First I tells them what I am going to tell them.

John Daintree: “What you expect to work will work.”

Deitrich Bonhoeffer – Jesus verses the pharisees. The essential difference was that the pharisees spent all their time drawing ever finer distinctions between right and wrong. Jesus just had to fulfill his calling without asking whether it was right or wrong.

Today I feel as if I have to play the role of a pharisee. I have to clarify what you have a right to expect.

That is the last religious reference until we reach the end.

How we got here. Visits to empty cathedrals. Parties, raves and other ephemeral events.

What 64 bit promises. What we are delivering with version 11.0

Where the work has been done. Data conversion issues.

Back in 1981 two guys in their early 30s.

Maidenhead. Railway bridge. Isambard Kingdom Brunel. Flat arch brick bridge.

Zilog, Z80, CP/M

Z8000, 16 bit, 2 segment, 64k code, 64k data

In C. No typedefs and no Enums.

Launched version 1 at the conference in Washington in April 1983

Before that we had done our first port. To a 32 bit minicomputer called a Gould. I can remember hardly anything about it except that it needed rigorous alignment. In particular floating point doubles had to be on 64 bit boundaries.

The code written for this paid enormous benefits later when the RISC revolution occurred.

Over the next few years we ported to a large number of Unix machines

Most were based on Motorola 68000 chips. Including a machine from IBM. They marketed it as a scientific workstation. I forget its model number/name and I could not find anything about it on the net.

The Xenix 286 port was an attempt to get into the mass market. 286 boxes were selling well. Xenix was a Microsoft product. However, it was less advanced than the Zilog and delivered a poor APL environment. After this we can forget all about 16 bit Unix machines.

Contrary to all of the current magazine articles. The first WIMP (windows, icons, mouse, pointer) machine that was on general sale was the Three Rivers Perq. It was at least a year ahead of the Apple Lisa which was the predecessor of the Apple Mac. Most magazines seem to jump from the research at Xerox Parc to the Lisa.

The Pyramid and the Ridge were based on AMD bit slice processors. The Amdahl is IBM 370 compatible. The Vax was all its own.

We have forgotten about Unix 16 bit. It now exists only in the memory of a few grey haired old men.

DOS 286 boxes were selling well.

Opus produced a board that enabled you to run Unix on top of DOS. It used NatSemi 32032 chips, or 32016 if you were parsimonious.

Since this meant adding a more powerful processor with its own memory it was not cheap. Its market was destroyed when Intel introduced the 386

The 386 changed everything. Now there was a 32 bit Intel chip with the market behind it.

SCO had taken over Xenix from Microsoft. The APL on SCO was quite successful

DOS was still a 16 bit operating system. It didn't see any difference between a 286 and a 386.

PharLap produced a product that enabled the switch into 32 bit mode

Together with Metaware who produced a C compiler they enabled us to produce a quite powerful APL environment.

On the Unix 32 bit side RISC chips began to dominate

The Sequent, which was based on multiple 386 boards was the exception.

There now enters a new column. 64 bit Unix. DEC introduced the Alpha. I have not met anybody who encountered this who was not an enthusiast for it.

Our APL for the Alpha involved sweeping edits of the code forcing all “int” to become “long int”. It was not possible to bring the changes back to the main development stream. Nevertheless the Alpha port really did deliver on its promises.

Meanwhile Microsoft produced its own DOS extender which enabled 32 bit code to run on its 16 bit operating system.

John Daintree used this to produce our first Windows version

Microsoft had recruited a team from DEC to write a new operating system. I was an enthusiast for NT from the moment I saw it. It provided a proper platform on which to use APL.

MainWin provided a Win32API compatible set of libraries that enabled us to port the GUI interface back to the Unix environment

Now we get our chance to add another column. Microsoft has at last given us a 64 bit environment.

We have been using it and an IBM RS6000 to develop a new fully integrated 64 bit port. 14 years after the one for the DEC Alpha.

The address space afforded by 64 bits is 18EB. Exa is 10*18.

The operating system won't give you that much but it will give you far more than you need. Yet. You can buy Windows boxes with 16GB. Unix boxes with much more.

The Alpha version did give us large arrays

We have currently restricted array sizes to 4G elements and individual shape elements also to 4G

The Alpha version did give us large integers but with some problems. It made them []CT sensitive and you could not rely on converting an integer to double and back again without loosing precision. For this release we have ducked these issues.

This common code is known as “partner”.

It is used where individual APL processes need to exchange data. Those processes need not always be on similar machines.

Not all of the exchanges carry information about the originating architecture.

Some do. Up until now the only one used has been APL sockets.

Socket interactions are typically two way

Version 10.1 supports socket interactions which involve byte swapping between big endian (sane) and little endian (insane) boxes.

Now we are in a world where that exchange may involve changing the data between a 32 bit APL and a 64 bit APL.

Worse we may have to do both byte swapping and width changing at the same time.

With version 11 APL at both ends – this all works.

Components as files worked well on operating systems with hashed directories. For us this meant BSD on the SUN

For other machines accessing one of many components became a slow operation

Customers and even Dyadic staff didn't like using the single []FF system function. So 1985 saw the release of a completely rewritten file system

The DEC Alpha had 64 bit everything including files. However, its files could not be read or written by anything else.

Files were interchangeable between 68000, SPARC, RS6000, HP PA-RISC and Silicon Graphics MIPS

Files were also interchangeable between Intel, NatSemi and DEC MIPS

However, this interchangeability was something of a fluke just because the architectures were the same

Whilst components now knew what had written them interoperability was still constrained to similar architectures. Which by this time meant two entrenched camps. Big endian Sun Solaris and IBM AIX vs Little endian Linux and Windows

Now for large files only there is full interoperability. Not only are components self conscious but the APLs can read each others components

Version 10.1 supports 64 bit component files. Do not confuse this with a 64 bit APL.

I will refer to these as large files.

Version 10.1 still wrote 32 bit data into these files.

We did take the opportunity when they were introduced to improve the information held in the encapsulation for a component. The sort of information reported by []FRDCI

Version 11 like Version 10.1 still creates small files by default.

For small files:

Components are forced to be 32 bit on write.

64 bit APL narrows the data on write.

Byte swapping on write is not supported.

We were going to be harsh and only write simple data. No []ORs. However, our QAs were riddled with tests for []ORs. It was easier to put in the support than change the QAs.

For large files:

Each APL writes data in its native format. In the same way as each APL writes socket data in its native format.

The reader decides if it needs to do translation as the data is read.

This means that a component file can contain components written by different APLs

Obviously there is information on the file that is used to find individual components.

This information is maintained in the format of the machine that created the file. If necessary it is converted on output as well as input.

There are obviously some problems when some users are running 10.1 and other users are running a 64 bit version 11

A 64 bit version 11 will write components to a small file.

It cannot put the necessary encapsulation information into the component. So it converts the data on output. Only narrowing is supported. So writing to small files requires you to be on a compatible architecture.

Full interoperability will require the use of the large file structure.

Version 10.1 will have an update to enable it to read components written by version 11.

In other words it will narrow and if necessary swap 64 bit components on read

This table summarises the position.

Currently workspaces saved on a windows box can be loaded on a Unix box.

And vice versa

This continues to be the case

If the architecture is different then we don't preserve extant GUI objects

What applies to APs also applies to DLLs ([]NA)

The end!

Ctrl+Shift+F12