Creating Symbolic Links

The function **CreateSymbolicLink** allows you to create symbolic links using either an absolute or relative path.

Symbolic links can either be absolute or relative links. Absolute links are links that specify each portion of the path name; relative links are determined relative to where relative–link specifiers are in a specified path. Relative links are specified using the following conventions:

- Dot (. and ..) conventions—for example, "..\" resolves the path relative to the parent directory.
- Names with no slashes (\)—for example, "tmp" resolves the path relative to the current directory.
- Root relative—for example, "\Windows\System32" resolves to the "*current drive*:\Windows\System32". directory
- Current working directory-relative—for example, if the current working directory is "C:\Windows\System32", "C:File.txt" resolves to "C:\Windows\System32\File.txt".

Note If you specify a current working directory–relative link, it is created as an absolute link, due to the way the current working directory is processed based on the user and the thread.

A symbolic link can also contain both junction points and mounted folders as a part of the path name.

Symbolic links can point directly to a remote file or directory using the UNC path.

Relative symbolic links are restricted to a single volume.

Example of an Absolute Symbolic Link

In this example, the original path contains a component, '*x*', which is an absolute symbolic link. When '*x*' is encountered, the fragment of the original path up to and including '*x*' is completely replaced by the path that is pointed to by '*x*'. The remainder of the path after '*x*' is appended to this new path. This now becomes the modified path.

X: "C:\alpha\beta\absLink\gamma\file"

Link: "absLink" maps to "\\machineB\share"

Modified Path: "\\machineB\share\gamma\file"

Example of a Relative Symbolic Links

In this example, the original path contains a component '*x*', which is a relative symbolic link. When '*x*' is encountered, '*x*' is completely replaced by the new fragment pointed to by '*x*'. The remainder of the path after '*x*', is appended to the new path. Any dots (..) in this new path replace components that appear before the dots (..). Each set of dots replace the component preceding. If the number of dots (..) exceed the number of components, an error is returned. Otherwise, when all component replacement has finished, the final, modified path remains.

X: C:\alpha\beta\link\gamma\file

Link: "link" maps to "..\..\theta"

Modified Path: "C:\alpha\beta\..\..\theta\gamma\file"

Final Path: "C:\theta\gamma\file"

Related topics

Symbolic Links
Hard Links and Junctions
Naming Files, Paths, and Namespaces