

TOFO

Tandem Object File Optimizer

07. April 2015

Version 101

We found out, that the new compilers of Itanium systems produce a lot of slack in object files, consuming large amounts of disk space.

In the 'good old times', a compiler allocated one large extent, able to hold the entire object file. This was efficient in terms of disk space usage, and load times.

This seems to have changed.

The problem now is, that objects are stored in files, where the first extent is pretty small, and the needed secondary extents are pretty big. The small first extent is unable to hold the entire object file, so a second extent has to be allocated.

Because the secondary extents are really big, a large amount of disk space is wasted.

This results in the fact, that an object file looks small (small EOF), but requires a large portion of allocated disk space.

To optimize the disk space, needed to hold an executable file, we developed the Tandem Object File Optimizer (TOFO) product. TOFO copies an existing object file, having more than 1 extent, into a file with a primary extent, which is able to hold the entire object file.

This saves a lot of disk space, and optimizes the creation time of a new process, started from this object.

Command syntax

```
[run] TOFO [/OUT <file>/] -HELP  
[LIST [<template>]]  
OPTIMIZE [<template>]
```

where:

-HELP

displays the command syntax.

[LIST [<template>]]

displays the allocation situation of all files, matching <template>.

When <template> is missing, the object files of the users current location are listed.

Running TOFO without any parameter causes it to LIST the situation of the object files of the users current location.

OPTIMIZE [<template>]

optimizes the files, matching <template>.

When <template> is missing, the files, matching <template> in the users current location are optimized.

Installation

TOFO comes as a set of executable files with file code 100, 700 and 800.

- Code 100 can be used on K, S and Itanium machines
- Code 700 can be used on K and S machines only
- Code 800 can be used on Itanium machines only

Simply place the appropriate version of TOFO into the search path of your system, e.g. `$SYSTEM.SYSTEM.TOFO`

Security Settings

TOFO does not need to become licensed, nor to be PROGIDed.

The owner of TOFO should be set to the system owner, e.g. `SUPER.SUPER`.

The security should be set to: `OAO`, or the equivalent in `SAFEGUARD`.

Usage

Best is, to incorporate TOFO in the compilation batch files, and to execute it, when all object file manipulations (such as `AXCEL`, `BIND`, `nld`, `eld` etc.) are done.

e.g.:

```
eptal/in source/object;suppress,optimize 2
eld -o object object -s -set systype guardian
TOFO optimize object
```

This minimizes the disk space, occupied by the object file, dramatically (see real life example below), and shortens process creation times.

Restrictions

TOFO works with unstructured files only, having these file codes:

- 100 = non native object files
- 180 = C files
- 500 = native NonStop X object files
- 700 = native MIPS [K and S systems] object files
- 800 = native Itanium object files

Example

LIST

```
$GHS1 TOFO 23> tofo list object
TOFO (101) - T7172H06 - (07Apr2015) System \GINKGO, running NSK H06
Copyright (c) GreenHouse Software & Consulting 2007,2015
  File Name                Allocated      In Use      %Waste
$GHS1.TOFO.OBJECT          1,118,208      64,640      95
Num processed: 1
Allocated:                1,118,208
In use:                    64,640
```

OPTIMIZE

```
$GHS1 TOFO 24> tofo optimize object
TOFO (101) - T7172H06 - (07Apr2015) System \GINKGO, running NSK H06
Copyright (c) GreenHouse Software & Consulting 2007,2015
  File Name                Old Allocated  New Allocated
$GHS1.TOFO.OBJECT          1,118,208      86,016
Num processed: 1
```

LIST

```
$GHS1 TOFO 25> tofo list object
TOFO (101) - T7172H06 - (07Apr2015) System \GINKGO, running NSK H06
Copyright (c) GreenHouse Software & Consulting 2007,2015
  File Name                Allocated      In Use      %Waste
$GHS1.TOFO.OBJECT          86,016         64,640      25
Num processed: 1
Allocated:                86,016
In use:                    64,640
$GHS1 TOFO 26>
```

In the example above, OBJECT was compiled with the epTAL compiler. The first LIST command displays the disk space allocation, done by the epTAL compiler. It shows, that the compiler allocated 1,1 MB of disk space for a file, that requires 64 KB.

The OPTIMIZE command then compresses the object file. It as well displays the newly allocated space, which now is 86 KB.

A subsequent LIST command verifies this.


FAQ's

Why does TOFO not compress the file as much as possible?

This depends on the system it is running on:

- On K and S systems, the compression is much better. Unstructured files were expanded from 512 to 514 bytes in 197x by adding a 16 bit checksum. This required Tandem special formatted disk drives. The advantage is that this checksum was calculated in the controller, and transparently written to disk. It as well ensured that the required disk space was optimally used! The disadvantage is that the disk drives are not 'industry compatible'.
- On Itanium systems, disk drives are standard disks, formatted with 512 byte records. To keep the record check sum of unstructured files, these files require some structure and space somewhere within the file. In other words: An unstructured file is no longer really an unstructured file, but needs an internal structure to keep the checksums. To do so, in minimum 14 extents are allocated for the smallest file. And here is another disadvantage of this new behaviour: Allocating disk space by setting the EOF to the end of an unstructured file may take quite some time – because the system needs to write the structure for the unstructured allocated space – even if it is empty. Depending on the file size, this may take minutes!

In case you stumble into problems, please let us know.

Carl Weber
GreenHouse Software & Consulting 
07Apr2015