64 bit Dyalog APL

- A few bits of history
- Benefits
- Interoperability
  - Sockets
  - Component files
  - Workspaces
  - APs

A few bits of history

	Unix
Year	16 bit
1981	Zilog Z8000

A few bits of history

	Unix	
Year	16 bit	32 bit
1981	Zilog Z8000	
1983		Gould

Launched at Washington Conference

A few bits of history

Year	16 bit	32 bit
1981	Zilog Z8000	
1983		Gould
Launched at Washington Conference		
		Three Rivers/ICL Perq
		Cadmus
	Xenix 286	
		DataIndustria
		Altos
		Sun 1
		Apollo
		IBM
		Masscomp
		Apple Lisa
		Silicon Graphics
		Vax
		Pyramid
		Ridge
		Amdahl
		AT&T/3b2

A few bits of history

Year	Unix/Linux 32 bit	Dos/Windows 16/32 bit
1985	Opus/Natsemi 32032	Opus/NatSemi 32032
	SCO Unix 386	
1985		DOS/PharLap Intel 386/387

A few bits of history

	Unix/Linux		Dos/Windows
Year	32 bit	64 bit	16/32 bit
1985	Opus/Natsemi 32032 SCO Unix 386		Opus/NatSemi 32032
1985			DOS/PharLap Intel 386/387
1987	IBM PC/RT IBM PS2/AIX HP/9000		
1989			DOS/PharLap Intel 486
1990	Sun/SPARC		
1990	IBM/RS6000 DEC workstation MIPS Silicon Graphics MIPS Sequent		
1992		DEC Alpha	Windows 3 Watcom
	HP/PA-RISC		

A few bits of history

	Unix/Linux		Dos/Windows	Windows
Year	32 bit	64 bit	16/32 bit	32 bit
1992		DEC Alpha	Windows 3 Watcom	
	HP/PA-RISC			
1994				NT 3.51
1995			Windows 95	
				NT 4.0
1998			Windows 98	
MainWin based Unix ports				
	Solaris/SPARC			
	IBM AIX/p4			
	Linux/x86			

A few bits of history

	Unix/Linux		Windows	
Year	32 bit	64 bit	32 bit	64bit
1992		DEC Alpha		
	HP/PA-RISC			
1994			NT 3.51	
1995				
			NT 4.0	
1998				
MainWin based Unix ports				
	Solaris/SPARC			
	IBM AIX/p4			
	Linux/x86			
2006		IBM AIX/p5	XP	XP64

Benefits

- Large address space
  - Can map many files
  - Can load many DLLs
- Large workspace
- Large arrays
  - Not with 11.0 but with 11.1
- Large integers
  - Not with 11.0 but probably with 11.1

Interoperability

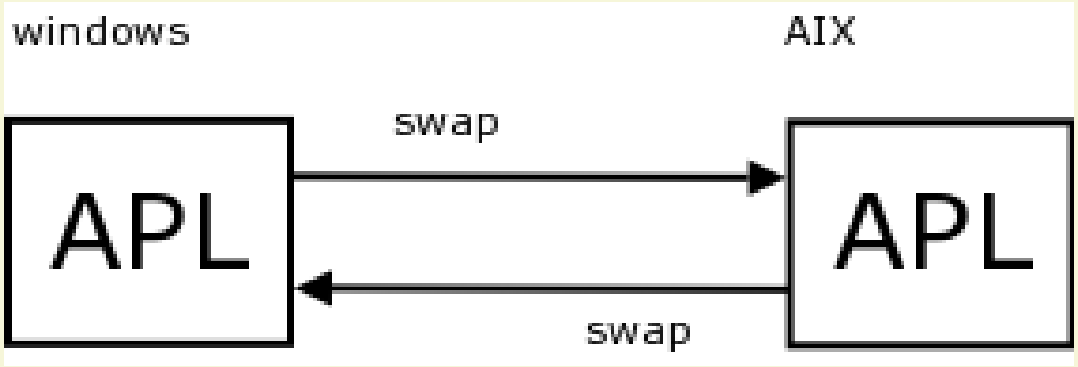
- Some aspects of APL's interaction with external elements use common code
  - APs
  - Components
  - Sockets with style APL
  - DDE between APLs
  - Clipboard

Interoperability

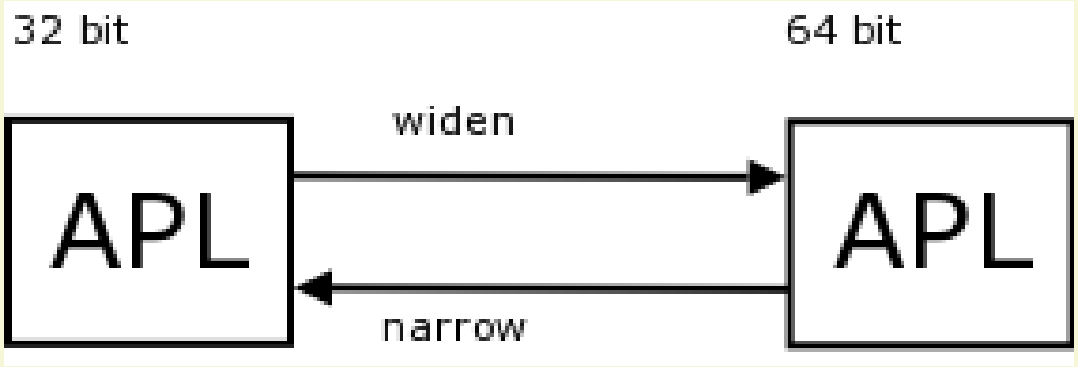
- Some mechanisms have encapsulation that identifies the architecture that wrote the information
  - Components
  - APL style sockets
- These can interact between APL's

Interoperability
Sockets

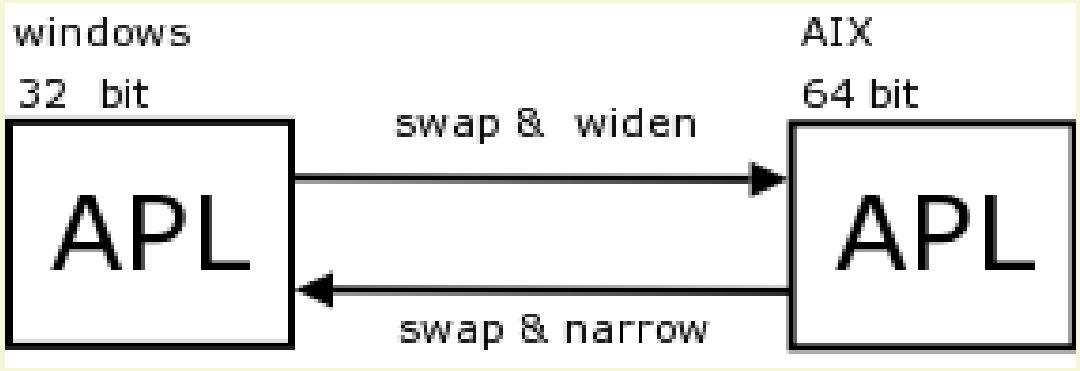
Interoperability Sockets



Interoperability
Sockets



Interoperability
Sockets



Component file history

- 1981
  - Components were individual files in a directory
  - Components had 16 bit header information
- 1985
  - New indexing structure with components inside a file
  - Entirely 32 bit
  - Machines with similar architecture could read/write each others component files

Component file history

- 1992
  - DEC Alpha: Entirely 64 bit. Only the Alpha version could read/write these component files
- 2004
  - Extended []FCREATE to support large files

```
'large' []FCREATE 1 64
[]FSIZE 1
1 1 120 1.844674407E19
```

  - Component data still 32 bit but is self conscious

Component file history

- 2006
  - Files are now interoperable
  - Individual components may have been written by different architectures

Interoperability Components

- 64 bit files vs 64 bit APL
- File size vs Component Size
  - Version 10.1 supports large files but not large components.
  - Version 11 supports large components.
- Small files carry architecture information at the file level
- Large files tag components with architecture

Interoperability Components (Large files)

- Components are written natively
  - A 64 bit APL puts 64 bit data into a component
  - A 32 bit APL puts 32 bit data into a component
  - A big endian APL (AIX) puts big endian data into a component
  - A little endian APL (windows) puts little endian data into a component
- Components are translated as they are read

Interoperability Components (Large files)

- Individual components can come from different APLs
- The overall indexing structure is defined by the APL that created the file
- An APL that updates the file maintains the original index structure

Interoperability Components (Small files)

- Components are written traditionally
  - A 64 bit APL puts 32 bit data into a component
  - A 32 bit APL puts 32 bit data into a component
  - If the file was not of the same endianness as the APL then the file cannot be updated
  - An AIX APL cannot update a Windows small component file
- Components are translated as they are read
- Components may be translated during write

Interoperability Components

- There is an update to version 10.1 called 10.1.5
  - It can read components written by a 64 bit different endian APL
  - Provided they were written to a large file
- i.e. A 10.1.5 Windows APL can read components written by a 64 bit AIX APL
- Customers who don't need this can avoid 10.1.5. 10.1.2 will be maintained as well.

Interoperability Components

APL	File		
	Large	Small	
		Same Endian	Different Endian
10.1.2	same architecture	↔	X
10.1.5	↔	↔	←
11.0	↔	↔	←

Interoperability workspaces

- Workspaces are written natively
- Workspaces are translated as they are read
- GUI objects are silently destroyed
  - This is consistent with version 10.1

Interoperability
Auxiliary Processors

- The code for exchanging data with APs will NOT be updated
- APs will have to be re-compiled for 64 bit
- A 64 bit APL cannot use a 32 bit AP
- A 32 bit APL cannot use a 64 bit AP

- Ruthie Foster
- Stages
- track 11
- Church