Pattern Summaries – Small Memory Software

Major Technique: Small Architecture

How can you manage memory use across a whole system? Make every component responsible for its own memory use.

Memory Limit How can you share out memory between multiple competing components? Set limits for each component and fail allocations that exceed the limits.

Small Interfaces How can you reduce the memory overheads of component interfaces? Design interfaces so that clients control data transfer.

Captain Oates How can you fulfil the most important demands for memory? Sacrifice memory used by less vital components rather than fail more important tasks.

Read-Only Memory What can you do with read-only code and data? Store read-only code and data in read-only memory.

Hooks How can you change information in read-only storage? Access read-only information through hooks in writable storage and change the hooks to give the illusion of changing the information.

Major Technique: Secondary Storage

What can you do when you have run out of primary storage? Use secondary storage as extra memory at runtime.

Application Switching How can you reduce the memory requirements of a system that provides many different functions? Split your system into independent executables, and run only one at a time.

Data File Pattern What can you do when your data doesn't fit into main memory? Process the data a little at a time and keep the rest on secondary storage.

Resource Files Pattern How can you manage lots of configuration data? Keep configuration data on secondary storage, and load and discard each item as necessary.

Packages How can you manage a large program with lots of optional pieces? Split the program into packages, and load each package only when it's needed.

Paging Pattern How can you provide the illusion of infinite memory? Keep a system's code and data on secondary storage, and move them to and from main memory as required.

Major Technique: Compression

How can you fit a quart of data into a pint pot of memory? Use a compressed representation to reduce the memory required.

Table Compression Pattern How do you compress many short strings? Encode each element in a variable number of bits so that the more common elements require fewer bits.

Difference Coding Pattern How can you reduce the memory used by sequences of data? Represent sequences according to the differences between each item.

Adaptive Compression Pattern How can you reduce the memory needed to store a large amount of bulk data? Use an adaptive compression algorithm.

Major Technique: Small Data Structures

How can you reduce the memory needed for your data? Choose the smallest structure that supports the operations you need.

Packed Data How can you reduce the memory needed to store a data structure? Pack data items within the structure so that they occupy the minimum space.

Sharing How can you avoid multiple copies of the same information? Store the information once, and share it everywhere it is needed.

Copy-on-Write How can you change a shared object without affecting its other clients? Share the object until you need to change it, then copy it and use the copy in future.

Embedded Pointer How can you reduce the space used by a collection of objects? Embed the pointers maintaining the collection into each object.

Multiple Representations How can you support several different implementations of an object? Make each implementation satisfy a common interface.

Major Technique: Memory Allocation

How do you allocate memory to store your data structures? Choose the simplest allocation technique that meets your need.

Fixed Allocation How can you ensure you will never run out of memory? Pre-allocate objects during initialisation.

Variable Allocation How can you avoid unused empty space? Allocate and deallocate variable-sized objects as and when you need them.

Memory Discard How can you allocate temporary objects? Allocate objects from a temporary workspace and discard it on completion.

Pooled Allocation How can you allocate a large number of similar objects? Pre-allocate a pool of objects, and recycle unused objects.

Compaction How do you recover memory lost to fragmentation? Move objects in memory to remove unused space between them.

Reference Counting How do you know when to delete a shared object? Keep a count of the references to each shared object, and delete each object when its count is zero.

Garbage Collection How do you know when to delete shared objects? Identify unreferenced objects, and deallocate them.