FontCache

Thomas Richter

COLLABORATORS			
	TITLE : FontCache		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Thomas Richter	June 25, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FontCache

1.1 FontCache Guide

FontCache Guide

Guide Version 1.10 FontCache Version 1.8

This release contains a new version of the "FixFonts" program. Never ever try to run this, nor the original "FixFonts", on an "AFS" partition. The AFS implementation of ExAll() is buggy and might either crash or hang. This is not a "FixFonts" problem! Either re-install the FFS, or any other *working* filing system.

Table of Contents

An additional tip: You can avoid caching directories by creating a file ".nocache" in the "_bullet" directory of the fonts directory. The contents doesn’t matter. However, this file is "global" for the complete FONTS: assign, it does not disable parts of a multi-assign.

Thomas Richter

Rühmkorffstraße 10A

12209 Berlin

Germany

EMail: thor@einstein.math.tu-berlin.de

WWW: http://www.math.tu-berlin.de/~thor/thor/index.html

1.2 The THOR-Software Licence

The THOR-Software Licence

This License applies to the computer programs known as "FontCache" and "FixFonts 40.2". The "Program", below, refers to such program.

The programs and files in this distribution are freely distributable under the restrictions stated below, but are also Copyright (c) Thomas Richter.

Distribution of the Program by a commercial organization without written permission from the author to any third party is prohibited if any payment is made in connection with such distribution, whether directly (as in payment for a copy of the Program) or indirectly (as in payment for some service related to the Program, or payment for some product or service that includes a copy of the Program "without charge"; these are only examples, and not an exhaustive enumeration of prohibited activities). However, the following methods of distribution involving payment shall not in and of themselves be a violation of this restriction:

(i) Posting the Program on a public access information storage and retrieval service for which a fee is received for retrieving information (such as an on-line service), provided that the fee is not content-dependent (i.e., the fee would be the same for retrieving the same volume of information consisting of random data).

(ii) Distributing the Program on a CD-ROM, provided that the files containing the Program are reproduced entirely and verbatim on such CD-ROM, and provided further that all information on such CD-ROM be redistributable for non-commercial purposes without charge.

Everything in this distribution must be kept together, in original and unmodified form.

Limitations.

THE PROGRAM IS PROVIDED TO YOU "AS IS," WITHOUT WARRANTY. THERE IS NO WARRANTY FOR THE PROGRAM, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IF YOU DO NOT ACCEPT THIS LICENCE, YOU MUST DELETE ALL FILES CONTAINED IN THIS ARCHIVE.

1.3 About FontCache

"FontCache" is a patch applied to the diskfont.library. It patches the AvailFonts() function, resulting in a very quick font requester initialisation. The patched library keeps the complete database of available fonts in a file on disk, and reads it instead of parsing the FONTS: directory tree; thus opening a font requester is much faster, dependending on the number of fonts you keep - this IS NO additional RAM cache as the one supplied by the ASL or reqtools library, the cache is kept on disk.

For my approx. 300 fonts, building the font requester takes now 5 seconds instead of half a minute!

This is of course only useful if you have

a) a HD to keep the cache.

b) a big bunch of fonts.

Of course the cache is updated when you add new fonts to the font directory, you don't need to do that manually. The cache is completely transparent to the user, no need to worry about it at all.

The 1.6.1 distribution contains, too, a rewrite of the FixFonts program. This 40.2 release fixes the font directories, as well as rewrites the cache files.

The 1.8 distribution adds another feature that improves the speed of the reqtools and asl requesters even more, even if the font list has been scanned already.

Read here about caveats.

1.4 FontCache caveats

If you installed FontCache, the FIRST font requester won't be faster since the disk cache is not yet created - it will be written back as soon as the font list has been read.

Due to an undocumented feature of the format of the diskfont data, some very old programs won't work appropriate with FontCache, namely the 1.3 programs "FED" and "NotePad". I think nobody will miss them. Newer programs that use the ASL or reqtools requester or do the avail font parsing properly don't have these problems and work fine. In fact, I haven't noticed any other incompatibilities.

This incompatibility to the antique 1.3 might have been gone with the new 1.3 release anyways, but I haven't checked!

Adding of font directories:

The 1.3 version of the cache allows adding additional font directories with

Assign FONTS: <directory> add

something the old version had problems with. However, if you add assigns to FONTS: after the cache has already been written, the contents of the additional directories WILL NOT enter the cache. They are supposed to be "temporary font sources" that aren't worth to enter the cache. Their contents is scanned manually, without using the cache.

If you want to add a font directory permanently to your FONTS: assign, delete the caches with

Delete FONTS:_bullet/FontCache.#?

and add the necessary assign command to your startup-sequence. Then reboot. The next program that scans the font list will rewrite the cache for the enlarged font directory.

This mechanism has, of course, a caveat:

When installing new fonts and rewriting the cache, make sure that the FONTS: assign contains ONLY your system FONTS: directory and nothing else.

If the FONTS: assign contains additional directories at the time the cache is written, these assitional directories will enter the cache as well.

Changing the FONTS: assign completely will work, however, provided you flush the libraries to remove the RAM cache that ASL and reqtools keep. But that's not a problem of the FontCache at all.

The FontCache works fine, too, with the BetterOpenLibs program of the same author. Private fonts can be kept together with the program that need them - if you've installed this patch.

Where is the cache kept:

The cache itself is kept in the files

FONTS:_bullet/FontCache.<x>

where <x> is the mode AvailFonts() was invoked with. If you want to flush the disk cache explicitly, remove these files. If you only want to install a new font, it is not required to remove these files manually - FontCache will notice the change automatically and rewrite the cache as soon as a program requires a font list. You can just install the patch and forget about it.

FixFonts and the AFS:

Neither run the 40.2 in this distribution nor any other "FixFonts" version on AFS partitions, including the official CBM versions that came with your workbench. All releases of "FixFonts" use the ExAll() dos library function, which is not correctly integrated into AFS. Running it may either hang, loop infinitely or crash.

1.5 Installing FontCache

On installation time, the diskfont library file is patched. To undo the patch or to be able to upgrade to a newer version of the FontCache program

PLEASE KEEP THE ORIGINAL diskfont library in a safe place!

To apply the patch, you need the original CBM 39.3 version of the diskfont library. It is part of the Workbench 3.1 and can't be supplied in this archive for copyright reasons. The patch WON'T WORK WITH "CRUNCHED" OR OTHERWISE MODIFIED VERSIONS OF THE DISKFONT.LIBRARY. YOU MUST USE THE ORIGINAL 3.1 VERSION.

O.K., now the steps to install the patch:

- Open a shell.

- Copy the diskfont.library to a safe place. (! IMPORTANT !)

- Unpack the archive to "RAM:". (You might have already done this since you're reading this guide. :-)

A directory called "FontCache" should have been created.

- Add "RAM:FontCache" to the path.

- Enter the following command:

spatch -oRAM:diskfont.library -pRAM:FontCache/diskfont.pch LIBS:diskfont.library

(do not enter the line breaks you see above, they are created by MultiView! Enter just one single line, please.)

- Copy the output file "RAM:diskfont.library" back to LIBS:

- Copy the "FixFonts" program in this distribution where it belongs, i.e. to SYS:System. Additionally to fixing the font directories, the 40.2 version of this program re-writes the font cache. This is not provided as a patch because it is a complete rewrite of the original workbench program.

If the installation procedure fails, please check if you have the latest version of the diskfont.library! You need an original 39.3 version!

- If you're updating from a previous version of the FontCache program, you might want to remove the old cache files manually. This is not strictly necessary because the algorithm will detect an obsolete cache file itself and will replace it; later releases, though, might require fewer or different cache files, thus, some of them won't be seen, touched or read at all. Furthermore, the 1.8 release organizes the cache a bit differently on request, giving an additional speedup.

The following line will remove all caches:

delete FONTS:_bullet/FontCache.#?

- If you're using the reqtools font requester, you may want to speed up this requester even more. Enter the following command to install the speedup:

copy RAM:FontCache/descending to FONTS:_bullet/.sort

Afterwards, you should flush the cache manually as described above. This ".sort" file tells the FontCache program to sort the fonts descending, which is optimal for reqtools; it does not change the order of the fonts presented in the requester, though. Other font requesters might be faster if you copy the "ascending" file on top of ".sort", though. Just go and try.

If you have further problems, write me !

REMARK: Do NOT apply this patch to the FontCache 1.0 to 1.7 patched diskfont.library. The original CBM version must be used instead, as always! (You kept the original version, didn't you?)

1.6 Contacting the Author

Here's my EMail address:

thor@einstein.math.tu-berlin.de

Thomas Richter

You may also want to visit my web page, latest versions of all my programs (plus more) are available there:

WWW: http://www.math.tu-berlin.de/~thor/thor/index.html

The selection is quickly expanding, check in monthly!

1.7 Developer information

Since nobody seems to read the RKRMs, here's again the correct algorithm how to use the AvailFonts() function:

int afShortage, afSize; struct AvailFontsHeader *afh;

...

afSize = 400; do { afh = (struct AvailFontsHeader *) AllocMem(afSize, 0); if (afh) { afShortage = AvailFonts(afh, afSize, AFF_MEMORY|AFF_DISK); if (afShortage) { FreeMem(afh, afSize); afSize += afShortage; } } else { fail("AllocMem of AvailFonts buffer afh failed\n"); break; } } while (afShortage);

/* * if (afh) non-zero here, then: * 1. it points to a valid AvailFontsHeader * 2. it must have FreeMem(afh, afSize) called for it after use */

Please note that you've to recall AvailFonts() as long as it returns a non-zero result. IT DOES NOT SUFFER TO CALL IT ONLY TWICE, using a memory block enlarged by the return value of the first call as buffer for the second call. THE SECOND CALL MIGHT ASK FOR MORE MEMORY AS WELL, so you have to restart over, like shown in the example above.

The additional calls might be required, not only by the FontCache, but also by the original diskfont.library functions.

The meaning of the files used by the FontCache patch:

FONTS:_bullet/.nocache: This is a dummy file. It is just checked if it is there, and if it is, the font cache is disabled.

FONTS:_bullet/FontCache.2: The cache file for the default "TextAttr" structures. The format is intentionally undocumented.

FONTS:_bullet/FontCache.10002: The cache file for the "TTExtAttr" structures. It contains additional tag information if required.

Future releases might require additional or less cache files, do not interpret these files yourself!

FONTS:_bullet/.sort: This file contains a four byte ID describing how the font cache file shall be sorted when it is created. Once the cache is build, this file has no meaning. If this file is not present, the font cache is simply in directory order. The following four byte IDs have been defined:

0x00000000 : directory order 0x00000001 : case insensitive by name, ascending 0xffffffff : case insensitive by name, descending

All other IDs are reserved and should not be used.

1.8 The Credits and Thank You Page

People that contributed to the "FontCache" and I'd like to thank:

Ernst Besser for using even the first releases. (-;

Dirk Neubauer for continously testing it and providing really helpful bug reports. Most of the newer releases are due to his work and testing.

Lyster E. Wick Jr. for providing additional ideas for the 1.5 version of the cache, especially for recommendation of using semaphores now.

Stuart 'Kyzer' Caie - Kyzer/CSG for providing a patch to FixFonts and for the idea letting FixFonts rewrite the font cache. I haven't used your code here, though, because I thought it would be better to rewrite the program completely.

1.9 History

Version 1.0: First AmiNet release.

Version 1.1: Removed problems with programs showing up some fonts twice. They depend, however, on an undocumented feature of the diskfont.library, which is now emulated by FontCache as well. Thanks goes to Iain Hyslop for pointing out this incompatibility with FontCache and Scala/Pagestream. This should be removed now.

Version 1.2: Rewrote parts of the patch to make it more conformal to the BetterOpenLibs patch. The cache files got shorter, too.

Version 1.3: Fixed a long standing problem with assign-added font directories which are now respected by the cache. Thanks to Dirk Neubauer for the hints. Fixed a bug in the 1.2 code that wrote invalid caches from time to time and which resulted in fewer "cache hits" as possible.

Version 1.4: The FontCache compares now the date in the cache with the latest directory in the FONTS: (multi-)assign instead of just checking the first FONTS: directory. Should avoid problems with multiassigned directories. Furthermore, it is checked first whether the cache is up to date, to avoid unnecessary calls of the AvailFonts function.

Version 1.5: Fixed a bug in the cache creation code in case a program requested TTextAttrs, i.e. font information with tags. Added semaphore protection to avoid disk trashing. Rewrote a part of the code that passed parameters thru various levels of the program - that's now much cleaner due to the semaphore. Added checking for a file ".nocache" in "_bullet" that avoids caching if present.

Version 1.6: Fixed an AMOS compatibility problem; strange, I fixed that one some time ago, but I made the same mistake again...

Version 1.7: Added a workaround against a bug in Scala: It does not check the return code of AvailFonts and expects a proper font list even though the function returned an error condition. The FontCache patch will now try to fill in as most fonts as possible before failing.

Version 1.8: Added the option to keep the cache file sorted to speed up the reqtools requester even if the font list is already in the internal cache of reqtools. This more or less bypasses the naive "insertion sort" algorithm by reqtools and replaces it by a smarter "quicksort".