Лекции №14
АРХИВАЦИЯ ДАННЫХ

ОБЩИЕ ПОНЯТИЯ

Одним из наиболее широко распространенных видов сервисных программ являются программы, предназначенные для архивации, упаковки файлов путем сжатия хранимой в них информации.

> **Сжатие информации** — это процесс преобразования информации, хранящейся в файле, к виду, при котором уменьшается избыточность в ее представлении и соответственно требуется меньший объем памяти для хранения.

Сжатие информации в файлах производится за счет устранения избыточности различными способами, например за счет упрощения кодов, исключения из них постоянных битов или представления повторяющихся символов или повторяющейся последовательности символов в виде коэффициента повторения и соответствующих символов.

Целью процесса сжатия, как правило, есть получение более компактного выходного потока информационных единиц из некоторого изначально некомпактного входного потока при помощи некоторого их преобразования.

Основными техническими характеристиками процессов сжатия и результатов их работы являются:

* степень сжатия (compress rating) или отношение (ratio) объемов исходного и результирующего потоков;
* скорость сжатия - время, затрачиваемое на сжатие некоторого объема информации входного потока, до получения из него эквивалентного выходного потока;
* качество сжатия - величина, показывающая на сколько сильно упакован выходной поток, при помощи применения к нему повторного сжатия по этому же или иному алгоритму.

Все способы сжатия можно разделить на две категории: обратимое и необратимое сжатие.

Под необратимым сжатием подразумевают такое преобразование входного потока данных, при котором выходной поток, основанный на определенном формате информации, представляет, с некоторой точки зрения, достаточно похожий по внешним характеристикам на входной поток объект, однако отличается от него объемом. Данный подход реализован в популярных форматах представления видео и фото информации, известных как JPEG и JFIF алгоритмы и JPG и JIF форматы файлов. Необратимое сжатие невозможно применять в областях, в которых необходимо иметь точное соответствие информационной структуры входного и выходного потоков.

Обратимое сжатие всегда приводит к снижению объема выходного потока информации без изменения его информативности, т.е. - без потери информационной структуры. Более того, из выходного потока, при помощи восстанавливающего или декомпрессирующего алгоритма, можно получить входной, а процесс восстановления называется декомпрессией или распаковкой и только после процесса распаковки данные пригодны для обработки в соответствии с их внутренним форматом

Применяются различные алгоритмы подобного сжатия информации.

Все алгоритмы сжатия данных качественно делятся на

1) алгоритмы сжатия без потерь, при использовании которых данные на приемной восстанавливаются без малейших изменений, и
2) алгоритмы сжатия с потерями, которые удаляют из потока данных информацию, незначительно влияющую на суть данных, либо вообще невоспринимаемую человеком (такие алгоритмы сейчас разработаны только для аудио- и видео- изображений).

АЛГОРИТМЫ АРХИВАЦИИ

1. Сжатие способом кодирования серий (RLE)

Наиболее известный простой подход и алгоритм сжатия информации обратимым путем - это кодирование серий последовательностей (Run Length Encoding - RLE).

Суть методов данного подхода состоит в замене цепочек или серий повторяющихся байтов или их последовательностей на один кодирующий байт и счетчик числа их повторений.

Например:

44 44 44 11 11 11 11 11 01 33 FF 22 22 - исходная последовательность

03 44 04 11 00 03 01 03 FF 02 22 - сжатая последовательность

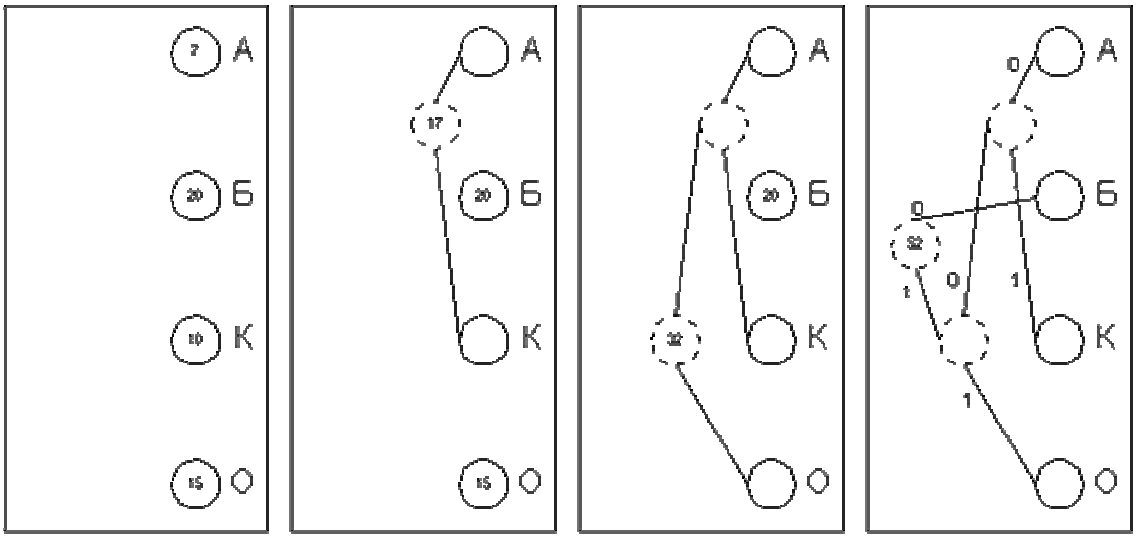
Первый байт указывает сколько раз нужно повторить следующий байт. Если первый байт равен 00, то затем идет счетчик, показывающий, сколько за ним следует неповторяющихся данных.

Описанный метод является простым и очень эффективным способом сжатия файлов. Однако он не обеспечивает большой экономии объема, если обрабатываемый текст содержит небольшое количество последовательностей повторяющихся символов. Недостатком метода RLE является достаточно низкая степень сжатия.

2. Алгоритм Хаффмана

Это так называемый оптимальный префиксный код или кодирование символами переменной длины. Код переменной длины позволяет записывать наиболее часто встречающиеся символы и фразы всего лишь несколькими битами, в то время как редкие символы и фразы будут записаны более длинными битовыми строками.

Алгоритм основан на том факте, что некоторые символы из стандартного 256-символьного набора в произвольном тексте могут встречаться чаще среднего периода повтора, а другие, соответственно, – реже. Сжимая файл по алгоритму Хаффмана первое что мы должны сделать - это необходимо прочитать файл полностью и подсчитать сколько раз встречается каждый символ из расширенного набора ASCII. Выпишем их вертикально в ряд в виде ячеек будущего графа по правому краю листа (рис. 1а).



а б в г

Рис.1

Выберем два символа с наименьшим количеством повторений в тексте (если три или большее число символов имеют одинаковые значения, выбираем любые два из них). Проведем от них линии влево к новой вершине графа и запишем в нее значение, равное сумме частот повторения каждого из объединяемых символов (рис.1б). Отныне не будем

принимать во внимание при поиске наименьших частот повторения два объединенных узла (для этого сотрем числа в этих двух вершинах), но будем рассматривать новую вершину как полноценную ячейку с частотой появления, равной сумме частот появления двух соединившихся вершин. Будем повторять операцию объединения вершин до тех пор, пока не придем к одной вершине с числом (рис.1в и 1г). Для проверки: очевидно, что в ней будет записана длина кодируемого файла. Теперь расставим на двух ребрах графа, исходящих из каждой вершины, биты 0 и 1 произвольно – например, на каждом верхнем ребре 0, а на каждом нижнем – 1. Теперь для определения кода каждой конкретной буквы необходимо просто пройти от вершины дерева до нее, выписывая нули и единицы по маршруту следования.

Код Хаффмана является префиксным, то есть код никакого символа не является началом кода какого-либо другого символа. А из этого следует, что код Хаффмана однозначно восстановим получателем, даже если не сообщается длина кода каждого переданного символа. Получателю пересылают только дерево Хаффмана в компактном виде, а затем входная последовательность кодов символов декодируется им самостоятельно без какой-либо дополнительной информации.

3. Арифметическое кодирование

Совершенно иное решение предлагает т.н. арифметическое кодирование. Арифметическое кодирование является методом, позволяющим упаковывать символы входного алфавита без потерь при условии, что известно распределение частот этих символов и является наиболее оптимальным, т.к. достигается теоретическая граница степени сжатия.

Предполагаемая требуемая последовательность символов, при сжатии методом арифметического кодирования рассматривается как некоторая двоичная дробь из интервала [0, 1). Результат сжатия представляется как последовательность двоичных цифр из записи этой дроби.

Идея метода состоит в следующем: исходный текст рассматривается как запись этой дроби, где каждый входной символ является "цифрой" с весом, пропорциональным вероятности его появления. Этим объясняется интервал, соответствующий минимальной и максимальной вероятностям появления символа в потоке.

Пример.

Пусть алфавит состоит из двух символов: a и b с вероятностями соответственно 0,75 и 0,25.

Рассмотрим наш интервал вероятностей [0, 1). Разобьем его на части, длина которых пропорциональна вероятностям символов. В нашем случае это [0; 0,75) и [0,75; 1). Суть алгоритма в следующем: каждому слову во входном алфавите соответствует некоторый подинтервал из интервала [0, 1) а пустому слову соответствует весь интервал [0, 1). После получения каждого следующего символа интервал уменьшается с выбором той его части, которая соответствует новому символу. Кодом цепочки является интервал, выделенный после обработки всех ее символов, точнее, двоичная запись любой точки из этого интервала, а длина полученного интервала пропорциональна вероятности появления кодируемой цепочки.

Применим данный алгоритм для цепочки "aaba":

Шаг	Цепочка	Интервал
0	нет	[0;1)
1	a	[0;0,75)
2	aa	0.75*0.75=0.5625 [0;0,5625)
3	aab	0.5625*0.25=0,140625-0,5625=-0,421875 [0,421875;0,5625)
4	aaba	0,140625*0.75=0,10546875

Границы интервала вычисляются так берется расстояние внутри интервала (0,5625-0,421875=0,140625), делится на частоты [0; 0,10546875) и [0,10546875; 1) и находятся новые границы [0,421875; 0,52734375) и [0,52734375; 0,5625).

В качестве кода можно взять любое число из интервала, полученного на шаге 4, например, 0,43.

Алгоритм декодирования работает аналогично кодирующему. На входе 0,43 и идет разбиение интервала.

Продолжая этот процесс, мы однозначно декодируем все четыре символа. Для того, чтобы декодирующий алгоритм мог определить конец цепочки, мы можем либо передавать ее длину отдельно, либо добавить к алфавиту дополнительный уникальный символ - "конец цепочки".

4. Алгоритм Лемпела-Зива-Велча (Lempel-Ziv-Welch - LZW)

LZW - История этого алгоритма начинается с опубликования в мае 1977 г. Дж. Зивом (J. Ziv) и А. Лемпелем (A. Lempel) статьи в журнале "Информационные теории" под названием "IEEE Trans". В последствии этот алгоритм был доработан Терри А. Велчем (Terry A. Welch) и в окончательном варианте отражен в статье "IEEE Computer" в июне 1984. В этой статье описывались подробности алгоритма и некоторые общие проблемы, с которыми можно столкнуться при его реализации. Позже этот алгоритм получил название - LZW (Lempel - Ziv - Welch).

Алгоритм LZW представляет собой алгоритм кодирования последовательностей неодинаковых символов. Возьмем для примера строку "Объект TSortedCollection порожден от TCollection.". Анализируя эту строку мы можем видеть, что слово "Collection" повторяется дважды. В этом слове 10 символов - 80 бит. И если мы сможем заменить это слово в выходном файле, во втором его включении, на ссылку на первое включение, то получим сжатие информации. Если рассматривать входной блок информации размером не более 64К и ограничится длинной кодируемой строки в 256 символов, то учитывая байт "флаг" получим, что строка из 80 бит заменяется 8+16+8 = 32 бита. Алгоритм LZW как-бы "обучается" в процессе сжатия файла. Если существуют повторяющиеся строки в файле, то они будут закодированы в таблицу. Очевидным преимуществом алгоритма является то, что нет необходимости включать таблицу кодировки в сжатый файл. Другой важной особенностью является то, что сжатие по алгоритму LZW является однопроходной операцией в противоположность алгоритму Хаффмана (Huffman), которому требуется два прохода.

Данный алгоритм отличают высокая скорость работы, как при упаковке, так и при распаковке, достаточно скромные требования к памяти и простая аппаратная реализация.

Недостаток - низкая степень сжатия по сравнению со схемой двухступенчатого кодирования.

Пример.

Предположим, что у нас имеется словарь, хранящий строки текста и содержащий порядка от 2-х до 8-ми тысяч пронумерованных гнезд. Запишем в первые 256 гнезд строки, состоящие из одного символа, номер которого равен номеру гнезда.

Алгоритм просматривает входной поток, разбивая его на подстроки и добавляя новые гнезда в конец словаря. Прочитаем несколько символов в строку s и найдем в словаре строку t - самый длинный префикс s.

Пусть он найден в гнезде с номером n. Выведем число n в выходной поток, переместим указатель входного потока на length(t) символов вперед и добавим в словарь новое гнездо, содержащее строку t+c, где c - очередной символ на входе (сразу после t). Алгоритм преобразует поток символов на входе в поток индексов ячеек словаря на выходе.

При практической реализации этого алгоритма следует учесть, что любое гнездо словаря, кроме самых первых, содержащих одно-символьные цепочки, хранит копию некоторого другого гнезда, к которой в конец приписан один символ. Вследствие этого можно обойтись простой списочной структурой с одной связью.
Пример: ABCABCABCABCABCABC - 1 2 3 4 6 5 7 7 7

1	A
2	B
3	C
4	AB
5	BC
6	CA
7	ABC
8	CAB
9	BCA

5. Двухступенчатое кодирование. Алгоритм Лемпела-Зива

Гораздо большей степени сжатия можно добиться при выделении из входного потока повторяющихся цепочек - блоков, и кодирования ссылок на эти цепочки с построением хеш таблиц от первого до n-го уровня.

Метод, о котором и пойдет речь, принадлежит Лемпелю и Зиву и обычно называется LZ-compression.

Суть его состоит в следующем: упаковщик постоянно хранит некоторое количество последних обработанных символов в буфере. По мере обработки входного потока вновь поступившие символы попадают в конец буфера, сдвигая предшествующие символы и вытесняя самые старые.

Размеры этого буфера, называемого также скользящим словарем (sliding dictionary), варьируются в разных реализациях кодирующих систем.

Экспериментальным путем установлено, что программа LHarc использует 4-килобайтный буфер, LHA и PKZIP - 8-ми, а ARJ - 16-килобайтный.

Затем, после построения хеш таблиц алгоритм выделяет (путем поиска в словаре) самую длинную начальную подстроку входного потока, совпадающую с одной из подстрок в словаре, и выдает на выход пару (length, distance), где length - длина найденной в словаре подстроки, а distance - расстояние от нее до входной подстроки (то есть фактически индекс подстроки в буфере, вычтенный из его размера).

В случае, если такая подстрока не найдена, в выходной поток просто копируется очередной символ входного потока.

В первоначальной версии алгоритма предлагалось использовать простейший поиск по всему словарю. Однако, в дальнейшем, было предложено использовать двоичное дерево и хеширование для быстрого поиска в словаре, что позволило на порядок поднять скорость работы алгоритма.

Таким образом, алгоритм Лемпела-Зива преобразует один поток исходных символов в два параллельных потока длин и индексов в таблице (length + distance).

Очевидно, что эти потоки являются потоками символов с двумя новыми алфавитами, и к ним можно применить один из упоминавшихся выше методов (RLE, кодирование Хаффмана или арифметическое кодирование).

Так мы приходим к схеме двухступенчатого кодирования - наиболее эффективной из практически используемых в настоящее время. При реализации этого метода

необходимо добиться согласованного вывода обоих потоков в один файл. Эта проблема обычно решается путем поочередной записи кодов символов из обоих потоков.
Пример:
Первая ступень
abcabcabcabcabc - 1 a 1 b 1 c 3 3 6 3 9 3 12 3
Вторая ступень - исключение большой группы повторяющихся последовательностей
1 a 1 b 1 c 12 3
и сжатие RLE, кодирование Хаффмана, арифметическое кодирование

РАЗНОВИДНОСТИ АРХИВАТОРОВ

1. Архиваторы дисков – Double Space, Stacker
2. Архиваторы exe и com файлов : PKLITE, DIET, LZEXE, EXEPACK, AINEXE и др. Упакованные exe и com файлы имеют те же расширения (exe и com) и сохраняют свою способность к исполнению, в отличие от архивных файлов. Они занимают значительно меньше места на диске, чем неупакованные файлы.
3. Архиваторы файлов
4. Специализированные архиваторы (необратимые), для специальных типов информации, например, графические, аудио, видео файлы.

ОБЩИЕ СВЕДЕНИЯ ОБ АРХИВАЦИИ ФАЙЛОВ

Понятие процесса архивации файлов

Сжиматься могут как один, так и несколько файлов, которые в сжатом виде помещаются в так называемый архивный файл или архив.

> **Архивный файл** — это специальным образом организованный файл, содержащий в себе один или несколько файлов в сжатом или несжатом виде и служебную информацию об именах *файлов, дате и времени их создания или модификации, размерах и т.п.*

Целью упаковки файлов обычно являются обеспечение более компактного размещения информации на диске, сокращение времени и соответственно стоимости передачи информации по каналам связи в компьютерных сетях. Кроме того, упаковка в один архивный файл группы файлов существенно упрощает их перенос с одного компьютера на другой, сокращает время копирования файлов на диски, позволяет защитить информацию от несанкционированного доступа, способствует защите от заражения компьютерными вирусами.

Степень сжатия зависит от используемой программы, метода сжатия и типа исходного файла. Наиболее хорошо сжимаются файлы графических образов, текстовые файлы и файлы данных, для которых степень сжатия может достигать 5 - 40%, меньше сжимаются файлы исполняемых программ и загрузочных модулей — 60 - 90%. Почти не сжимаются архивные файлы. Программы для архивации отличаются используемыми методами сжатия, что соответственно влияет на степень сжатия.

> **Архивация (упаковка)** — помещение (загрузка) исходных файлов в архивный файл в сжатом или несжатом виде.
> **Разархивация (распаковка)** — процесс восстановления файлов из архива точно в таком виде, какой они имели до загрузки в архив. При распаковке файлы извлекаются из архива и помещаются на диск или в оперативную память.

Программы, осуществляющие упаковку и распаковку файлов, называются *программами-архиваторами.*

Большие по объему архивные файлы могут быть размещены на нескольких дисках (томах). Такие архивы называются *многотомными.* Том — это составная часть многотомного архива. Создавая архив из нескольких частей, можно записать его части на несколько дискет.

Основные виды программ-архиваторов

В настоящее время применяется несколько десятков программ-архиваторов, которые отличаются перечнем функций и параметрами работы, однако лучшие из них имеют примерно одинаковые характеристики. Из числа наиболее популярных программ можно выделить:

ARJ, PKPAK, LHA, ICE, HYPER, ZIP, PAK, ZOO, EXPAND, разработанные за рубежом, а также AIN и RAR, разработанные в России.

Обычно упаковка и распаковка файлов выполняются одной и той же программой, но в некоторых случаях это осуществляется разными программами, например, программа PKZIP производит упаковку файлов, а PKUNZIP — распаковку файлов.

Перечень программ сжатия с кратким указанием алгоритмов их работы.

PKZIP 1.10:

Метод Shrinked -- модифицированный алгоритм LZW с частичной очисткой словаря и переменной длиной кода.

Метод Imploded -- модифицированный алгоритм Лемпела-Зива и статическое кодирование Хаффмана.

LHArc:

Алгоритм Лемпела-Зива и динамическое кодирование Хаффмана.

LHA:

Алгоритм Лемпела-Зива и статическое кодирование Хаффмана.

ARJ:

Алгоритм Лемпела-Зива и оригинальный метод кодирования

Программы-архиваторы позволяют создавать и такие архивы, для извлечения из которых содержащихся в них файлов не требуются какие-либо программы, так как сами архивные файлы могут содержать программу распаковки. Такие архивные файлы называются самораспаковывающимися.

> **Самораспаковывающийся архивный файл** — это загрузочный, исполняемый модуль, который способен к самостоятельной разархивации находящихся в нем файлов без использования программы-архиватора.

Самораспаковывающийся архив получил название SFX-архив (SelF-eXtracting). Архивы такого типа в обычно создаются в форме .EXE-файла.

К **основным функциям** архиваторов относятся:

- архивация указанных файлов или всего текущего каталога;
- извлечение отдельных или всех файлов из архива в текущий каталог (или в указанный каталог);
- просмотр содержимого архивного файла (состав, свойства упакованных файлов, их каталожная структура и т.д.);
- проверка целостности архивов;
- восстановление поврежденных архивов;
- ведение многотомных архивов;
- вывод файлов из архива на экран или на печать.