



Technology Group

GEAR

Map File Analyzer Tool

28 May 2008



1. TABLE OF CONTENT

1.	Table of content	2
2.	List of tables.....	3
3.	List of figures.....	3
4.	Document history.....	4
5.	Purpose.....	4
6.	Audience	4
7.	Description.....	4
7.1	Command line usage	4
8.	Options.....	5
9.	Filters	6
10.	Sorting.....	7
11.	Grouping	7
12.	Scripting.....	8
12.1	Command list:	8
13.	Example usage	9
14.	Tips	9
15.	Known Limitations	9

2. LIST OF TABLES

Error! No table of figures entries found.

3. LIST OF FIGURES

No table of figures entries found.

4. DOCUMENT HISTORY

Revision	Date	Author	Reference	Description
000	April 10 th 2008	Aouellet	None	First Draft
001	May 22 nd	Fjoanis	None	Added documentation for new features
002	May 28 th	Pjette	None	Corrections and page layout

5. PURPOSE

The purpose of this document is to serve as an instruction manual for the MapFileAnalyzer tool. As such, it will explain how to use the tool but not the design architectures and decisions taken throughout its development.

6. AUDIENCE

This document is intended to Gear clients.

7. DESCRIPTION

The map file interpreter is a tool that analyzes a compiler-generated map file and outputs it in a human-readable format. Once a map file is loaded in the tool, many options are available to the user (such as filters and sorting) in order to extract the exact piece of information that is needed. Such a tool is useful to find what part of an executable takes the more space in memory and can be used to track the impact of modifications on the code size.

This tool can be used both with a GUI and as command line (which means it can easily be automated as part of building/testing process). Arguments are passed by command line options or by using a script. If you enter MapFileInterpreter.exe without any arguments it will start the GUI, otherwise it will start the command line tool.

As of now, only Nintendo DS map files are supported.

7.1 Command line usage

To get a brief summary of available commands, you can type:

```
./MapFileAnalyzer.exe -h
```

And the output will be

Usage: <map file> [options]

Options:

- h Show this help message
- n <count> Limit the output to <count> lines max
- o <file> Output the result to <file>. If not specified, the result is output to stdout

- f<filter> Apply a filter on column (refer to documentation for the filter format)

- g <column> Group results for each different value found in <column>

- s <column> Sort results according to <column> order
- sd <column> Sort descending results according to <column> order
- r <file> Run a script file
- c <mask> Semicolon-separated list of columns to show. By default, all columns are shown

- m <file> Used to specify the path to the MHFCompanion.xml File. The file is used for the module column. Without that file specified, module column won't be usable. Valid column names: address, size, section, library, file, module and symbol.

The syntax you should use to pass parameters on the command line will differ depending on which shell you are using. In Windows cmd.exe, whenever you need to supply a parameter that contains spaces (such as a filter expression), you must enclose it between double quotation marks (""). If you are using bash/cygwin, it's preferable to use single quotation marks (') instead. The reason for this is that it will prevent some part of the content being interpreted by the shell (e.g. the !~ operator that means "does not match" in the tool, but confuses bash).

Whenever you see something enclosed between brackets (<>) in the help message, it means that it's a mandatory option. The first parameter must be the map file to load, followed by options (if needed). When only a map file is specified, the entire content is outputted without being sorted, filtered or grouped.

8. OPTIONS

-n <count> Use this parameter to limit the line count that should be outputted. Only the first <count> lines will show on the console.

-o <file> By default, results are outputted to stdout. If the -f options is specified, followed by a file name, the content will be written to the file.

-f <filter> Apply a filter expression. There are eight (8) supported operators: ==, !=, <, <=, >, >=, ~~, !~. You can apply as many filters as you want, just make sure that they are separated by semicolons. For more information, see the "filter" section of this document.

-g <column> This option allows to group column content based on fields in <column>. For every different field in <column>, other columns are merged. You can only group by section, symbol, module, library and file. The concept is similar to the sql command GROUP BY.

-s <column> Ascending sort of all lines from values in <column>

-sd <column> Descending sort of all lines from values in <column>

-r <file> Run the script in <file>. The available commands are explained below in the section "scripting"

-c <mask> Used to specify a semicolon-separated list of column to show. By default, all columns are shown. Valid column name: address, size, section, symbol, library, file, module

-m <file> used to specify the path to the file MHFCompanion.xml for the module column. Without that file specified, module columns won't be usable.

9. FILTERS

Filters are of the form <column name> <operator> <value>. You can apply a filter on any column. There are eight (8) supported operators:

Equivalence/difference: ==, !=, ~~ , !~

Relation: <, <=, >, >=

Relation operator can only be applied on numerical columns (address and size). The == and != operators can be applied on either numerical or string columns. Also, if they are used to match content on string column, the comparison is case-sensitive. Note : size and addresses values can be entered in decimal or hexadecimal (With the 0x prefix in front of them).

The ~~ and !~ operators are used with perl-like regular expression. ~~ is semantically equivalent with ==, while !~ is the opposite (not-match). If the <value> part of the expression contains spaces and you are using the command line, don't forget to enclose it with quotation marks (" or '). Regular expressions are not case-sensitive.

Example:

size>32

library==gear_nitro.lib

symbol~~"Gear.*tag"

When more than one filter is specified, they cumulate. In our example, only symbols that are more than 0x32 bytes AND are from the gear_nitro.lib library AND begin by Gear AND end by tag will be included in the final results.

They can be combined like this:
`size>32;library==gear_nitro.lib;symbol~~"Gear.*tag"`

For a quick introduction to perl-like regular expression, you can read <http://www.cs.tut.fi/~jkorpela/perl/regexp.html>

10. SORTING

To sort results based on a column, simply use its name. You can sort ascendingly or descendingly.

11. GROUPING

Equivalent to the sql GROUP BY command. It allows to get relevant information such as:

- Space occupied in each section (.text, .data, ...)
- Space occupied in each object file
- ...

When columns are grouped, the size column is cumulative. If all other column values are the same for a particular group, you'll see their value in the output, if not, you'll see a dash. More details in the examples below.

The merge is done like this:

- The size field is additive for all symbols in the same group
- For any other field, if all the symbols in a group have the same value for each column, the value is preserved; otherwise, it becomes a dash.

Example: Let's say we want to group the following symbols by section (the third column)

02000000	2128 .text	libsyscall.a	_svc_mw	.text
02000850	340 .text	empty_tests	crt0	_start
020009a4	28 .text	empty_tests	crt0	INITi_CpuClear32
02000bf0	36 .rodata	empty_tests	crt0	_start_ModuleParams
02000bf0	36 .rodata	empty_tests	crt0	_start_FindDrives

The result will be:

-	2496	.text	-	-	-
-	72	.rodata	empty_tests	crt0	-

All entries which section was equivalent to the string ‘.text’ had their sizes added. Therefore, what we see (2496) is the total size of those entries (2128 + 340 + 28 == 2496). The same goes for the ‘.rodata’ entries.

Entries of the same group that have different values in their respective columns have those values replaced with a dash; otherwise values that were all the same, are preserved. For example, this means that if we want to group values with regards to their ‘section’ string, if the values do not all share the same address, we will not display any address in order not to confuse the client.

12. SCRIPTING

You can do the exact same thing that you can do at the command line by writing a script. Each command line option has a scripting equivalent. You specify one command per line, followed by its parameters.

12.1 Command list:

```
filter <filters>
column <column>
output <file>
line <count>
sort <column>
sortdescending <column>
groupby <column>
module <MHFCompanion.xml>
```

Once your script is written, you call it by using the -s command line option, followed by the script's file name. You can use both command line options and script simultaneously. Please be cautious about the order in which the options are specified. If you load the script using -s, and then redefine some options, they will override the previous ones. The opposite situation happens if you load the script at the end. The only exception are filters, as they get appended to the existing list of filters.

13. EXAMPLE USAGE

You will see an example of results for each command.

Ten biggest symbols from Gear namespace, show only size, file and symbol column:

```
MapFileAnalyzer.exe TestRunner.nef.xMAP -n 10 -sd size -c 'size;file;symbol' -f 'symbol~~Gear::~*'
```

Ten biggest symbols from Gear namespace, grouped by object file, show only size and file column

```
MapFileAnalyzer.exe TestRunner.nef.xMAP -n 10 -sd size -c 'size;file' -f 'symbol~~Gear::~*' -g file
```

Summary of size by section

```
MapFileAnalyzer.exe TestRunner.nef.xMAP -sd size -c 'size;section' -g section
```

Show code size of a particular Gear module (filter + group file)

```
MapFileAnalyzer.exe 03empty_tests.nef.xMAP -c 'size;file' -g file -f 'file==diskdevice'
```

Top 10 of files that use the more space

```
MapFileAnalyzer.exe 03empty_tests.nef.xMAP -c 'size;file' -g file -n 10 -sd size
```

14. TIPS

- ✓ It might be useful to disable the linker dead-symbol stripping.
- ✓ The mapfile generated by the CW linker can be made smaller by removing useless information. You can uncheck the "show transitive closure".

15. KNOWN LIMITATIONS

The CW linker truncates filename to 15 character when it generates the map file. Take this into account if you try to match a filter against a file.