Handle data type

```
#include <Types.h>

typedef Ptr *Handle;          /* pointer to a pointer to a signed char */
```

Notes: This is the generic form of double indirection used everywhere in Macintosh programming. A Handle is a 32-bit pointer to a 32-bit pointer to a char. It is the return value of such functions as **NewHandle** and **GetResource**. The Handle itself points somewhere inside a block of Master pointers. At that address is the address of some data. This clever system lets the Memory Manager move data around in memory without affecting your programs access to the data.

To access handle data, use double indirection; eg:

```
theHandle= NewHandle( 1000 );     /* allocate 1000-byte storage */
strcpy( *theHandle, "copy me" );  /* single indirection to get addr */
byte0 = **theHandle;              /* double indirection to get data */
byte5 = **theHandle[4];
DisposHandle( theHandle );        /* free the allocation */
```

To access the fields of a structure:

```
typedef struct {
    char    myByte;
    short   myInt;
} MyStruct, *MyStructPtr, ** MyStructHandle;

MyStructHandle theHandle; /* create a pointer to a pointer to a MyStruct
*/
MyStructPtr     thePtr;

theHandle=(MyStructHandle)NewHandle(sizeof(MyStruct));
            /* allocate */

    /* Fill the handle with data */

theByte = (*theHandle)->myByte;   /* fetch a data element */
theByte = (**theHandle).myByte;   /* alternate syntax; same action */
thePtr = *theHandle;              /* dereference Handle, see below */
HLock( theHandle );               /* lock the data in place */
theByte = thePtr->myByte;         /* more direct access to the data */
HUnlock( theHandle );             /* let the data float */

theByte = thePtr->myByte;         /* Dangerous!, Handle is unlocked!
*/
```

Before accessing the data at a Handle, you must be sure that the data won't be moved as you work with it. As long as you use double indirection; eg ...

(handleName).fieldName** ...or... **(*handleName)->fieldName**

... you'll always be OK; however, it can be more efficient to get a pointer directly to the data, as in the final example above. In that case, be sure to lock the data in place via **HLock** before assigning the value of the pointer. See **About the Moves Memory Icon** for more on determining when memory may move.

Nearly all system-defined data records are typedef'd with a specific handle name. For instance a PicHandle is a pointer to a pointer to a Picture structure. Such specific names make programming easier since you need not coerce the pointer to tell the compiler what type of data is being addressed.

However, be aware of the following:

(**aHandle) = something that moves memory

is not safe C. C compilers are allowed to "partially dereference" the expression **aHandle, that is, they can dereference it once, evaluate the right hand side of the expression (move memory), and then do the second dereference. This would make **aHandle point to some random place in memory.

MS-DOS and Windows folks: You will rarely need (or want) to access toolbox data structures directly. For instance, use **LCellSize**(n, listHandle) rather than (*listHandle)->cellSize=n.