Computer Systems and Networks

ECPE 170 – Jeff Shafer – University of the Pacific

Endianness

Lab Schedule

Activities

- **Today**
  - Endianness
  - Python
- **Thursday**
  - Network programming
  - **Lab 8 – Network Programming**

Assignments Due

- **Lab 8**
  - **Due by Mar 30th 5:00am**
- **Lab 9**
  - **Due by Apr 6th 5:00am**

Endianness

- **In typical computer memory,
each address (location) stores one byte**
- If we have a one-byte integer, how is that stored in memory?
- If we have a two-byte integer, how is that stored in memory?
- If we have a four-byte integer, how is that stored in memory?

Endianness = Byte Ordering

Endianness Example

- 32-bit hexadecimal number
  `0x12345678`
- Composed of 4 bytes:
  `0x12 0x34 0x56 0x78`
  (MSB) *(LSB)*
- Two possible arrangements:

Address	"Option A"	"Option B"
0	0x12	0x78
1	0x34	0x56
2	0x56	0x34
3	0x78	0x12

Endianness Example

- 32-bit hexadecimal number
  0x12345678
- Composed of 4 bytes:
  0x12 0x34 0x56 0x78
  (MSB) *(LSB)*
- Two possible arrangements:
  - **Big Endian**
  - **Little Endian**

Address	Big Endian	Little Endian
0	0x12 (MSB)	0x78 (LSB)
1	0x34	0x56
2	0x56	0x34
3	0x78	0x12

Endianness

- **How is DEADBEEF$_{16}$ stored in little and big endian formats at address 21C$_{16}$?**
  - Little endian
    - 21C$_{16}$=EF$_{16}$
    - 21D$_{16}$=BE$_{16}$
    - 21E$_{16}$ =AD$_{16}$
    - 21F$_{16}$=DE$_{16}$
  - Big endian
    - 21C$_{16}$=DE$_{16}$
    - 21D$_{16}$=AD$_{16}$
    - 21E$_{16}$ =BE$_{16}$
    - 21F$_{16}$=EF$_{16}$

Big Endian –vs– Little Endian

Big-Endian CPU

- **Most significant byte (MSB) comes first** (stored in lower memory address)
- Examples
  - Motorola 68000
  - Java virtual machine
  - IBM PowerPC (by default, can also be little endian)

Little-Endian CPU

- **Least significant byte (LSB) comes first** (stored in lower memory addresses)
- Examples
  - Intel x86/x86-64
  - DEC Alpha
  - ARM (by default, also can be big endian)

Etymology of "Endiann"

- Origin in 1980s
- Reference to Swift's *Gulliver's Travels*, in which the Lilliputians were divided into two camps:
  - Those who ate their eggs by opening the 'big' end
  - Those who ate them by opening the 'little' end
- In other words, **a trivial distinction**

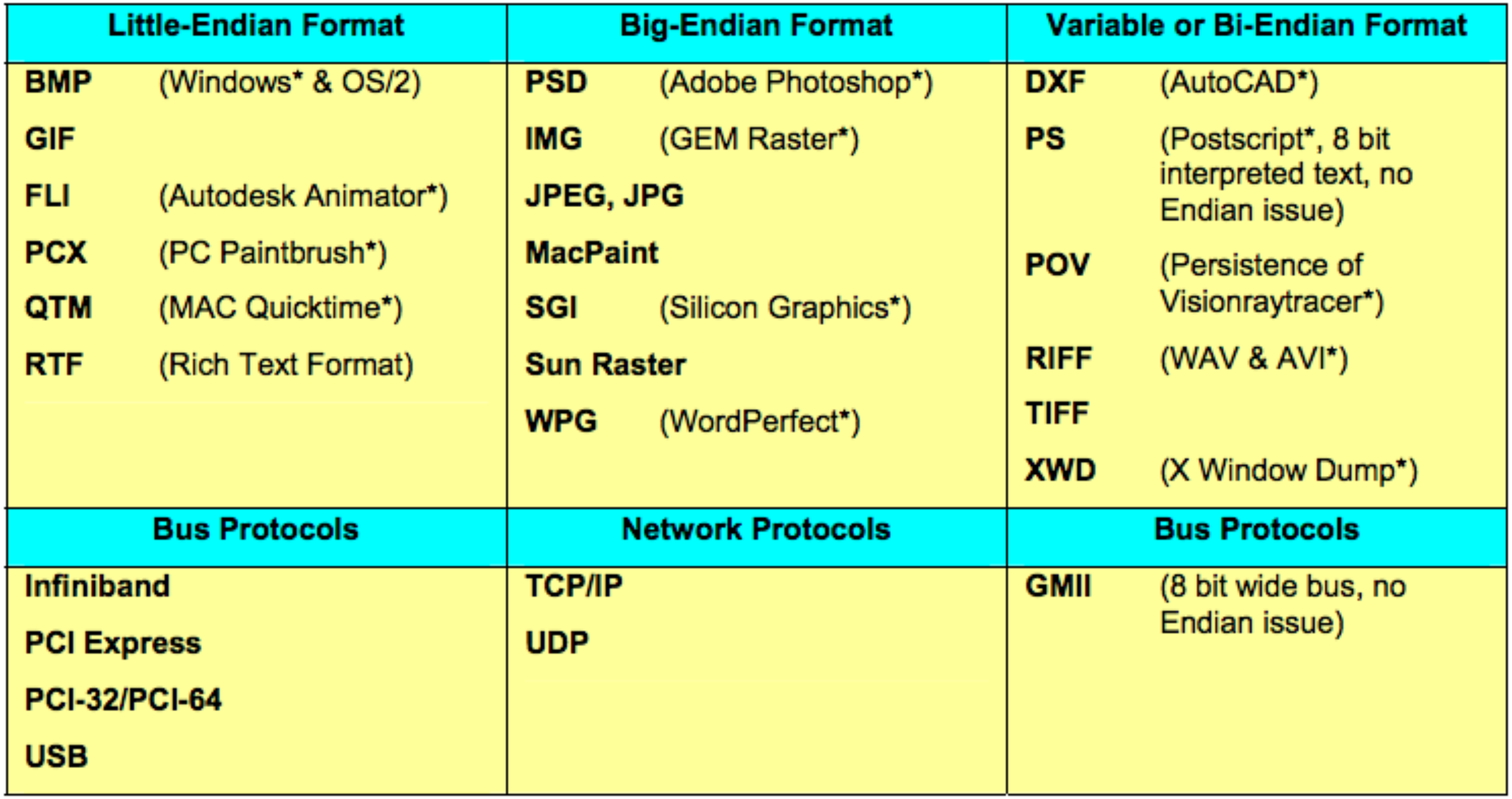
Do I Care?

- **When do I need to care that some computers are big-endian and others are little endian?**
  - What happens if I open big-endian data on a little-endian computer?
- Endianness must be considered whenever you are **sharing data** between different computer systems
  - Reading/writing data files to **disk**
  - Reading/writing data files to **network**

Best Practices

- **Pick one format and stick with it!**
  - Example: Data sent over the network will always be in *big-endian* format regardless of who sends it
    - *Networks are big-endian "by tradition"*
  - Example: Data written to disk will always be in *little-endian* format regardless of who writes it
- **Convert between data storage/transfer format and internal representation as needed**
  - Example: Little-endian machines convert to big-endian before sending data onto the network (and convert back upon receiving data from the network)

Examples in Industry



Little-Endian Format	Big-Endian Format	Variable or Bi-Endian Format
BMP (Windows* & OS/2)	PSD (Adobe Photoshop*)	DXF (AutoCAD*)
GIF	IMG (GEM Raster*)	PS (Postscript*, 8 bit interpreted text, no Endian issue)
FLI (Autodesk Animator*)	JPEG, JPG	
PCX (PC Paintbrush*)	MacPaint	POV (Persistence of Visionraytracer*)
QTM (MAC Quicktime*)	SGI (Silicon Graphics*)	
RTF (Rich Text Format)	Sun Raster	RIFF (WAV & AVI*)
	WPG (WordPerfect*)	TIFF
		XWD (X Window Dump*)
Bus Protocols	**Network Protocols**	**Bus Protocols**
Infiniband	TCP/IP	GMII (8 bit wide bus, no Endian issue)
PCI Express	UDP	
PCI-32/PCI-64		
USB		

Table 2- Common file formats

http://download.intel.com/design/intarch/papers/endian.pdf