ListRec structure

#include <Lists.h>

typedef struct **ListRec** {		Size	Offset	Description
Rect	rView;	8	0	Display rectangle, in local coords **(LSize)**
GrafPtr	port;	4	8	Where list resides (usually WindowPtr**)**
Point	indent;	4	12	Horiz and vert offset for text in each cell
Point	cellSize;	4	16	Width and height of each cell **(LCellSize)**
Rect	visible;	8	20	Currently-visible cells, in cell coords
ControlHandle	vScroll;	4	28	Leads to ControlRecord for vert scroll bar
ControlHandle	hScroll;	4	32	Leads to ControlRecord for horiz scroll bar
char	selFlags;	1	36	Mouse-selection options (see Notes)
Boolean	lActive;	1	37	TRUE when list active (**LActivate)**
char	lReserved;	1	38	(reserved, internal flags)
char	listFlags;	1	39	Auto scroll: 0=none,1=horiz,2=vert,3=both
long	clikTime;	4	40	Time of last click (ticks since startup)
Point	clikLoc;	4	44	Location of previous click, in local coords
Point	mouseLoc;	4	48	Current mouse location, in local coords
ProcPtr	lClikLoop;	4	52	Addr called while mouse is down **(LClick)**
Cell	lastClick;	4	56	Last cell clicked (cell coords)(**LLastClick)**
long	refCon;	4	60	Available for application use
Handle	listDefProc;	4	64	Leads to custom LDEF code (0=standard)
Handle	userHandle;	4	68	Available for app or list definition proc
Rect	dataBounds;	8	72	Bounds of data cells, in cell coords
DataHandle	cells;	4	80	Leads to cell contents (see **LFind** for layout)
short	maxIndex;	2	84	(used internally)
short	cellArray[1];	*n*	86	Array of offsets to the cell data
} **ListRec**;		86+*n*		*n* is(dataBounds width*height*2) bytes long

typedef ListRec ***ListPtr**;
typedef ListRec ****ListHandle**;

typedef Point **Cell**; (OK to use **PtInRect**, et al. with cell coords)

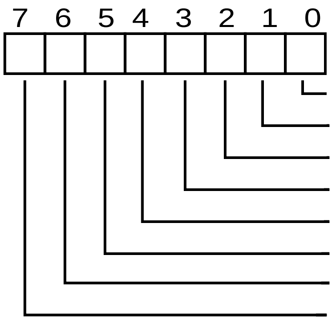
typedef char **DataArray**[32000];
typedef DataArray ***DataPtr** ;
typedef DataArray ****DataHandle**;(the cells field is this data type);

Notes: A Handle leading to a **ListRec** structure is used in all List Manager functions.

Many attributes of how the list is displayed are determined by fields of the GrafPort structure (eg, txFont, txFace, etc.). If you want to change some fields from their defaults, it is best to do so right after calling **LNew** but before displaying any data.

The indent.v field normally gets set to the ascent + descent of the current font and indent.h gets arbitrarily set to 4. You may want to modify these values (eg, set indent.h to a larger value to make room for a small icon).

The selFlags field determines how selection is performed with the mouse. By default, click deselects all cells and selects the current one, Shift-click and Shift-drag extends the selection as a rectangular 'range', Command-click or Command-drag toggles, according to the state of the initial cell. Options include:



0x01	(reserved)	
0x02	lNoNilHilite	1= Don’t highlight empty cells
0x04	lUseSense	1= Shift senses state of initial cell
0x08	lNoRect	1= Don’t expand selection as a rectangle
0x10	lNoExtend	1= Shift won’t extend selection
0x20	lNoDisjoint	1= Click deselects all previous selections
0x40	lExtendDrag	1= Drag extends without Shift key down
0x80	lOnlyOne	1= Only one cell can be selected at a time

You may need to play with various options to find the best method for your application. A common variation is:

```
    (*theList)->selFlags = lUseSense | lNoRect | lNoExtend;
```

which makes the Shift key do what the Command key does (and therefore requires no explanation to the user).

The cellArray portion of the structure is formatted in row-major order (cells 0...*n*) of row 0, (cells 0...*n*) of row, etc. The high bit of each word identifies if the cell is currently selected. The low 14-bits identify the offset of the start of the cell data, within the data area identified by the cells Handle. It is undocumented, but the word directly lower in memory (at offset-2) is the length of the data. See **LFind** for example code which accesses the data directly.

IM (and other sources) call lActive a Boolean (ie, a 16-bit integer). It is a pascal-style BOOLEAN; a one-byte value.