

Dual Guide

- Introduction.....2
- Concept.....2
- Status.....2
- External patches.....2
- Commands.....2
 - Patch creation.....3
 - C command: create patch.....3
 - Patch management.....3
 - A command: apply patch.....3
 - U command: undo patch.....3
 - V command: verify patch.....3
 - Patch analyses.....3
 - X command: display compatibility.....3
 - M command: display offsets.....3

Introduction

Dual is a patcher designed for the dynamic management of multiple patches in a single target file. It allows to check, apply, undo or analyze patches in a safe way.

It follows a different direction from standard patchers and it isn't recommended for the creation of small files for distribution. However, external compression and external patchers can be used simultaneously with dual for this purpose.

Dual is compiled in the C language and only uses the standard C library and it should have high portability between different operating systems. The binaries included are for the linux systems but the source files are included for compilation.

Concept

Dual checks the difference between two files of the same size, called the original file and the modified file. Unlike standard patchers, it stores the bytes of both files when they differ in the same file position: the undo bytes, relative to the original file, and the applied bytes, relative to the modified file.

When a patch is applied to the target file, the undo bytes are used as a security check. If the undo bytes don't match the actual bytes of the target file, the patch isn't applied. This procedure avoids the collision of applied patches.

Dual can analyze the patches and display the compatibilities between them. With this information, the user can dynamically undo and apply patches for a specific setup of patches.

Status

When the patches are checked against the target file, the following status are evaluated:

- **X status:** The undo bytes and the applied bytes don't match the bytes of the target file. The main reason of this status is that the patch is designed for a different target file. Also, if another patches were applied, the new patch isn't compatible with the already applied patches.
- **U status:** the undo bytes match the bytes of the target file. The patch can be safely applied to the target file.
- **A status:** the applied bytes match the bytes of the target file. The patch can be safely undo to the target file.

External patches

Dual only needs the original and the modified files to create a patch in dual format. To convert another patch in a different format, it is possible to apply the foreign patch in a clean copy of the original file. Then, a new dual patch can be created between the original file and the copy modified by the foreign patch. This procedure can be used to convert between many patch with different formats.

Commands

A summary of the commands is displayed if dual is called with the -h flag:

- dual -h

Patch creation

C command: create patch

- Syntax: `dual c original_file modified_file patch_file`
- Example: `dual c alpha.bin beta.bin patch.dua`

Creates a new dual patch. The patch stores the bytes of the original and modified files when they differ in value for the same position.

The original and modified files must have the same size and be below 4GiB.

Patch management

A command: apply patch

- Syntax: `dual a target_file patch_file [patch_file...]`
- Example: `dual a target.bin patch.dua`

Try to apply all referenced patches in the target file. Each patch is checked against the target file and only is applied if the patch has the U status. Otherwise, the patch isn't applied.

U command: undo patch

- Syntax: `dual u target_file patch_file [patch_file...]`
- Example: `dual u target.bin patch.dua`

Try to undo all referenced patches in the target file. Each patch is checked against the target file and only is undo if the patch has the A status. Otherwise, the patch isn't undo.

V command: verify patch

- Syntax: `dual v target_file patch_file [patch_file...]`
- Example: `dual v target.bin *.dua`

Check all patches against the target file and display their status (X, U or A). The main reason for this command is to discover the patches that were already applied in the target file.

Patch analyses

X command: display compatibility

- Syntax: `dual x patch_file patch_file [patch_file...]`
- Example: `dual x *.dua`

Analyze all referenced patches and display the incompatibilities between them. The main reason for this command is to get information to build a specific set of patches without collision between them.

M command: display offsets

- Syntax: `dual m patch_file [patch_file...]`
- Example: `dual m *.dua`

Display a list of the patches segments for all referenced patches. It is mainly used for debugging purposes.

The number of bytes by segment is actually the internal value stored in the patch. It corresponds to the individual size of the undo or applied bytes. Together, the effective number of bytes stored by segment is the double of the displayed size.