR project under J4.03. The new release has made some significant improvements to the system. The major change is the editor. Prior to this release J relied on Window's text controls for editing. The problem with this approach is the limited (¡20K) size of the files you can handle and the nagging differences between various flavors of windows. NT allows much bigger text controls than 3.1/95/98 and WinCE. The new editor makes it possible to edit large J files on all flavors without change. The editor also supports syntax coloring and brings J up the level of the spiffy C++, Visual Basic and Java editors that programmers are getting used to.

July 20 1999

1. **Bug:** the delete command breaks when deleting groups that have no members in the put dictionary. **FIXED** this error had nothing to do with the `del` command and instead was caused by bug in my code compression tool. It was not properly excluding single character names and the compressed code was overwriting some single char nouns with wrong values.

September 29 1999

1. It's been month's since I touched JAR. I've been busy with other things. Mostly a long vacation and the near breakdown on my marriage.

   Don't you just love these programming notes?

2. As for JAR. I've discovered a **Bug:** in the access control mechanism. The first time a JAR dictionary is loaded after immediately loading JAR. The master file doesn't seem to get updated properly. Hence, if you start another JAR task you can open the same dictionary in the second task read/write. This is not the way it's supposed to work.

October 14 1999

1. The access bug I discovered is not a bug. My JAR profile was set to wipe the status bits when a new task started. It was annoying to have to constantly reset the master file because I forgot to close dictionaries before leaving J. When more than one JAR task is active you don't want to reset this file. The bits were being properly set and reset. A better approach would be to find some way to determine if another J task is executing. If the no task - go ahead and reset the bits else leave them alone.

October 15 1999

1. Yesterday and most of last night I had a good old fashioned *codeathon* I started working on JAR and just kept going. I found and fixed a bug in the compresson routine. Changed the `JARutil` class to include the compression tool instead of augmenting the class later on. This is a simpler way of doing things. It blows a few bytes of memory and

storage but simplifies the compressor. I added a set locale facility for the JAR edit procedure. I edited documentation in words. Modified the contents of various dictionary groups. Revamped the build scripts that are used by the J project manager. Fixed a sorting discrepancy in `getobjects`. I created the order of magnitude test dictionary: a dictionary roughly 10 times larger than a reasonable code dictionary. The order of magnitude dictionary contains roughly 12,000 words. The `put` command was able to write all 12,000 words from a name, class and script table. This is beyond what I wanted it to do. I tested the `packd` verb on the order of mag dictionary. It worked despite not using a work loop. Basically this stuff is already pretty capable and robust but I'm aiming for something more.

2. **Problem:** I noticed while tracking down a problem in my JAR build scripts that different JAR commands often return things in different orders. This is not good. This broke my JAR build script when I started developing JAR with `jardev` and `jar` on the path. Prior to yesterday I mostly had only one dictionary on the path. Since the system is designed to use more than one dictionary I thought it prudent to start working with more than one. This is eating your own dog food as they say. Because objects are returned in path order sorting problems show up more. I am going to change all get operations to return objects in the order requested if the request is in the form of a boxed list of names. Obviously when I ask for a group I will impose an order but when a command is given a list it will always return objects in exactly the way they are asked for on the list. This is a simple easy to comphrehend rule.

October 16 1999

1. I think I've fixed all the occurrences of `get` so that all objects are returned in the order requested.

2. **Bug:** I noticed that the word count reported by `did` for my order of magnitude dictionary (OMD) did not match `#}.dnl''` After some experimenting I found that when `put` is given word lists with duplicates the count includes the duplicates. I changed `putwords` to nub it's names. This fixes the problems for `putwords` but I'm not certain what happens with the other put routines. I will have to check them out.

As I do I will add test cases to insure such errors never occur again. Adding test cases also allows me to test the test handling parts of the system. More dog food.

3. I checked the other verbs and discovered that duplicate put names are dangerous. When putting tables they can corrupt your directory. I accidently did this when putting name,class,value tables back into `jardev`. I had to restore the last backup I had made of `jardev` with `packd`. I lost a little work. But I have changed the system to:

   (a) Tolerate duplicates where they can't do any harm. In the case of multiple word puts where many names are refering to the same thing I just nub the name list and proceed. The same for deletions where many names refer to the same dictionary object.

   Test Case: It is possible for a single object to have many names in J. For example:

```
NB. object word is defined in a class object which
NB. is instantiated.

word_class_    NB. refers to object
word__obj      NB. refers to object
word_n_        NB. where n is some integer refers to object

word    NB. if the current locale is the class or object
        NB. the name by itself also refers to the object
```

   JAR should bar all word references that that contain locale references and only admit the last case.

   (b) In the case of name value tables you cannot do this because the same name can have different values. When you are replacing existing words the last value is used. For new words the directory gets trashed. In either case the result is confusing so I return an error and let the user fix his argument table.

4. I still have to finish `makedump` to complete the non-GUI part of JAR. `makedump` will just dump words, test cases and documentation from

the put dictionary. More elaborate dumps will have to be programmed using the systems commands.

October 29 1999

1. This last week I installed JAR on a number of Adaytum computers in Toronto. I also made a number of changes — mostly to display and edit utilities `ed` and `disp`. **Note:** `ed` and `disp` should display a high degree of symmetry. Generally if you you can display it you will probably want to edit it. I will have to build a number of test cases to illustrate this principal.

2. I am planning on using JAR as a tool and as an example of a non-trivial, well documented, J system for others in the office to study and learn from. JAR reflects some of my programming guidelines, ie:

   - Programs and Test Cases are of equal importance.
   - Test suites should be at least 10 times larger than the system it tests. *The order of magnitude maxim.*
   - Documentation and organization is crucial for large programs.
   - When ever possible programming tools should be used to construct themselves.
   - When in doubt "defeature." There isn't a chunk of software on Earth that couldn't be improved by removing unneccessary, redudant or clunky features.
   - Good systems run well on datasets 10 times larger than a typical dataset and continue to function without breaks on datasets 100 times larger than normal. *The two orders of magnitude rule.*

   Of course this applies to all programming tasks but my programmers never see it consistently applied and tend to cut corners.

3. I've cloned the JAR development environment at Adaytum with the one exception of this log file. I'm sure I don't want people reading about my marital problems or raging bisexuality.

November 5 1999

1. Another week in Toronto and more JAR work. It always helps to move a program to a new system environment. It always exposes problems. This week I spent sometime in the U of T residence hotel writing test scripts to generate the order of magnitude and second order of magnitude test dictionaries. Generating and then regenerating the 2nd order dictionary exposed a bug. The Total J memory used reported by the `9!:` verb overflows and wrap when consumed memory goes above 2 gigabytes. The memory used is reported as negative number which hosed the next verb. For my app the only important thing about this number is that it's unpredicatable. I simply floored and took absolute value. I'll report this to Roger because it is a genuine J *anomaly*.

2. I revamped the distribution `*.bat` files to include the `jardev` dictionary. So many changes have been put into `jardev` in the last three weeks that generating `JAR` source from the JAR dictionary alone produce out of date source.

3. Revised the JAR lab to use `jardev`. It better illustrates how to use this system anyway.

4. Used Word 2000 to generate `*.htm` versions of JAR documentation. If your Word document is simple this works quite well. There are things that could be improved, like tables of contents turning into links to the sections they refer to! Oddly this feature works for footnotes but not for tables of contents or page references, which incidently lose meaning in the `*.hmt` format. Probably another case of Microsoft selling something before it's done, or if you want to be cynical, withholding an obvious and necessary feature to force customers to buy Word 2001.

5. I gave JAR to Chris Burke and he installed it on this machine. There is a good chance JAR will be included in the next general release of J.

November 21 1999

1. More JAR work. I have changed the way I generate unique extended precision integer `DIDNUM`'s. I'm now calling the Windows `GUID` *global unique identifier* dll. The GUID is based on the machine's network card identifier and the system clock and Microsoft claims it will be unique worldwide for at least a hundred years. This change also sidesteps the

overflow reported above. The GUID algorithm is slower than my home grown method (about a millisecond more) but it's *probably* a better unique id.

2. I've thought of a way to use the DIDNUM's to solve a nagging problem with master file resets. If you open a JAR dictionary `read/only` the master file is set with a this dictionary is open bit. If you forget to explicitly close the dictionary when you exit J the dictionary remains marked read only. This blocks you and other users. For sometime I've been issuing a global reset in my JAR profile to overcome this irritant. Unfortunately global resets will not be appreciated when the master file is on a network drive and more than one user is using it. What I plan to do is to change the reset so that there are `RESETME`, `RESETRW` and `RESETALL` options. The `RESETME` will use the current `JAROBID DIDNUM` and a list of the last ten or so `JAROBID`'s to reset only those dictionaries that have been opened read only on the user's machine. Hence the importance of implementing a `DIDNUM` that will never clash on a network.

3. I'm also making one more change to make JAR more multiuser friendly. I'm going to provide `dpset` option that will mark the current put dictionary as `read/only`. This setting will make it impossible to open the dictionary read/write after it is closed. This will make it far more difficult to accidently damage network library dictionaries. To insure that I don't lock myself out of such dictionaries I will store the DIDNUM of the task that set the dictionary read-only in the JAR directory of the machine that set readonly on. Only a `RESETALL` will clear such locks. `RESETRW` will clear all dictionaries that are not marked read only.

December 6 1999

1. JAR will now be included in the next official release of J. Chris Burke will include it. This is a great deal for ISI. They get a nice little utility for free. I also benefit as my work will be visible to the entire J world. Hey it's not a big world but it's composed of people with a clearly demonstrated taste in software.

2. To prepare JAR for official release I have to make a number of changes and finish the `*.pdf` documentation — especially for the utilities.

3. changes to make include:

   (a) Implement `RESETME dpset` option.

   (b) Implement `READONLY` and `READWRITE dpset` options.

   (c) Change `newd` to create dictionaries in a default J user directory, eg:

   ```
   \verb?c:\j404\user\jardicts\dictionary
   ```

   (d) Complete documentation

   (e) Modify lab to reflect changes in locations and install procedures.

   (f) Modify install to reflect fact that jar will be a default load unit.

   (g) Change initial object creation verb to test z locale for clashes.

January 9 2000

1. It's a new millennium, (in origin zero of course), and I am still working on JAR. All changes I listed in the previous entry have been made as well as some changes to fix a rather serious `regd` bug. It turned out that I couldn't register dictionaries that were placed on read-only drives because the registration process attempted to write to the parameter component of the word file. This also exposed a bug in the J `jwrite` verb. It crashes when attempting to write to a read-only file instead of properly returning a negative return code. If the JAR dictionary is set to `READONLY` it now properly registers dictionaries on read-only drives. It still crashes because of the `jread` bug when the dictionary is `READWRITE` but should return an error once Chris has had a chance to fix `jread`.

2. JAR did not make it into the first J 4.04a release but should make it into 4.04b or one of the 4.05x releases.

3. **Scope Change:** I have abandoned my JAR GUI browser plans for now. My original plan was to use the J grid class to implement a JAR GUI browser. I am not going to do this. If I ever get back to this I will take the ATL COM object route that my current employer **Adaytum Software** is taking with the J component engine in **e.planning** and

e.analyst. In this approach a J system is basically given a C++ ATL COM interface which turns the code into a true COM object. This would be a good little project for me to learn a not more about C++, COM and ATL.

February 7 2000

1. Last week I discovered a bug in `jnfrblcl` when saving some of Bob Bernecky's code in JAR. He was using a complex argument `_1j1` to specify a format option. My name extractor saw the bogus name `j1` as an apparent global. There is code in `jnfrblcl` to scrub out tokens that begin with numerals but I forgot to include the `_` sign meaning negative J values like `_1j1` would survive to confuse name extraction. By adding one character to my kill list I fixed this problem.

April 3 2000

1. **Bug Fix:** I found a genuine bug in JAR while using it last week. If you delete a word that has references from the put dictionary and then latter insert a word with exactly the same name in a dictionary lower on the path the old references in the put dictionary are picked up. You must delete put dictionary word references when you delete a word. I remember considering this case when I was agonizing over what to do about broken references. When I decided to allow broken references I obviously overlooked this simple confusing cleanup.

2. My handling of group and script texts upon regrouping needs some work. If a deep path group, with a group script, is regrouped in the put dictionary the group script is not copied to the put dictionary. This is a "feature" not a bug. I choose to avoid copying to speed up the group operation and to allow maximum flexibility. Having been bitten a few times with this "feature" I've decided that I should copy group and suite scripts when:

   A group exists on the path but not in the put dictionary.

May 9 2000

1. I made a **Change:** to the `od` verb today to display the complete root paths of dictionaries. There is now an option 4 code, for example

```
NB. display dictionary names and root paths

4 od ''
```

March 19 2002

1. Useful software just keeps on getting used. I thought I had retired this log two years ago but here I am in 2002 adding new entries. What drove me to resurrect these files — thankfully written in TEX and consequently immune to software *upgrades* — and dust off the skeletons?

   It's simple. JAR is very useful. I use it almost every day. When I am coding in J it gets a heavy workout. The system is very stable and reliable. I have never lost anything I put into JAR and despite heavy use the binary database files have never been corrupted or damaged. My decision to keep files closed at all times and buffer directory objects has proven very sound. I take a performance hit when I open and close files but I am now running JAR on a 1.7GHZ machine with a gigabyte of memory. JAR performed very well on much slower machines. On current hardware it flies.

   It's primarily because of new hardware that this log is now active again. New PCs are now being shipped with CD-RW drives and Windows XP provides direct support for CD burning. You drag files onto the CD and press the burn button to make a CD. Sneakernet has returned big time! For moving masses of data burning CDs is now competitive with high speed networks. I have been keeping my working JAR development directory in synch on a number of machines by burning CDs. It's simple, it evades corporate security polices, it's cheap and you always have a stable backup disk.

   There is only one thing I find irritating with this brave new CD burning world. Windows XP sets all the file attributes on burned files to *read-only*. This exposed a number of bugs in JAR. JAR expects to find its files in a *read-write* state. I seldom dick around with file attributes. It's a swamp I have long avoided! Unfortunately, if I forget to reset attributes on files copied off XP CDs JAR immediately crashes when I attempt to read, write or register databases. This annoyed me so

much I fixed JAR to check attributes when dictionaries are opened, registered or set to `READWRITE`. It was a simple fix but it distressed me to find such a *glaring* error in chunk of software I wrote and depend on!

This is something I have seen over and over. When software migrates to new OSes and new hardware long latent bugs often crawl out. The J system itself is now under going through such a process. Iverson software is investing considerable energy into getting a high quality port of J to Linux. Numerous long standing anomalies have now been flushed out as Eric and Roger attempt to make J behave the same way in Windows and Linux environments. This will be good for J.

2. *Anywhoo* my little attribute bug was not the only thing I've fixed in JAR during the last two years. Here is a short list from my incomplete notes.

   (a) Now using the `/jrt` parameter in the install file to suppress the display of the session manager - makes things a little cleaner and hides the fact that a J script is run by the JAR install. Requires J4.05 or later so will hold off for now. **DONE in JAR2.EXE**

   (b) Added a section on (packd) and (restd) to the main JAR introduction log. (DONE)

   (c) Fixed the programming JAR utilities lab so the embedded locale references error does not come up. (DONE)

December 17 2002

1. J5.01 finally shipped. Porting everything to Linux and making it work turned out to be a much bigger job than anticipated. Every software project is bigger than anticipated. Some of the changes in J5.01 broke JAR. A few weeks ago I made a new install executable but just a few minutes ago I discovered an issue with my distributed scripts. So dear software diary I thought it was time to make a few notes of recent JAR changes.

   - Updated distributed JAR labs to run in the 5.01 enviroment.
   - Modified the install scripts to run with `JCONSOLE.EXE`. This is far cleaner, simpler and more portable that using the the `/jrt` parameter of `j.exe`

- Changed the startup scripts to use new `jsystempath` verb.
- Compiled new install.
- Added headers to distributed utility scripts to set the load locale. This was necessary because of changes in the way the J `load` verb.

2. Aside from these minor tweaks JAR runs like a race horse under J5.01.

June 11 2003

Another major version of J (Version 5.02) has been released and once again I have to crack open the JAR and update the system. This time changes to `load` path handling in the new J release broke JAR.

For this update I:

1. **Removed the `*.exe` install program.**

   About half of the J users liked the `*.exe` install and the other half hated it. The haters didn't like the idea of running an `*.exe` on their pure systems. Who knows what evil lives in the hearts of `*.exe`'s. These same people don't mind running arbitrary J scripts. A J script can easily destroy entire networks of computers and would probably be the evil genius weapon of choice if only more evil geniuses knew about J.

2. Updated my regression tests. The `[.` conjunction was removed from J. This broke a number of word definitions in my master test data file `allwords.ijs`.

3. Modified all JAR labs to run with the new path system and to setup the JAR system if it hasn't been initialized on user systems. This is the Jsoftware approved method of initializing and documenting a J system addon.

4. Wrote a master JAR macro to run all my tests: `REGRESSIONTESTS`.

5. Changed my build procedure to create zip files that can be unzipped into `addons\jar`.

June 25 2003

There was a minor clean-up release of `J502a` today and as usual it broke something in JAR. This time it was my fault. I forget to include some `jpath`'s in one of my labs, e.g.

```
NB. load project manager
NB. Note: must use (jpath) to reliably resolve J relative
NB. path references which are denoted with the leading ~ character
require jpath '~system\extras\util\project.ijs'

NB. set werewolf project file and start project manager
openp_jproject_ jpath '~addons\jar\jarprj\werewolf.ijp'
jijs_runprojman_button_jijs_ ''
```

I've fixed this nuisance for myself. I won't send an update for to Eric until `J502b` is released.

June 27 2003

I have made some changes to my JAR tools. JAR is designed to be highly programmable. Over the last few years I have written a number of small tool programs that call the core facilities of JAR to perform various tasks. My tools were spread over a number of group scripts. I have consolidated all my tools into one new class script that extends a core JAR class when it loads.

The new `jartools` class also includes a new tool that fixes one of my long standing JAR irritants and also integrates JAR generated scripts into the standard J system `load` system.

When I designed JAR I angonized over whether aggregates like groups and tests should have both a preamble and a postamble. My first instinct was to have both but I restrained myself and only implemented preambles. Usually the "when in doubt throw it out" design maxim negates kludge construction but it can also suppress useful features. Postambles are definitely useful. The typical structure of a J load script is:

```
NB. header (preamble) lines to declare load class, target
NB. locales et cetera.  The JAR preamble handles this region
 ...
 ...

NB. (definitions) that are typically but not always passive. JAR
NB. generated definitions in this region are passive. Passive
NB. means code is not executed here.  General J scripts can
NB. of course RULE THE UNIVERSE here.
 ...
 ...
```

```
NB. tail (postamble) used to initialize and activate previous
NB. definitions. JAR does not directly support this region
 ...
 ...
```

I have been handling postambles by defining a related script as a JAR macro and then using various script kludges to glue generated groups and this postamble together.

The new verb `mls` (make load script) now does this automatically. It generates a JAR group, looks for `POST_` macros and appends if found, then `mls` writes the generated script to the JAR script directory. Finally, for the default case, it modifies the J system `scripts.ijs` file to make the JAR generated script immediatly useful as a standard J `load` script, eg:

```
NB. load jar and tools
load 'jar'
load 'jartools'

NB. generate JARaddon class script and put on scripts.ijs
mls 'JARaddon'

NB. can now load JARaddon like any standard J script
load 'JARaddon'
```