NOTE: This Technical Note has been retired. Please see the Technical Notes page for current documentation.

Technical Note ME11

Using a PurgeProc

CONTENTS

This Technical Note discusses the use of the `purgeProc` field of an application's heap zone.

[Aug 01 1989]

Introduction

Most applications will never need to use a `purgeProc`. However, if your application requires the ability to maintain purgeable handles containing data, or you need to have special notification when a certain handle is purged, a purgeProc might help you.

What exactly is a purgeProc?

The purgeProc, which is documented in very briefly in *Inside Macintosh* , Volume 2, page 2-23, is the mechanism which allows the Memory Manager to alert your application that it is getting ready to purge a given purgeable handle. This warning is given so that you can save the data, or note for any special reason that the data no longer exists. The `purgeProc` is passed the handle that is being purged. It is up to you to determine if any action should be taken on the handle. The Pascal interface to the `purgeProc` looks something like this:

```
PROCEDURE MyPurgeProc ( theHandle:Handle );
```

In C it would look like this:

```
pascal void myPurgeProc(Handle theHandle);
```

Each zone has its own `purgeProc` pointer in its zone header. Each time the Memory Manager prepares to purge a handle, it checks the zone header of the zone that the handle belongs to, to determine if your application has installed a purgeProc. If this field is not NIL, it calls the routine pointed to with the handle being purged. This occurs for each handle being purged (not every purgeable handle necessarily). When your routine is called, test the handle passed to be sure that it is a handle you care about, and then act on it. Keep in mind that all handles that pass through your `purgeProc` may not be expected, since your application can create purgeable handles in a few ways, like calling `_HPurge` on an existing handle, or loading a resource (the resource could have the purgeable attribute set), or by calling a routine that could load a resource.

Installing a purgeProc

You install your own `purgeProc` by setting the field in your zone header. Here is a sample routine that installs a `purgeProc`:

```
PROCEDURE InstallProc;

VAR    myZone:THz;

BEGIN
    myZone:=GetZone; { recover my applications zone }
    gOldProc:=myZone^.purgeProc; { save the old purgeProc }
    myZone^.purgeProc:=@myPurgeProc;
END;
```

You must be sure to follow these rules regarding what a purge proc must, can, and cannot do.

- Do not rely on A5 being set properly to your application's globals. (See Technical Note #208.)
- Do not cause memory to be moved or purged.
- Do not open a file (but you can write to an already open file).
- Do not dispose of or change the purge status of the passed handle.
- Only use `purgeProcs` when absolutely needed.
- Avoid using `purgeProcs` if you are also using the `SetResPurge(true)` feature.
- Do write any data to a data file synchronously.

- Do preserve the contents of all registers except A0-A2/D0/D1.
- Do use the `FindFolder` feature of System 7 to locate the temporary folder on the user's hard disk if you are creating a temporary file to hold the contents of purged handles.
- Do test the handle state first to determine if the handle belongs to the Resource Manager, to weed out most handle purges you do not care about
- Do not take too long to determine if the passed handle is in need of purge notification (many programmers do not realize just how many purgeable handles come and go, or how often their purgeProc might be called for a single new handle).

Here is a pseudo code sample that illustrates one possible use of the purge warning procedure: saving the contents of the handle to a file before purging.

```
Procedure PurgeWarning(theHandle:myHType);
begin
  SetUpApplicationA5;     { see Inside Mac and Technotes for how to do this }
  IF BAND(hGetState(theHandle),$20)=0 then
    BEGIN {If we get here the handle does not belong to a resource}
      IF InSaveList(theHandle) then WriteData(theHandle);
    END;
  RestoreOldA5;
END;
```

Remember, the save file should probably be open at this point because opening a file can cause memory to move. You will have to maintain a save list to indicate purgeable handles that need saving.

Note:
If you plan to use the `SetResPurge(true)` option that allows you to modify purgeable resources (not a normal thing for an application to do), don't patch the `purgeProc` pointer. If you do, remove your `purgeProc`, call `SetResPurge`, then re-install the `purgeProc`, being sure that it calls through to the Resource Manager's routine after it is finished.

Back to top

Conclusion

`PurgeProcs` can be used when an application needs to better manage low memory situations, and easily take advantage of large memory conditions. Be very careful when using them, however, keeping in mind that you are in the middle of a Memory Manager routine when you are being called, and you may be called often.

Back to top

References

Inside Macintosh , Volume II, Memory Manager chapter

Technical Note M.OV.A5 -- Setting and Restoring A5

Inside Macintosh , Volume VI, Finder Interface chapter (FindFolder)

Back to top

Downloadables

Acrobat version of this Note (K) Download

Back to top