PEMANFAATAN FITUR TERSEMBUNYI PADA NTFS BAGI KOLEKTOR MEDIA DIGITAL

Choerun Asnawi

Program Studi D3 Manajemen Informatika
STMIK Jenderal Achmad Yani Yogyakarta

c.asnawi@gmail.com

ABSTRAK

NT File System (NTFS) merupakan sistem berkas yang saat ini paling banyak digunakan pada sistem operasi Microsoft Windows. NTFS memiliki banyak fitur yang tidak banyak dipublikasikan sehingga banyak pengguna yang tidak mengetahuinya. Beberapa dari fitur tersebut dapat dimanfaatkan untuk memudahkan penataan koleksi media digital, semacam film, foto, musik, buku elektronik (e-book) atau software. Hal ini tentu saja sangat berguna bagi seorang kolektor media digital. Fitur-fitur yang bermanfaat itu antara lain adalah Alternate Data Streams (ADS), hard link dan directory junction. ADS dapat digunakan untuk menyimpan metadata dari setiap item atau kelompok item dalam koleksi. Hard link dan directory junction dapat digunakan untuk mereferensi sebuah item atau kelompok item dalam koleksi dengan lebih dari satu nama atau lokasi, tanpa harus melakukan duplikasi data.

Kata Kunci: NTFS, Alternate Data Streams, *hard link*, *directory junction*, koleksi, media digital.

PENDAHULUAN

Hobi mengkoleksi meliputi berbagai aktivitas mulai dari mencari, menemukan, memperoleh, menata, mengkatalogkan, menampilkan, menyimpan, dan memelihara sekelompok item yang menjadi minat dari sang kolektor. Bagi seorang kolektor, menjadi suatu keharusan baginya untuk dapat menata koleksinya dengan seefektif dan seefisien mungkin. Tujuan utamanya adalah supaya lebih mudah untuk menelusuri dan menemukan kembali item-item dalam koleksinya tersebut. Penataan yang baik juga diharapkan akan mengurangi atau menghindari adanya duplikasi item, terutama saat ukuran koleksi sudah sangat besar. Selain itu koleksi yang tertata dengan baik akan lebih mudah untuk disampaikan atau diperlihatkan atau "dipamerkan" kepada orang lain yang memiliki minat yang sama. Dengan penataan koleksi yang rapi maka akan menjadi lebih mudah jika akan melakukan pertukaran koleksi.

Duplikasi item dan kesulitan dalam menemukan item merupakan masalah utama dalam kegiatan mengkoleksi. Hal tersebut juga berlaku pada koleksi media digital. Untuk menghindari permasalahan tersebut dapat dilakukan dengan menggunakan program aplikasi tertentu yang memang bertujuan untuk melakukan penataan koleksi. Program aplikasi semacam itu kemungkinan besar

akan menyimpan data koleksi ke dalam suatu basis data, dengan menggunakan format basis data sendiri atau memanfaatkan salah satu produk sistem manajemen basis data (DBMS) yang ada.

Dalam tulisan ini akan ditunjukkan cara menata dan memeilihara koleksi media digital tanpa harus menggunakan DBMS tertentu. Penataan koleksi media digital bisa dilakukan dengan memanfaatkan sistem "basis data" bawaan yang sudah disediakan langsung oleh sistem operasi yang digunakan pada komputer, yaitu sistem berkas atau *filesystem*. Hanya saja untuk lebih mengefektifkan dan mengefisienkan penataan koleksi, maka dibutuhkan sistem berkas yang memiliki fitur-fitur tertentu. Dalam tulisan ini akan digunakan sistem berkas NTFS pada sistem operasi Microsoft Windows dan cara memanfaatkan fitur tersembunyi yang ada di dalamnya.

DASAR TEORI

Media Digital

Media digital, sebagai kebalikan dari media analog, biasanya berupa media elektronik yang menggunakan kode digital. Media digital yang dimaksudkan dalam tulisan ini antara lain berupa: video (film), foto (citra atau gambar), animasi, dan musik digital, dokumen-dokumen digital semacam buku elektronik (*e-book*) atau jurnal elektronik (*e-journal*), serta segala bentuk media yang dapat disimpan dalam bentuk berkas komputer.

Sistem Berkas NTFS

Sistem berkas (*file system* atau *filesystem*) merupakan salah satu bagian dari sistem operasi yang berguna untuk mengatur cara penamaan, penyimpanan, serta penataan berkas komputer dan data yang terkandung di dalamnya agar mudah untuk diakses dan diperoleh kembali. Sistem berkas dapat menggunakan perangkat penyimpan (*storage device*) seperti *hard disk* atau CD-ROM dan terlibat langsung dalam memelihara lokasi fisik berkas, atau hanya menyediakan akses menuju data pada server berkas dengan bertindak sebagai client dari suatu protokol jaringan semacam *Network File System* (NFS) atau *Common Internet File System* (CIFS).

Kebanyakan sistem operasi yang ada menyediakan satu atau lebih sistem berkas. Bahkan saat ini sebuah sistem berkas telah menjadi bagian tak terpisahkan dari setiap sistem operasi modern. Pada sistem operasi Microsoft Windows, dikenal adanya dua sistem berkas utama, yaitu *File Allocation Table*

(FAT) dan *New Technology File System* (NTFS). FAT adalah sistem berkas yang telah ada sebelum masa sistem operasi Windows dan merupakan sistem berkas bawaan untuk keluarga sistem operasi Windows 9x. Sedangkan NTFS merupakan sistem berkas baru yang ditujukan untuk keluarga sistem operasi Windows NT, termasuk versi-versi sesudahnya seperti Windows 2000, Windows XP, Windows Server 2003, Windows Server 2008, Vista serta Windows 7.

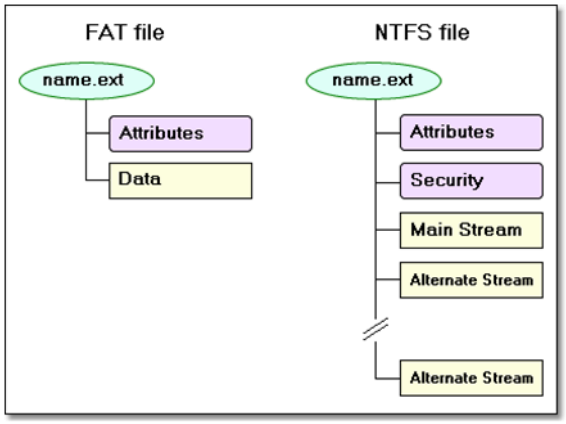
NTFS didesain sebagai pengganti sistem berkas FAT, yang memang fiturnya sudah tidak sesuai lagi untuk diterapkan pada sistem operasi modern. Sementara itu NTFS banyak menawarkan fitur-fitur yang disesuaikan dengan kebutuhan sebuah sistem operasi modern. Beberapa fitur yang ada pada NTFS merupakan hasil adopsi dari berbagai sistem berkas yang ada pada sistem operasi lain. Sebagai contoh, fitur pengaturan akses (*access control*), *volume mount point*, *disk quota*, *hard link*, *symbolic link* dan *directory junction* merupakan fitur yang diadopsi dari sistem berkas di keluarga sistem operasi UNIX. Fitur *Update Sequence Number* (USN) *Journal* merupakan fitur yang sudah ada di sistem berkas semacam *Journaled File System* (JFS) buatan IBM. Sementara fitur *Alternate Data Streams* (ADS) merupakan fitur tambahan untuk menjaga kompatibilitas dengan *Hierarchical File System* (HFS) pada Apple Macintosh.

Dari berbagai fitur yang disediakan dalam NTFS, tidak semuanya dapat diketahui dengan mudah oleh penggunanya. Banyak di antaranya yang tidak dipublikasikan serta tidak disertai dengan *tool* untuk pengguna, sehingga hanya menjadi fitur tersembunyi untuk pengguna. Dan hal itu bukannya tanpa alasan. Fitur-fitur tersebut memang tidak ditujukan langsung untuk pengguna umum, namun lebih ditujukan untuk administrator sistem. Selain itu kompatibilitas dengan sistem berkas sebelumnya (FAT) juga masih perlu diperhitungkan, terutama karena FAT saat ini masih banyak digunakan. Perlu diketahui bahwa berbagai produk USB *flash drive*, sebagai perangkat penyimpanan yang sangat umum dipakai saat ini, kebanyakan masih menggunakan FAT sebagai sistem berkas bawaannya, Penggunaan yang tidak tepat dikhawatirkan dapat mengganggu kestabilan sistem atau menghilangkan data pengguna.

Namun demikian, di dalam artikel ini akan ditunjukkan bagaimana caranya memanfaatkan beberapa fitur NTFS tersebut agar seorang kolektor media digital dapat menata koleksinya dengan lebih efektif dan efisien. Fitur NTFS yang akan dibahas dan digunakan di sini adalah fitur ADS, *hard link*, *reparse point* serta *directory junction*.

Alternate Data Streams (ADS)

Untuk menggambarkan keberadaan ADS, dapat dilakukan dengan membandingkan struktur penyimpanan berkas pada NTFS dengan FAT. Hal tersebut dapat dilihat pada gambar 1. Pada gambar tersebut dapat dilihat bahwa sebuah berkas pada NTFS dapat menyimpan lebih dari satu data, atau dalam hal ini dinamakan sebagai *stream*. Pengguna biasa secara langsung hanya dapat melihat bagian *stream* utama (*main stream*), yang dapat disamakan dengan bagian data pada berkas FAT. Sementara itu data atau *stream* alternatif tidak dapat diakses secara langsung tanpa menggunakan *tool* khusus tertentu. Dan seperti telah disampaikan di depan bahwa *tool* khusus tersebut memang sengaja tidak disediakan secara langsung oleh Microsoft sebagai pembuat NTFS.



Gambar 1: Konsep penyimpanan berkas pada NTFS dibandingkan dengan FAT (Inv Softworks, tanpa tahun)

Setiap *stream* alternatif memiliki ID yang berupa nama unik dalam berkas induknya. Karena itulah *stream* alternatif dapat disebut juga sebagai *stream* yang memiliki nama atau *stream* bernama. Sementara itu *stream* utama yang dapat diakses langsung menggunakan nama berkas tidak membutuhkan dan memang tidak memiliki suatu ID, sehingga disebut juga sebagai *stream* tanpa nama. *Stream* utama ini merupakan suatu keharusan dan selalu tersedia. Jika seorang pengguna membuat sebuah *stream* alternatif namun berkas induknya tidak ada, maka sistem akan secara otomatis membuat *stream* utama dengan panjang nol. Jika pengguna menghapus *stream* utama maka sistem menganggapnya sebagai permintaan untuk menghapus keseluruhan berkas, dan semua *stream* alternatif yang ada akan terhapus juga.

Saat sebuah program membuka berkas NTFS, sebenarnya yang dibuka adalah *stream* utama yang terkandung dalam berkas itu. Untuk mengacu pada *stream* alternatif caranya adalah menambahkan huruf titik dua (*colon*) diikuti

nama *stream* di belakang nama berkas. Sebagai contoh, **namafile.eks** mengacu pada *stream* utama pada berkas atau keseluruhan berkas dengan nama **namafile.eks**, tergantung dari konteksnya. Sementara itu teks **namafile.eks:idstream** mengacu pada *stream* alternatif dengan nama **idstream** di dalam berkas **namafile.eks**.

Sebuah direktori juga dapat memuat *stream* alternatif, yang dapat diakses seperti halnya *stream* di dalam berkas. Namun demikian, direktori tidak diperkenankan untuk memiliki *stream* tanpa nama. Setiap usaha untuk mengakses *stream* tanpa nama pada direktori akan menyebabkan kesalahan berupa penolakan akses (*access denied*).

Tautan Keras (*Hard Link*)

Sebuah *link* atau tautan pada dasarnya adalah nama alternatif untuk sebuah berkas atau direktori. Bagi yang mengenal baik keluarga sistem operasi UNIX, pasti mengetahui tentang perintah `ln` dan kegunaannya. Dalam instalasi Linux atau BSD terdapat banyak sekali tautan, dan perintah `ln` merupakan salah satu perintah yang sangat umum dipakai. Windows, terutama dengan sistem berkas NTFS-nya, juga mengenal tautan, namun sungguh mengherankan bahwa tidak tersedia *tool* yang memiliki antarmuka grafis, untuk membuat dan mengatur tautan. Microsoft hanya menyediakan sebuah *tool* berupa baris perintah (*command line*) bernama `fsutil`, yang tidak didokumentasikan dengan baik serta tidak dipublikasikan sehingga banyak *user* yang tidak tahu keberadaannya.

Tautan ada dua macam, yaitu *hard link* atau tautan keras serta *soft link* atau lebih dikenal sebagai tautan simbolis (*symbolic link*). Tautan keras menggunakan entri *Master File Table* (MFT) yang sama dengan berkas aslinya. Menambahkan sebuah tautan keras akan menciptakan sebuah atribut nama baru dan menaikkan hitungan (*counter*) tautan keras. Untuk sebuah berkas yang baru dibuat, nilai hitungan ini adalah satu. Penghapusan sebuah tautan keras akan menghilangkan nama yang terkait dan menurunkan hitungan tautan keras. Saat hitungan menjadi nol, sistem akan menghapus berkas, membebaskan ruang penyimpan yang teralokasi dan menghapus entri MFT-nya. Semua atribut nama bersifat mandiri, sehingga menghapus, memindahkan, atau mengganti nama berkas tidak akan berpengaruh terhadap tautan keras lainnya. Oleh karena semua tautan harus mengacu pada entri MFT yang sama, maka tautan keras tidak dapat dibuat pada *drive* atau *volume* yang berbeda. Keterbatasan lainnya adalah bahwa tautan keras pada NTFS hanya dapat dibuat untuk berkas, tapi tidak untuk direktori.

Reparse Point dan *Directory Junction*

Reparse point menyediakan cara lain untuk membuat tautan. Jika sebuah berkas atau direktori memiliki sebuah *reparse point* yang melekat padanya, maka sistem akan memanggil sebuah filter sistem berkas, yang diindikasikan dengan sebuah *tag reparse point*. Filter tersebut dapat mengimplementasikan sembarang metode untuk mengakses data sebenarnya. *Hierarchical Storage Management* (HSM) adalah contoh bagus pemanfaatan *reparse point*.

Tautan ke direktori yang menggunakan mekanisme *reparse point* dinamakan sebagai *directory junction* atau cukup disingkat *junction*. *Junction* ini dapat digunakan untuk mengatasi keterbatasan tautan keras yang hanya dapat digunakan pada berkas. Hanya saja perlu diingat bahwa *junction* ini sesungguhnya merupakan tautan simbolis. Akibatnya jika direktori yang diacu terhapus atau dipindahkan, maka *junction* yang mengacunya akan menjadi kehilangan acuan. Di sisi lain, *junction* memiliki kelebihan tidak dibatasi pada direktori yang berada pada *drive* yang sama, namun dapat menunjuk pada direktori di *drive* atau *volume* yang berbeda, atau bahkan kepada *drive* itu sendiri.

Dalam hal ketersediaan *tool* untuk pengguna, nasib *junction* hampir sama dengan ADS. Walaupun perintah `fsutil` dapat digunakan untuk berurusan dengan *reparse point*, namun perintah tersebut tidak dapat digunakan untuk membuatnya. Fasilitas yang ditawarkan oleh perintah `fsutil` hanyalah untuk mengecek keberadaan (*query*) dan menghapus *reparse point*. Untungnya hal ini diperbaiki saat Windows Vista dirilis dengan disertakannya sebuah perintah bernama `mklink`. Perintah `mklink` ini bahkan dapat digunakan untuk membuat tautan keras, tautan simbolis dan *junction* sekaligus.

PEMBAHASAN

DBMS Versus Sistem Berkas

Dalam menata sebuah koleksi media digital, setidaknya ada dua hal yang harus dilakukan, yaitu penentuan lokasi dan struktur penyimpanan item-item koleksi serta pembuatan katalognya. Lokasi dan struktur penyimpanan berkaitan dengan kemudahan dalam menelusuri dan melihat-lihat isi koleksi. Sementara pembuatan katalog berkaitan dengan kemudahan untuk mencari dan menemukan suatu item di dalam koleksi. Pada umumnya, item-item koleksi media digital berbentuk berkas komputer dan disimpan di dalam sistem berkas yang ada serta disusun menggunakan struktur direktori. Sementara untuk katalog

biasanya menggunakan sistem basis data yang terpisah, baik menggunakan format basis data sendiri maupun menggunakan DBMS yang ada.

Pada kenyataannya terdapat tiga macam cara dalam kaitannya dengan lokasi item serta katalognya, yaitu:

1. menggunakan sistem berkas yang tersedia untuk penyimpanan item koleksi namun menggunakan sistem basis data sebagai pencatatan katalognya;
2. baik item koleksi maupun pencatatan katalognya disimpan menjadi satu di dalam sebuah sistem basis data; dan
3. sepenuhnya memanfaatkan sistem berkas yang ada untuk penyimpanan item koleksi sekaligus katalognya.

Cara yang pertama merupakan cara yang paling mudah diimplementasikan, terutama untuk item-item koleksi yang berukuran besar seperti video. Sistem berkas sendiri sebenarnya dapat dianggap sebagai sebuah sistem basis data yang dikhususkan untuk menata penyimpanan berkas komputer. Sebagai sistem basis data bawaan asli dari sistem operasi, dapat diasumsikan bahwa sistem berkas memiliki ketahanan yang lebih baik dibandingkan produk sistem manajemen basis data yang hanya “menumpang” di atas sistem berkas. Oleh karena itu, item koleksi yang dapat disimpan sebagai berkas komputer menjadi sangat sesuai untuk ditempatkan di dalam sistem berkas itu sendiri. Sementara itu dengan menyimpan katalognya ke dalam sebuah DBMS, terutama yang bersifat relasional (RDBMS), maka akan diperoleh sebuah desain katalog yang solid namun fleksibel serta mudah dikelola. Kelemahan dari cara ini adalah terjadinya pemisahan antara katalog dengan item koleksi, sehingga integritasnya kurang dapat dijaga. Proses penambahan, pengubahan dan penghapusan item harus disertai dengan pemrosesan data di katalog, yang merupakan proses secara terpisah. Jika salah satu dari dua proses tersebut tidak dilakukan, maka dapat dipastikan integritas data akan jadi kacau. Sementara kalau harus terus menerus melakukan kedua proses tersebut tentu akan merepotkan bagi sang kolektor dan menghabiskan banyak waktunya.

Dua cara terakhir utamanya bertujuan untuk menutupi kelemahan cara pertama yang disebutkan di atas. Hal tersebut dapat dilakukan dengan menggabungkan lokasi penyimpanan item dan katalog koleksi menggunakan tempat atau metode yang sama. Dengan mempertimbangkan cara pertama di atas maka ada dua kemungkinan yang dapat dilakukan, yaitu menyatukan

keduanya di dalam DBMS yang sama atau memakai fasilitas yang disediakan sistem berkas untuk mengelola keduanya. Jika menggunakan DBMS secara penuh, maka data item dapat disimpan dalam sebuah *field* yang bertipe data *Blob*. Data bertipe *Blob* adalah sekumpulan data biner yang ditaruh di dalam DBMS sebagai satu entitas. Dibandingkan dengan menggunakan sistem berkas, penggunaan *Blob* setidaknya akan memiliki dua keuntungan berikut:

- Pekerjaan untuk mem-*backup* katalog dan data item koleksi dalam satu waktu menjadi lebih mudah.
- Integritas transaksional saat melakukan perubahan pada katalog dan isi dari koleksi dapat dijaga dengan baik. Kebanyakan sistem berkas tidak transaksional.

Namun untuk koleksi item yang rata-rata berukuran besar seperti video, cara ini sangat tidak dianjurkan. Ada beberapa alasan untuk tidak menaruh data biner di dalam basis data, antara lain:

- Maksud utama penyimpanan data dalam sebuah DBMS adalah untuk membuat semacam struktur dan pengurutan terhadap data, selain untuk maksud kemudahan pencarian data. Sayangnya hampir semua DBMS yang ada saat ini tidak atau belum menyediakan fasilitas pengurutan berdasarkan data biner maupun fasilitas pencarian di dalam data biner, apalagi untuk data yang memiliki format tertentu. Tentu saja tidak ada artinya kalau hanya mencari sederetan data biner mentah yang tak bermakna di dalam data *Blob*. Pencarian akan menjadi lebih berarti jika berkaitan langsung dengan konten yang terkandung dalam *Blob*, misalnya mencari garis dalam data citra, atau mencari *frame* dengan isi tertentu di dalam video. Kekurangan pada DBMS tersebut membuat penggunaan *Blob* menjadi berkurang nilainya.
- Untuk data yang besar, menaruh data biner di dalam basis data akan memperbesar ukuran basis data dengan sangat cepat, sehingga akan menyulitkan pengendalian ukuran basis data.
- Proses penyimpanan, pengubahan, dan pengambilan data akan menjadi lebih lambat dibandingkan dengan langsung menggunakan sistem berkas.
- Bahasa *query* yang biasanya digunakan dalam DBMS (contohnya SQL) pada dasarnya tidak diperuntukkan untuk menuliskan nilai data biner, sehingga akan merepotkan saat akan memasukkan data ke dalam tabel.

Cara yang terakhir adalah dengan memanfaatkan sistem berkas yang telah disediakan oleh sistem operasi sebagai tempat penyimpanan item sekaligus katalog dari koleksi. Cara inilah yang akan dibahas secara mendetail dalam tulisan ini, meliputi alasan pemakaiannya, keuntungan dan kekurangannya, serta cara mengoptimalkannya menggunakan sistem berkas NTFS. Sebagai tempat penyimpanan media digital, sistem berkas merupakan tempat yang paling pas. Namun untuk pembuatan katalog, yang dapat dimanfaatkan dari kebanyakan sistem berkas hanyalah struktur dan nama direktori serta nama berkas.

isbn	id_pengarang
0321474074	1
0131498622	2
0596523696	3
0596523696	4
0130463469	5

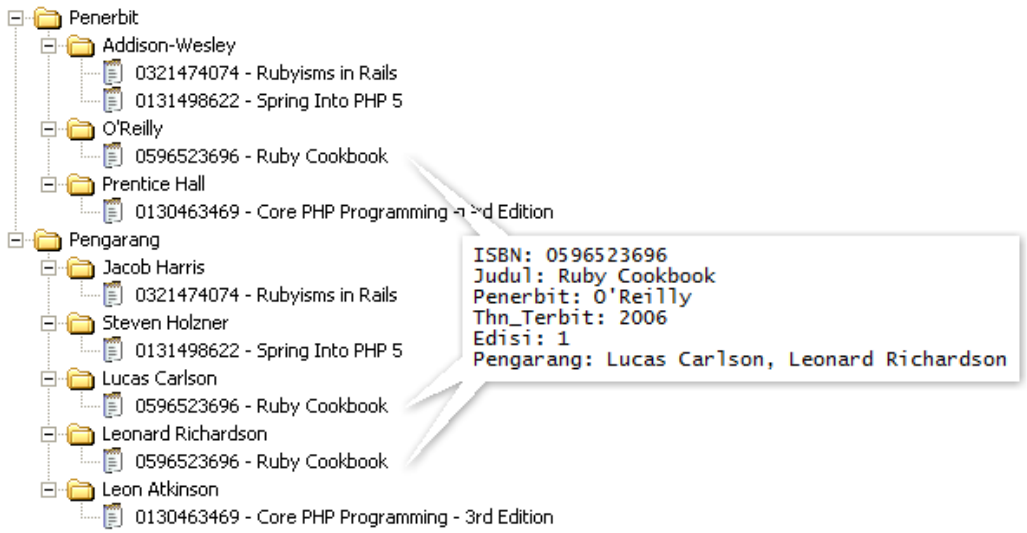
id_pengarang	nama_pengarang
1	Jacob Harris
2	Steven Holzner
3	Lucas Carlson
4	Leonard Richardson
5	Leon Atkinson

id_penerbit	nama_penerbit
1	Addison-Wesley
2	O'Reilly
3	Prentice Hall

isbn	judul	id_penerbit	thn_terbit	edisi
0321474074	Rubyisms in Rails	1	2006	1
0131498622	Spring Into PHP 5	1	2005	1
0596523696	Ruby Cookbook	2	2006	1
0130463469	Core PHP Programming	3	2003	3

Gambar 2: Contoh penggunaan RDBMS untuk menyimpan katalog

Untuk lebih jelasnya dapat dilihat pada contoh yang tergambar pada gambar 2 dan gambar 3. Contoh tersebut mengambil kasus sebuah koleksi buku elektronik (*e-book*). Gambar 2 memperlihatkan contoh penyimpanan katalog koleksi buku (tanpa data isi dari bukunya) menggunakan RDBMS. Sedangkan pada gambar 3 diperlihatkan struktur direktori dalam sistem berkas untuk menyimpan katalog koleksi buku yang sama.



Gambar 3: Contoh penggunaan sistem berkas untuk menyimpan katalog

Setiap *e-book* dalam koleksi tersebut diidentifikasi menggunakan sebuah ISBN atau *International Standard Book Number*. Pada gambar 2 dapat dilihat bahwa kolom `isbn` memang dijadikan kunci utama pada tabel yang menyimpan data buku. Sedangkan pada gambar 3 sebenarnya nama berkas yang dipakai untuk menyimpan data per buku dapat disingkat cukup hanya menggunakan ISBN-nya saja, tapi di sini ditambahkan dengan judul buku dengan tujuan untuk memudahkan pencarian.

Bagi seseorang yang memahami konsep basis data, dengan melihat secara sekilas pun sudah dapat berkesimpulan bahwa penyimpanan katalog menggunakan RDBMS akan lebih baik hasilnya daripada menggunakan sistem berkas. Dengan DBMS, terutama yang mendukung bahasa *query* semacam SQL, dapat diperoleh fasilitas pencarian data yang kinerjanya tinggi dan sangat fleksibel. Misalnya untuk mencari data buku berdasarkan tahun terbitnya, tinggal memasukkan *query* yang tepat maka akan segera diperoleh data yang diinginkan. Sementara itu dengan menggunakan sistem berkas, proses akan menjadi lebih lambat karena harus membaca satu persatu setiap berkas yang berisi data buku. Hal ini tetap berlaku sekalipun sudah ada fasilitas dari sistem operasi untuk melakukan pencarian ke dalam konten dari berkas. Selain itu dengan menggunakan sistem berkas, maka akan mengakibatkan adanya duplikasi data, yang ditandati dengan adanya dua atau lebih berkas yang sama tapi terletak pada direktori yang berbeda. Sebagai contoh, dapat dilihat pada buku “Ruby Cookbook” yang muncul hingga tiga kali. Lebih jauh lagi, dikarenakan adanya dua pengkategorian buku (berdasarkan penerbit dan pengarang) maka dapat dipastikan bahwa setiap buku akan tersimpan datanya minimal di dua berkas. Buku “Ruby Cookbook” dapat muncul tiga kali karena pengarangnya lebih dari satu. Seandainya nantinya mau ditambah kategori lainnya lagi (misalnya berdasarkan tahun terbit) maka akan muncul duplikasi lagi.

Dengan mempertimbangkan berbagai hal tersebut di atas maka perlu ditinjau kembali keuntungan-keuntungan yang dapat diperoleh dengan menggunakan sistem berkas. Selain itu juga harus dicarikan jalan keluar untuk mengatasi berbagai kelemahan yang disebutkan di atas. Berikut ini adalah daftar hal yang dapat dijadikan alasan untuk mempertimbangkan penggunaan cara terakhir tersebut:

1. Proses pembacaan data item koleksi relatif lebih cepat menggunakan sistem berkas. Bahkan untuk item koleksi yang berukuran besar,

menggunakan sistem berkas akan terasa jauh lebih cepat daripada menggunakan DBMS. Hal ini dikarenakan data item harus diekstrak dulu dari dalam DBMS baru bisa dibaca. Pada item koleksi yang besar maka proses ekstraksi data membutuhkan waktu yang tidak sebentar.

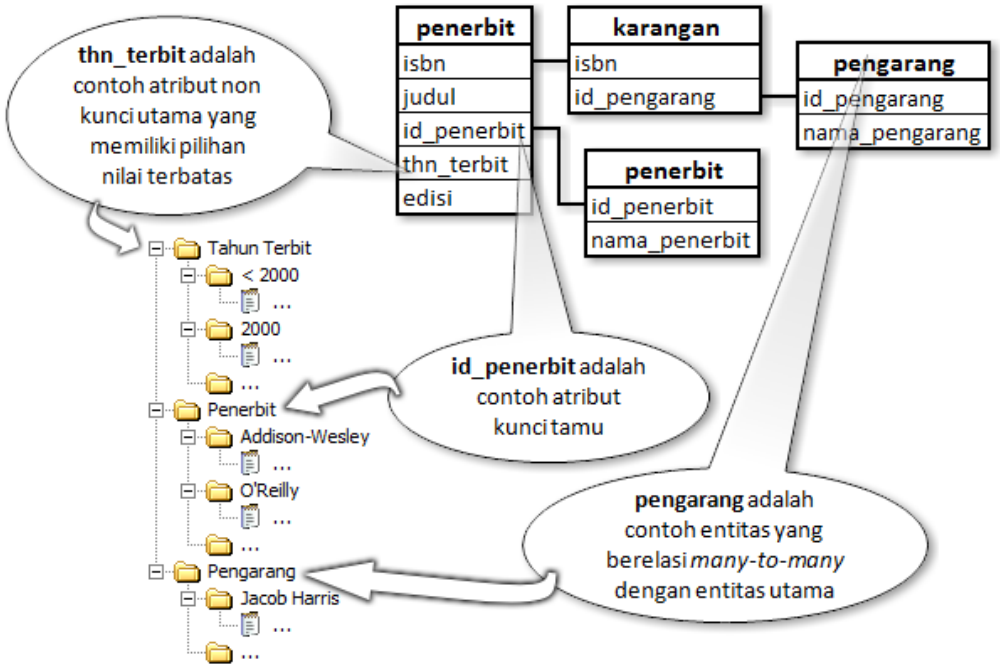
2. Penggunaan DBMS membutuhkan ruang memori yang lebih besar daripada menggunakan sistem berkas, terutama untuk data item koleksi yang berukuran besar. Sehingga untuk sistem-sistem komputer dengan memori yang pas-pasan, sistem berkas menjadi pilihan yang lebih tepat.
3. Bagi kebanyakan pengguna komputer, proses penataan koleksi jauh lebih mudah menggunakan sistem berkas daripada menggunakan DBMS. Menata berkas pada sebuah sistem berkas dapat dilakukan tanpa menggunakan *tool* atau aplikasi khusus selain peramban berkas (*file browser*) yang telah disediakan sejak awal oleh sistem operasi. Berbeda dengan kalau menggunakan DBMS, selain membutuhkan aplikasi khusus, untuk memasukkan data item koleksi ke dalam tabel database (berupa data *Blob*) memerlukan kerja ekstra.
4. Untuk penggunaan personal, sistem bekas relatif lebih tinggi tingkat ketahanannya dibandingkan dengan DBMS. Proses pem-*backup*-an data koleksi pun akan lebih mudah dilakukan bagi kebanyakan pengguna.
5. Duplikasi data yang muncul jika menggunakan sistem berkas secara penuh, dapat diatasi dengan menggunakan tautan (*link*). Untuk itu memang diperlukan sistem berkas yang mendukung penggunaan tautan, seperti NTFS.
6. Dengan menyatukan metode atau lokasi penyimpanan data item dan katalog maka akan didapatkan integritas yang tinggi. Memang akan berkurang tingkat integritasnya apabila penyimpanan data katalog dan item koleksi tersebar pada lokasi atau berkas yang berbeda, tetapi hal ini dapat dihindari dengan menggunakan fitur ADS yang tersedia pada NTFS. Dasar pemikirannya adalah bahwa data katalog tambahan dapat disimpan di dalam *stream* alternatif.
7. Jika didasarkan pada data yang berada pada katalog tambahan, memang pencarian item koleksi menjadi sulit dilakukan tanpa aplikasi khusus. Tetapi untuk pencarian yang hanya berdasarkan nama item atau kategori utama, tetap dapat dilakukan menggunakan aplikasi pencarian berkas yang telah tersedia.

Dengan melihat berbagai alasan yang tertera di atas, maka penulis menganggap bahwa manajemen koleksi dengan hanya memanfaatkan fitur pada sistem berkas patut untuk diterapkan. Selain itu perlu juga untuk dikembangkan suatu metode yang dapat memudahkan manajemen koleksi menggunakan sistem berkas secara penuh.

Penyusunan Kategori Menggunakan Struktur Direktori

Sebelumnya perlu ditekankan lagi bahwa untuk kategorisasi koleksi yang kompleks, penggunaan DBMS menjadi lebih unggul dibandingkan dengan menggunakan sistem berkas langsung. Namun untuk kategorisasi koleksi yang sederhana, dapat dibuat dengan cara memanfaatkan susunan direktori yang ada pada sistem berkas. Contoh sederhananya sudah terlihat pada gambar 3, yaitu berupa koleksi buku yang dikategorikan berdasarkan penerbit dan pengarang.

Dasar dari penyusunan kategori ini adalah dengan melihat atribut-atribut non kunci utama pada entitas utama yang nilainya merupakan anggota dari suatu himpunan nilai. Ini berarti atribut dengan pilihan nilai terbatas (*integer*, *boolean*, *enum*, dll.) atau atribut kunci tamu. Contohnya adalah atribut penerbit dan tahun terbit pada entitas buku. Selain itu juga ditambah dengan melihat entitas lain yang memiliki relasi *many-to-many* dengan entitas utama. Sebagai contoh adalah entitas pengarang yang berelasi secara *many-to-many* dengan entitas utama, yaitu buku.



Gambar 4: Contoh konversi susunan kategori dari struktur tabel ke struktur direktori

Setiap kategori yang diperoleh kemudian diwujudkan sebagai direktori tingkat satu atau subdirektori langsung dari direktori akar (*root directory*) koleksi. Untuk selanjutnya direktori yang mewakili suatu kategori tertentu dinamakan direktori kategori. Direktori kategori tertinggi (pada tingkat pertama) dinamakan sebagai direktori kategori utama. Jika suatu kategori masih dapat dibagi lagi ke dalam subkategori maka cukup dengan menambahkan direktori-direktori kategori di dalam direktori kategori yang terkait. Direktori kategori yang berada di dalam direktori kategori yang lain dinamakan direktori subkategori. Di dalam setiap direktori kategori tersebut kemudian dibuatlah direktori-direktori yang mewakili setiap nilai dalam kategori yang terkait. Jenis direktori yang terakhir disebutkan ini dinamakan sebagai direktori nilai kategori. Untuk lebih jelasnya dapat dilihat pada gambar 4.

Menghilangkan Duplikasi dengan *Hard Link* dan *Junction*

Salah satu permasalahan yang muncul saat menyimpan koleksi multi kategori menggunakan sistem berkas adalah adanya duplikasi item. Setiap item yang tersimpan di dalam berkas memiliki salinan di beberapa kategori sekaligus. Sebagai contoh, sebuah buku akan terdaftar di dalam kategori penerbit berdasarkan nama penerbitnya, juga terdaftar di bawah kategori tahun terbit berdasarkan tahun terbitnya, dan juga terdaftar di bawah kategori pengarang di bawah nama pengarangnya.

Duplikasi data akan menyulitkan dalam pemeliharaan koleksi. Perubahan di salah satu item haruslah memiliki efek terhadap setiap salinannya. Hal inilah yang kadang membuat repot si kolektor. Setiap kali melakukan perubahan di suatu item, maka diharuskan melakukan perubahan juga di setiap salinan dari item tersebut. Jika proses ini dilupakan maka dapat dipastikan bahwa integritas koleksi akan menjadi terganggu.

Penggunaan Tautan Keras

Para pengguna sistem operasi keluarga UNIX tentu sangat familiar dengan penggunaan *link* (tautan) untuk mengatasi duplikasi berkas seperti di atas. Cara peniadaan duplikasi menggunakan tautan adalah dengan membuat setiap salinan dari item koleksi sebagai tautan keras, bukan sebagai salinan berkas. Dengan menggunakan tautan, maka diperoleh paling tidak 2 (dua) keuntungan, yaitu mengakses tautan sama saja dengan mengakses berkas aslinya serta mengubah data yang ada dalam berkas akan berefek pada semua tautan yang ada. Hal ini dikarenakan tautan hanyalah sebuah referensi menuju

data yang sama dengan berkas aslinya. Sehingga kerepotan untuk mengubah data di semua salinan akan dapat dihilangkan sama sekali.

Tautan sendiri merupakan salah satu fitur dasar pada sistem berkas bawaan sistem operasi keluarga UNIX yang mulai dikenal di Windows setelah Microsoft memperkenalkan sistem berkas NTFS. Setidaknya ada 4 (empat) cara untuk membuat tautan keras di Windows, yaitu:

1. Menggunakan perintah bawaan `fsutil`.

   Untuk membuat tautas keras menggunakan perintah `fsutil`, pengguna harus memiliki *privilege* sebagai administrator. Cara penggunaannya adalah sebagai berikut:

   ```
   fsutil hardlink create path_ke_link path_ke_berkas
   ```

2. Menggunakan perintah bawaan `mklink` (Windows Vista ke atas).

   Keuntungan dari perintah `mklink` dibandingkan perintah `fsutil` adalah dapat dipakai untuk membuat macam-macam tautan yang didukung pada Windows. Penggunaannya untuk membuat tautan keras adalah sebagai berikut:

   ```
   mklink /H path_ke_link path_ke_berkas
   ```

3. Menggunakan aplikasi yang dibuat oleh pihak ketiga (*3rd party*).

   Sehubungan dengan sangat kurangnya *tool* bawaan untuk mengatur penggunaan tautan di Windows, maka terdapat beberapa pihak ketiga yang berinisiatif untuk membuat program untuk memudahkan pemakaian tautan. Contohnya adalah **NTFS Link** dan **Link Shell Extension** (LSE). Program-program ini menawarkan antarmuka grafis atau integrasi dengan Windows Explorer yang akan memudahkan penanganan tautan. NTFS Link merupakan sekelompok *shell extensions* pada Windows, yang menyediakan fungsionalitas tambahan untuk membuat dan menggunakan tautan keras dan *junction point* di sistem berkas NTFS (Elsdörfer, tanpa tahun). LSE menyediakan fasilitas pembuatan tautan keras, *junction*, *volume mountpoints*, dan tautan simbolis (secara kolektif disebut sebagai tautan) pada Windows Vista atau yang lebih baru, dan sebuah proses penggandaan (*cloning*) yang memanfaatkan tautan keras dan simbolis sekaligus (Schinagl, 2010).

4. Menulis program aplikasi sendiri, memanfaatkan Win32 API.

   Win32 *Application Programming Interface* (API) dimulai dari Windows 2000 ke atas menyertakan berbagai fungsi untuk menangani tautan keras. Untuk membuat tautan keras disediakan fungsi `CreateHardLink`. Untuk menghapus tautan keras sama dengan menghapus berkas, menggunakan fungsi `DeleteFile`. Sementara untuk mengetahui banyaknya tautan keras yang terkait dengan suatu berkas digunakan fungsi `GetFileInformationByHandle`.

Penggunaan *Junction*

Jika tautan keras digunakan untuk menghilangkan duplikasi pada berkas atau item koleksi, maka untuk menghilangkan duplikasi kategori dan nilai kategori digunakanlah *junction*. Seperti telah disebutkan di atas bahwa pembuatan kategori dan nilai kategori pada sistem berkas adalah dengan menggunakan direktori. Duplikasi yang dimaksud di sini adalah adanya direktori, baik itu direktori kategori maupun direktori nilai, yang memiliki isi yang sama. Sebagai contoh, buku-buku tentang Microsoft Access yang tersimpan di dalam direktori nilai Microsoft Access, bisa dimasukkan di bawah kategori *Office Applications*, tapi bisa juga dimasukkan di bawah kategori *Database*.

Apabila yang dibuat adalah salinan direktori jelas akan membuat duplikasi semua berkas atau item yang ada dalam direktori tersebut. Terlebih lagi, jika nantinya akan ditambahkan berkas atau item baru pada direktori tersebut maka diharuskan untuk menambah item yang sama di direktori salinannya. Untuk itu salah satu solusi untuk masalah ini adalah dengan menggunakan *junction*, sehingga menghilangkan duplikasi dan modifikasi di satu tempat otomatis akan memodifikasi di tempat lain yang terkait.

Setidaknya ada 3 (tiga) cara untuk membuat *junction*, yaitu:

1. Memanfaatkan **desktop.ini**, *shortcut* dan atribut *system* pada direktori. Metode pembuatan *junction* ini merupakan sebuah metode yang tak terdokumentasi, sehingga jarang ada yang mengetahui. Caranya adalah dengan langkah-langkah sebagai berikut:
   a. Membuat sebuah berkas teks yang terkonfigurasi khusus di dalam direktori yang akan dibuat sebagai *junction*, dengan nama **desktop.ini**, yang berisi teks berikut:

```
[.ShellClassInfo]
CLSID2={0AFACED1-E828-11D1-9187-B532F1E9575D}
```

Baris `[.ShellClassInfo]` merupakan bagian kepala yang mengindikasikan bahwa pasangan entri dan nilai yang ada di bawahnya akan meng-kustomasi kelakuan dari *folder* (direktori). Entri `CLSID2` diberi nilai `{0AFACED1-E828-11D1-9187-B532F1E9575D}`, yang menunjuk pada suatu kunci *registry* yang mengidentifikasikan suatu pintasan *folder* spesial dengan nama `Target`.

b. Membuat pintasan (*shortcut*) ke direktori yang akan dijadikan target dari *junction*, di dalam direktori yang akan dibuat sebagai *junction*, yang diberi nama `Target`.

c. Menambahkan atribut *system* pada direktori yang digunakan sebagai *junction*, dengan perintah `attrib`. Untuk menghilangkan *junction* cukup dengan meniadakan atribut *system* tersebut.

2. Menggunakan perintah bawaan `mklink` (Windows Vista ke atas), yaitu dengan sintaks sebagai berikut:

```
mklink /J path_ke_link path_ke_direktori
```

3. Menggunakan aplikasi yang dibuat oleh pihak ketiga.
   Beberapa pihak ketiga telah berinisiatif untuk membuat program untuk memudahkan pemakaian *junction*. Contohnya adalah Sysinternals dengan program **Junction**, Rekenwonder Software dengan **Junction Link Magic** (JLM), serta **Link Shell Extension** (LSE). Program-program ini menawarkan antarmuka grafis atau integrasi dengan Windows Explorer yang akan memudahkan penanganan *junction*. JLM adalah sebuah program utilitas gratis buatan Rekenwonder Software yang dapat digunakan untuk membuat *junction point* pada Windows 2000, XP, Vista, Windows Server 2008 dan Windows 7 (Rekenwonder, tanpa tahun).
4. Menulis program aplikasi sendiri, memanfaatkan Win32 API.
   Untuk membuat, menghapus dan mengecek keberadaan *junction* dalam Win32 API cukup hanya menggunakan satu fungsi saja, yaitu fungsi `DeviceIoControl`.

Pemanfaatan *Alternate Data Stream* (ADS)

ADS di sini utamanya digunakan untuk menyimpan data atau informasi tambahan untuk sebuah kategori atau nilai kategori atau item koleksi sendiri. Misalkan untuk penerbit buku, ADS dapat digunakan untuk menyimpan alamat

atau logo dari setiap penerbit. Contoh lainnya adalah untuk menyimpan gambar sampul dari setiap buku. Alih-alih menyimpannya dalam berkas yang terpisah, dengan ADS maka informasi tersebut dapat diintegrasikan dengan menulisnya ke dalam *stream* alternatif.

Setidaknya ada 3 (tiga) cara yang dapat digunakan untuk memanfaatkan fitur ADS ini. Ketiga cara itu adalah:

1. Menggunakan “fasilitas bawaan” yang disediakan oleh sistem operasi.

   Walaupun Microsoft tidak menyediakan satu pun *tool* untuk menangani ADS, namun untungnya dukungan dari sistem operasi di aras rendah sangat *solid*. Setiap *system call* yang berurusan dengan berkas, seperti membuka, membaca dan menulisi berkas, semuanya mendukung penggunaan *stream* alternatif. Sayangnya pada aras yang lebih tinggi (tingkat *shell*, baik baris perintah maupun antarmuka grafis) dukungan itu tidak ada sama sekali atau sangat terbatas. Sebagai contoh, pengguna tidak dapat hanya menyalin salah satu *stream* alternatif saja dari suatu berkas menggunakan Windows Explorer. Menyebutkan nama *stream* alternatif sebagai parameter di baris perintah juga akan menyebabkan kesalahan, seperti terlihat di gambar 5.

```
C:\>copy some.txt stream.dat:alt
The filename, directory name, or volume label syntax
        0 file(s) copied.
```

Gambar 5: Kesalahan yang muncul akibat penggunaan ADS di baris perintah

   Kabar baiknya bahwa pemakaian *Input/Output redirection* memiliki dukungan penuh terhadap ADS. Sehingga setiap program yang melakukan pembacaan masukan dari *standard input* atau melakukan penulisan ke *standard output* dapat memanfaatkan fitur ADS ini secara penuh. Pada baris perintah, hal ini berarti penggunaan operator > untuk *output redirection* dan operator < untuk *input redirection*, seperti terihat di gambar 6. Sayangnya perintah-perintah yang dapat dimanfaatkan dan sudah tersedia di Windows lebih banyak ditujukan untuk data teks, bukan data biner, sehingga sulit jika *stream* alternatif berisi data biner.

```
C:\>echo This is just some text. >stream.dat:text

C:\>more <stream.dat:text
This is just some text.

C:\>_
```

Gambar 6: Contoh penggunaan *I/O redirection* di baris perintah

Pada contoh tersebut, perintah `echo` merupakan perintah yang biasa digunakan untuk menuliskan ke *standard output*, sehingga dapat dimanfaatkan untuk menuliskan data langsung dari layar ke *stream*. Perintah lainnya seperti `type` juga dapat digunakan untuk tujuan yang sama, tapi dapat mengambil data dari dalam berkas. Sementara itu perintah `more` merupakan perintah yang dapat membaca dari *standard input*, sehingga dapat digunakan untuk membaca isi dari *stream*. Sayangnya seperti telah disebutkan di atas, perintah `more` ini hanya cocok untuk menangani data teks, bukan data biner.

2. Menggunakan program buatan pihak ketiga.
   Beberapa contoh program aplikasi untuk menangani ADS antara lain **AlternateStreamView** dari NirSoft dan **FlexHex** dari Inv Softworks LLC. AlternateStreamView merupakan sebuah utilitas kecil yang dapat digunakan untuk memindai suatu drive NTFS, dan menemukan serta mengekstrak semua *stream* alternatif yang disimpan di dalamnya (Sofer, 2009). Sedangkan FlexHex sebenarnya merupakan sebuah *hex editor* yang memiliki kemampuan lebih, salah satunya untuk melakukan operasi *stream*.
3. Menulis program sendiri, dengan memanfaatkan Win32 API.
   Fungsi Win32 API yang digunakan untuk mengecek apakah suatu drive mendukung ADS adalah `GetVolumeInformation`. Untuk melakukan enumerasi *stream* alternatif di dalam berkas digunakan fungsi `NtQueryInformationFile`. Sedangkan untuk melakukan operasi pembuatan, penghapusan, pembacaan dan penulisan *stream* sama caranya dengan melakukan operasi terhadap berkas. Fungsi-fungsi seperti `CreateFile`, `DeleteFile`, `OpenFile`, `CloseFile`, `ReadFile`, dan `WriteFile` tetap dapat digunakan untuk *stream*. Tetapi jika yang dibutuhkan hanya untuk membuat, menulis dan membaca *stream* saja, maka sudah cukup dengan membuat program untuk membaca dan menulis ke *standard input/output* saja.

KESIMPULAN

Penataan koleksi dengan hanya menggunakan struktur direktori pada sistem berkas akan mempermudah pekerjaan seorang kolektor. Hal ini karena penggunaan sistem berkas tidak membutuhkan pengetahuan teknis yang terlalu tinggi, cukup dengan menggunakan aplikasi peramban berkas (*file browser*) yang sudah tersedia dari awal pada sistem operasi yang digunakan. Proses pencarian juga cukup dengan menggunakan fasilitas pencarian berkas yang sudah tersedia. Dengan demikian kebanyakan pengguna akan dapat dengan cepat dan mudah untuk mempelajari teknik ini.

Fitur pembuatan tautan pada NTFS dapat dimanfaatkan untuk menangani kemungkinan terjadinya duplikasi item dalam koleksi. Untuk memakainya, bagi kebanyakan pengguna yang diperlukan hanyalah meng-*install* dan atau mempelajari program yang akan digunakan untuk menangani tautan tersebut. Sementara itu untuk pengguna yang tahu tentang pemrograman, dapat menulis program khusus yang disesuaikan dengan kebutuhannya sendiri. Sementara itu untuk memanfaatkan dengan maksimal fitur ADS haruslah dengan membuat program khusus yang digunakan untuk membaca isi dari ADS.

DAFTAR PUSTAKA

Boswell, William, 2003, *Inside Windows Server 2003*, Addison-Wesley.

Elsdörfer, Michael, *NTFS Link*, http://elsdoerfer.name/=ntfslink.

Inv Softworks, *NTFS Alternate Streams: What, When, and How To*, http://www.flexhex.com/docs/articles/alternate-streams.phtml.

Inv Softworks, *NTFS Hard Links, Directory Junctions, and Windows Shortcuts*, http://www.flexhex.com/docs/articles/hard-links.phtml.

Rekenwonder, *Junction Link Magic*, http://www.rekenwonder.com/linkmagic.htm.

Russinovich, Mark, 2007, *Microsoft TechNet: Windows Sysinternals Junction v1.05*, http://technet.microsoft.com/en-us/sysinternals/bb896768.aspx.

Schinagl, Hermann, 2010, *Link Shell Extension*, http://schinagl.priv.at/nt/hardlinkshellext/hardlinkshellext.html.

Shultz, Greg, 2004, *Manually creating junction points in Windows XP*, http://articles.techrepublic.com.com/5100-10878_11-5388706.html.

Sofer, Nir, 2009, *AlternateStreamView v1.12 - View/Copy/Delete NTFS Alternate Data Streams*, http://www.nirsoft.net/utils/alternate_data_streams.html.

_____, 2007, *How to create and manipulate NTFS junction points*, http://support.microsoft.com/?kbid=205524.

_____, 2010, *MSDN Library: Hard Links and Junctions*, http://msdn.microsoft.com/en-us/library/aa365006(VS.85).aspx.

_____, *Windows Symbolic and Hard Links*, http://shell-shocked.org/article.php?id=284.