16-bit Data Types Integral data type

#include <Types.h>

integer	word or longword range: same as range of short or long
unsigned integer	unsigned word or longword range: same as range of short or long
short	signed word range: -32768...32767
unsigned short	unsigned word range: 0...65535
OSErr	signed word range: -32768...32767 (see Error Codes)
Boolean	signed word range: 0...1, 0...non-zero, FALSE...TRUE
SmallFract	unsigned fraction between 0 and 1
LangCode	signed word range: -32768...32767
ScriptCode	signed word range: -32768...32767

Notes: The MC68xxx is a "Little Endian" meaning that the first byte in a 2-byte word (the lower of the two bytes in memory) contains the most significant part of the value (contrasted to "Big Endians," such as the 80x86 which have the lowest byte of all data types as the least significant part). Thus, to use a byte pointer to obtain a 16-bit word on the Mac, you could:

```
Byte    *bp;
short   theShort;

theShort = (short)(bp[0] << 8) + bp[1];
```

The MC68000 (used in the original Mac and the Plus) is different from the 80x86 with regards to odd-address word fetches (i.e., when you use a MOV.W to read memory from an address which is not evenly-divisible by 2, you get a CPU exception error and your program fails). Thus, you might need to use code like the above on unformatted data area. Your compiler will not watch for errors such as:

```
Byte      *bytePtr;
short     *shortPtr, x;
Byte      mixedBuffer[1000];

bytePtr = mixedBuffer;
bytePtr++;
shortPtr = (short *)bytePtr;   /* points to an odd address now */
x = *shortPtr;                 /* System Error ID = 2 on old Macs */
```

Macintosh compilers will word align data structures so that you will not get odd address errors by dereferencing a multiple word field that happens

to follow a 1 byte sized field. For example:

```
struct {
        char x;
        short y;
        char z;
} foo;
```

sizeof() would return a value of 6 for this structure. The short field starts on an even byte (word) boundary, and the structure itself is padded to take up an even number of bytes.

Integers are either 2 or 4 bytes in size depending on your compiler and its configuration.