AN ASYNCHRONOUS GARBAGE COLLECTOR FOR THE CAP FILING SYSTEM

A.D. Birrell and R.M. Needham

University of Cambridge Computer Laboratory

The CAP filing system [Needham and Birrell 1977] is able to produce garbage or 'lost objects' on disc, because its directory structure is a general naming network containing, in principle, cyclic substructures. Such substructures may become inaccessible as a result of deletion of capabilities or file directory entries. Up to now the tidy-up of the disc which occurs on system restart has been relied on to recover such space; this is somewhat inelegant and makes one nervous about leaving the system running for long periods. Accordingly an asynchronous garbage collector has been introduced.

Basic management of the disc system is in terms of _system internal names_ or SINs [Needham and Walker 1977]. File dictionaries relate text names to SINs, and a reference count for each SIN is maintained in a central SIN-table. The count shows how many file directories refer to the SIN in question, and also indicates whether the relevant object is active in the current virtual memory. The reference counts are only an incomplete guide to material which should be discarded, because of the existence of looped structures as mentioned above - a standard problem with reference counts. However, the SIN-table also contains information on the _type_ of the object referred to: specifically whether it is a directory or not. This type information is the key to garbage collection, since it enables us to limit the area within which the garbage collector must search for garbage. We know that objects of type 'segment' cannot contain references to further objects, and thus that the reference count system will work correctly for them - that is, a segment object may be deleted if, and only if, its reference count is zero. We accordingly pursue a mixed approach, in which reference counts are used when appropriate and a scanning type of garbage collection used otherwise. Since segments predominate (it would be an unusual system in which most of the objects were directories) the cost of the garbage collection is modest.

The action of the garbage collector will now be described. It runs as a separate system process intended to be permanently active. At the start of its cycle it scans the SIN directory, setting up a data structure with an entry for every object of type 'dictionary'. These entries record the state of the garbage-collector's knowledge of the dictionary; they have three values - 'marked', 'needs examination', and the initial state 'not visited'. The garbage collector determines the SIN of the master directory and of each currently active directory, marks their entries as 'needs examination', and proceeds through a scan of accessible dictionaries starting with them. As any directory is scanned, entries in it referring to other directories are noted. Any directory thus referred to has its entry changed if need be: state 'not visited' is changed to 'needs examination', and if there was no record of it at all an entry is created in state 'needs examination'. When a particular directory has been scanned its state is changed to 'marked'. While the garbage collector is running the rest of the system is busy and will be creating new directories or retrieving capabilities for existing ones. These actions are the province of the SIN table manager SINMAN [Needham and Walker 1977] in each process. It is necessary that the garbage collector be informed of such events, since a directory which is created or retrieved is certainly accessible. Accordingly SINMAN, using the standardly available inter-process message facilities, sends a (non-reply) message to the garbage collector detailing any such event. The message is sent before retrieval or creation, and the communication facility guarantees that the message is put forthwith on the garbage collector's input queue. On receipt of such a message, the garbage collector amends the recorded state of the relevant directory if necessary.

The algorithm terminates when the collector has no record of any directory requiring examination, and there are no messages on its input queue. At this point, all objects of type "directory" which were identified in the initial scan of the SIN-directory and which are still in state "not visited" are garbage and may be disposed of. Any directory which is still accessible or active will either have been noted in the scan of the directory structure or have been recorded as becoming active in the current virtual memory. This follows from the fact that a directory capability may only be preserved in an existing directory if it is active at the time - compare the movement of books, where you cannot reshelve a book without pulling it from its original shelf. The principle is familiar in the design of on-the-fly garbage collectors.

In the particular context of the CAP filing system it has been possible to take great advantage of centralised knowledge that certain objects, numerically predominant, do not contain pointers and may therefore be managed by a straightforward reference count system. This combined approach gives a great practical advantage in that only a small proportion of the material on disc need be looked at.

Although only directories are inspected, the garbage collector gives rise to substantial disc traffic. It is therefore specially arranged to run rather slowly, examining up to ten directories each minute; in our present circumstances this means that a cycle takes somewhat less than an hour, with little interference with ordinary users. If let run at full speed, it would take some 1¾ minutes, during which user response is severely affected by disc traffic and to some extent by main memory use.

Having identified certain directories as garbage, it remains to dispose of them. Such directories are scanned, and the reference counts of any directories referred to from them is decremented. At the conclusion of this exercise, all garbage directories should have zero reference count. If any do not, something has gone seriously wrong and the whole operation is aborted. This checking stage is there for prudential reasons only; it seems sensible to have it there because the installation of the garbage collector revealed one or two errors in old code which had had no ill effect previously. There could be more. If the test is passed, then another scan of the garbage directories decrements the reference counts of all non-directory objects referred to and finally deletes the directories themselves. The availability of this test is another advantage of maintaining a parallel scheme of reference counts.

Terminological Note: We have used the term 'directory' where strictly in the CAP system we should have said 'file directory or procedure control block'. Both types of structure contain references to other objects which may be directories.

Acknowledgements: The CAP Project is supported by the U.K. Science Research Council

References:

R.M. Needham and A.D. Birrell, 'The CAP Filing System', SOSP6, 1977
R.M. Needham and R.D.H. Walker, 'The Cambridge CAP Computer and its Protection System', SOSP6, 1977.