MemMan Psuedocode View

(starting from the point where MemMan has been called and passed a parameter, containing the location of a loadable object from the filesystem)

0) Read 'param' section of object. Read params:

0a) Check for '.type' parameter. Should be one of the following:

- bin\_exec (read-only, executable)

- bin\_data (read-write, not executable)

- text\_exe (read-only, executable)

- text\_dat (read-write, not executable)

0b) Check for '.rtip' parameter (RunTime Initialization Param). Should be either:

- on\_boot (loaded after kernel-space - once)

- service (loaded after kernel-space - continuous)

- instance (loaed after user-space - once)

- on\_kill (loaded after user-space stops - dies)

0c) Check for '.rtpp' parameter (RunTime Persistance Param). Should be either:

- \_root (system permission)

- admin (elevated permission)

- \_user (limited permission)

0d) Check for '.rtsp' parameter (Runtime segment Size Param). Should be either:

- \_static (may not request more memory)

- dynamic (may use 'burst' limit)

0e) Check for '.rtmp' parameter (RunTime Memory Param). Should be a number no larger than 5% of total RAM.

- 'burst' size allows for an extra 2% in addition to this. However, this limit may never be exceded.

0f) Check for '.name', '.desc' and other object descriptors

1) Get size of 'param' section.

2) Get size of 'code' section.

3) Get size of (static, initialized) 'data' section.

4) Get size of (static, uninitialized) 'bss' section.

5) Get total object size (sum of the previous sections) and return 'object size'+'.rtmp' as a memory request, and set '.state' to 'queued'

6) If memory is available, check priority. Objects with the highest permission level and persistance are loaded first.

7) If no memory is available, object remains 'queued' and is pushed to persistent storage as a queued chunk.

8) Otherwise, it's sent into RAM. Its '.state' is set to 'active'.

- Once loaded into RAM, r/w/e remains persistent, and may not change until it leaves volatile memory.

9) When an object is waiting for a certain event or condition, and is not actively performing an instruction, its '.state' is changed to 'inactive'.

10) When an object runs out of memory, MemMan will attempt to delete orphanned/leaked data. Object '.state' changes to 'paused'.

10a) If unable to free enough space for continued execution, change Object '.state' to 'stopped'

- Object's stack contents are pushed to persistent memory, in the form of Read-Only debug file

- Throw 'Out of Memory' Exception

- Remove Object from RAM

10b) If enough free space is produced, change Object '.state' to 'active'

- Object is unpaused - execution continues