发明名称
解压压缩文件时减小存储需求的方法和系统

摘要
本发明提供了一种对压缩文件进行解压缩的方法和系统。根据本发明的方法，包括：输入步骤，从所述压缩文件中读入一压缩的数据块；解压缩步骤，对所述压缩的数据块进行解压缩；输出步骤，将解压缩的数据存储到解压缩文件中；删除步骤，从所述压缩文件中删除所述已经解压缩的压缩数据块。本发明提供的方法和系统减少了在压缩格式和解压缩格式之间数据的不必要的重复，不仅能够在解压缩过程中减少存储空间需求，而且不会对现有的压缩/解压缩算法作重大改变，很容易被集成到现有的压缩/解压缩工具中。
权利要求书

1. 一种对压缩文件进行解压缩的方法，包括：
   输入步骤，从所述压缩文件中读入一压缩的数据块；
   解压缩步骤，对所述压缩的数据块进行解压缩；
   输出步骤，将解压缩的数据存储到解压缩文件中；
   删除步骤，从所述压缩文件中删除所述已经解压缩的压缩数据块。

2. 根据权利要求 1 所述的方法，其中，所述压缩文件中包含了压缩数据的分块信息。

3. 根据权利要求 2 所述的方法，其中，其中所述压缩数据的分块信息通过所述压缩文件中的数据块表或者所述压缩文件中的数据块之间的压缩标志表示。

4. 根据权利要求 3 所述的方法，其中所述压缩文件在压缩时是分数据块独立压缩的，数据块之间没有相关性。

5. 根据权利要求 1 所述的方法，其中所述解压缩步骤进一步包括：
   对所述压缩的数据块进行解压缩时，只获得该压缩的数据块的分解压缩结果；
   对未在所述压缩的数据块完成解压缩的压缩数据，加入到下一个待解压缩的数据块。

6. 根据权利要求 1 或 5 所述的方法，其中在所述删除步骤之前还包括：
   判断后续解压缩过程是否需要先前的数据；
   响应于后续解压缩过程需要先前的数据，存储后续压缩过程需要的先前的数据。

7. 根据权利要求 5 所述的方法，其中所述未在所述压缩的数据块完成解压缩的压缩数据采用输入数据的偏移地址表示。

8. 根据权利要求 1 所述的方法，其中删除步骤和其它步骤是在不同的流程中实现的，其中，所述其它步骤在解压缩流程完成，所述删除步骤
在删除流程完成，所述其它步骤包括所述输入步骤，所述解压缩步骤和所述输出步骤。

9. 根据权利要求 8 所述的步骤，其中所述解压缩流程在所述压缩文件中加入标志，该标志表明该标志前的数据可以被删除，所述解压缩流程和删除流程通过所述标志交换信息。

10. 一种对压缩文件进行解压缩的系统，包括：
输入装置，用于从所述压缩文件中读入一压缩的数据块的装置；
解压缩装置，用于对所述压缩的数据块进行解压缩的装置；
输出装置，用于将解压缩的数据存储到解压缩文件中的装置；
删除装置，用于从所述压缩文件中删除所述已经解压缩的压缩数据块的装置。

11. 根据权利要求 10 所述的系统，其中，所述压缩文件中包含了解压缩数据的分块信息。

12. 根据权利要求 11 所述的系统，其中，所述压缩数据的分块信息通过所述压缩文件中数据块表或者所述压缩文件中的数据块之间的压缩标志表示。

13. 根据权利要求 12 所述的系统，其中所述压缩文件在压缩时是分数据块独立压缩的，数据块之间没有相关性。

14. 根据权利要求 10 所述的方法，其中所述解压缩装置的处理步骤还包括：
对所述压缩的数据块进行解压缩时，只获得该压缩的数据块的解压缩结果；
对未在所述压缩的数据块完成解压缩的压缩数据，加入到下一个待解压缩的数据块。

15. 根据权利要求 10 或 14 所述的系统，其中还包括：
判断装置，用于判断后续解压缩过程是否需要先前的数据的装置；
预处理装置，响应于后续解压缩过程需要先前的数据，用于存储后续压缩过程需要的先前的数据的装置。
16. 根据权利要求14所述的系统，其中所述未在所述压缩的数据块完成解压缩的压缩数据采用输入数据的偏移地址表示。

17. 根据权利要求10所述的系统，其中所述解压缩装置在所述压缩文件中加入标志，该标志表明该标志前的数据可以被删除。

18. 根据权利要求17所述的系统，其中所述删除装置和所述其它装置之间独立工作，所述删除装置和所述其它装置之间通过所述标志交换信息，其中，所述其它装置包括所述输入装置、所述解压缩装置和所述输出装置。
解压缩文件时减小存储需求的方法和系统

技术领域

本发明涉及计算机软件技术，特别地，涉及解压缩文件时减小存储设备空间需求的方法和系统

背景技术

文件压缩是一种通过对原始文件中的信息进行编码，从而使用较少字节的文件来表达该原始文件中的信息的过程。文件压缩是非常有用的，因为文件压缩大大减少了资源的消耗，例如，信息保存时，压缩文件可以减少对存储资源的需求，信息传输时，压缩文件可以减少对带宽的需求，等等。由于这些需求，压缩技术发展很快，涌现出大量的压缩算法，不同的压缩算法会采用不同的压缩格式，例如微软的 Window 操作系统中常用的 ZIP、RAR 格式，Linux/Unix 操作系统常用的 gz、bz 格式等。

文件压缩可以将多个文件压缩成一个文件，因此，当对较多的文件进行压缩时，或者对一个非常大的原始文件进行压缩时，压缩文件仍然很大。当解压缩作为一个整体过程，被压缩的数据在整个解压缩过程中，在存储设备（例如磁盘或磁带）中保持不变，这样，为了成功地解压缩一个大小为 S 的压缩文件，在解压缩完成之前，至少需要 2S 大小的存储空间。这样的存储空间要求对于较大的压缩文件是很难满足的。

现有技术已经提出了一些对压缩文件减少存储空间需求的解决办法。第一种方法是解压缩完成后删除被压缩的数据；这种方法的典型例子是 Linux 中的 bunzip2，其当解压缩完全完成后，默认将自动删除压缩文件。

第二种方法是对压缩文件进行选择性地解压缩。WinRAR 及 WinZip 就是采用该方法的典型例子，这两种软件允许用户在未解压缩文件前浏
览该压缩文件，当压缩文件中的一个或多个文件被选择并且被打开时，解压方法就直接启动。

但是，这两种方法实际上都是将解压的文件存储到存储设备上，但是对于某些领域的数据，如银行交易数据，石油业的监测数据，气象观测涉及的数据、科学计算涉及的数据，或者多媒体数据等，这些数据的数据量非常大，备份的压缩文件也非常大，例如地震行业中超过 100G 字节的数据库是非常常见的。这样大的备份压缩文件在解压缩时就需要非常大的存储空间，加上解压后的文件，仅仅这两个文件的大小，存储空间也常常不能满足这种要求。

发明内容

因此，需要一种在解压缩过程中减少存储空间需求的方法和系统，该方法和系统能解决现有方法不能满足解压缩过程中减少存储空间需求的问题。

根据本发明的一个方面，提供了一种对压缩文件进行解压缩的方法，包括：输入步骤，从所述压缩文件中读入一压缩的数据块；解压缩步骤，对所述压缩的数据块进行解压缩；输出步骤，将解压缩的数据存储到解压缩文件中；删除步骤，从所述压缩文件中删除所述已经解压缩的压缩数据块。

根据本发明的另一个方面，提供了一种对压缩文件进行解压缩的系统，包括：输入装置，用于从所述压缩文件中读入一压缩的数据块的装置；解压缩装置，用于对所述压缩的数据块进行解压缩的装置；输出装置，用于将解压缩的数据存储到解压缩文件中的装置；删除装置，用于从所述压缩文件中删除所述已经进行解压缩的压缩数据块的装置。

本发明提供的方法和系统减少了在压缩格式和解压缩格式之间数据的不必要的重复，不仅能够在解压缩过程中减少存储空间需求，而且不对现有的压缩/解压缩算法作重大改变，很容易被集成到现有的压缩/解压缩工具中。
附图说明

通过对附图中本发明示例实施例方式的更详细描述，本发明的上述、以及其它目的、特征和优势将变得更加明显，其中，相同的参考标号通常代表本发明示例实施例方式中的相同部件。

图 1 示意性地示出了根据本发明的一个优选的实施方式的解压缩方法流程图；
图 2 示意性地示出了一个分块压缩的方法流程图；
图 3 示意性地示出了根据图 2 的压缩方法对应的解压缩方法的流程；
图 4 示意性地示出了具有压缩标志的压缩数据；
图 5 示例性地示出了压缩数据并生成压缩标志的方法流程；
图 6 示意性地示出了使用图 5 压缩流程的压缩文件解压缩过程；
图 7 示例性地示出了解压缩数据缓冲区和输入数据的偏移地址的关系示意图；
图 8 示例性地示出了方案 3 的解压缩流程；
图 9a 示例性地示出了一个优选实施方式的解压缩过程；
图 9b 示意性地示出了图 9a 解压缩过程对应的压缩数据删除过程；以及
图 10 示例性地示出了本发明所用的解压缩系统。

具体实施方式

将参照附图更加详细地描述本发明的优选实施方式，在附图中显示了本发明的优选实施例。然而，本发明可以以各种形式实现而不应该理解为被这里阐述的实施例所限制。相反，提供这些实施例是为了使本发明更加透彻和完整，并且，完全将本发明的范围传达给本领域的技术人员。

通常的解压缩技术在完成整个解压缩过程中，会在存储空间（例如硬盘或磁带等）保持原始压缩数据不变，如果原始压缩数据过大，解压缩过程中存储空间不足，会造成解压缩失败。

本发明提出了一种在解压缩过程中减少存储空间需求的方法，该方法
在解压缩时，全部的压缩数据被分成若干块，每个数据块的大小可以相同，
也可以不同，分块可以由用户指定大小分块或不同大小分块，这样就
can be divided into smaller blocks. In this case, the size of the blocks can be
different or the same, which allows for more flexibility in compression.

图 1 示意性地示出了根据本发明的一个优选的实时方式的解压缩方法
流程图，该流程在解压缩过程中能够减少存储空间的需求。在图 1 中，首
先在步骤 S101，开始解压缩过程，该过程也可认为是一个初始化过
程，用户可以在开始的初始化过程得到分块信息，也可能是用户自己对数
can be divided into smaller blocks. In this case, the size of the blocks can be
different or the same, which allows for more flexibility in compression.

图 1 示意性地示出了根据本发明的一个优选的实时方式的解压缩方法
流程图，该流程在解压缩过程中能够减少存储空间的需求。在图 1 中，首
先在步骤 S101，开始解压缩过程，该过程也可认为是一个初始化过
程，用户可以在开始的初始化过程得到分块信息，也可能是用户自己对数
data 分块等。这里假设全部压缩数据的大小为 M。在步骤 S102，从全部压
缩文件中读入一个数据块，假设该数据块的大小为 M1(M1＜M)，在步骤
S103，使用现有的解压缩方法解压读入的该压缩的数据块；在步骤 S104，
将解压后的数据输出到解压缩文件中，假设解压后的该数据块大小为 N1，
全部压缩的数据解压后的数据大小为 N(N1＜N)，然后在步骤 S105，从压缩
文件（大小为 M）中删除已经解压的压缩数据块（大小为 M1），然后
返回步骤 S102，继续读入下一压缩数据块进行解压。由上面的解压过程描
述可以看出，原始的压缩文件占用的存储空间在不断减小，解压后的数据
文件占用的存储空间不断加大，在读入第 k 个数据块时，整体占用的存储
空间为 M-M1-...-Mk+N1+...+Nk。其所占用的存储空间比现有技术少
M1+...+Mk。当原始压缩文件全部解压完毕时，压缩文件占用的存储空间
将全部释放出来，这时该方法整体占用的存储空间为 N，所占用的存储空
间比现有技术少 M。这

由于压缩/解压缩方法比较特殊，很多算法要利用数据之间的相关性，
因此，上述方法的具体实现中还有很多具体的要求，例如，数据分成多少
个数据块，每个数据块如何处理等。下面将提供四种不同的具体实施方
案，来实现图 1 中的方法。当然，本领域技术人员在本发明提供的技术方
案下，可以知道还有多种其它的实现方法，具体数据如何分块，并不影
响本发明的实现。因此，本发明提供的具体实施例并不构成对本发明的限
制，相反，在本发明构思下的其它技术方案都被认为包含在本发明的保护

8
范围之内。

方案 1:

在该技术方案中，为了进行解压缩，在压缩时将原始数据分块，原始数据块之间是相互独立的，因此，可以对每个原始数据块单元分别压缩，然后将压缩结果融合在一起，成为压缩后的数据，同时，压缩后的数据还记录了如何对数据进行分块，这样，解压缩时就可以根据压缩文件中的数据分块信息，对每个压缩的数据块单元信息分别解压缩，每当解压缩一个压缩的数据块单元，就可以从原始压缩文件中删除该压缩的数据块单元信息，从而达到解压缩时减少原始压缩文件占用的存储空间的目的。

每一个数据块单元的长度可由用户指定，例如，用户可以指定相同大小的数据块单元，也可以指定不同大小的数据块单元，或者，也可以采用系统默认的数据块大小，例如，1 兆字节等等。通常来说，小的数据块更能减少对存储空间的需求，但是，会降低压缩的效率，因为，分成更多的相互独立的数据块减小了数据之间的相关性。因此，用户可以根据自己的需求，灵活分块。

数据分块的信息在压缩过程中，以各种形式存储在压缩后的文件中，优选地，可以采用表 1 所示的的数据块表，放在压缩后的文件的开始、结尾或者任何指定的地方。表 1 中，每一个数据块占用一行。每一行的条目包括数据块的 ID，在压缩后的文件中该数据块的大小和数据块的偏址，也可以记录数据块起始位置和结束位置，或者是其它相关信息。通过这些信息能够导出在压缩后的文件中该数据块的起始位置和结束位置的信息。在该表中，用户还可以包括其它信息。

<table>
<thead>
<tr>
<th>数据块 ID</th>
<th>数据块大小</th>
<th>数据块偏址</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x472A</td>
<td>0x00000000</td>
</tr>
<tr>
<td>1</td>
<td>0x5D24</td>
<td>0x0000427A</td>
</tr>
<tr>
<td>2</td>
<td>0x4057</td>
<td>0x00009F9E</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
图 2 示意性地示出了一个分块压缩的方法流程图。该压缩流程中，在步骤 S201，开始压缩过程。在步骤 S202，将数据分成数据块。该步骤可以通过接收用户的分块信息来进行分块，也可以自动分块，例如，将每个数据块的大小定义为 1 兆字节等等。在步骤 S203，对于每个数据块，采用现有的压缩算法进行压缩，并将压缩后的数据加入到压缩文件中，就可以得到该压缩后的数据块在压缩后的文件中的位置，包括其起始位置和结束位置。然后在步骤 S204，更新每个数据块在压缩后的文件中的位置信息，接着，在步骤 S205，所有的数据都压缩完毕，将得到的数据分块信息加入到压缩文件中。最后，在步骤 S206，输出压缩后的文件，就完成了分块压缩数据过程。

图 3 示意性地示出了根据图 2 的压缩方法对应的解压缩方法的流程。图 3 所示的方法流程中，步骤 S301 还要获得压缩数据分块信息，这是和图 1 略有不同的地方，其他步骤和图 1 是一致的，这里，不再赘述。

这样，通过在压缩时对数据分块，解压缩时利用分块信息解压缩，逐步删除解压后的压缩信息，就实现了解压缩时减少存储空间需求的目的。该技术方案不依赖于压缩算法，可以用于任何压缩算法。

方案 2：

在该技术方案中，采用方案 1 中的分块压缩方式，但是，不在压缩文件中直接保留数据分块信息，而是通过将压缩标志放入压缩文件的方式，来间接提供给解压缩方法数据的分块信息。任意两个相邻的压缩标志之间的数据被认为是分块压缩的，没有相关性。图 4 示意性地示出了具有压缩标志的压缩数据。

图 5 示意性地示出了压缩数据并生成压缩标志的方法流程。和图 2 相比，步骤 S204 和 S205 更新为图 5 中的步骤 S504，将压缩后的数据块加入到压缩文件后，加入压缩标志，其它如 S201 和 S501、S202 和 S502、S203 和 S503、S206 和 S505 步骤是相同的，这里不再赘述。

对应图 5 的压缩流程，图 6 示意性地示出了使用图 5 压缩流程的压缩文件解压缩过程。这时读入的数据块就是两个压缩标志之间的数据。
骤 S602 所示。其余步骤，如 S101 和 S601、S103 和 S603、S104 和 S604、S105 和 S605 是相同的，这里不再赘述。

该技术方案不依赖于压缩算法，可以用于任何压缩算法。

3 方案 3

为了说明该方法，首先要定义以下几个概念:

解压缩数据缓冲区(Buffer): 该数据缓冲区是用于解压缩过程的一个数据缓冲区，通常是内存中的一个存储区域。该数据缓冲区可以有固定的大
小，例如，1兆字节等，也可以由用户在合适的范围内设定其大小，并且，
不同的解压缩算法也可以采用不同的缓冲区大小。数据缓冲区大小也就是
对应着图 1 的数据块的大小，解压缩算法就对该解压缩数据缓冲区中的数
据进行解压缩。这里需要注意的是：压缩算法常常利用数据之间的相关性
因此，对解压缩数据缓冲区中的数据进行解压缩时，并非全部数据都能够
解压缩，通常，其最后的一部分数据，可能会由于其相关的数据没有完全读
入解压缩缓冲区，从而使得该部分数据不能够被正确解压缩。

输入数据的偏移地址(Offset): 解压缩数据缓冲区可能出现只有部分数
据被解压缩的情况，因此，在下一次将压缩数据读入到下一解压缩缓冲
区时，需要将没有被解压的压缩数据再次读入到该下一压缩数据缓冲区,
Offset 为没有被解压的压缩数据在 Buffer 中的起始偏移地址，从 0 到
offset 之间的数据已经被正确解压缩。如果 offset 值最后等于 Buffer
的大小，则所有的输入数据都被成功解压缩。

将某一解压缩缓冲区的全部或部分数据解压缩后，就要考虑压缩文件中
已经解压缩的数据对应的压缩数据，如果该数据不影响后续解压缩过程,
就将该数据删除；如果该数据影响后续解压缩过程，可以将该数据利用另
外的数据结构保留以备后续解压缩过程使用，但是可以在硬盘之类的存储
空间将该数据删除。在解压缩算法中，与先前数据有依赖关系的含义，可以
是和解压后的先前数据有依赖关系，也可以是和未解压的先前数据之间有
依赖关系，要根据不同的算法在另外的数据结构中存储对应需要的依赖数
据。
图 7 示例性地示出了解压缩数据缓冲区和输入数据的偏移地址的关系示意图。

图 8 示例性地示出了方案 3 的解压缩流程。在步骤 S801，开始解压缩过程。在步骤 S802，从压缩文件起始点读入一部分压缩数据到一个解压缩数据缓冲区。这个解压缩数据缓冲区就对应图 1 中的数据块，缓冲区大小可以指定，也可以由用户设定。然后在步骤 S803，使用现有的解压缩方法解压读入的该部分数据，获得 OFFSET，在解压缩数据缓冲区中，OFFSET 的数据已经被完全解压缩了。但是需要在步骤 S805 判断后续解压缩过程是否需要先存数据？如果需要，可以在步骤 S806 使用另外的内存存储 OFFSET 之间的后续解压过程需要的依赖数据，然后进入步骤 S807；如果没有数据依赖关系，就可以直接进入步骤 S807，从压缩的数据中删除压缩文件起始点到 OFFSET 之间的数据。步骤 S804 是输出 OFFSET 之间的解压后的数据，该步骤需要在步骤 S803 后执行，但是可以在步骤 S807 之前任意步骤，甚至在循环之间的步骤 S807 之后执行。

下面将以压缩算法 DEFLATE 为例进行相应的说明。DEFLATE 压缩算法是 LZ77 压缩算法和哈夫曼编码相结合的压缩算法，被 zip、gzip、zlib 及 png 档案广泛使用。DEFLATE 算法首先采用 LZ77 算法进行重复出现“字符串”（注：这里的“字符串”并不仅仅局限于可显示的字符串，而是任意的 bit 流）的压缩，之后进一步采用哈夫曼编码进行压缩。

简单的讲，LZ 算法被认为是字符串匹配的算法。例如：在一段文本中某字符串经常出现，并且可以通过前面文本中出现的字符串指针来表示。当然这个算法的前提是指针应该比字符串本身要短。如下例所示：

待压缩的文本字符串：Blah blah blah blah blah!

^ ^

1,2,…6

经过 LZ77 扫描结果：Blah b[D=5, L=18]！

其中字符串“lah b”重复出现，因此 LZ77 算法采用[D=5,L=18]（通
常被认为是字符指针来压缩后续重复的字符串。其中 D 表示距离，L
表示长度。其意义就是从插入 [D, L] 的插入点（第 7 个字符处）开始往后的
18 个字符与 “插入点减去 D 的值” 开始往后的 18 个字符一模一样，即：

Blah blah blah blah blah!

1, 7………………24!

从插入点往后的 18 个字符串是：lah blah blah blah

从 “插入点减去 D 的值” 即第二个字符往后的 18 个字符串是：lah blah

由于 LZ77 限定 D 最长为 256 字节（需要 8bits），L 最长是 32Kbytes（需
要 15bits），那么一个 [D, L] 配对就需要至少 23bits，即 3bytes。因此，
如果重复的字符串的长度大于 3 即可达到压缩的目的。

为了进一步减少信息冗余，DEFLATE 算法对 LZ77 扫描的结果进行
哈夫曼编码，最终压缩的数据的第一部分哈夫曼树包括两个哈夫曼树信息，
其中一颗哈夫曼树信息用于对字符串及其对应的长度进行编码；另外一颗
哈夫曼树则对插入的 [D, L] 对中的 D 信息进行编码。

在解压文件之前，两颗哈夫曼树的信息将被首先读入内存进行构建解
压所需的哈夫曼树（通常，哈夫曼树信息远远小于压缩数据信息），之后
读取压缩数据（采用哈夫曼编码的数据）进行解压。由哈夫曼编码规则可
知，每个哈夫曼编码都是可以唯一识别信息的编码，因此，压缩程序在成
功识别一个哈夫曼码之后即可将其从压缩文件中删除。

举个简单的哈夫曼编码的例子:

待编码字符串：ABBBACDCAA

其中 A(4) B(3) C(2) D(1) 分别表示 A 出现了 4 次，B 出现了 3 次，C
出现了 2 次，D 出现了 1 次。

其哈夫曼编码如下所示:

x(10)

x(6) A(4)

x(3) B(3)
D(1)    C(2)
A:  1
B:  01
C:  001
D:  0000

依据哈夫曼压缩之后的字符串为:
1010101100100000 0111（一共不到 3Bytes）
此时，如果 Buffer=10bit 如下所示:
1010101100100000 0111
    ^=10th Bit
那么，在 0 到 Buffer 结束之间最后一个识别的哈夫曼码是:
1 01 01 01 1 0 0100000 0111
    ^=9th Bit
因为前 8bits 已经被成功解压，所以可以删除，即 OFFSET=8bit；而
OFFSET 到 Buffer 结束之间的 2bit 没有被解压缩，故需要输入下个 Buffer。

方法 4

在本技术方案中，关键点在于在解压缩过程中，原始的压缩文件中
插入 TO-BE-REMOVED 标志。该标志表明该压缩标志以前的数据都已经
被解压缩了，可以被删除，从而达到一边解压缩原始压缩文件，一边删除
部分已经解压缩的压缩文件的内容，从而减少在解压缩过程中需要的存储
空间。

该方案在解压缩过程包含了两个流程，第一个流程是解压缩过程，
第二个流程是删除流程，这两个流程是并行的，通过 TO-BE-REMOVED
标志来交换信息。

图 9a 示意性地示出了一个优选实施方式的解压缩过程，在该解压缩过
程中，首先在步骤 S901，开始解压缩过程，解压缩过程也是分块进行解压
缩的，假设数据块的大小为 N，则在步骤 S902，从最后一个
TO-BE-REMOVED 标志开始读入 N 字节的压缩数据，对于第一个数据
块，由于没有 TO-BE-REMOVED 标志，可以在压缩文件的开始加入一个
TO-BE-REMOVED 标志，这样该数据块的处理就和其它的数据块处理完
全相同了。然后在步骤 S903，解压缩该 N 个字节的 M 个字节压缩数据，
正如方案 3 中所陈述的，对解压缩数据缓冲区中的数据进行解压缩时，可
能并非全部数据都能够解压缩，可能出现最后的一部分数据，由于其相关
的数据没有完全读入解压缩缓冲区，该部分数据最后不能够解压缩。因此,
这里我们假设大小为 N 的数据块中只有 M 个数据被成功解压缩 (M 小于等
于 N)。然后在步骤 S905，判断后续解压缩过程是否需要先前数据？如果
需要，可以在步骤 S906 使用另外的内存存储后续解压过程需要的依赖数
据，然后进入步骤 S907；如果没有数据依赖关系，就可以直接进入步骤
S907，在步骤 S907，将 TO-BE-REMOVED 标志插入到压缩文件的偏移
地址，在该步骤之前，需要执行步骤 S904，得到被删除的 M 个数据结
束的偏移地址，该步骤可以在步骤 S903 之后，步骤 S907 之前的任何一步执
行。这样，一个独立的解压缩过程就可以单独运行了，其在原始的压缩文
件中插入了若干 TO-BE-REMOVED 标志，表明该标志之前的的数据都可以
被删除。

图 9b 示意性地示出了图 9a 解压缩过程对应的压缩数据删除过程。该
过程首先在步骤 S9011 开始，在步骤 S9012 扫描压缩文件，在步骤 S9013
判断是否在压缩文件中发现 TO-BE-REMOVED 标志，如果发现该标志，
进入步骤 S9014，将压缩文件从开始到该标志的数据删除，然后可以继续
在步骤 S9012 扫描压缩文件，如果考虑节省 CPU 资源，也可以休眠若干
时间后，从新开始扫描压缩文件。如果在步骤 S9013 没有发现该标志，可
以继续在步骤 S9012 扫描压缩文件。

在同一个发明构思下，图 10 示意性地示出了本发明所用的解压缩系统
1001，该系统包括，输入装置 1004，解压缩装置 1005，输出装置 1006 和
删除装置 1007。输入装置 1004 用于从压缩文件 1002 中读入一压缩的数据
块；解压缩装置 1005 用于对所述压缩的数据块进行解压缩；输出装置 1006
用于将解压缩的数据存储到解压缩文件 1003 中；删除装置 1007 用于从所
述压缩文件 1002 中删除已经完成解压缩的压缩数据块。

在一种实施方式中，压缩文件 1002 中包含了压缩数据的分块信息，优选实施方式中，该分块信息通过所述压缩文件中的数据块表或者所述压缩文件中的数据块之间的压缩标志表示。

在另一种实施方式中，压缩文件 1002 在压缩时是根据数据块独立压缩的，数据块之间没有相关性。

在另一种实施方式中，解压缩装置 1005 的处理步骤还包括：对所述压缩的数据块进行解压缩时，只获得该压缩的数据块的部分解压缩结果；对未在所述压缩的数据块完成解压缩的压缩数据，加入到下一个待解压缩的数据块。

在又一种实施方式中，解压缩装置 1005 还包括：判断装置 1008，用于判断后续解压缩过程是否需要先前的数据；以及预处理装置 1009，响应于后续解压缩过程需要先前的数据，用于存储后续压缩过程需要的先前的数据。

在另一种实施方式中，由于对解压缩数据缓冲区中的数据进行解压缩时，可能并非全部数据都能够解压缩，可能出现最后的一部分数据，由于其相关的数据没有完全读入解压缩缓冲区，该部分数据最后不能够解压缩，此时对所述未在所述压缩的数据块完成解压缩的压缩数据可以采用输入数据的偏移地址表示。

在又一种实施方式中，解压缩装置 1005 可以在压缩文件 1002 中加入 TO-BE-REMOVED 标志，该标志表明该标志前的数据可以被删除。这时，删除装置 1007 和其它装置（包括输入装置 1004、解压缩装置 1005 和输出装置 1006）之间独立工作，删除装置 1007 和所述其它装置之间通过所述 TO-BE-REMOVED 标志交换信息，从而在压缩文件 1002 中删除已经被完全解压缩的数据。

本发明的系统中的模块可以采用由诸如超大规模集成电路或门阵列、诸如逻辑芯片、晶体管等的半导体、或者诸如现场可编程门阵列、可编程
逻辑设备等的可编程硬件设备的硬件电路实现，也可以用由各种类型的处理器执行的软件实现，也可以由上述硬件电路和软件的结合实现。

本发明还提供一种程序产品，包含实现以上所有方法的程序代码以及承载该程序代码的承载介质。

虽然这里参照附图描述了本发明的示例性实施例，但是应该理解本发明不限于这些精确的实施例，并且在不背离本发明的范围和宗旨的情况下，本领域普通技术人员能对实施例进行各种变化的修改。所有这些变化和修改意欲包含在所附权利要求中限定的本发明的范围内。
图 1
图 2
图 3

...... 第 i 块 压缩标志 第 i+1 块 压缩标志 ......
图 5

用户分块信息

开始

将数据分成数据块

S501

S502

对于每个数据块，采用现有的压缩算法进行压缩

S503

将压缩后的数据块加入到压缩文件后，加入压缩标志

S504

输出压缩后的文件

S405

图 6

开始

S601

从全部压缩的数据中读入到下一个压缩标志之前的数据

S602

使用现有的解压缩方法解压读入的该部分数据

S603

输出解压后的该部分的数据

S604

从压缩的数据中删除已经被解压的数据

S605
图 7

S801 开始

S802 从压缩文件起始点读入一部分压缩数据到一个解压缩数据缓冲区

S803 使用现有的解压缩方法解压读入的该部分数据，获得 OFFSET

S804 输出解压后的该部分的数据

S805 判断后续解压缩过程是否需要先前数据？

S806 需要

S807 存储后续解压过程需要的依赖数据

S808 从压缩的数据中删除压缩文件起始点到 OFFSET 之间的数据

图 8
图9a

图9b
图 10