

in

Stefan Fuchs

COLLABORATORS

	<i>TITLE :</i> in		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Stefan Fuchs	July 20, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	in	1
1.1	Patch.library Documentation	1
1.2	Features	1
1.3	Installation	2
1.4	Programmers	2
1.5	Example programs	3
1.6	Support programs	3
1.7	Distribution	4
1.8	History	4
1.9	The future of patch.library	5
1.10	The Author	6

Chapter 1

in

1.1 Patch.library Documentation

patch.library 4.0 -- High-level handling of SetFunction()

Copyright © Stefan Fuchs, February 1996. Freely distributable.

Click on one of the following topics for more detailed information:

Features	Why using patch.library
Installation	How to install patch.library
Programmers	What programmers should know
Examples	How to use the example programs
Support	What programmes supporting patch.library exist
Distribution	Conditions for redistribution
History	Information about past and present versions
Future	What is planned for future versions
Author	How to get in touch with the author

Click on the CONTENTS button at the top to return here at any time.

1.2 Features

FEATURES:

- In general patch.library offers you an easy way of installing and removing your own patches for library functions.
 - Patch.library checks, if your patch can safely be removed (e.g.: when no other task is running in the patchcode).
 - In general patches may only be removed in LIFO order. Patch.library
-

provides a mechanism that allows you to remove any patch you installed with patch.library any time.

- In addition it provides you a priority system, which allows you to specify the sequence of the patches, if more than one patch has to be installed for one library function.
- Patch.library also provides you a way to check, if your patch is already installed by supplying a name field for every patch.
- Patch.library automatically flushes caches, when needed. So software that uses patch.library will continue to work on systems with an 68040 or 68060
- With patch.library you have no problems patching pre-V36 dos.library functions, since patch.library automatically detects these nonstandard functions and takes the appropriate steps to patch these functions.
- The tasks using the patch may be limited to certain tasknames and/or taskIDs.
- But always remember that installing and especially removing patches can never be coded without any risk of crashing the machine. Although patch.library does everything possible to minimize the chances of a crash, do not use its functions without good care (e.g. minimize the number of install and remove operations) says who? in V5 we'll see!

1.3 Installation

To install patch.library simply copy the file libs/patch.library to libs:

Patch.library will work on any Amiga with any Kickstart version, but will take advantage of features only provided with higher versions (e.g.: memory pools).

1.4 Programmers

The patchcode you install must obey these rules:

- Patchcodes MUST be reentrant! (i.e. library-functions can always be used by multiple tasks at the same time)
 - Patchcodes MUST preserve ALL registers!
(If replacing a library function a valid returncode must be set as documented in the Autodocs)
 - Patchcodes MUST be pc-relative, if NewCodeSize > 0!
 - Patchcodes should use as little stack as possible, because tasks using your code may otherwise run out of stack.
 - Patchcodes can be ended either with the rts-instruction, which is the normal method, or with the FALLBACK macro provided in 'Patch.i'. The FALLBACK macro allows You to end the patch and execute the original library function, if you replaced the library function (priority = 0). The FALLBACK macro functions like a 'rts'-instruction, if priority <> 0.
 - WARNING: The FALLBACK macro may be replaced by a better solution in V5,
-

and may eventually no longer work in later versions. Please E-Mail me, if you use the FALLBACK macro.

Some more notes and warnings, if you want to use patch.library:

- Some library functions use in-line code (e.g.: exec.library/GetCC())
Patching these functions will always fail.
- Patching exec.library/Switch() with patch.library is not possible and will crash the system, because this call switches between stacks.
(This function is patched by XOper)
Should do no harm because this function is PRIVATE anyway.
- There is always a chance (very small) that memory will be deallocated, while instructions from it will still executed (-> crash).
- Patching functions will use some stack (at least 4 Bytes) from programs calling the patched function. This might crash the machine, if a program has only a very small stack reserve.
- The patches should be written in assembler - support for C-patches is currently not fully provided, but it should be possible to write the appropriate stub.

More infos for programmers can be found in the accompanying file 'patch.doc' and in the example programs.

1.5 Example programs

The package includes two programs with assembler source to show you how to use the patch.library:

1. CPUClr: Installs a patch routine for graphics.library/BltClear(), which uses the CPU instead of the BLITTER to clear Chipmem.
It shows how easy and save it can be to install and remove a patch, without wasting memory or CPU time.
With the program CPUClrTEST you can check how much faster memory clearing becomes.
For more informations about CPUClr see CPUClr.doc.
2. ShowNeededFiles: (Simple SnoopDos)
Installs some patch routines to monitor dos.library/Open(), Lock() and LoadSeg() functions. It shows how the priority system of patch.library works.
Note that this program works with all versions of the dos.library.
Break this program with CTRL-C.

1.6 Support programs

The following programs support patch.library and should be available on Aminet:

- PatchSetFunc
It patches the exec.library/SetFunction() to use patch.library instead of the normal OS-function.

- PatchSupervisor

It patches the `exec.library/OldOpenLibrary()`, `OpenLibrary()`, `OpenDevice()` functions and allows disk-based libraries to be expunged (in other words: removed from memory), if no task uses it (`Opencount` is zero), but patches are still installed. If the library (or device) is opened again, the patches will automatically be reinstalled.

So this program may save you some precious memory (on my system more than 150 KB), especially when used in conjunction with `PatchSetFunc`.

1.7 Distribution

This material is © Copyright 1993-96 by Stefan Fuchs. All rights reserved.

It may be distributed freely as long as the following restrictions are met:

- The distributor may charge a fee to recover distribution costs. The fee for diskette distribution should not be more than the cost to obtain the same diskette from Fred Fish.
- The distributor agrees to cease distributing the programs and data involved if requested to do so by the author.
- You may copy and distribute verbatim copies of the program's executable code and documentation as you receive it, in any medium, provided that you conspicuously and appropriately publish only the original, unmodified program, with all copyright notices and disclaimers of warranty intact and including all the accompanying documentation, example files and anything else that came with the original.
- If you are interested in including any of this material in a commercial product, you should contact the author for his permission.
- The author will not be liable for any damage arising from the failure of the programs or the library to perform as described, or any destruction of other programs using the library residing on a system. While I know of no damaging errors, the user of this package uses it at his or her own risk.

This package may be distributed in PD-series (e.g.: the Fred Fish library) or on the Aminet.

1.8 History

NEWS FOR V4:

- New functions for V4:
 - * `CreatePatchProject()`

- Introduces the new project management system
- * `RemovePatchProject()`
 - Remove many patches with one function call
- `Patch.library` now provides a hook mechanism, which will be called, when a patch is permanently removed from the system.
- `FindPatchTags()` has now tags, which allow case-independent search and searching for multiple patches of the same name
- Patches can now be disabled/enabled
- Caches will now be cleared with all Kickstart versions, when required. Also fixes a bug, which could crash the machine on 68040 and above (Made a bad assumption about the order of the bytes in a cache being written out to memory)
- `Patch.library` now supports the external `PatchSupervisor` program, which helps to save memory, when a library is patched, but not used by applications.
- `Patch.library` now uses memory pools, when running on Kickstart 3.0 or higher.

NEWS FOR V3:

- New functions for V3:
 - * `FindPatchTags()`
 - can be used instead of the old `FindPatch()` function, but takes a taglist as parameter for future enhancements.
 - * `SetPatch()`
 - Makes it possible to limit the tasks, which will use a patch. (e.g.: A `OpenWindow()` function patch will only apply, if `CygnusED` opens a window / A `DisplayBeep()` function patch will be used for all tasks, but not for the task, which has the TaskID \$007c835a). Some other attributes of a patch may also be set.
 - * `GetPatch()`
 - This function will, return some attributes and lists of a patch.
 - * `PatchFreeVec()`
 - Frees memory returned by `GetPatch()`.

NEWS FOR V2:

- New functions for V2:
 - * `RemovePatchTags()`
 - replaces the obsolete functions `RemovePatch()` and `WaitRemovePatch()` patches can now be removed ANY TIME
 - * `InstallPatchTags()`
 - replaces the obsolete function `InstallPatch()` patching of libraries, devices and resources is now possible
- Many of the internal structures have been made public, so the lists may be scanned by application programs. Note that these structures were rearranged for V2, so make sure that at least `patch.library V2` is installed. Also make sure to use the semaphore protection, when scanning these lists.
- Some bugs have been removed and a few possible locking problems have been solved.

1.9 The future of patch.library

What will be done in the future to enhance patch.library:

- Create AmigaGuide documentation for all programmes.
- Write a commodity similar to the Commodity Exchange program that allows manipulations of the IncludeTask, ExcludeTask lists of the patches and a way to load and save these settings (already under development). This will however need some new features in patch.library like a notification system, which will only be available in the next patch.library release.
- Implement some special code, which makes removing patches 100% save (at least in theory).
- Program more applications using the patch.library.
- Bug fixing

1.10 The Author

To contact the author for bugreports, hints, ideas, donations,

write to:

Stefan Fuchs
Oskar-von-Miller-Str. 49

E-Mail: snfuchs@sokrates.franken.de

D - 90478 Nürnberg
GERMANY

AND FINALLY: If you use patch.library in any of your projects, please send me a copy, of your program, so I can make sure that everything works fine with future versions of patch.library.
