УДК 681.326

А.М. Шашлов

МЕТОД ВОССТАНОВЛЕНИЯ ФРАГМЕНТИРОВАННЫХ ФАЙЛОВ ПРИ УТРАТЕ СВЕДЕНИЙ О ФРАГМЕНТАЦИИ

Рассмотрены ограничения известных алгоритмов восстановления данных при утрате сведений о фрагментации файлов, состоящих более чем из одного фрагмента. Приведена классификация повреждений файловых систем. Предложен новый алгоритм, обеспечивающий для ряда форматов файлов восстановление информации о фрагментации файла на основе данных, содержащихся в файле.

E-Mail: anthon2k@mail.ru

Ключевые слова: *файловые системы, восстановление данных, логические повреждения.*

Методы восстановления данных при логических повреждениях файловых систем в целом являются исследованными [1], а возможность восстановления данных зависит от вида и объема повреждения в каждом конкретном случае.

Классификация повреждений файловых систем. Рассмотрим обобщенную классификацию повреждений файловых систем [1–3] по виду повреждений.

Повреждения файловой системы:

- повреждения области данных – поврежденные файлы и структура поврежденных каталогов не могут быть восстановлены;
- утрата сведений о фрагментации – фрагментированные файлы не могут быть восстановлены, если нет резервных копий данных о фрагментации (структуры, содержащие сведения о фрагментации имеют резервную копию, фрагменты могут быть найдены в файле «подкачки», файле «спящего режима»;
- повреждения корневого каталога – корневой каталог может быть восстановлен путем поиска каталогов и файлов, которые не имеют внешних ссылок;
- повреждения загрузочного сектора – загрузочный сектор может быть восстановлен на основании резервной копии, а в случае ее отсутствия – на основании сведений системы разделов или сохранившихся элементов логики файловой системы и спецификации файловой системы.

Приведенная классификация относится к файловым системам FAT16 / FAT32 / NTFS4 / NTFS5 / NTFS6 / EXT3 / EXT4. Повреждения информации средств контроля доступа указанных файловых

систем, а также файловых систем с криптографической защитой информации выходят за пределы исследования настоящей статьи.

Следует отметить, что в случае, если утрачены сведения о последовательности кластеров, которые составляют фрагментированные файлы, данные, содержавшиеся в этих файлах, не могут быть восстановлены существующими средствами автоматизированного восстановления данных. Невозможность восстановления таких файлов обусловлена необходимостью решения многокритериальной и многофакторной задачи по подбору последовательности кластеров диска, составляющих такой файл.

Оценка возможности перебора фрагментов файла. Для успешного восстановления сведений о фрагментации файла необходимо обеспечить возможность проверки тождественности найденного линейно упорядоченного множества Q_i кластеров исходному линейно упорядоченному множеству K кластеров, которые составляли восстанавливаемый файл.

Проверка тождественности найденного упорядоченного множества Q_i кластеров исходному линейно упорядоченному множеству K кластеров, которые составляли восстанавливаемый файл, может быть осуществлена для тех форматов файлов, которые предусматривают контрольную сумму, имитовставку или электронную подпись для обеспечения проверки целостности данных, содержащихся в файле.

Наиболее простым решением, позволяющим обеспечить поиск искомого линейно упорядоченного множества K кластеров, которые составляли восстанавливаемый файл, является полный перебор кластеров в адресном пространстве накопителя с проверкой для каждой возможной комбинации целостности найденной цепочки кластеров до подтверждения тождественности найденного линейно упорядоченного множества Q_i кластеров исходному линейно упорядоченному множеству K кластеров, которые составляли восстанавливаемый файл. Оценим число комбинаций и затраты времени для такого перебора.

В случае, если информация о размере восстанавливаемого файла была утрачена и не может быть определена на основе сведений, содержащихся в структуре файла, число комбинаций, которые подлежат перебору для подбора цепочки кластеров файла, будет определяться числом перестановок кластеров области данных для цепочек разной длины:

$$k = n_p^{'}!\, n_p^{'},$$

где $n_p^{'}$ – размер области данных накопителя в кластерах, определяется из известного размера области данных накопителя в секторах и из-

вестного размера кластера в секторах, $n_p^{'} = n_p / c$ (c – размер кластера (число секторов в кластере)). Таким образом, поиск числа комбинаций, которые подлежат перебору, будет сводиться к нахождению числа перестановок:

$$k = n_{\text{р}}^{'}! n_{\text{р}}^{'} = n_p / c! \; n_p / c.$$

Произведем оценку времени T, необходимоего для осуществления такого перебора даже для небольшого логического раздела накопителя размером 20 972 825 КБ (около 20 ГБ) с файловой системой NTFS. Размер сектора накопителя равен 512 байт, а размер кластера – 4 096 байт (8 секторов). Системные области такого раздела занимают 102 317 КБ, а область данных состоит из 20 870 512 секторов. Таким образом, имеем следующие условия: число секторов области данных раздела: $n_{\text{р}} = 20\,870\,512$; $g = 512$ – размер сектора (число байт в секторе); $c = 8$ – размер кластера (число секторов в кластере); L – множество секторов раздела

Рассчитаем число комбинаций, которые подлежат перебору:

$$k = n_p^{'}! n_p^{'} = \left(\frac{n_p}{c}\right)! \frac{n_p}{c} = \left(\frac{20870512}{8}\right)! \frac{20870512}{8} = 2608814! \cdot 2608814.$$

Время, которое тратится на проверку каждой комбинации, зависит от формата файла и построения оценочной функции, которая осуществляет вычисление и сверку контрольной суммы/хэш-функции/имитовставки или проверяет электронную подпись.

Вместе с тем очевидно, что даже в случае перебора нескольких сотен комбинаций в 1 с такой метод восстановления данных фрагментированных файлов не будет приемлемым для промышленной эксплуатации.

Если информация о размере восстанавливаемого файла не была утрачена или может быть определена на основе сведений, содержащихся в структуре файла, число комбинаций, которые подлежат перебору для подбора цепочки кластеров файла, будет определяться числом размещений из числа кластеров области данных по числу кластеров в восстанавливаемом файле:

$$k = A_{n_p'}^{b'} = \frac{n_p'!}{\left(n_{p'} - b'\right)!},$$

где b' – размер файла в кластерах, определяется из известного размера файла в секторах и известного размера кластера диска в секто-

рах, $b' = b/c$; n'_p – размер области данных накопителя в кластерах, определяется из известного размера области данных накопителя в секторах и известного размера кластера в секторах, $n'_p = n_p/c$.

Таким образом, поиск числа комбинаций, которые подлежат перебору, сводится к нахождению числа размещений:

$$k = A^{b'}_{n_p'} = \frac{n_p'!}{(n_p' - b')!} = \frac{(\frac{n_p}{c})!}{\left(\frac{n_p}{c} - \frac{b}{c}\right)!}.$$

Произведем оценку времени *T*, необходимого для осуществления такого перебора для логического раздела накопителя с приведенными выше параметрами. Примем размер файла равным 6 291 456 байт (6 мегабайт). Такой файл будет занимать 12 288 секторов на диске, т. е. b = 12 288. Найдем число комбинаций, которые подлежат перебору:

$$k = A^{b'}_{n_p'} = \frac{n_p'!}{(n_p' - b')!} = \frac{(\frac{n_p}{c})!}{\left(\frac{n_p}{c} - \frac{b}{c}\right)!} = \frac{\left(\frac{20870512}{8}\right)!}{\left(\frac{20870512}{8} - \frac{12288}{8}\right)!} =$$

$$= \frac{(2608814)!}{(2608814 - 1536)!} = \frac{(2608814)!}{(2607278)!}.$$

Как и в предыдущем случае, даже в случае перебора нескольких сотен комбинаций в 1 с такой метод восстановления данных фрагментированных файлов не приемлем для промышленной эксплуатации. При этом число комбинаций, подлежащих перебору, будет нелинейно возрастать при увеличении размера файла.

Синтез оценочной функции для проверки упорядоченной пары кластеров. Сократить время, необходимое для перебора для ряда форматов файлов, можно просинтезировав оценочную функцию, которая позволяет осуществлять проверку не всего файла целиком, а упорядоченной пары кластеров файла на предмет их принадлежности к искомому файлу и корректности порядка следования.

В качестве примера построения такой функции рассмотрим случай, когда файл, требующий восстановления, имеет формат XML. Одной из особенностей формата XML является то, что для двух произвольно взятых соседних фрагментов документа, как правило, можно установить их принадлежность к одному документу и очередность.

Учитывая, что документ XML имеет открывающие и закрывающие теги, соответствие тегов, схожесть имен тегов в двух фрагментах документа XML определяет возможность или невозможность их принадлежности к одному документу. Таким образом, имеем оценочную функцию:

$$u(a_i, a_j) = \begin{cases} 1 \text{ при } j = i+1; \\ 0 \text{ при } j <> i+1. \end{cases}$$

Когда существует возможность синтеза оценочной функции, позволяющей проверить не весь файл целиком, а упорядоченную пару кластеров файла на предмет их принадлежности к данному файлу и корректности порядка следования, число комбинаций, которые подлежат перебору, существенно меньше, чем при полном переборе:

$$k = \sum_{d=0}^{b'} n_p' - d.$$

Произведем оценку времени T, необходимого для осуществления такого перебора для логического раздела накопителя и файла с параметрами, приведенными выше. Найдем число комбинаций, которые подлежат перебору:

$$k = \sum_{d=0}^{b'} n_p' - d = \sum_{d=0}^{b/c} \frac{n_p}{c} - d = \sum_{d=0}^{\frac{12288}{8}} \frac{20870512}{8} - d = \sum_{d=0}^{1536} 2\,608\,814 - d.$$

Время, затрачиваемое на проверку каждой комбинации, зависит от формата файла и построения оценочной функции, осуществляющей вычисление и сверку контрольной суммы/хэш-функции/имитовставки или проверку электронной подписи.

Следует отметить, что число комбинаций в данном случае несоизмеримо меньше, чем значения, которые были получены ранее при оценке числа комбинаций полного перебора, однако оно остается слишком большим для того, чтобы такой метод мог непосредственно использоваться в промышленной эксплуатации.

По мнению автора данный результат является приемлемым при принятии дополнительных мер, направленных на ограничение множества кластеров, которые могут включаться в перебираемые пары кластеров.

Разработка алгоритма восстановления сведений о фрагментации файла. Решение задачи восстановления фрагментированных

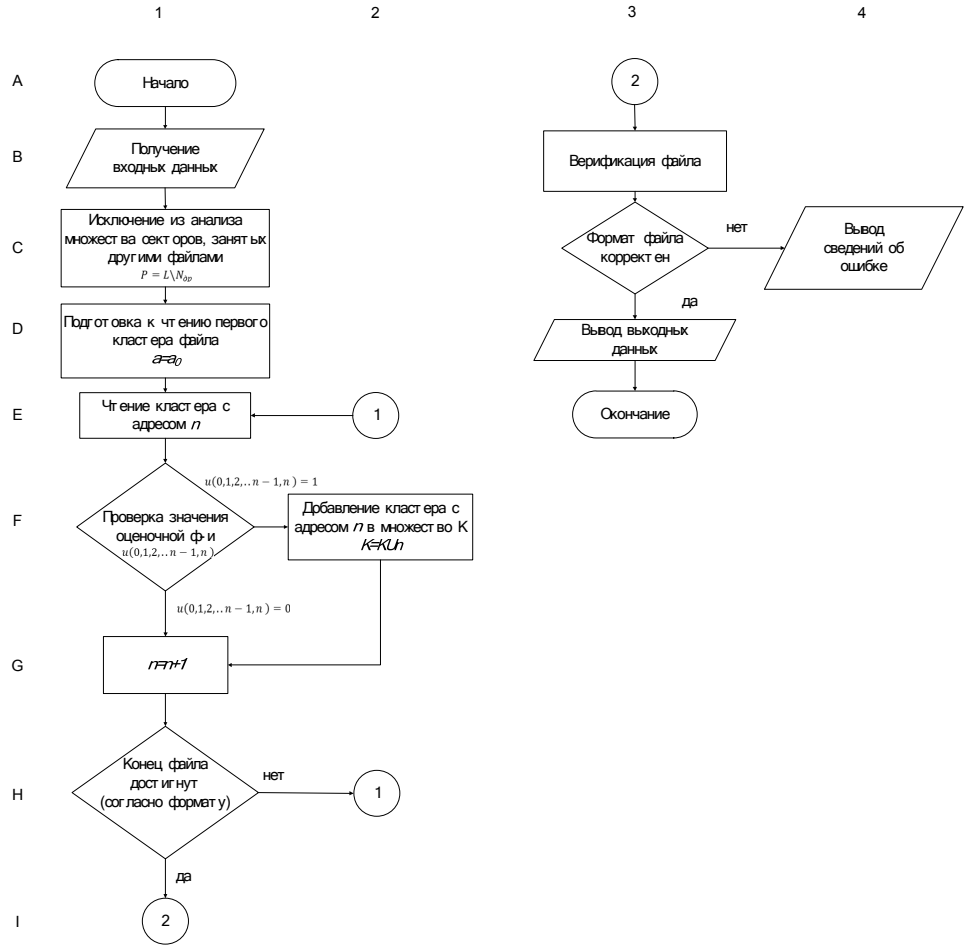
файлов при утрате сведений о фрагментации предполагает нахождение за приемлемое время T линейно упорядоченного множества K адресов секторов накопителя, относящихся к исходному файлу, т. е. стоит также подзадача условной оптимизации. Обязательным набором входных данных для алгоритма являются следующие данные: n_p – число секторов области данных раздела; m_n – номер первого сектора искомого файла; g – размер сектора (число байт в секторе) c – размер кластера (число секторов в кластере); $L_i = l(i), i = \overline{0, n_p}$ – данные (упорядоченные множества байт), хранимые в секторах адресного пространства логического раздела накопителя; L – множество секторов раздела; $N_{\text{др}}$ – множество секторов раздела, занятых другими файлами. В зависимости от характера повреждений также может быть известно b – общее число секторов искомого файла (размер файла в секторах).

Выходными данными для алгоритма является множество K секторов раздела, содержащих данные искомого файла. При этом: $K = k\left(n_p, m_n, g, c, L, N_{\text{др}}\right)$ или $K = m(n_p, m_n, g, c, L, N_{\text{др}}, b)$ в зависимости от характера и объема повреждений. Рассмотрим алгоритм в виде блок-схемы, представленной на рисунке (см. стр. 182).

Один из путей обеспечения приемлемого времени перебора T – ограничение множества кластеров, которые используются в переборе. Вместе с тем возможность применения такого подхода зависит от состояния накопителя (т.е наличия данных, позволяющих ввести такие ограничения).

В случае сохранности информации о размещении первых секторов части других фалов и/или их фрагментации, есть возможность исключить из перебора кластеры раздела накопителя, которые относятся к другим файлам. Даже при сложных смешанных повреждениях и утрате логических структур, хранящих сведения о фрагментации файлов, число таких кластеров может быть значительным. Исключения этих кластеров необходимо проанализировать в найденных в адресном пространстве накопителя структур каталогов, а также логическую структуру корневого каталога, если она сохранилась, исключив первые кластеры всех файлов, на которые они ссылаются, из дальнейшего анализа. Таким образом, для определения множества секторов, используемых в переборе, из множества L секторов области данных раздела необходимо вычесть множество $N_{\text{др}}$ секторов, которые будут отнесены к другим файлам в данном разделе.

$$P = L \setminus N_{\text{др}}.$$



Алгоритм восстановления сведений о фрагментации файлов

Далее необходимо производить поиск всех кластеров, которые могут относиться к данному файлу. В случае, если формат файла позволяет использовать оценочную функцию, которая позволяет осуществлять проверку не всего файла целиком, а упорядоченной пары кластеров файла на предмет их принадлежности к данному файлу и корректности порядка следования, поиск кластеров файла сводится к поиску второго кластера в каждой паре (первым кластером в паре является ранее найденный кластер или первый кластер файла, который известен).

Для каждой пары кластеров, подлежащих проверке, необходимо проверять значение оценочной функции. При значении оценочной функции $u(a_1, a_2) = 1$ следует добавить найденный кластер в множество найденных кластеров исходного файла, прекратить итерационный перебор и продолжать подбор следующего кластера файла, используя найденный кластер как первый кластер в паре.

Необходимо отметить, что драйверы файловых систем FAT 16 /FAT 32 /NTFS 4 /NTFS 5 / NTFS 6 производят кэширование логического адреса сектора, запись в который осуществлялась последним [4]. Таким образом, при наличии на диске свободного пространства каждый следующий кластер, принадлежащий восстанавливаемому файлу, как правило, будет иметь бóльший адрес, чем предыдущий. Исключением являются случаи, когда сектора кластеров, имеющих большие адреса, принадлежат другим файлам.

Следует отметить, что каждый фрагмент файла в большинстве случаев состоит более, чем из одного кластера [5].

Поэтому представляется целесообразным при подборе кластеров файла каждый раз начинать перебор с кластера, следующего за последним найденным кластером файла и продолжать его в сторону увеличения логического адреса первого сектора кластера в адресном пространстве накопителя.

На завершающем этапе необходимо осуществить финальную верификацию структуры файла, расчет контрольной суммы/хэш-функции/имитовставки/проверку электронной подписи в зависимости от того, какие средства контроля целостности предусмотрены форматом файла. Финальная верификация файла позволяет подтвердить соответствие найденного множества секторов K исходному множеству секторов.

СПИСОК ЛИТЕРАТУРЫ

1. Kane Pamela, Hopkins Andy. The Data Recovery Bible, Preventing and Surviving Computer Crashes/Book and Disk. Brady, May 1993. – 512 p.
2. McKusick, M., K. Bostic et al. The Design and Implementation of the 4.4 BSD Operating System. – MA: Addison-Wesley, 1996. – 580 p.
3. Mark G. Sobell A Practical Guide to Ubuntu Linux. – Ann Arbor, Michigan, Prentice Hall, August 2010. – 1320 p.
4. Solomon David A., Russinovich M. Inside Microsoft Windows 2000. – Redmond, WA: Microsoft Press, 2000. – 944 p.
5. Dominic Giampaolo Practical File System Design, San Francisco, California: MORGAN KAUFMANN PUBLISHERS, INC., 1999. – 237 p.

Статья поступила в редакцию 14.05.2012