RTrack Documentation

Boris Folgmann

COLLABORATORS			
	TITLE : RTrack Documentation		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Boris Folgmann	February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 RTrack Documentation **1**

1.1 RTrack . . . 1
1.2 copyrights . . . 1
1.3 description . . . 2
1.4 requirements . . . 2
1.5 contents . . . 2
1.6 installation . . . 3
1.7 usage . . . 3
1.8 functions . . . 4
1.9 hints . . . 5
1.10 history . . . 5

Chapter 1

RTrack Documentation

1.1 RTrack

RTrack - Resource Tracking Link Library

Freeware © 1995 by PROXITY SOFTWORKS

Development by Boris Folgmann

RTrack 0.1 (15.8.95) User Manual

Copyrights Copyright information

Disclaimer Legal stuff

Description What is it for?

System requirements What is needed?

Contents Archive contents

Installation How to install

Usage How is it used?

Functions Available functions

Hints Further information

History What's new?

Support How to contact us

Update Where to get new releases

Credits Thanks to some persons

1.2 copyrights

COPYRIGHTS

Unless otherwise noted, all files are

Freeware © 1995 by PROXITY SOFTWORKS

All Rights Reserved.

MagicWB © 1994 Martin Huttenloher

Kickstart and Workbench are Copyright

© 1985-1995 by Amiga Technologies

1.3 description

DESCRIPTION

RTrack.lib is a link library for resource tracking. All resource allocations are maintained in a linked list and automatically freed before exiting. The linked list is located in a memory pool, so the small allocations don't fragment memory. One RTrack list node is only 12 Bytes in size and the list handling is so fast that you won't have any overhead worth mentioning.

This version works 100% but the set of supported routines is not complete yet. Please contact the author if you need a specific function in your program.

1.4 requirements

SYSTEM REQUIREMENTS

Kickstart 2.04

Workbench 2.0

amiga.lib V40 for memory pool functions

1.5 contents

CONTENTS

This software package consists of the following files:

RTrack.lib

The RTrack link library.

include/proto/rtrack.h

include/clib/rtrack_protos.h

Headerfiles for using RTrack.lib.

test.c

SMAKEFILE

SCOPTIONS

Example program.

RTrack.guide

This AmigaGuide document for Multiview.

PSI

Proxity Softworks information text.

Some icons are part of MagicWB and included with permission of the author.

1.6 installation

INSTALLATION

If you use SAS/C simply copy the includes to your INCLUDE: directory and
RTrack.lib to your LIB: directory.

copy include include: all

copy RTrack.lib lib:

Place this documentation where you want.

1.7 usage

USAGE

This version of the link library is in SAS/C object module library format.
RTrack uses some auto initialisation and termination functions (beginning
with _STI and _STD) which are supported by SAS/C only. Therefore you have
to link with SAS/C startup-code.
If you use no startup-code or your own custom startup-code you must call
_STI_RTrackInit() before and _STD_RTrackCleanup() after your main()
function. Note that _STI_RTrackInit() returns non zero for failure, in
this case you should not call your main function but _STD_RTrackCleanup()
and then exit your program with RETURN_FAIL (returncode 20).

```
int YourProgram()
{
int rc;
... /* Init stuff, open dos etc. */
if (! _STI_RTrackInit())
rc = main();
else
rc = RETURN_FAIL;
_STD_RTrackCleanup();
... /* Cleanup stuff, close dos etc. */
return rc;
}
```

If you compile with SAS/C standard startup-code this is all done
automatically. Make sure that dos.library is opened before
_STI_RTrackInit() is called because ERROR_NO_FREE_STORE is set as secondary
result code with SetIoErr() if memory pool allocation fails.
If you compile with a different ANSI-C compiler which can link with the
SAS/C link library it should work like this:

```
int main(void)
{
atexit(_STD_RTrackCleanup); /* Let exit() call the cleanup function! */
/* Needed for RTracks autoexit function. */
if (_STI_RTrackInit()) /* exit() on init error */
exit(RETURN_FAIL);
... /* go on with your stuff */
return your_rc; /* _STI_RTrackCleanup is called */
/* automatically! */
}
```

1.8 functions

FUNCTIONS
All functions of RTrack.lib begin with 'rk'.
e.g. allocate some mem:
mem = rkAllocVec(100, MEMF_ANY);
If you want to exit on failures automatically then call at any time:
rkAutoExit(TRUE);
If any RTrack call fails RTrack will not return NULL but call
exit(RETURN_FAIL);
This behaviour can be disabled at any time:
rkAutoExit(FALSE);
You need to have your own exit() function, if you don't link with
startup-code! If you want to free a resoure manually before exiting use
the appropriate function for explicit freeing:
rkFreeVec(mem);
RTrack will not call FreeVec() if the supplied pointer is not in the
allocated resources list but give an error message and continue the
program.
Have a look at the test.c program. Compile with:
sc test.c LIB RTrack.lib
RTrackDump() is for debugging purposes only and prints the contents of the
list.
If you use RTrack remember to only autofree resources which are
needed during the whole program run. Others should be freed as
soon as they are no longer needed with the appropriate rkFunction.
If your executable is linked with catch.o the resources will also be freed
if your program crashes.

1.9 hints

HINTS

As a bonus you are able to use RTracks private memory pool from within your program. Therefore you don't have to handle your own memory pool if you are satisfied with a pool with MEMF_ANY and a puddle size of 1K (this is the current value and may change in the future). The functions are called RTrackAllocAny() and RTrackFreeAny(). They are easy to use e.g. for storing strings or other small data structures. Don't use it for public data which needs to be shared with other tasks since you need MEMF_PUBLIC in this case.

Note: Standard AllocMem() and FreeMem() are not provided as RTrack functions since RTrack must keep track of the size of the memory anyway, so use RTrackAllocVec() and RTrackFreeVec() instead.

1.10 history

HISTORY

0.1 (15.8.95) Release

First public release.