

i-effect[®] *ZIP Grundlagen

Algorithmus

Der „**deflation Algorithmus**“, der von zip und gzip benutzt wird ist eine Variation des LZ77 (Lempel-Ziv 1977, siehe Referenz weiter unten). Er findet mehrfach vorkommende Zeichenfolgen im Eingabedatenstrom. Das zweite Vorkommen einer Zeichenfolge wird durch einen Zeiger ersetzt, der auf das erste Vorkommen zeigt, und zwar in der Form einer paarweisen Angabe (Entfernung, Länge). Die Entfernungen sind begrenzt auf 32K Bytes, die Längen auf 258 Bytes. Wenn eine Zeichenfolge in den vorherigen 32K Bytes nicht vorkommt wird sie als Zeichen-Bytefolge ausgegeben. (In dieser Beschreibung ist Zeichenfolge als eine Reihenfolge von Bytes zu verstehen, die nicht notwendigerweise druckbare Zeichen enthalten muss.)

Zeichen-Bytefolgen oder Treffer-Längen werden mit einem Huffman-Baum komprimiert und Treffer-Entfernungen werden mit einem anderen Baum komprimiert. Diese Baumstrukturen werden in kompakter Form am Anfang eines jeden Blocks gespeichert. Die Blöcke können jede beliebige Größe haben (außer der Beschränkung, dass die komprimierten Daten für einen Block in den zur Verfügung stehenden Hauptspeicher passen müssen). Ein Block wird beendet, wenn zip feststellt, dass es nützlich wäre, einen neuen Block mit leeren Baumstrukturen zu beginnen.

Doppelte Zeichenfolgen werden mit Hilfe eines Hash-Table gefunden. Alle Eingabe-Zeichenfolgen der Länge 3 werden in diesen Hash-Table eingefügt. Ein Hash-Index wird für die nächsten 3 Bytes berechnet. Wenn die Hash-Kette für diesen Index nicht leer ist werden alle Zeichenfolgen in dieser Kette mit der momentanen Eingabezeichenfolge verglichen und die größte Übereinstimmung wird ausgewählt.

Die Hash-Ketten werden beginnend mit der aktuellen Zeichenfolge durchsucht, um kurze Enterungen und folglich die Vorteile der Huffman-Kodierung bestmöglich ausnutzen zu können.

Die Hash-Ketten sind einfach verknüpft. Es werden keine Hash-Ketten gelöscht, sondern der Algorithmus entfernt Einträge, die zu alt sind.

Um eine problematische Situation zu vermeiden, werden sehr lange Hash-Ketten willkürlich auf eine bestimmte Länge gekürzt. Festgelegt wird diese durch die Aufrufoption der Geschwindigkeit (1 bis 9). Also findet zip nicht immer die längste mögliche Übereinstimmung, in der Regel aber eine Übereinstimmung, die lang genug ist.

Zip verschiebt ebenfalls die Auswahl an Übereinstimmungen mit einem trägen Schätzalgorithmus.

Nachdem eine Übereinstimmung der Länge N gefunden wurde, sucht zip nach einer längeren Übereinstimmung im nächsten Eingabebyte. Wird eine längere Übereinstimmung gefunden, wird die vorherige Übereinstimmung auf eine Länge von eins gekürzt. (So wird ein einzelnes Byte erzeugt) und die längere Übereinstimmung wird nachfolgend ausgegeben. Ansonsten wird die Original-Übereinstimmung beibehalten und die Suche nach der nächsten Übereinstimmung wird nur N Schritte später eingeleitet.

Die Lazy-Match Berechnung wird außerdem durch Laufzeitparameter beeinflusst. Wenn die momentan gefundene Übereinstimmung lang genug ist reduziert zip die Suche nach einer längeren Übereinstimmung, sorgt also dafür, dass die Verarbeitungsgeschwindigkeit steigt. Wenn die Kompressionsrate wichtiger ist als Geschwindigkeit versucht ZIP eine komplette zweite Suche, auch wenn die erste Übereinstimmung schon lang genug war.

Die Lazy-Match Berechnung wird nicht für schnelle Komprimierungen durchgeführt (Geschwindigkeit 1-3). Bei diesen Methoden werden neue Einträge in die Hash-Table nur dann eingefügt, wenn keine Übereinstimmung gefunden wird, oder wenn die Übereinstimmung nicht lang genug ist. Das führt zu einer verschlechterten Komprimierungsrate, spart aber Zeit, weil sowohl weniger Einfügungen, als auch weniger Suchen durchgeführt werden müssen.

nach *Jean-loup Gailly*
gzip@prep.ai.mit.edu
jloup@gzip.org

Gzip-Dateiformat

Das pkzip-Format bürdet eine Menge an Overhead in verschiedenen Headern auf, die für ein Archivprogramm sehr sinnvoll sind nicht jedoch für die Komprimierung einer einzelnen Datei. Gzip benutzt eine wesentlich einfachere Struktur: Zahlen werden im „**little endian**“ Format gespeichert und Bit 0 ist das „**least significant**“ Bit. Eine Gzip- Datei ist eine Folge komprimierter Teildateien. Jede Teildatei hat folgende Struktur:

2 Bytes	magic header 0x1f, 0x8b (\037 \213)
1 Byte	Kompressions-Methode (0..7 reserviert, 8 = deflate)
1 Byte	Flags
	Bit 0 gesetzt: Datei wahrscheinlich ascii Text
	Bit 2 gesetzt: Extra Feld vorhanden
	Bit 3 gesetzt: Original Dateiname vorhanden
	Bit 4 gesetzt: Dateikommentar vorhanden
	Bit 5 gesetzt: Datei ist verschlüsselt.
	Bit 6,7: Reserviert
4 Bytes	Dateiänderungsdatum in Unix Format

1 Byte	Extra Flags (Abhängig von Komprimierungsmethode)
1 Byte	Betriebssystem auf dem die Komprimierung durchgeführt wurde.
2 Bytes	Optional Teile Nummer (Zweiter Teil = 1)
2 Bytes	Optional Extra-Feld Länge
? Bytes	Optional Extra-Feld
? Bytes	Optional Original Dateiname, X'00' am Ende
? Bytes	Optional Dateikommentar, X'00' am Ende
12 Bytes	Optional Verschlüsselungskopf
? Bytes	Komprimierte Daten
4 Bytes	crc32
4 Bytes	Unverdichtete Eingabedateilänge modulo 2^{32}

Das Format wurde entwickelt

- (a) um Komprimierungen im „Single-pass“ durchführen zu können ohne notwendiges Rückwärtslesen und
- (b) ohne dass vorher die unkomprimierte Länge der Eingabedatei oder der zur Verfügung stehende Platz auf dem Ausgabemedium bekannt sein muss. Wenn die Eingabe nicht von einer regulären Plattendatei kommt wird das Modifikationsdatum auf das Datum des Beginns der Komprimierung gesetzt.

Die Zeitmarke ist hauptsächlich sinnvoll, wenn eine gzip Datei über ein Netzwerk transportiert wird. In diesem Fall würde es nicht genügen, nur die Besitzattribute zu bewahren. Im vorliegenden Falle werden die Attribute von gzip beim Komprimieren/Dekomprimieren bewahrt, eine Zeitmarke Null wird ignoriert.

Bit 0 in den Flags ist nur ein optionaler Anzeiger, welcher durch ein kurzes Vorlesen in der Eingabedatei gesetzt werden kann. Im Zweifelsfall wird das Bit gelöscht um binäre Daten anzuzeigen. Auf Systemen, die unterschiedliche Dateiformate für ASCII Text und Binärdaten haben, kann der Dekompressor dieses Flag benutzen, um das richtige Format auszuwählen.

Das Extra-Feld, wenn vorhanden, muss aus einem oder mehreren Unterfeldern bestehen, jedes mit dem folgenden Format:

Unterfeld ID	2 Bytes
Unterfeld Größe:	2 Bytes (little-endian Format)

Unterfeld Daten

Die Unterfeld ID kann aus zwei Zeichen mit symbolischer Bedeutung bestehen. Id's mit einem zweiten Nullbyte sind reserviert für spätere Verwendung. Die folgenden Id's sind zur Zeit definiert:

Ap (0x41, 0x70): Apollo Dateityp Information

Die Unterfeldgröße ist die Größe der Unterfelddaten und beinhaltet nicht die ID und die Größe selber.

Das Feld „**Extra Feld Größe**“ ist die gesamte Länge des Extrafeldes, inklusive der Unterfeld-ID und Unterfeldgröße.

Es muss möglich sein, das Ende der komprimierten Daten zu ermitteln und zwar unabhängig von der Komprimierungsmethode und der tatsächlichen Größe der komprimierten Daten. Wenn die komprimierten Daten nicht in eine Datei passen (besonders bei Verwendung von Disketten/Bändern) beginnt jeder Teil mit dem oben beschriebenen Kopf, aber nur der letzte Teil hat die Angaben „**crc32**“ und „**unkomprimierte Länge**“. Ein Dekompressor kann für solche in mehreren Teilen komprimierte Dateien Benutzeranforderungen für weitere benötigte Teile senden.

Es ist wünschenswert aber nicht notwendig, dass einzelne Teile einer solchen Datei unabhängig voneinander dekomprimiert werden können falls ein Teil beschädigt ist. Dies ist nur möglich, wenn der Stand der Komprimierung nicht von einem Teil in den anderen übernommen wird.

Wenn die Datei auf einem System komprimiert wird, in dessen Dateisystem mit Klein- und Großbuchstaben gearbeitet wird, muss der Originaldateiname in Kleinbuchstaben umgewandelt werden. Wenn Daten von der Standard-Eingabe komprimiert werden, gibt es keinen Originaldateinamen.

Die Kompression wird immer durchgeführt, selbst dann, wenn die komprimierte Datei etwas größer als das Original ist. Im schlimmsten Fall kann die Vergrößerung ein paar Bytes für den gzip Dateikopf, plus 5 Bytes für jeden 32K Block betragen. Dies entspricht einem Faktor von 0.015% für große Dateien.

In jedem Fall wird die aktuell benötigte Anzahl von Speicherblöcken dadurch so gut wie niemals erhöht.

Die Verschlüsselung ist die von zip 1.9. Für die Verschlüsselungsprüfung wird das letzte Byte des entschlüsselten Verschlüsselungskopfes untersucht.

Es muss Null sein. Die Zeitmarke einer verschlüsselten Datei darf auf Null gesetzt werden, um zu vermeiden, dass daraus der Aufbau des Kopfes abgeleitet werden könnte.

nach *Jean-loup Gailly*
gzip@prep.ai.mit.edu
jloup@gzip.org

Referenzen

[LZ77] Ziv J., Lempel A., „A Universal Algorithm for Sequential Data Compression“, IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.
 APPNOTE.TXT documentation file in PKZIP 1.93a. It is available by ftp in ftp.cso.uiuc.edu:/pc/exec-pc/pkz193a.exe [128.174.5.59]
 Use „unzip pkz193a.exe APPNOTE.TXT“ to extract (note: unzip, not gunzip).

ZIP-Dateiformat

Zusätzlich zum vorher beschriebenen gzip-Format unterstützt i-effect® *ZIP auch das auf vielen Plattformen vorzufindende Archivformat ZIP. Auf Windowsplattformen beispielsweise ist dies durch Produkte wie WINZIP™ bekannt geworden.

Ein ZIP-Archiv ist eine Datei, in der die komprimierten Daten von mehreren Dateien untergebracht werden. Entsprechende Funktionen in der benutzten Software erlauben das Hinzufügen, Löschen, Extrahieren oder Anzeigen der komprimierten Inhalte.

i-effect® ist mit seinem im optionalen Modul *ZIP erhältlichen Funktionsumfang voll kompatibel zu dem allgemeinen ZIP Format und erlaubt so den Austausch von Archivdateien mit anderen Plattformen (z.B. WINZIP™).

Innerhalb einer Archivdatei (.ZIP-Datei) können unterschiedliche Objektarten komprimiert sein. Zur Zeit werden unterstützt: Physische Dateien (PF-DTA), Quelldateien (SRC-PF), Sicherungsdateien (SAVF), Dateien aus dem IFS-Dateisystem und Spooldateien.

Werden diese Objektarten mit i-effect® *ZIP auf einer IBM Power Systems wiederhergestellt, dann erhalten Sie die tatsächlichen Ursprungsobjektarten. Spooldateien werden als tatsächliche Spoleinträge in Ausgabewarteschlangen angelegt; Komprimierte Sicherungsdateien werden zu richtigen Sicherungsdateien u.s.w.

EDIFACT-Grundlagen

Das EDIFACT-Regelwerk liefert eine umfangreiche Grundlage an Nachrichtendefinitionen für die verschiedensten Handelsdokumente. Hinter dem Akronym EDIFACT verbirgt sich der Anspruch, Austauschvereinbarungen für Dokumente aus den Bereichen Behörden, Handel und Transport unter Verwendung einheitlicher Richtlinien zur Verfügung zu stellen:

Electronic
 Data
 Interchange
 For
 Administration
 Commerce and
 Transport

Maßgeblicher Träger der Normierungsarbeit sind die United Nations. Dokumente, die aus der Organisationsstruktur der United Nations entstammen, werden auch UN/EDIFACT benannt. Neben diesen Definitionen existieren auch Dokumentvorgaben aus anderen Institutionen (z. B. Odette der Automobilindustrie) die Ihre Dokumente nach gleichen Kodierungsregeln aufbereiten, aber in Ihrer gesamten Darstellung nicht den UN-Dokumenten entsprechen.

i-effect® ist aufgrund seiner flexiblen Tabellenlogik in der Lage, sämtliche Dokumente die nach EDIFACT-Syntax aufbereitet sind, zu konvertieren, sofern eine passende Umsetzungs-vorschrift hinterlegt ist. Die Dokumente müssen den Syntaxregeln laut ISO 9735 entsprechen. Diese Fassung gibt es auch als deutsche Übersetzung und ist beim DIN unter der Veröffentlichung 16557 zu beziehen.

Weitere grundlegende Informationen zu EDIFACT allgemein und deren praktische Anwendungen erhalten Sie über Ihr EDI-Beratungshaus oder über die Organisationen, die sich aktiv an der Normierung von EDIFACT-Dokumenten beteiligen, z.B.:

DIN	EDIFACT allgemein
DEDIG	EDIFACT praxisbezogen
CCG / GS1	Konsumgüterwirtschaft
BSL	Transport und Logistik
VDA	Automobilproduktion

und vielen weiteren Organisationen und Verbänden aus dem Bereich Textil, Möbel, Banken etc.

EDIFACT-Regeln zur Konvertierung

Im Folgenden werden auf allgemeine formaltechnische EDIFACT-Regeln eingegangen, die für eine Konvertierung relevant sind.

Die kleinste Informationseinheit im EDIFACT ist das Datenelement, bzw. das Datenunterelement, sofern es sich um eine Zusammensetzung von mehreren Datenelementen handelt. Somit stellen die Datenelemente die Bausteine dar, aus denen Informationen aufgebaut werden, entweder als einfaches Datenelement oder als zusammengesetztes Datenelement, auch Composit genannt. Jedes Datenelement hat gewisse Eigenschaften wie z.B.:

Länge	z.B. 4
Längenart	Fix, Variabel
Datentyp	Numerisch, Alphanumerisch, Alphabetisch
Semantik	Inhaltliche Beschreibung

Bei der Semantik, also der Beschreibung, was sich inhaltlich in diesem Feld befindet, weicht EDIFACT von üblichen Codierungsformen ab. Die Beschreibung von Datenbankfeldern oder Satzartfeldern gehen davon aus, dass in diesem Feld auch ein eindeutiger Wert eingetragen ist. Für eine Telefonnummer wird ein Feld reserviert und für eine Faxnummer ein weiteres. In EDIFACT erfolgt die eindeutige Beschreibung erst durch die Interpretation eines weiteren Datenfeld, einem sogenannten Bezeichner (Qualifier). Das Beispiel mit den Telefon- und Faxnummern würde in EDIFACT folgendermaßen dargestellt:

Kontaktnummer: 0226123990	Qualifier: TE
Kontaktnummer: 0226191845	Qualifier: FX

Die Kontaktnummer stellt ein Datenfeld für beliebige Verbindungsarten zu dem beschriebenen Kontaktpartner dar, der zu nutzende Dienst (Telefon, Fax, Mailbox, Telex, BTX, Cityruf, GSM-Netz) wird erst durch den Qualifier zugewiesen. Diese Darstellungsform wurde genutzt, um bei neuen Informationsarten lediglich die Codeliste (= Übersicht aller Qualifier des Datenelementes) zu erweitern und die bestehende Struktur der Nachricht beizubehalten.

Die Zusammenstellung mehrerer Datenelemente und Datenelementgruppen ergibt ein Segment. Jedes Segment erhält eine eindeutige, dreistellige Bezeichnung (Segment-Tag). Es beschreibt eine bestimmte Teilinformation eines Dokumentes, z.B.

NAD	Namen und Adressen
DTM	Datumsangaben
MOA	Geldbeträge
TDT	Transportmittel

Mehrere Segmente können zu Segmentgruppen zusammengefasst werden, sofern sie inhaltlich zusammengehören. Die Beschreibung einer Artikelposition in einer Bestellung erfordert in einigen Fällen eine ausführliche Angabe von Zusatzinformationen, die dann in einer Segmentgruppe zusammengefasst werden:

Segmentgruppe LIN-IMD-PIA-API...

LIN	Artikelposition
IMD	weitere Artikelbeschreibung
PIA	Ersatz und Referenzangaben
API	zusätzliche Preisangaben

Die Zusammenstellung aller Segmente und Segmentgruppen ergibt dann die eigentliche Nachrichtendefinition:

INVOIC	Rechnung
ORDERS	Auftrag
IFTMIN	Speditionsauftrag

Derzeit sind weit über 200 Dokumente in EDIFACT definiert und bieten die Grundlage für einen intensiven beleglosen Austausch zwischen Unternehmen, Behörden und anderen am Geschäftsvorgang beteiligten Partnern.

Die EDIFACT -Nachrichten unterliegen einer definierten Struktur, wobei zwischen Service- und Nutzdatensegmenten zu unterscheiden ist.

Bei EDIFACT wird davon ausgegangen, dass die erzeugten Nachrichten per Datenfernverarbeitung von Partner A zu Partner B übertragen werden. Daher wurden spezielle Segmente für die Übertragung (Transmission) einer physischen Datei definiert. Mit den Segmenten

UNB
...
UNZ

wird eine physische Datei umklammert und beinhaltet Mitteilungen für einen Adressaten, der im UNB-Segment angegeben wird. Innerhalb dieser Übertragungsdatei können verschiedene Dokumententypen auftreten, zum Beispiel Lieferscheine und Rechnungen. Die Dokumententypen werden mit den Segmenten

UNG
....
UNE

voneinander abgetrennt. Die eigentliche Nachricht steht zwischen den Segmenten

UNH
....
UNT

Die Gesamtstruktur einer Übertragungsdatei kann dann wie folgt aussehen:

UNB	Beginn der Übertragungsdatei
UNG	Lieferscheine
UNH	Lieferschein A
...	
UNT	Ende Lieferschein A
UNH	Lieferschein B

...	
UNT	Ende Lieferschein B
UNE	Ende Lieferscheine
UNG	Rechnungen
UNH	
...	
UNT	Ende Rechnung A
UNH	Rechnung B
...	
UNT	Ende Rechnung B
UNE	Ende Rechnungen
UNZ	Ende der Übertragungsdatei

Für jede Nachricht, die der Konverter bearbeiten soll, muss eine eindeutige Umsetzungsanweisung hinterlegt sein, die eine Beziehung (Mapping) zwischen der Inhouse-Seite und der EDIFACT-Seite herstellt. Diese Zuweisung erfolgt für jedes einzelne Datenelement innerhalb der Segmente und für jeden Segmenttyp innerhalb einer Gruppe, sofern dieses Segment an unterschiedlichen Positionen unterschiedlich zugeordnet wird. Die einzelnen Tabellenoptionen werden im folgenden Kapitel beschrieben.

