SCRIPT LANGUAGE

Like a cross between C and awk. Suck it and see. Variables are created on the fly. Comments are introduced with #. **break** and **continue** are not supported. Storage is allocated each time an expression is evaluated and only freed when the script is complete, so you will run out of memory if you loop a few thousand times.

BUILT-IN FUNCTIONS

int **active**(char *_appName_)

Returns 1 if _appName_ is currently the active application, else 0. "active" is used in the NeXTStep sense. Use **running()** to determine if the app is alive.

int **atoi**(char *_string_)

See manual pages.

void **flushEvents**()

Flush collected events. Events generated by calls to **postKeyEvent()**, **postMouseEvent()** etc are buffered and only actually played back when **flushEvents** is called. A call to **flushEvents** is automatically made at the end of a script.

char ***getenv**(char *_name_)

See manual pages.

char ***getParameter**()

CyberMan has a parameter window. This call returns it's contents.

int **launch**(char *_appName_)

Ask the Workspace manager to launch the application _appName_.

char ***malloc**(int _size_)

See manual pages. The space is automatically freed when the script completes.

char ***map**(char *_fileName_)

Opens the file and maps it in. Returns a pointer to the mapped data.

int **newWindow**(char *_appName_)

This call, in conjunction with **refreshWindowList** can be used to get the local window number of a new window which has been added to _appName_ since the last call to **refreshWindowList**. For example, if you call **refreshWindowList**, then click **Mail's** send button, **newWindow("Mail")** will return the window number of the Send window.

void **outputToWindow**(char *_buf_)

CyberMan has an output window. This call displays the contents of _buf_ in that window. If _buf_ appears to contain RTF, it will be displayed as such.

char ***pasteIn**(int _pasteBoard_)

Returns a pointer to a buffer containing the contents of the given _pasteBoard_. _pasteBoard_ may either be **USER_PB** or **SELECTION_PB**. **USER_PB** may be used if the script has been run as a service, in which case it

refers to the requesting application's pasteboard. **SELECTION_PB** refers to the standard selection pasteboard. Only Ascii and Filename pasteboard types are currently recognized.

void **pasteOut**(int *pasteBoard*, char **buf*)

Copies *buf* to the specified pasteboard. See above for description of the pasteboard types. If the script is run from a service and if a return type has been specified in the options file in the script directory, then **USER_PB** may be used to replace the contents of the original selection.

int **pipe**(char **input*, char **cmd*)

Passes *input* through a pipe to the executable *cmd*. See the **pipe** manual page for more details.

void **postActivate**(char **appName*)

Posts an event to make the application the active app.

void **postDeactivate**(char **appName*)

Posts an event to deactivate the given app.

void **postKeyEvent**(char **appName*, int *flags*, char **keyString*)

Posts to *appName* a series of key down and key up events for the given *keyString.* If *flags* is **COMMAND**, the key strokes will be generated as if the Command key was held down. Note that events are not played until **flushEvents** is called.

void **postMouseEvent**(char **appName*, int *event*, int *click*, int *window*,

int *x*, int *y*, int *relativeTo*)
int *x*, int *y*, int *relativeTo*)

Posts to *appName* the given mouse event. *event* may be one of: NX_LMOUSEDOWN , NX_LMOUSEUP, NX_RMOUSEDOWN, NX_RMOUSEUP, NX_MOUSEMOVED, NX_LMOUSEDRAGGED, NX_RMOUSEDRAGGED, NX_MOUSEENTERED, NX_MOUSEEXITED, DRAG_WINDOW, SINGLE_CLICK, DOUBLE_CLICK or TRIPLE_CLICK. SINGLE_CLICK is shorthand for two events, NX_LMOUSEDOWN followed by NX_LMOUSEUP, and similarly for DOUBLE_CLICK and TRIPLE_CLICK. *click* is normally 1 for mouse down and 0 for mouse up, unless it is part of a single, double or triple click in which case both down and up clicks should have 1 for the first pair, 2 for the second and so on. *window* specifies the **local** window number of the application. This may be detemined using the supplied modified version of **Winfo**. The negative window values **NX_KEYWINDOW (-1)**, **NX_MAINWINDOW (-2)** may also be used (see appkit/NXJournaler.h). *x* and *y* are the coordinates of the mouse event relative to one of the corners of the given window. Which corner is specified by the value of *relativeTo* which may be **TOP_LEFT**, **BOTTOM_LEFT**, **TOP_RIGHT** or **TOP_LEFT**. If a negative pseudo window number is given, *relativeTo* must be **BOTTOM_LEFT** because **CyberMan** cannot determine the window size. **DRAG_WINDOW** specifies a drag operation on the given window in which case *x* and *y* are delta offsets from the current position.

void **postponeEvent**(int *interval*)

All events have a time stamp. This increments the time stamp by the specified interval. The interval is in units of about 66 milliseconds. The next event will consequently be delayed by the specified amount.

void **printf**(char **fmt*, ...)

This call is for debugging purposes. It does a normal printf and outputs to the console. A maximum of ten parameters may be supplied.

void **refreshWindowList**()

Updates **CyberMan's** list of processes and their window numbers. This

needs to be called in conjunction with **newWindow** and also may need to be called if **active** and **running** are to work accurately.

int **running**(char *_appName_)

Returns 1 if the application is running, else 0. May be used with **launch** if the app is not currently running.

void **setEventInterval**(int _interval_)

Events are seperated from each other by a small delta time interval. By default, this is zero. The interval delta may be set using this call.

void **sleep**(int _seconds_)

Makes **CyberMan** sleep for the given number of seconds.

void **sprintf**(char *_buf_, char *_fmt_, ...)

See manual page. A maximum of ten parameters may be given.

int **strlen**(char *_string_)

See manual pages.

int **system**(char *_cmd_)

Execute the given command. See **system** manual page for more details.

int **unhide**(char *_appName_)

Unhides the given application. The only difference between this and **postActivate** is that it happens immediately by sending a request to the Workspace manager.

BUILT-IN IDENTIFIERS

int **numProcs**

The number of processes in **procList**.

char ***procList[]**

An array of character pointers containing the names of all the processes that **CyberMan** knows about.

char ***SCRIPT_DIR**

Pathname of the directory from which this script was run.