

# 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 determined 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.