(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년02월20일
(11) 등록번호 10-1365861
(24) 등록일자 2014년02월14일

(51) 국제특허분류(Int. Cl.)
*G06F 17/30* (2006.01) *G06F 17/40* (2006.01)
(21) 출원번호 10-2013-7029338(분할)
(22) 출원일자(국제) 2008년02월12일
심사청구일자 2013년11월05일
(85) 번역문제출일자 2013년11월05일
(65) 공개번호 10-2013-0130876
(43) 공개일자 2013년12월02일
(62) 원출원 특허 10-2009-7016865
원출원일자(국제) 2008년02월12일
심사청구일자 2012년08월27일
(86) 국제출원번호 PCT/FR2008/000177
(87) 국제공개번호 WO 2008/113921
국제공개일자 2008년09월25일
(30) 우선권주장
0753222 2007년02월13일 프랑스(FR)
(56) 선행기술조사문헌
US20040236761 A1
US5742817 A
US6687716 B1
US20060168450 A1

(73) 특허권자
**에스떼제 엥떼락티브**
프랑스, 빠리 75016, 아브뉘 모짜르트 29
(72) 발명자
**타마스, 알렉시스**
프랑스, 파리 에프-75016, 뤼 보시오, 3
**그랭베르, 아마우리**
프랑스, 파리 에프-75003, 뤼 노트르담 드 나자렛, 18
(74) 대리인
**김 순 영, 김영철**

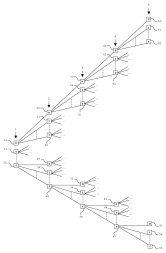
전체 청구항 수 : 총 9 항 심사관 : 손준영

(54) 발명의 명칭 **파일 관리 방법**

## (57) 요 약

본 발명은, 각각 N개의 디렉토리(directory)를 가지고 M은 1보다 큰 실수인 M개의 레벨(level)을 가진 디렉토리의 트리를 생성하는 단계를 포함하는 데이터 파일의 베이스(base)를 조직화하기 위한 첫 번째 단계와; 데이터 파일을 기록하기 위한 단계로서, 해쉬 함수(hash function)를 기록될 데이터 파일($F_i$)의 식별자(identifier)에 적용하는 단계, 상기 적용 단계의 결과에 따라 복수의 레벨의 트리 내에서 목적 디렉토리(destination directory)($R_{di}$)의 경로(path)를 결정하는 단계 및 데이터 파일의 식별자에 의존하는 위치에서, 상기 해쉬 함수에 의해 결정된 상기 디렉토리($R_{di}$)에 데이터 파일을 기록하는 단계를 포함하는 데이터 파일을 기록하기 위한 단계; 및 데이터 파일을 판독하기 위한 단계로서, 동일한 해쉬 함수를 판독될 데이터 파일($F_j$)의 식별자에 적용하는 단계, 상기 적용 단계의 결과에 따라 트리 내에서 목표 디렉토리(target directory)($R_{cj}$)의 경로를 결정하는 단계, 및 데이터 파일의 식별자에 의존하는 위치에서, 상기 해쉬 함수에 의해 결정된 상기 디렉토리($R_{cj}$)에서 데이터 파일을 판독하는 단계를 포함하는 데이터 파일을 판독하기 위한 단계를 포함하는 것을 특징으로 하는 파일 관리 방법에 관한 것이다.

*대 표 도* - 도1



## *특허청구의 범위*

**청구항 1**

M 개의 레벨을 포함하는 디렉토리(directory)의 트리를 구성하는 단계로서, 상기 M은 1보다 큰 정수이며, 상기 트리는 1 레벨의 디렉토리들의 세트를 포함하되, N은 1보다 큰 정수이며, 1부터 M-1까지의 각 X 레벨에 대하여, X 레벨의 디렉토리들 각각은 X+1레벨의 N개의 디렉토리들의 세트를 포함하는, 상기 M 개의 레벨을 포함하는 디렉토리의 트리를 생성하는 단계; 및

제 1 식별자를 갖는 제 1 파일을 저장하는 단계로서,

해쉬 함수(hash function)를 제 1 식별자에 적용하여 제 1 해쉬를 생성하는 단계;

상기 제 1 해쉬에 응답하여, 상기 트리를 트래버스(traverse)함으로써 M 레베로부터 목적 디렉토리(destination directory)를 선택하는 단계; 및

상기 목적 디렉토리로 상기 제 1 파일을 저장하는 단계;에 의해 상기 제 1 식별자를 갖는 상기 제 1 파일을 저장하는 단계;

를 포함하는, 방법.

**청구항 2**

제1항에 있어서,

상기 제 1 해쉬는 M개의 부분들을 포함하며, 상기 트리는 상기 M개의 레벨들 각각에 대해서, 상기 M개의 부분들 중 대응하는 하나의 부분을 이용하여 디렉토리를 선택함으로서 트래버스되는, 방법.

**청구항 3**

제2항에 있어서,

상기 M개의 부분들 각각은 B비트를 포함하며, B는 1보다 큰 정수이고, N은 $2^B$와 동일한, 방법.

**청구항 4**

제3항에 있어서,

B는 4이며, N은 16이고, N개의 디렉토리들에 대한 각각의 세트는 16개의 16진법 숫자를 이용하여 지칭되고, 상기 세트의 각 디렉토리에는 하나의 16진법 숫자가 대응하는, 방법.

**청구항 5**

제2항에 있어서,

상기 M개의 부분들 각각은 T개의 값들 중 하나로 정해질 수 있으며, N과 T가 동일한 경우, N개의 디렉토리들에 대한 각각의 세트는 상기 T개의 값들을 이용하여 지칭되고, 각 디렉토리에는 상기 T개의 값들 중 하나가 대응하는, 방법.

**청구항 6**

제1항에 있어서,

상기 디렉토리들의 트리를 구성하는 단계는, 기본적인 파일 시스템을 이용하되, 상기 기본적인 파일 시스템을 수정하지 않으며 수행되는, 방법.

**청구항 7**

제1항에 있어서,

제 2 식별자를 갖는 제 2 파일을 검색하는 단계로서,

상기 해쉬 함수를 상기 제 2 식별자에 적용하여 제 2 해쉬를 생성하는 단계;

상기 제 2 해쉬에 응답하여 상기 트리를 트래버스함으로써 M 레벨로부터 목표 디렉토리를 선택하는 단계; 및

상기 목표 디렉토리로부터 상기 제 2 파일을 검색하는 단계;에 의해 상기 제 2 식별자를 갖는 상기 제 2 파일을 검색하는 단계를 추가로 포함하는, 방법.

**청구항 8**

제7항에 있어서,

상기 제 2 파일을 획득하는 단계로서,

기본적인 파일 시스템의 원자적 연산(atomic operation)을 이용하여 상기 목표 디렉토리로부터 상기 제 2 파일을 열고 잠그는 단계;

상기 제 2 파일의 헤더 부분으로부터 상태 파라미터를 판독하는 단계;

상기 상태 파라미터가 상기 파일이 이용가능함을 나타내는 것에 응답하여, 상기 제 2 파일의 상기 상태 파라미터를 이용불가능한 상태로 설정하고, 다른 프로세스가 상기 제 2 파일을 이용하지 못하도록 방지하는 단계; 및

상기 기본적인 파일 시스템의 원자적 연산을 이용하여 상기 제 2 파일을 닫고 잠그는 단계;에 의해 상기 제 2 파일을 획득하는 단계를 추가로 포함하는, 방법.

**청구항 9**

제1항에 있어서,

상기 제 1 식별자는 상기 제 1 파일의 파일 이름인, 방법.

# *명 세 서*

## *기 술 분 야*

[0001] 본 발명은 디지털 파일의 관리 분야, 좀더 자세하게는 기록 매체 상에서 디지털 파일의 기록, 판독 및 수정에 관한 것이다. 좀더 정확하게는, 파일 시스템 특히, 컴퓨터 운영 시스템과 상호 작용하는 데이터 파일 관리를 위한 방법 및 시스템에 관한 것이다.

## *배 경 기 술*

[0002] 보통, 새로운 데이터 파일은 예컨대, 하드 디스크와 같은 메모리 내의 위치에 메모리 공간(memory space)의 가용성(availability)에 따라 기록된다. 하드 디스크 상에 기록하는 경우, 최근의 데이터가 포맷 시에 초기화된 블록 내에서 조직화된다. 이러한 블록들은 포맷 시에 메모리의 물리적 섹터(sector) 상에 분배된다.

[0003] 컴퓨터 상에서 데이터 파일의 관리는 컴퓨터 운영 시스템의 파일 시스템에 의해 수행된다.

[0004] 운영 시스템의 파일 시스템은 기록 매체 상의 하나 이상의 파티션(partition)을 이용한다. 파티션은 임의의 데이터 파일을 기록하기 위한 기록 공간 및 기록 공간의 특성(characteristic) 특히, 기록 공간 내의 파일들의 주소가 기록되는 인덱싱(indexing) 공간을 포함한다.

[0005] 어떤 파일들은, 데이터 파일에 그리고 동일 파이션에 속한 디렉토리(directory)와 관련된 특성의 리스트들을 함께 분류(group)한 디렉토리들을 포함한다.

[0006] 큰 기록 용량을 가진 예컨대, 수 기가바이트의 기록 매체에 있어서, 하나의 동일한 디렉토리 내에서 기록, 판독 또는 수정될 데이터 파일의 숫자가 커질 경우, 파일 시스템의 인덱싱 시스템의 성능은 떨어진다. 이는 처리되어야 할 매우 많은 수의 발생(occurrence)을 가지는 리스트를 포함하는 파일의 조직 즉, 디렉토리는, 데이터의 숫자가 수 만에 달하는 경우, 파일에의 접속 시간(access time)을 엄청나게 증가시킨다. 이러한 성능 감소는 파일의 단편화(斷片化, fragmentation) 즉, 데이터 파일의 이름을 바꿔야(rename) 할 때나 삭제할 때에 발생하는 디렉토리에 의해 증가된다.

[0007] 리눅스 EXT3 시스템(Linux EXT3 system)을 통해 알려진 다른 조직화 모드는 B-트리(tree) 타입(type)의 내부(internal) 인덱싱 시스템 및, 수많은 데이터 파일을 포함하는 디렉토리 내의 검색을 가속화하기 위해, 각 데이

터 파일에 대하여, 해쉬 함수(hash function)의 결과로부터의 데이터 파일의 이름의 다이제스트(digest)의 계산을 포함하는 파일의 조직 즉, 디렉토리를 제안한다. 이러한 해결책은 동일 디렉토리 내의 수십만의 데이터 파일을 사용할 수 있게 한다. 하지만, 특정 임계값(threshold)을 넘어서면, 해쉬 함수의 한계는, 개별 이름을 가지는 두 개의 데이터 파일에 대해 동일한 다이제스트를 할당하게 되는 충돌(collision)을 더 이상 피할 수 없게 한다. EXT3 파일 시스템에 있어서, 이러한 한계는 리눅스 2.6 커널(Linux 2.6 kernel) 버전에서, 동일 디렉토리 내에 데이터 파일의 숫자가 약 500,000에 이르게 되면서 발생한다.

[0008] 종래 기술에서, 파일 서버를 기술하고, 처리 과정을 간략화하기 위한 목적의 파일 참조(consultation) 방법을 제안하는 미국 특허 US 5742817호가 알려져 있다. 이 문서는 파일 시스템에 의해 자동적으로 귀속(attribute)되는 INODE 식별(identification)의 처리를 통해 파일에의 접속을 가속화하는 것을 제안한다. 이러한 처리는 이러한 식별자(identifier)로부터 파일 시스템 내의 서브-디렉토리(sub-directory)에 대응하는 세 개의 16진(hexadecimal) 데이터 아이템(item)을 추출하는 단계를 포함한다.

[0009] 미국 특허 출원 US 2004/0236761호는 일련의 디렉토리 및 서브-디렉토리내에 파일을 기록하는 시스템을 기술한다. 트리의 마지막 레벨(level)에서, 파일이 기록된 "해쉬 슬롯(HASH SLOT)" 서브-디렉토리를 결정하기 위하여, 상기 파일 이름에 의존하는 다이제스트를 계산하는 단계를 포함하는 처리가 수행된다.

[0010] 달리 말하면, 메모리 공간은 재래식(conventional) 트리 형태로 조직화되고, 트리의 마지막 레벨에서, 처리는 단일 레벨의 세트(set)내에서 서브-디렉토리를 결정한다.

[0011] 이러한 해결책은 기록될 수 있는 파일의 수를 제한한다. 서브-디렉토리의 수는 파일 시스템의 처리 모드에 의해 제한된다. 예컨대, 유닉스(UNIX) 타입의 파일 시스템에 있어서 65,536으로 제한된다.

[0012] 더욱이, 동일한 한계는 상기 서브-디렉토리 내의 파일의 수에도 역시 적용된다.

[0013] 요약하면, 용량이 제한되고, 용량을 완전히 사용하기 위해서는, 이러한 "방탕자" 같은("rake"-like) 단일 레벨의 서브-디렉토리 내에서 파일 시스템의 수행을 제한하는 서브-디렉토리의 수를 최대화할 필요가 있다.

## *발명의 내용*

### *해결하려는 과제*

[0014] 본 발명의 목적은, 파일 시스템에 의해 데이터 파일의 이름 상에서 수행되는 처리 연산에 의해 제한되기 보다는, 메모리 내의 가용 용량에 의해 오로지 제한되는 다수의 데이터 파일을 이용할 수 있는 강력하고 빠른 파일 관리 방법을 제안함으로써, 이러한 결점을 개선하는데 있다. 따라서 본 발명에 따른 해결책은 용량이 큰 기록 메모리를 사용하는데 있어서, 용량에 제한되지 않는다. 유일한 제한은, 운영 시스템에 의해 제공되는 운영 모드 보다는 이러한 메모리들의 물리적 용량이다. 이는, 파일 서버와 같은 어플리케이션(application)에 있어서, 매우 많은 수의 데이터 파일을 관리할 수 있게 한다.

### *과제의 해결 수단*

[0015] 가장 포괄적인 관점에서, 본 발명은,

[0016] 각각 N개의 디렉토리를 가지고 M은 1보다 큰 실수인 M개의 레벨을 가진 디렉토리의 트리를 생성하는 단계를 포함하는 데이터 파일의 베이스(base)를 조직화하기 위한 첫 번째 단계;

[0017] 데이터 파일을 기록하기 위한 단계로서,

[0018] - 해쉬 함수를 기록될 데이터 파일($F_i$)의 식별자에 적용하는 단계;

[0019] - 상기 적용 단계의 결과에 따라 복수의 레벨의 트리 내에서 목적 디렉토리(destination directory)($R_{di}$)의 경로(path)를 결정하는 단계; 및

[0020] - 데이터 파일의 식별자에 의존하는 위치에서, 상기 해쉬 함수에 의해 결정된 상기 디렉토리($R_{di}$)에 데이터 파일을 기록하는 단계를 포함하는 데이터 파일을 기록하기 위한 단계; 및

[0021] 데이터 파일을 판독하기 위한 단계로서,

[0022] - 동일한 해쉬 함수를 판독될 데이터 파일($F_j$)의 식별자에 적용하는 단계;

[0023] - 상기 적용 단계의 결과에 따라 트리 내에서 목표 디렉토리(target directory)($R_{cj}$)의 경로(path)를 결정하는 단계; 및

[0024] - 데이터 파일의 식별자에 의존하는 위치에서, 상기 해쉬 함수에 의해 결정된 상기 디렉토리($R_{cj}$)에서 데이터 파일을 판독하는 단계를 포함하는 데이터 파일을 판독하기 위한 단계를 포함하는 것을 특징으로 하는 파일 관리 방법에 관한 것이다.

[0025] 저장 유닛(storage unit)을 동시에 이용하고 성능(접속 속도 및/또는 메모리 용량)을 향상 시키기 위하여, 일 특정 실시예에 따르면, 본 발명은, 상기 파일 관리 방법에 있어서, 데이터 파일은 Q개 이상의 저장 유닛에 분배되고, 각 저장 유닛은 N개의 디렉토리 중에서 P개의 디렉토리 레벨에 대응되는 것을 특징으로 하는 파일 관리 방법에 관한 것이다.

[0026] 미국 특허 출원 US 2004/0236761호의 내용과 대조적으로, 이러한 해결책은 트리의 각 레벨에서 디렉토리의 수를 제한하는 결과를 낳고, 이는 성능 특히, 2차원을 가진 트리 내의 데이터 파일로의 접속 속도를 향상시키며, 거의 무제한의 수의 데이터 파일이 기록되도록 할 수 있다. 한계는 파일 시스템의 한계에 더 이상 의존하지 않고, 저장 매체의 물리적 용량에만 의존하게 된다.

[0027] 종래 기술의 해결책과 대조적으로, 본 발명은 해쉬 함수를 적용함으로써, 복수의 레벨을 가진 트리 내에 경로를 귀속하고, 주어진 디렉토리 내에 위치를 귀속하지 않는다. 마지막 레벨의 서브-디렉토리 내의 위치는 해쉬 함수에 의해 귀속되지 않고, 데이터 파일의 식별자와 관련된 이름에 의해 귀속된다.

[0028] 유리하게는, N은 16과 같고, 해쉬 함수는 SHA-1 함수이다.

[0029] 파일 관리의 추가적인 문제점은 두 개의 어플리케이션에 의한 동일한 데이터 파일로의 동시 접속의 경우 데이터 완전성(integrity)의 보호에 관한 것이다.

[0030] 운영 시스템의 대다수에 있어서, 파일 시스템의 레벨에서 데이터 파일의 잠금(lock)을 속행할 수 있고, 기록 또는 판독되는 과정에서 주어진 순간에 데이터 파일로의 접속을 방지할 수 있다고 알려져 있다. 이러한 해결책들은 일반적으로 메모리에 기존의 잠금 테이블(locking table)로부터의 파일 시스템에 의해 관리된다. 이러한 해결책의 결점은, 네트워크를 통한 파일로의 접속의 경우, 복수의 원격 컴퓨터에 의해 동일한 파일로의 접속의 경우에 간섭(interference) 또는 에러(error)가 발생할 수 있다는 것이다.

[0031] 미국 특허 US 6850969호는 잠금의 사용을 피하기 위한 특정 해결책을 제시한다. 이 특허는 열린(open) 데이터 파일 내의 수정을 기록하는 어플리케이션에 대한 가능성의 사후적인(posteriori) 제어를 제안하고, 이러한 소위 원자적 연산(atomic operation) 동안 어떠한 어플리케이션도 동일 데이터 파일 상에 개입할 수 없도록 보증한다. 본 발명의 의미 내에서 원자적(atomic)은, 소위 원자적 연산의 모든 태스크(task)의 완료 전에 중단(interruption) 이나 제3의 연산(third-party operation)의 방해(interference)의 가능성 없이 연속적으로 수행되는 개별 태스크의 세트를 의미한다.

[0032] 이 해결책은, 복수의 어플리케이션이 공통의 파일의 세트 상에 동시에 처리 연산을 수행할 경우 일부 연산의 완료를 방해하므로 만족스럽지 못하다. 이 해결책은 확실히 기존의 데이터 파일에 대한 변경을 피할 수 있게 하지만 동시 안전 접속의 손실을 피할 수는 없다. 왜냐하면, 이 솔루션은 각 처리 과정들이 전 처리 과정이 완료된 후에 발생하도록 하기 때문이다. 따라서, 이 해결책은 예컨대, XML 포맷의 데이터와 같은 포함된 파일들 상의 처리 연산에 실질적으로 적응되지 못했고, 다른 경쟁 어플리케이션에 의해 언제든지 사용될 수 있다.

[0033] 이 때문에, 본 발명은 각 데이터 파일이 해더(header) 및 바디(body)를 가지는 것을 특징으로 하는 파일 관리 방법에 관한 것이다. 해더는 파일 관리 시스템 내의 데이터 파일의 A 상태 파라미터(state parameter)를 포함한다. 바디는 파일의 수정 가능한(modifiable) 내용을 포함한다. 본 방법은 새로운 접속을 방해하는 상태에 대한 상기 상태 파라미터 중 하나의 수정을 야기시키는 목표 데이터 파일에 대한 접속 단계를 포함한다.

[0034] 이러한 데이터 파일의 관리는 컴퓨터 운영 시스템의 파일 시스템에 독립적이다.

## *도면의 간단한 설명*

[0035] 본 발명은 첨부된 도면을 참조한, 한정하지 않는 실시예에 관한 아래의 기술을 읽음으로써 보다 잘 이해될 것이다.

도 1은 본 발명에 따른 파일 관리 방법에 의해 사용되는 트리(tree)의 개략적인 도면을 도시한다.

도 2 내지 도 8은 본 파일 관리의 함수의 도면을 도시한다.

도 9는 클라이언트/서버 모드(client/server mode) 에 관한 네트워크를 통한 파일 관리 시스템의 사용에 대한 도면을 도시한다.

## *발명을 실시하기 위한 구체적인 내용*

[0036] 도 1은 본 발명에 따른 파일 관리 방법에 의해 사용되는 트리(tree)의 개략적인 도면을 도시한다.

[0037] 기술된 예에서, 데이터 파일들은 M = 5개의 레벨(1 내지 5)로 서브-디바이드(sub-divide)된 구조에 조직화된다. 각 레벨(1 내지 4)의 각 디렉토리는 마지막 레벨(5)과 분리되어 있고, N=16개의 디렉토리(10 내지 12), (20 내지 25), (30 내지 35), (40 내지 45)를 포함한다. 16개의 디렉토리의 각각의 이름은 0과 f 사이의 16진수 숫자와 관련된다. 상기 구조에 포함된 디렉토리의 총 숫자는 $16+16^{2}+16^{3}...+16^{M}$이다.

[0038] 기술된 예에서, 트리는,

[0039] 데이터 파일을 포함하지 않고 서브-디렉토리만을 포함한, 1 내지 4 레벨의 69,904개의 디렉토리; 및

[0040] 각 디렉토리가 데이터 파일을 저장하기 위한 파일을 포함한, 레벨 5의 1,048,576개의 디렉토리(50 내지 55)를 포함한다.

[0041] 마지막 레벨(5)의 각 디렉토리(50 내지 55)는 데이터 파일의 한정된 숫자 L을 포함할 수 있고, L은 예컨대, 약 1000으로 작게 설정되어, 컴퓨터 운영 시스템의 파일 시스템이 무엇이든, 빠른 접속 시간을 허락한다.

[0042] 데이터 파일에 대한 메모리 위치의 귀속

[0043] 새로운 데이터 파일이 생성되고, 문자열(character string)에 의해 형성된 단일 사용 이름(use name)이 지정될 때, 본 방법은 사용 이름을 나타내는 이러한 문자열에 대해 해쉬 함수($H_i$)를 적용하는 단계로 구성되고, 그 결과는 16진수 형태로 나타나는 값으로 구성될 것이다. 해쉬 함수 SHA-1에 있어서, 16진수 형태의 결과는 0과 F 사이의 40개의 16진수 숫자를 포함할 것이다.

[0044] 본 방법은 상기 정의된 디렉토리 트리의 문맥(context) 내 및 해쉬 함수의 이러한 결과로부터 상기 트리의 디렉토리 경로를 결정하는 단계로 구성된다.

[0045] 해쉬 함수의 결과의 첫 번째 M개의 숫자의 각 순위(rank)는 디렉토리 트리의 레벨에 대응될 것이다.

[0046] 사용 이름 "myfile"의 예를 들면, SHA-1 함수의 어플리케이션은 "b3580ab45cb088ba47ff070aa81c2dae1be56ca2"를 반환한다.

[0047] 5개의 레벨(1 내지 5)에 대해서, 레벨 1의 디렉토리는 SHA-1 값의 첫 문자에 대응하는 이름 "b"를 지니게 될 것이고, 레벨 2의 디렉토리는 두 번째 문자에 대응하는 "3"이 될 것이며, 그리고 이와 같이 계속 될 것이다. 마지막 디렉토리(5)에서의 위치는 사용 이름을 구성하는 문자의 각 바이트(byte)의 2개 숫자의 16진수 표시로 구성될 것이다. 이름 "myfile"에 있어서, 파일의 위치 및 이름은 "6d7966696c65"일 것이다. "6d"는 문자 "m"의 2개 숫자의 16진수 표시에 대응되고, "79"는 문자 "y"의 2개 숫자의 16진수 표시에 대응되고, "66"은 문자 "f"의 2개 숫자의 16진수 표시에 대응되며, 이와 같이 계속 된다.

[0048] 복수의 바이트로 코딩(coding)이 필요한 문자에 있어서, 모든 바이트는 이러한 형태로 성공적으로 표시될 수 있다.

[0049] 데이터 파일의 사용 이름의 유일성(uniqueness) 때문에, 파일의 지정은 해쉬 함수로부터 계산된 위치 및 사용 이름으로부터 유일하게 계산된 데이터 파일 이름의 조합으로부터 비롯되므로 반드시 유일할 것이다.

[0050] 이 해결책은 디렉토리 트리의 레벨 M의 디렉토리 내에 데이터 파일을 같은 확률로 최대 분산으로 분배할 수 있게 한다. 오직 레벨 M만이 데이터 파일을 포함하고, 사이에 들어있는(intermediate) 레벨은 데이터 파일을 포함하지 않으며, 오직 레벨 M의 디렉토리 내에 파일의 분배에 사용된다.

[0051] 레벨 M의 각 디렉토리 내에서, 데이터 파일의 숫자는 예컨대, 약 1,000정도로 알맞게 제한되고, 이는 빠른 접속 시간을 유도한다. 레벨 M의 트리 내에서 생성될 수 있는 데이터 파일의 최대 숫자는 $1,000 \times 16^{M}$이다. 예를 들

어, M=5일 때, 레벨 M의 디렉토리의 트리 내에서 대략 10억개의 데이터 파일을 생성하고, 읽거나 쓸 수 있다.

[0052] 데이터 파일의 생성, 기록, 판독 및 삭제

[0053] 첫 번째 단계는 앞에서 언급한 해쉬 함수를 적용함으로써, 그리고 사용 이름으로부터 파일의 이름을 계산함으로써, 파일의 사용 이름으로부터 트리 내의 접속 경로 및 관련 파일 이름을 결정하는 단계로 구성된다. 이 단계는 파일이 기록된 레벨 M의 디렉토리 및 이 디렉토리 내의 파일의 위치를 결정할 수 있게 한다.

[0054] 동일 데이터 파일에 대한 접속 경쟁

[0055] 데이터 파일에 대한 접속의 안전한 관리를 위해, 데이터 파일은 각각 해더 및 바디를 포함한다. 바디는 직접 접속 가능한 형태로 또는 압축된 또는 암호화된 형태로, 파일의 수정 가능한 내용을 포함하고, 이 바디는 파일 관리 시스템 내의 파일의 상태 파라미터를 포함한 해더를 앞에 둔다.

[0056] 체계적으로, 일련의 임의의 연산(기록, 판독, 수정 또는 삭제) 처리에 의한 파일로의 접속은, 다른 처리 연산에 의한 추가 접속을 방지하는 상태에 대한 해더의 상기 상태 파라미터 중 하나를 수정하는 단계로 구성된 첫 번째 획득(acquisition) 단계로 구성된다.

[0057] 자연히, 파일이 처리 연산에 의해 접속되거나 사용될 때, 다른 처리 연산은 이러한 초기 획득 단계 동안에 금지(prohibition) 또는 대기(waiting)의 통지와 마주치게 될 것이다.

[0058] 다른 처리 연산은 목표 파일이 다시 한번 접속 허용 상태가 될 때까지 또는 미리 결정된 시간 딜레이(delay)가 접속 시도의 중단을 야기시킬 때까지 이러한 초기 획득 단계를 되풀이한다.

[0059] 일단 일련의 연산들이 처리 연산에 의해 완료되면, 마지막 릴리스(release) 단계는 해더의 수정된 파라미터의 초기 상태를 회복한다. 따라서, 다른 처리 연산이 데이터 파일을 접속하도록 한다.

[0060] 도 2 내지 도 8은 본 발명에 의해 사용되는 함수의 구조를 나타낸다.

[0061] 기존의 데이터 파일을 획득하는 함수

[0062] 도 2는 기존의 데이터 파일을 획득하기 위해 필요한 태스크(task)의 연속을 도시한다.

[0063] 함수는 입력(150)으로 이전에 개시된 바와 같이 사용 이름으로부터 계산된 파일의 이름을 받는다.

[0064] 첫 번째 태스크(100)는 파일의 열기(opening) 및 잠금(locking)을 요청함에 의해, 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도하는 단계로 구성된다. 따라서, 이 파일의 닫기(closure) 및 잠금 해제(unlocking) 때까지, 차후의 처리 연산에 의한 이 파일의 열기 및 잠금은 보호된다.

[0065] 그 다음, 함수는 파일의 열기 및 잠금을 입증(verify)하기 위한 테스트(101)를 수행한다. 만약 파일이 열리지 않고 잠기지 않았다면, 테스트(101)의 결과는 부정적이고 함수는 파일이 시도(100) 동안에 존재했는지를 확인하는 두 번째 테스트(102)를 수행한다. 부정적인 응답의 경우, 함수는 파일이 존재하지 않는다는 것을 나타내는 에러(103)를 반환한다.

[0066] 긍정적인 응답의 경우, 테스트(104)는 시도(100)의 수가 임계값(threshold value)을 초과했는지, 또는 타임아웃(timeout) 기간이 경과되었는지를 확인한다. 부정적인 응답의 경우에 시간 딜레이(149) 후의 새로운 시도(100)를 착수한다. 긍정적인 응답의 경우, 함수는 시스템 에러(105)를 반환한다.

[0067] 테스트(101)의 결과가 긍정적이면, 함수는 태스크(106) 동안에 파일 해더의 상태 파라미터 내의 파일의 사용 가능(availability) 상태를 판독한다.

[0068] 그 다음, 함수는 상태 파라미터에 따른 파일의 사용 가능에 대한 테스트(107)를 수행한다.

[0069] 테스트(107)의 결과가 부정적이면, 사용 불가능(unavailability) 상태에 대응하여, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제의 태스크(108)를 수행한다.

[0070] 그 다음, 함수는 시도(100)의 수가 임계값을 초과했는지, 또는 타임아웃 기간이 경과되었는지를 확인하는 테스트(109)를 수행한다. 부정적인 응답의 경우에 시간 딜레이(149) 후의 새로운 시도(100)를 착수한다. 긍정적인 응답의 경우, 함수는 현재의 처리 연산에 의해 파일이 획득된 채로 남아있다는 것을 나타내는 에러(110)를 반환한다.

[0071] 테스트(107)의 결과가 긍정적이면, 함수는 파일 헤더의 상태 파라미터 내의 파일의 이용 불가능 상태를 기록하

는 태스크(111), 파일 획득 식별자(identifier)를 계산하는 태스크(112), 파일 헤더의 상태 파라미터 내에 이 식별자를 기록하는 태스크(113), 및 파일 바디를 판독하는 태스크(114)를 수행한다. 태스크(115)는 파일의 수정 가능한 내용을 추출하기 위한 파일 바디의 처리에 대응하는 것이다. 이 처리 과정은 파일 해더의 상태 파라미터에 따라 실행된다. 예를 들어, 파일의 수정 가능한 내용의 압축 해제(decompression) 또는 암호 해독(deciphering)에 대응하는 것이다. 마지막 태스크(116)는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제로 구성된다.

[0072] 함수는 그 완료 단계(151)에서 출력으로서 데이터 파일의 획득의 확인, 파일 획득 식별자 및 파일의 수정 가능한 내용을 반환한다.

[0073] 새로운 데이터 파일을 획득하는 함수

[0074] 도 3은 새로운 데이터 파일을 획득하기 위해 필요한 태스크의 연속을 도시한다.

[0075] 함수는 입력(250)으로 이전에 개시된 바와 같이 사용 이름으로부터 계산된 파일의 이름을 받는다.

[0076] 첫 번째 태스크(200)는 파일의 생성, 열기 및 잠금을 요청함에 의해, 운영 시스템의 파일 시스템의 레벨에서 새로운 파일의 동시적인 생성, 열기 및 잠금을 시도하는 단계로 구성된다. 따라서, 이 파일의 닫기 및 잠금 해제 때까지, 차후의 처리 연산에 의한 이 파일의 열기 및 잠금은 보호된다.

[0077] 그 다음, 함수는 파일의 생성, 열기 및 잠금을 입증하기 위한 테스트(201)를 수행한다. 만약 파일이 생성되지 않았거나 열리지 않았거나 잠기지 않았다면, 테스트(201)의 결과는 부정적이고 함수는 파일이 시도(200) 동안에 존재했는지를 확인하는 두 번째 테스트(202)를 수행한다.

[0078] 긍정적인 응답의 경우, 함수는 파일이 이미 존재한다는 것을 나타내는 에러(204)를 반환한다.

[0079] 부정적인 응답의 경우, 함수는 시스템 에러를 나타내는 에러(203)를 반환한다.

[0080] 테스트(201)의 결과가 긍정적이면, 함수는 파일의 비어있는 바디를 가진 파일 해더를 생성하고 기록하는 태스크(205), 파일 헤더의 상태 파라미터 내의 파일의 이용 불가능 상태를 기록하는 태스크(206), 파일 획득 식별자를 계산하는 태스크(207), 및 파일 헤더의 상태 파라미터 내에 이 식별자를 기록하는 태스크(208)를 수행한다. 마지막 태스크(209)는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제로 구성된다.

[0081] 상기 함수는 완료 단계(251)에서 출력으로서 데이터 파일 획득의 확인과 파일 획득 식별자를 반환한다.

[0082] 기존의 데이터 파일을 획득하는 것과 함께 파일이 존재하지 않으면 파일을 생성하는 함수

[0083] 도 4는 기존의 데이터 파일을 획득하는 것과 함께 파일이 존재하지 않으면 파일을 생성하기 위해 필요한 태스크의 연속을 도시한다.

[0084] 함수는 입력(350)으로 이전에 개시된 바와 같이 사용 이름으로부터 계산된 파일의 이름을 받는다.

[0085] 첫 번째 태스크(300)는 파일의 열기 및 잠금을 요청함에 의해, 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도하는 단계 및 파일이 존재하지 않으면 파일을 생성하는 단계로 구성된다. 따라서, 이 파일의 닫기 및 잠금 해제 때까지, 차후의 처리 연산에 의한 이 파일의 열기 및 잠금은 보호된다.

[0086] 그 다음, 함수는 파일의 열기 및 잠금을 입증하기 위한 테스트(301)를 수행한다. 만약 파일이 열리지 않고 잠기지 않았다면, 테스트(301)의 결과는 부정적이고 함수는 파일이 시도(300)의 시점에 존재했는지를 확인하는 두 번째 테스트(302)를 수행한다. 부정적인 응답의 경우, 함수는 시스템 에러를 나타내는 에러(303)를 반환한다.

[0087] 긍정적인 응답의 경우, 테스트(304)는 시도(300)의 수가 임계값을 초과했는지, 또는 타임아웃 기간이 경과되었는지를 확인한다. 부정적인 응답의 경우에 시간 딜레이(349) 후의 새로운 시도(300)를 착수한다. 긍정적인 응답의 경우, 함수는 시스템 에러(305)를 반환한다.

[0088] 테스트(301)의 결과가 긍정적인 경우, 함수는 시도(300)의 시점에서 파일이 생성되었는지를 확인하는 테스트(306)를 수행한다. 파일이 생성되지 않았다면, 테스트(306)의 결과는 부정적이고 함수는 태스크(307) 동안에 파일 해더의 상태 파라미터 내의 파일의 사용 가능 상태를 판독한다.

[0089] 그 다음, 함수는 상태 파라미터에 따른 파일의 사용 가능에 대한 테스트(308)를 수행한다.

[0090] 테스트(308)의 결과가 부정적이면, 사용 불가능 상태에 대응하여, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제의 태스크(309)를 수행한다.

[0091] 그 다음, 함수는 시도(300)의 수가 임계값을 초과했는지, 또는 타임아웃 기간이 경과되었는지를 확인하는 테스트(310)를 수행한다. 부정적인 응답의 경우에 시간 딜레이(349) 후의 새로운 시도(300)를 착수한다. 긍정적인 응답의 경우, 함수는 현재의 처리 연산에 의해 파일이 획득된 채로 남아있다는 것을 나타내는 에러(311)를 반환한다.

[0092] 테스트(308)의 결과가 긍정적이면, 함수는 파일 헤더의 상태 파라미터 내의 파일의 이용 불가능 상태를 기록하는 태스크(312), 파일 획득 식별자를 계산하는 태스크(313), 파일 헤더의 상태 파라미터 내에 이 식별자를 기록하는 태스크(314), 및 파일 바디를 판독하는 태스크(315)를 수행한다. 태스크(316)는 파일의 수정 가능한 내용을 추출하기 위한 파일 바디의 처리에 대응하는 것이다. 이 처리 과정은 파일 헤더의 상태 파라미터에 따라 발생한다. 예를 들어, 파일의 수정 가능한 내용의 압축 해제 또는 암호 해독에 대응하는 것이다. 태스크(317)는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제로 구성된다.

[0093] 함수는 그 완료 단계(351)에서 출력으로서 데이터 파일의 획득의 확인, 파일이 생성되지 않았다는 것을 나타내는 통지, 파일 획득 식별자 및 파일의 수정 가능한 내용을 반환한다.

[0094] 테스트(306)의 결과가 긍정적이면, 함수는 파일의 비어있는 바디를 가진 파일 해더를 생성하고 기록하는 태스크(318), 파일 헤더의 상태 파라미터 내의 파일의 이용 불가능 상태를 기록하는 태스크(319), 파일 획득 식별자를 계산하는 태스크(320), 및 파일 헤더의 상태 파라미터 내에 이 식별자를 기록하는 태스크(321)를 수행한다. 마지막 태스크(322)는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제로 구성된다.

[0095] 함수는 그 완료 단계(352)에서 출력으로서 데이터 파일의 획득의 확인, 파일이 생성되었다는 것을 나타내는 통지 및 파일 획득 식별자를 반환한다.

[0096] 데이터 수정 없이 획득된 데이터 파일을 릴리스하는 함수

[0097] 도 5는 데이터 수정 없이 획득된 데이터 파일을 릴리스(release)하기 위해 필요한 태스크의 연속을 도시한다.

[0098] 함수는 입력(450)으로 이전에 개시된 바와 같이 사용 이름으로부터 계산된 파일의 이름 및 데이터 파일의 이전의 획득 동안에 반환된 획득 식별자를 받는다.

[0099] 첫 번째 태스크(400)는 파일의 열기 및 잠금을 요청함에 의해, 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도하는 단계로 구성된다. 따라서, 이 파일의 닫기 및 잠금 해제 때까지, 차후의 처리 연산에 의한 이 파일의 열기 및 잠금은 보호된다.

[0100] 그 다음, 함수는 파일의 열기 및 잠금을 입증하기 위한 테스트(401)를 수행한다. 만약 파일이 열리지 않고 잠기지 않았다면, 테스트(401)의 결과는 부정적이고 함수는 파일이 시도(400) 동안에 존재했는지를 확인하는 두 번째 테스트(402)를 수행한다. 부정적인 응답의 경우, 함수는 파일이 존재하지 않는다는 것을 나타내는 에러(403)를 반환한다.

[0101] 긍정적인 응답의 경우, 테스트(404)는 시도(400)의 수가 임계값을 초과했는지, 또는 타임아웃 기간이 경과되었는지를 확인한다. 부정적인 응답의 경우에 시간 딜레이(449) 후의 새로운 시도(400)를 착수한다. 긍정적인 응답의 경우, 함수는 시스템 에러(405)를 반환한다.

[0102] 테스트(401)의 결과가 긍정적이면, 함수는 태스크(406) 동안에 파일 해더의 상태 파라미터 내의 파일의 사용 가능 상태를 판독한다.

[0103] 그 다음, 함수는 상태 파라미터에 따른 파일의 사용 가능에 대한 테스트(407)를 수행한다.

[0104] 테스트(407)의 결과가 긍정적이면, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제의 태스크(408)를 수행하고 파일이 획득되지 않았다는 것을 나타내는 에러(409)를 반환한다.

[0105] 테스트(407)의 결과가 부정적이면, 사용 불가능 상태에 대응하여, 함수는 태스크(410) 동안에 파일 해더 내의 획득 식별자를 판독한다.

[0106] 그 다음, 함수는 이 식별자가 입력으로 제출된 식별자와 상이한지를 확인하는 테스트(411)를 수행한다. 긍정적인 응답의 경우, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제의 태스크(412)를 수행하고 입력으로서 제출된 획득 식별자가 유효하지 않다는 것을 나타내는 에러(413)를 반환한다.

[0107] 부정적인 응답의 경우, 함수는 파일 해더의 상태 파라미터 내의 획득 식별자를 삭제하는 태스크(414) 및 파일 해더 내의 사용 가능 상태를 기록하는 태스크(415)를 수행한다. 마지막 태스크(416)는 운영 시스템의 파일 시스

템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제로 구성된다.

[0108] 함수는 그 완료 단계(451)에서 출력으로서 데이터 파일의 릴리스의 확인을 반환한다.

[0109] 데이터 수정과 함께 획득된 데이터 파일을 릴리스하는 함수

[0110] 도 6은 데이터 수정과 함께 획득된 데이터 파일을 릴리스하기 위해 필요한 태스크의 연속을 도시한다.

[0111] 함수는 입력(550)으로 이전에 개시된 바와 같이 사용 이름으로부터 계산된 파일의 이름, 데이터 파일의 이전의 획득 동안에 반환된 획득 식별자 및 파일의 수정 가능한 내용을 받는다.

[0112] 첫 번째 태스크(500)는 파일의 열기 및 잠금을 요청함에 의해, 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도하는 단계로 구성된다. 따라서, 이 파일의 닫기 및 잠금 해제 때까지, 차후의 처리 연산에 의한 이 파일의 열기 및 잠금은 보호된다.

[0113] 그 다음, 함수는 파일의 열기 및 잠금을 입증하기 위한 테스트(501)를 수행한다. 만약 파일이 열리지 않고 잠기지 않았다면, 테스트(501)의 결과는 부정적이고 함수는 파일이 시도(500) 동안에 존재했는지를 확인하는 두 번째 테스트(502)를 수행한다. 부정적인 응답의 경우, 함수는 파일이 존재하지 않는다는 것을 나타내는 에러(503)를 반환한다.

[0114] 긍정적인 응답의 경우, 테스트(504)는 시도(500)의 수가 임계값을 초과했는지, 또는 타임아웃 기간이 경과되었는지를 확인한다. 부정적인 응답의 경우에 시간 딜레이(549) 후의 새로운 시도(500)를 착수한다. 긍정적인 응답의 경우, 함수는 시스템 에러(505)를 반환한다.

[0115] 테스트(501)의 결과가 긍정적이면, 함수는 태스크(506) 동안에 파일 해더의 상태 파라미터 내의 파일의 사용 가능 상태를 판독한다.

[0116] 그 다음, 함수는 상태 파라미터에 따른 파일의 사용 가능에 대한 테스트(507)를 수행한다.

[0117] 테스트(507)의 결과가 긍정적이면, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제를 하는 태스크(508)를 수행하고 파일이 획득되지 않았다는 것을 나타내는 에러(509)를 반환한다.

[0118] 테스트(507)의 결과가 부정적이면, 사용 불가능 상태에 대응하여, 함수는 태스크(510) 동안에 파일 해더 내의 획득 식별자를 판독한다.

[0119] 그 다음, 함수는 이 식별자가 입력으로 제출된 식별자와 상이한지를 확인하는 테스트(511)를 수행한다. 긍정적인 응답의 경우, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제의 태스크(512)를 수행하고 입력으로서 제출된 획득 식별자가 유효하지 않다는 것을 나타내는 에러(513)를 반환한다.

[0120] 부정적인 응답의 경우, 함수는 파일 해더의 상태 파라미터 내의 획득 식별자를 삭제하는 태스크(514), 파일 해더 내의 사용 가능 상태를 기록하는 태스크(515) 및 파일 바디 안에 파일의 수정 가능한 내용을 삽입하기 위한 파일 바디의 처리에 대응하는 태스크(516)를 수행한다. 이러한 처리 과정은 파일 해더의 상태 파라미터에 따라 발생한다. 예를 들어, 파일의 수정 가능한 내용의 압축(compression) 또는 암호화(enciphering)에 대응하는 것이다. 그 다음, 함수는 파일 보디를 기록하는 태스크(517)를 수행하고, 마지막 태스크(518)는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제로 구성된다.

[0121] 함수는 그 완료 단계(551)에서 출력으로서 데이터 파일의 릴리스 및 파일 내용의 수정의 확인을 반환한다.

[0122] 파일의 삭제와 함께 획득된 데이터 파일을 릴리스하는 함수

[0123] 도 7은 파일의 삭제와 함께 획득된 데이터 파일을 릴리스하기 위해 필요한 태스크의 연속을 도시한다.

[0124] 함수는 입력(650)으로 이전에 개시된 바와 같이 사용 이름으로부터 계산된 파일의 이름 및 데이터 파일의 이전의 획득 동안에 반환된 획득 식별자를 받는다.

[0125] 첫 번째 태스크(600)는 파일의 열기 및 잠금을 요청함에 의해, 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도하는 단계로 구성된다. 따라서, 이 파일의 닫기 및 잠금 해제 때까지, 차후의 처리 연산에 의한 이 파일의 열기 및 잠금은 보호된다.

[0126] 그 다음, 함수는 파일의 열기 및 잠금을 입증하기 위한 테스트(601)를 수행한다. 만약 파일이 열리지 않고 잠기지 않았다면, 테스트(601)의 결과는 부정적이고 함수는 파일이 시도(600) 동안에 존재했는지를 확인하는 두 번째 테스트(602)를 수행한다. 부정적인 응답의 경우, 함수는 파일이 존재하지 않는다는 것을 나타내는 에러(60

3)를 반환한다.

[0127] 긍정적인 응답의 경우, 테스트(604)는 시도(600)의 수가 임계값을 초과했는지, 또는 타임아웃 기간이 경과되었는지를 확인한다. 부정적인 응답의 경우에 시간 딜레이(649) 후의 새로운 시도(600)를 착수한다. 긍정적인 응답의 경우, 함수는 시스템 에러(605)를 반환한다.

[0128] 테스트(601)의 결과가 긍정적이면, 함수는 태스크(606) 동안에 파일 해더의 상태 파라미터 내의 파일의 사용 가능 상태를 판독한다.

[0129] 그 다음, 함수는 상태 파라미터에 따른 파일의 사용 가능에 대한 테스트(607)를 수행한다.

[0130] 테스트(607)의 결과가 긍정적이면, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제를 하는 태스크(608)를 수행하고 파일이 획득되지 않았다는 것을 나타내는 에러(609)를 반환한다.

[0131] 테스트(607)의 결과가 부정적이면, 사용 불가능 상태에 대응하여, 함수는 태스크(610) 동안에 파일 해더 내의 획득 식별자를 판독한다.

[0132] 그 다음, 함수는 이 식별자가 입력으로 제출된 식별자와 상이한지를 확인하는 테스트(611)를 수행한다. 긍정적인 응답의 경우, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제의 태스크(612)를 수행하고 입력으로서 제출된 획득 식별자가 유효하지 않다는 것을 나타내는 에러(613)를 반환한다.

[0133] 부정적인 응답의 경우, 함수는 파일 해더의 상태 파라미터 내의 획득 식별자를 삭제하는 태스크(614) 및 파일 해더 내의 사용 가능 상태를 기록하는 태스크(615)를 수행한다. 마지막 태스크(616)는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제로 구성된다.

[0134] 함수는 그 완료 단계(651)에서 출력으로서 데이터 파일의 릴리스 및 그 파일의 삭제의 확인을 반환한다.

[0135] 파일의 획득 없이 기존의 데이터 파일을 단순 판독하는 함수

[0136] 도 8은 파일의 획득 없이 기존의 데이터 파일을 단순 판독하기 위해 필요한 태스크의 연속을 도시한다.

[0137] 함수는 입력(750)으로 이전에 개시된 바와 같이 사용 이름으로부터 계산된 파일의 이름을 받는다.

[0138] 첫 번째 태스크(700)는 파일의 열기 및 잠금을 요청함에 의해, 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도하는 단계로 구성된다. 따라서, 이 파일의 닫기 및 잠금 해제 때까지, 차후의 처리 연산에 의한 이 파일의 열기 및 잠금은 보호된다.

[0139] 그 다음, 함수는 파일의 열기 및 잠금을 입증하기 위한 테스트(701)를 수행한다. 만약 파일이 열리지 않고 잠기지 않았다면, 테스트(701)의 결과는 부정적이고 함수는 파일이 시도(700) 동안에 존재했는지를 확인하는 두 번째 테스트(702)를 수행한다. 부정적인 응답의 경우, 함수는 파일이 존재하지 않는다는 것을 나타내는 에러(703)를 반환한다.

[0140] 긍정적인 응답의 경우, 테스트(704)는 시도(700)의 수가 임계값을 초과했는지, 또는 타임아웃 기간이 경과되었는지를 확인한다. 부정적인 응답의 경우에 시간 딜레이(749) 후의 새로운 시도(700)를 착수한다. 긍정적인 응답의 경우, 함수는 시스템 에러(705)를 반환한다.

[0141] 테스트(701)의 결과가 긍정적이면, 함수는 태스크(706) 동안에 파일 해더의 상태 파라미터 내의 파일의 사용 가능 상태를 판독한다.

[0142] 그 다음, 함수는 상태 파라미터에 따른 파일의 사용 가능에 대한 테스트(707)를 수행한다.

[0143] 테스트(707)의 결과가 부정적이면, 사용 불가능 상태에 대응하여, 함수는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제의 태스크(708)를 수행한다.

[0144] 그 다음, 함수는 시도(700)의 수가 임계값을 초과했는지, 또는 타임아웃 기간이 경과되었는지를 확인하는 테스트(709)를 수행한다. 부정적인 응답의 경우에 시간 딜레이(749) 후의 새로운 시도(700)를 착수한다. 긍정적인 응답의 경우, 함수는 현재의 처리 연산에 의해 파일이 획득된 채로 남아있다는 것을 나타내는 에러(710)를 반환한다.

[0145] 테스트(707)의 결과가 긍정적이면, 함수는 파일 헤더의 상태 파라미터 내의 파일의 이용 불가능 상태를 기록하는 태스크(711), 및 파일의 수정 가능한 내용을 추출하기 위한 파일 바디를 처리하는 태스크(712)를 수행한다. 이 처리 과정은 파일 해더의 상태 파라미터에 따라 발생한다. 예를 들어, 파일의 수정 가능한 내용의 압축 해제

또는 암호 해독에 대응하는 것이다. 마지막 태스크(713)는 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제로 구성된다.

[0146] 함수는 그 완료 단계(751)에서 출력으로서 데이터 파일의 단순 판독의 확인 및 파일의 수정 가능한 내용을 반환한다.

[0147] 자세하게,

[0148] - 파일의 사용 가능 상태를 포함하는 데이터 파일의 해더의 상태 파라미터의 판독,

[0149] - 처리 연산에 의해 릴리스되지 않은 데이터 파일의 강제적인 릴리스,

[0150] - 트리의 서브셋(subset)에 의해 트리 내에 포함된 데이터 파일의 리스트와 같은 다른 함수들은 트리의 운영을 유용하게 한다.

[0151] 클라이언트-서버 모드

[0152] 도 9는 특별하고 바람직한 어플리케이션의 경우에 관한 것이다. 도면은 클라이언트-서버 모드(client-server mode)에 관한 네트워크를 통한 본 발명에 따른 파일 관리 시스템의 사용에 필요한 태스크의 연속을 도시한 것이다.

[0153] 이 경우, 파일 관리 시스템은 서버 소프트웨어(801)을 포함하는 서버 컴퓨터(800) 상에 설치된다. 서버 컴퓨터는 각각 클라이언트 소프트웨어(805 내지 807)을 포함하는 복수의 클라이언트 컴퓨터(802 내지 804)에 의해 접속 가능하다. 클라이언트 소프트웨어 및 서버 소프트웨어 간의 주고받음(exchange)은, 예컨대, HTTP 또는 바람직하게는 HTTPS 타입의 안전한 프로토콜(protocol)과 같은 알려진 타입의 네트워크 송신 프로토콜을 사용한다.

[0154] 이 경우, 본 발명은 "파일 획득" 또는 "파일 릴리스" 타입의 명령에 의해 본 발명에 따른 파일 관리 시스템의 함수를 호출하기 위해 어플리케이션 프로토콜(808)을 사용한다. 파일은 서버 컴퓨터의 레벨에서 복수의 서브셋(809 내지 811)으로 분배된다. 본 발명에 따른 트리에 대응한 각 서브셋은 "테이블(table)"이라 한다. 데이터 파일의 사용 이름은 프로토콜 내에서 "키(key)"에 의해 지정된다.

[0155] 첫 번째 태스크(812)는 클라이언트 소프트웨어에 의해 클라이언트 컴퓨터 중 하나 상에서 수행된다. 태스크(812)는 테이블 및 키에 관련된 획득, 릴리스 또는 단순 판독 함수를 요청하는 것으로 구성된다.

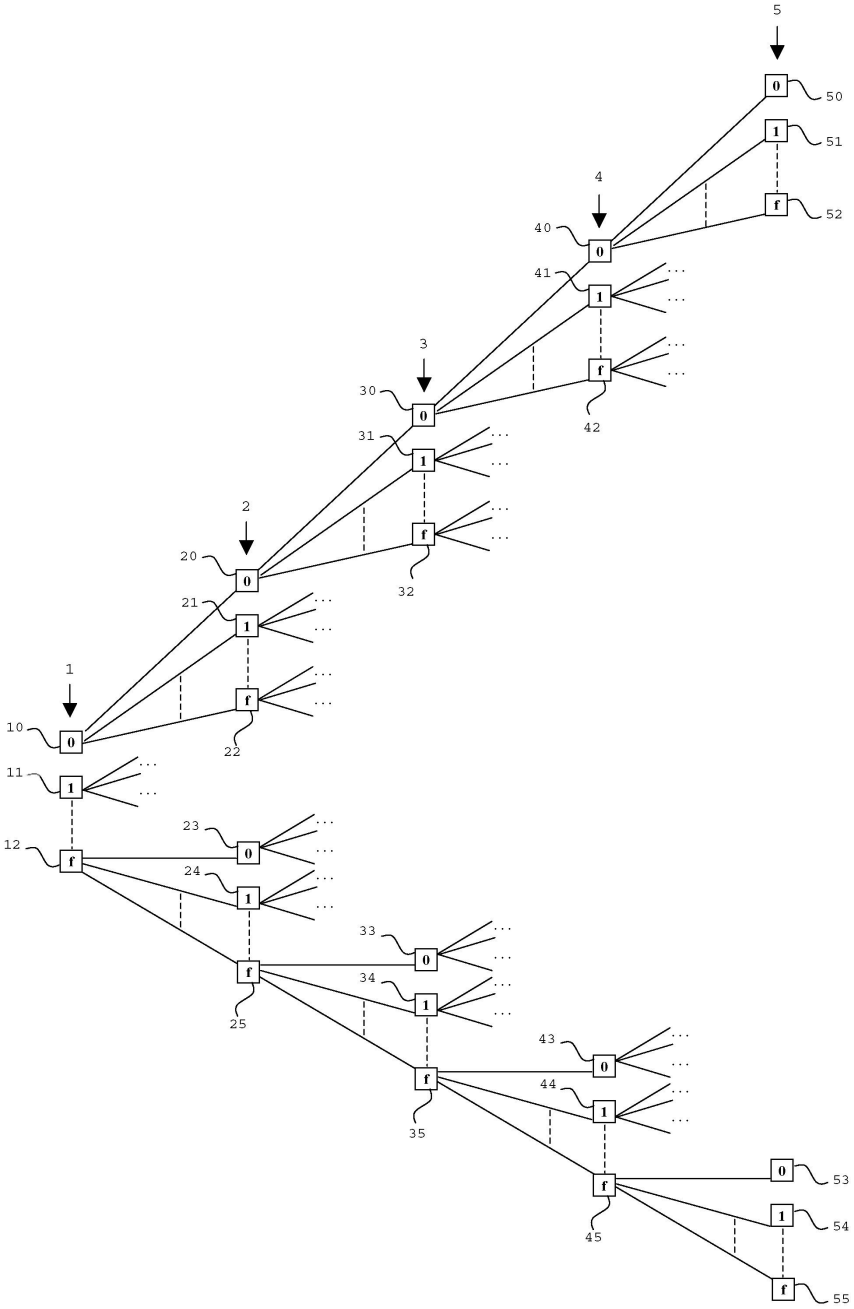
[0156] 두 번째 태스크(813)는 서버 컴퓨터 상에서 서버 소프트웨어에 의해 수행된다. 태스크(813)는 클라이언트 소프트웨어에 의해 요청된 함수를 수행하는 것으로 구성된다.

[0157] 세 번째 태스크(814)는 서버 컴퓨터 상에서 서버 소프트웨어에 의해 수행된다. 태스크(814)는 요청된 함수의 결과를 클라이언트 소프트웨어에 반환하는 것으로 구성된다. 선택적으로, 시스템은 데이터의 저장 용량 및 부담을 분배하기 위해 복수의 서버를 포함한다. 이 경우, 클라이언트 소프트웨어(805 내지 807)는 테이블 및 키에 따른 서버를 선택하기 위한 법칙(rule)을 포함한다.

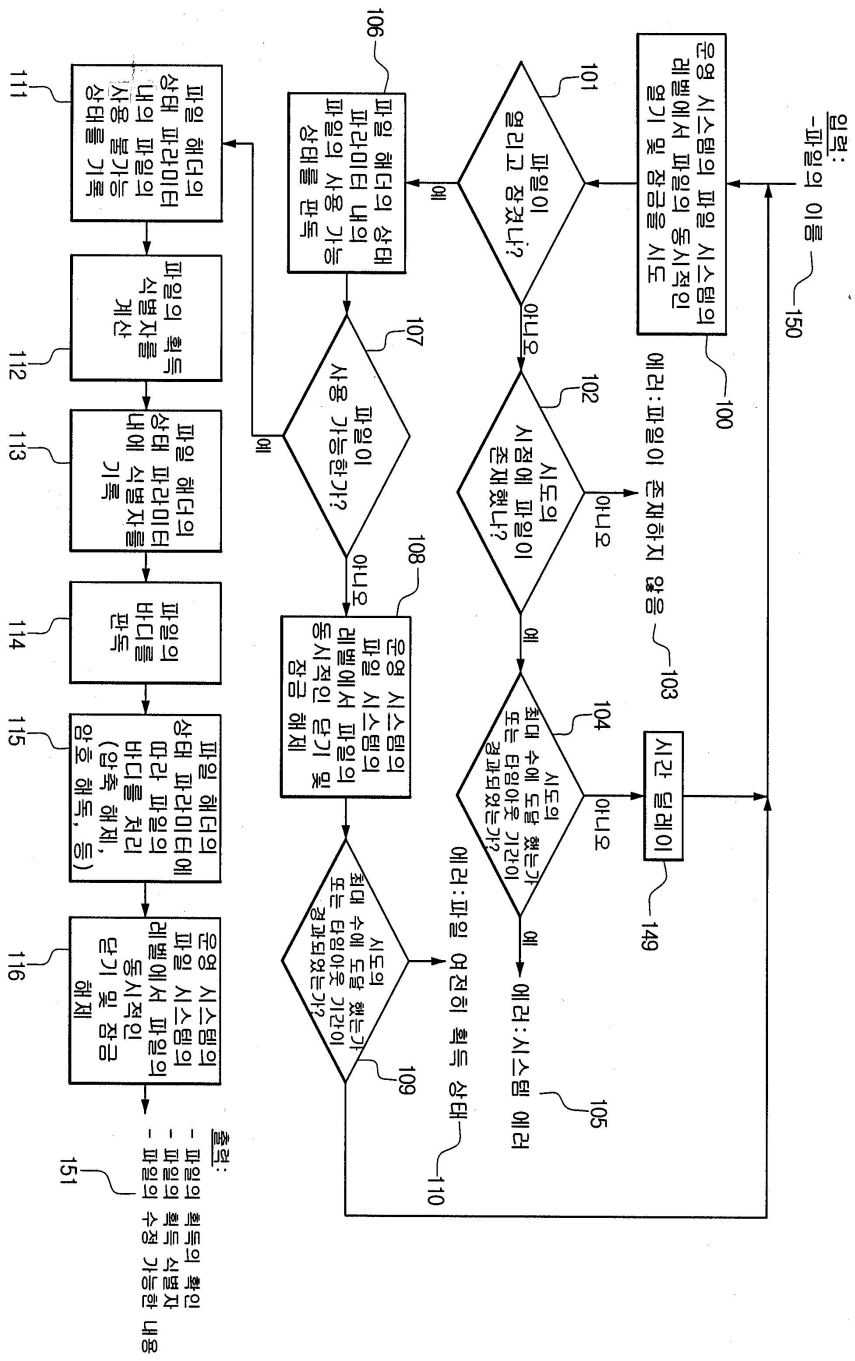
[0158] 서버 소프트웨어(801)는 또한 백업(backup) 컴퓨터 상에 트리의 완전한 사본(copy)을 만들지 않고, 백업 컴퓨터 상에 서버 컴퓨터의 트리의 증분 복구(incremental reconstruction)를 하도록 수행되는 연산의 로깅(logging)의 기능(functionality)을 포함할 수 있다.

## 도면

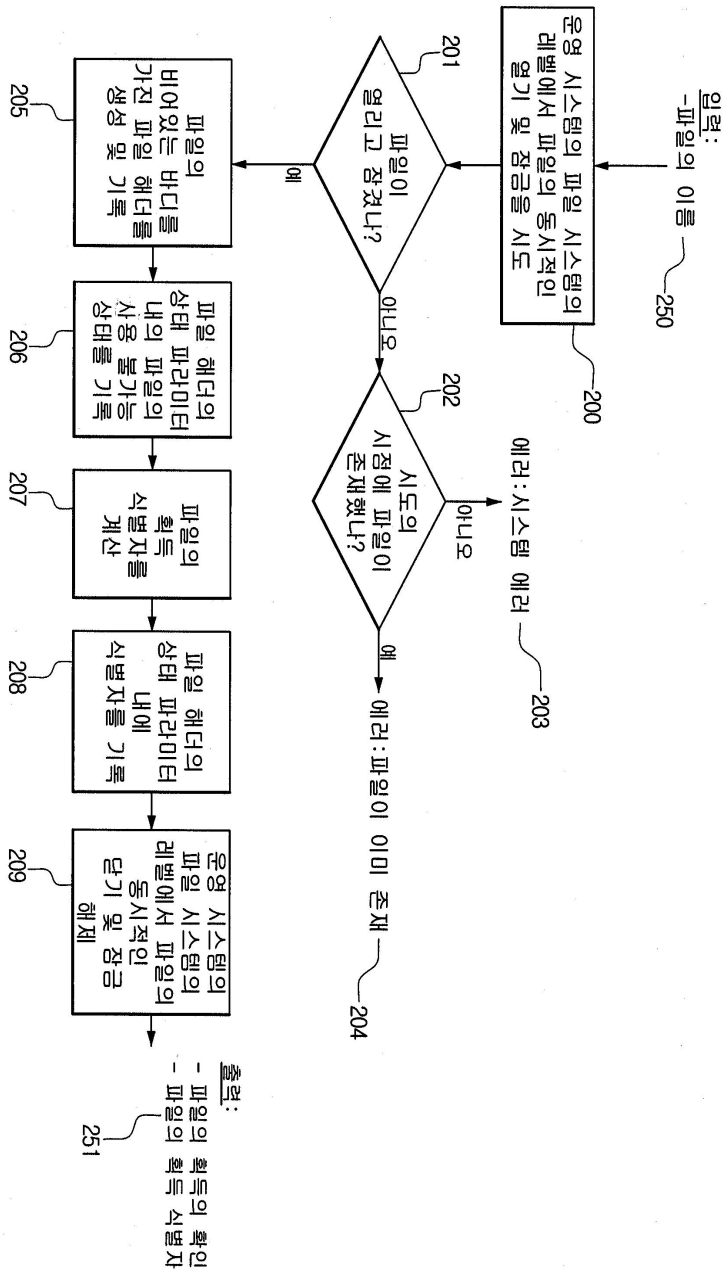
### 도면1


1
2
3
4
5
10
11
12
20
21
22
23
24
25
30
31
32
33
34
35
40
41
42
43
44
45
50
51
52
53
54
55
0
1
f

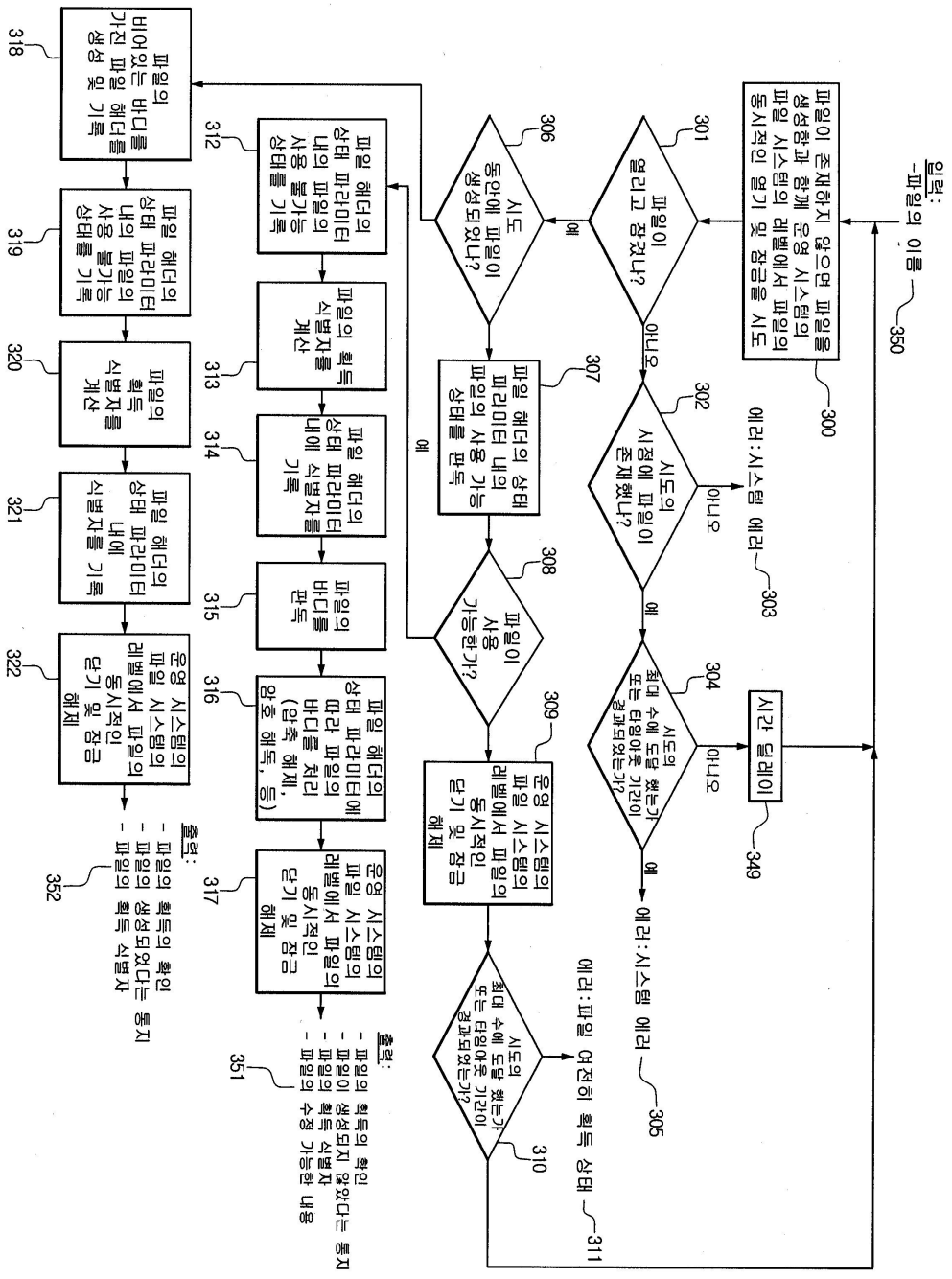
도면2


입력:
-파일의 이름
150
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도
100
파일이 열리고 잠겼나?
101
예
아니오
시도의 시점에 파일이 존재했나?
102
아니오
에러:파일이 존재하지 않음
103
예
시도의 최대 수에 도달 했는가 또는 타임아웃 기간이 경과되었는가?
104
아니오
시간 딜레이
149
예
에러:시스템 에러
105
에러:파일 여전히 획득 상태
110
파일 헤더의 상태 파라미터 내의 파일의 사용 가능한 상태를 판독
106
예
파일이 사용 가능한가?
107
아니오
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
108
시도의 최대 수에 도달 했는가 또는 타임아웃 기간이 경과되었는가?
109
파일 헤더의 상태 파라미터 내의 파일의 사용 불가능 상태를 기록
111
파일의 획득 식별자를 계산
112
파일 헤더의 상태 파라미터 내에 식별자를 기록
113
파일의 바디를 판독
114
파일 헤더의 상태 파라미터에 따라 파일의 바디를 처리 (압축 해독, 등)
115
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
116
출력:
- 파일의 획득의 확인
- 파일의 획득 식별자
- 파일의 수정 가능한 내용
151

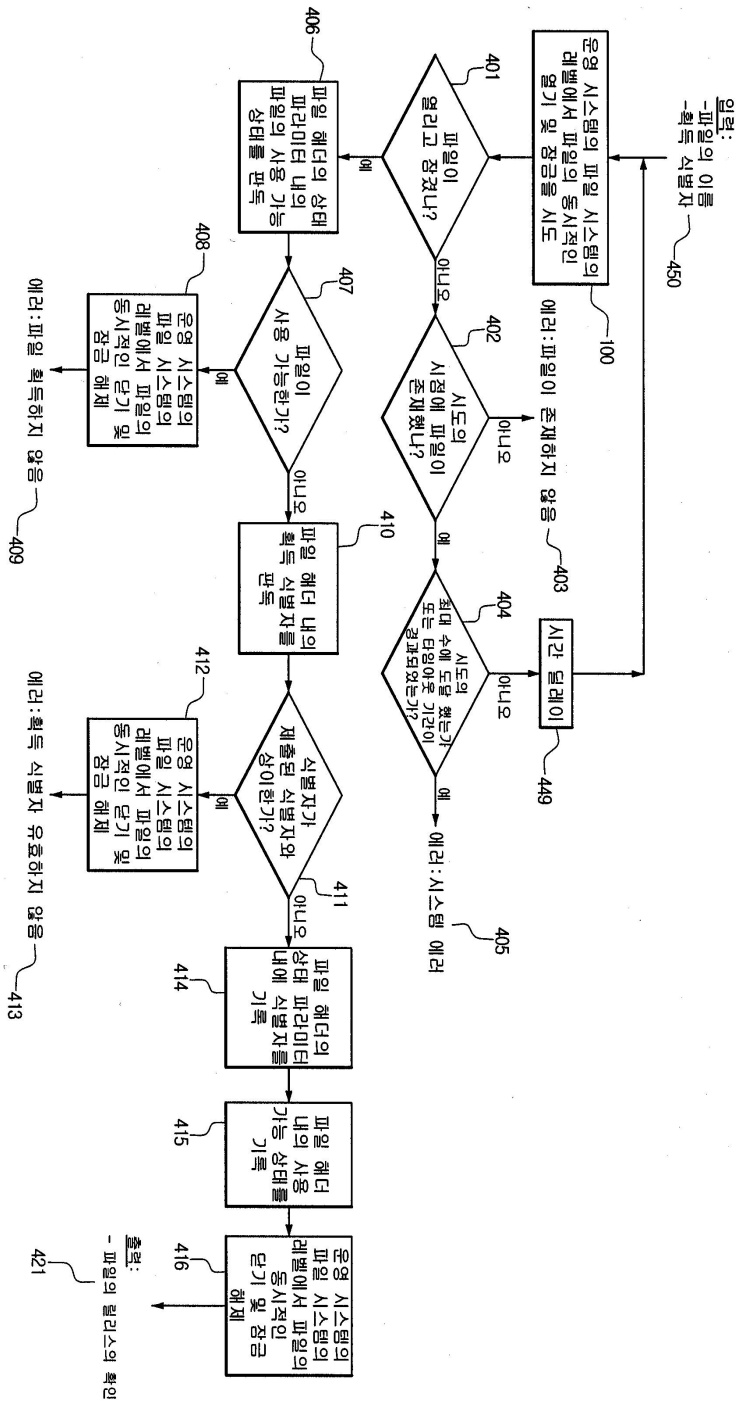
도면3


입력:
-파일의 이름
250
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도
200
201
파일이 열리고 잠겼나?
예
아니오
202
시도의 시점에 파일이 존재했나?
예
아니오
에러: 파일이 이미 존재
204
에러: 시스템 에러
203
205
파일의 비어있는 바디를 가진 파일 헤더를 생성 및 기록
206
파일 헤더의 상태 파라미터 내의 파일의 사용 불가능 상태를 기록
207
파일의 획득 식별자를 계산
208
파일 헤더의 상태 파라미터 내에 식별자를 기록
209
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
출력:
- 파일의 획득의 확인
- 파일의 획득 식별자
251

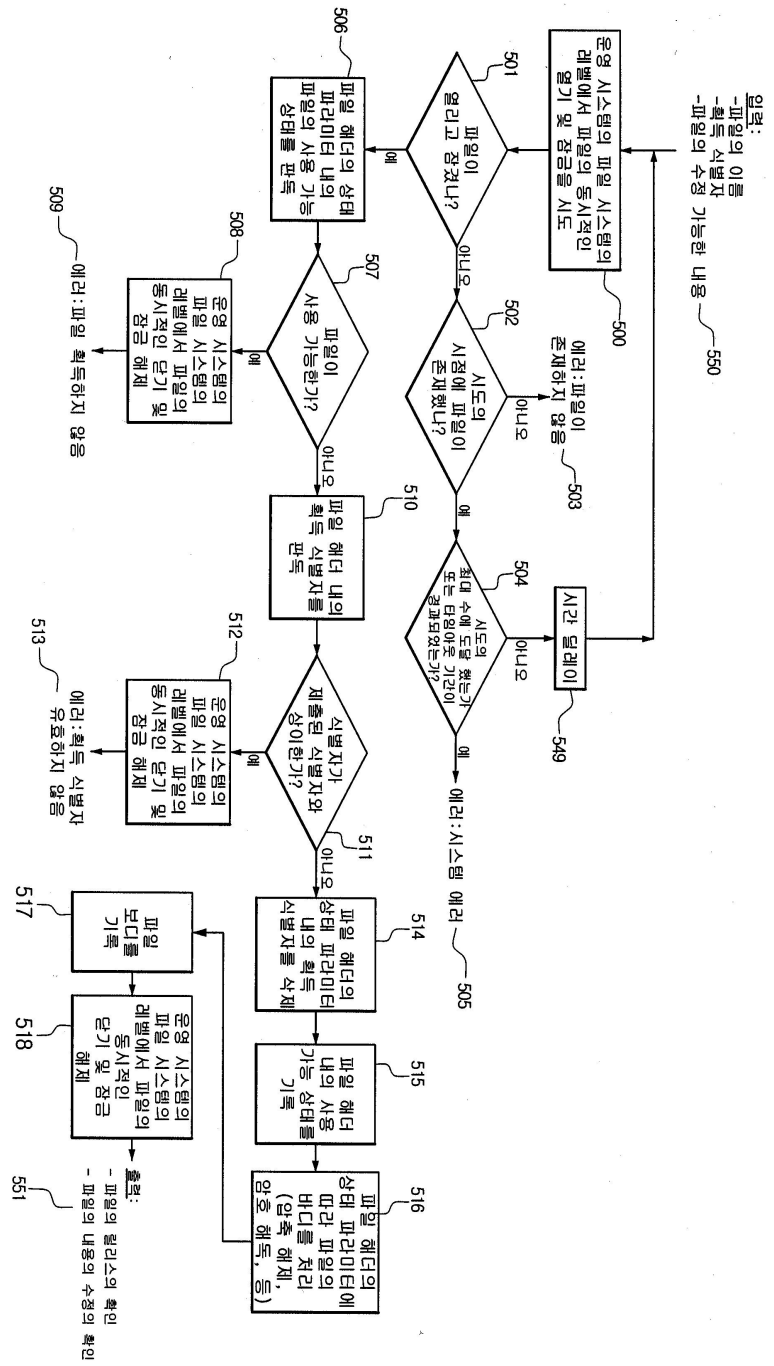
도면4


입력:
-파일의 이름
350
파일이 존재하지 않으면 파일을 생성함과 함께 운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도
300
파일이 열리고 잠겼나?
301
아니오
에러:시스템 에러
303
예
시도의 동안에 파일이 생성되었나?
306
예
파일 헤더의 상태 파라미터 내의 파일의 사용 불가능 상태를 기록
312
파일의 획득 식별자를 계산
313
파일 헤더의 상태 파라미터 내에 식별자를 기록
314
파일의 바디를 판독
315
파일 헤더의 상태 파라미터에 따라 파일의 바디를 처리 (암호 해독, 등)
316
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
317
출력:
- 파일의 획득의 확인
- 파일이 생성되었다는 통지
- 파일의 획득 식별자
- 파일의 수정 기능한 내용
351
파일의 비어있는 바디를 가진 파일 헤더를 생성 및 기록
318
파일 헤더의 상태 파라미터 내의 파일의 사용 불가능 상태를 기록
319
파일의 획득 식별자를 계산
320
파일 헤더의 상태 파라미터 내에 식별자를 기록
321
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
322
출력:
- 파일의 획득의 확인
- 파일이 생성되었다는 통지
- 파일의 획득 식별자
352
파일 헤더의 상태 파라미터 내의 파일의 사용 가능 상태를 판독
307
파일이 사용 가능한가?
308
예
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
309
시도의 동안에 파일이 존재했나?
302
아니오
에러:시스템 에러
303
예
최대 수에 도달 했는가 또는 타임아웃 기간이 경과되었는가?
304
아니오
시간 딜레이
349
예
에러:시스템 에러
305
최대 수에 도달 했는가 또는 타임아웃 기간이 경과되었는가?
310
에러:파일 여전히 획득 상태
311

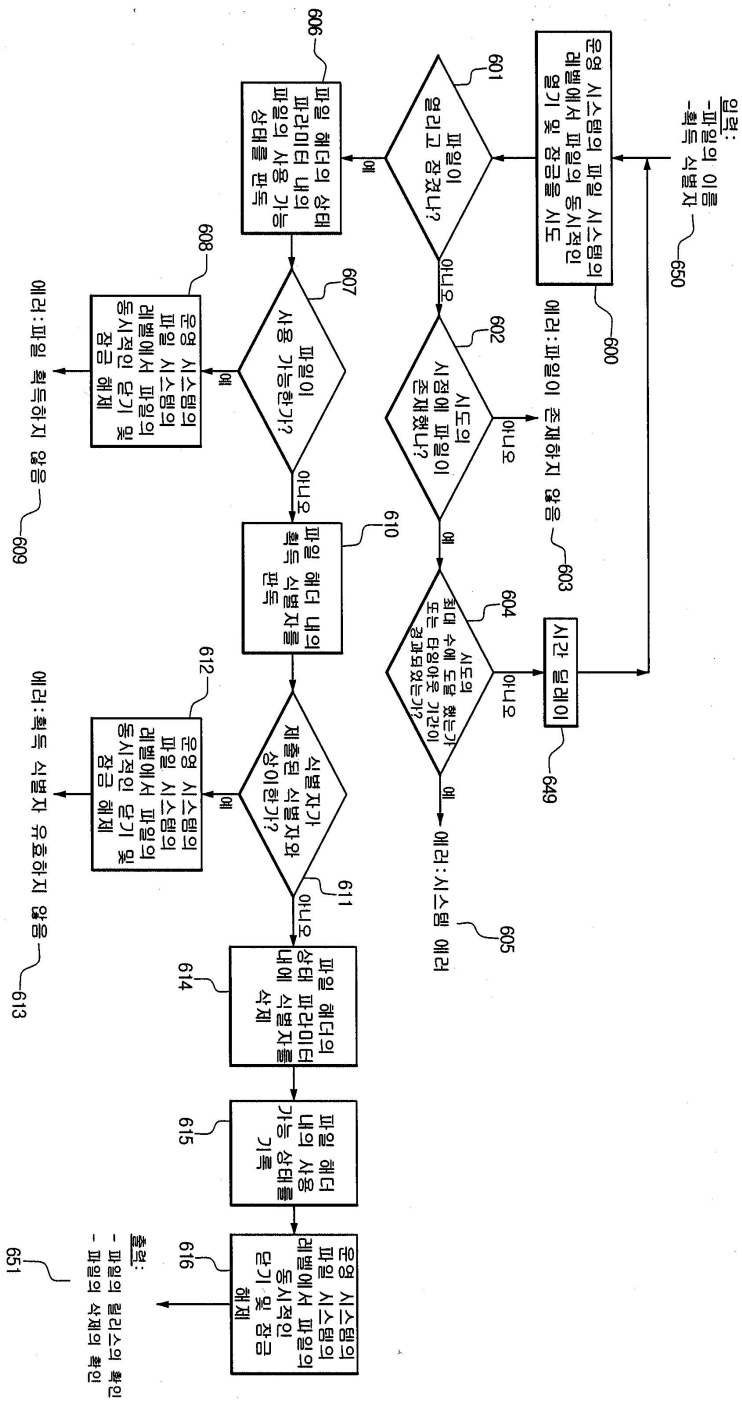
도면5


입력:
-파일의 이름
-획득 식별자
450
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도
100
401
파일이 열리고 잠겼나?
예
406
파일 헤더의 상태 파라미터 내의 파일의 사용 가능 상태를 판독
407
파일이 사용 가능한가?
예
408
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
409
에러: 파일 획득하지 않음
아니오
410
파일 헤더 내의 획득 식별자를 판독
411
식별자가 제출된 식별자와 같은가?
예
412
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
413
에러: 획득 식별자 유효하지 않음
아니오
414
파일 헤더의 상태 파라미터 내에 식별자를 기록
415
파일 헤더 내의 사용 가능 상태를 기록
416
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
421
출력:
- 파일의 클라이언트의 확인
아니오
402
시도의 시점에 파일이 존재했나?
예
404
시도의 최대 수에 도달 했는가 또는 타임아웃 기간이 경과되었는가?
예
405
에러: 시스템 에러
아니오
403
에러: 파일이 존재하지 않음
아니오
449
시간 딜레이

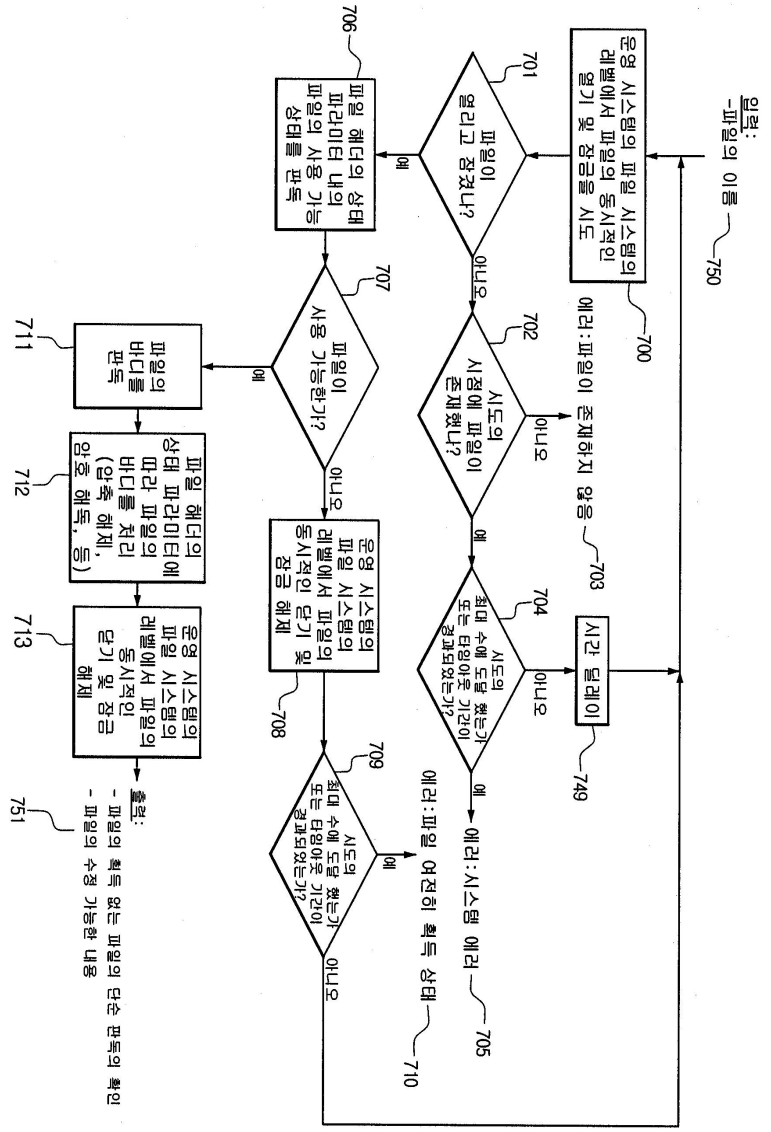
도면6


입력:
-파일의 이름
-획득 식별자
-파일의 수정 기능한 내용
550
운영 시스템의 파일 시스템의 레벨에서 파일을 열기 및 잠그는 시도
500
파일이 열리고 있었나?
501
예
아니오
시도의 시점에 파일이 존재했나?
502
아니오
에러: 파일이 존재하지 않음
503
예
최대 수에 도달 했는지 또는 타임아웃 경과되었는가?
504
아니오
시간 딜레이
549
예
에러: 시스템 에러
505
파일 헤더의 상태 파라미터 내의 파일의 사용 가능
506
예
파일이 사용 가능한가?
507
아니오
운영 시스템의 파일 시스템의 레벨에서 파일을 열기 및 잠금 해제
508
에러: 파일 획득하지 않음
509
파일 헤더 내의 획득 식별자를 판독
510
식별자가 제출된 식별자와 상이한가?
511
예
운영 시스템의 파일 시스템의 레벨에서 파일의 열기 및 잠금 해제
512
에러: 획득 식별자 유효하지 않음
513
아니오
파일 헤더의 상태 파라미터 내의 식별자를 삭제
514
파일 헤더 내의 사용 가능 상태를 기록
515
파일 헤더의 상태 파라미터에 따라 파일의 바디를 처리 (암호 해독, 등)
516
파일 바디를 기록
517
운영 시스템의 파일 시스템의 레벨에서 파일을 열기 및 잠금 해제
518
출력:
- 파일의 릴리스의 확인
- 파일의 내용의 수정의 확인
551

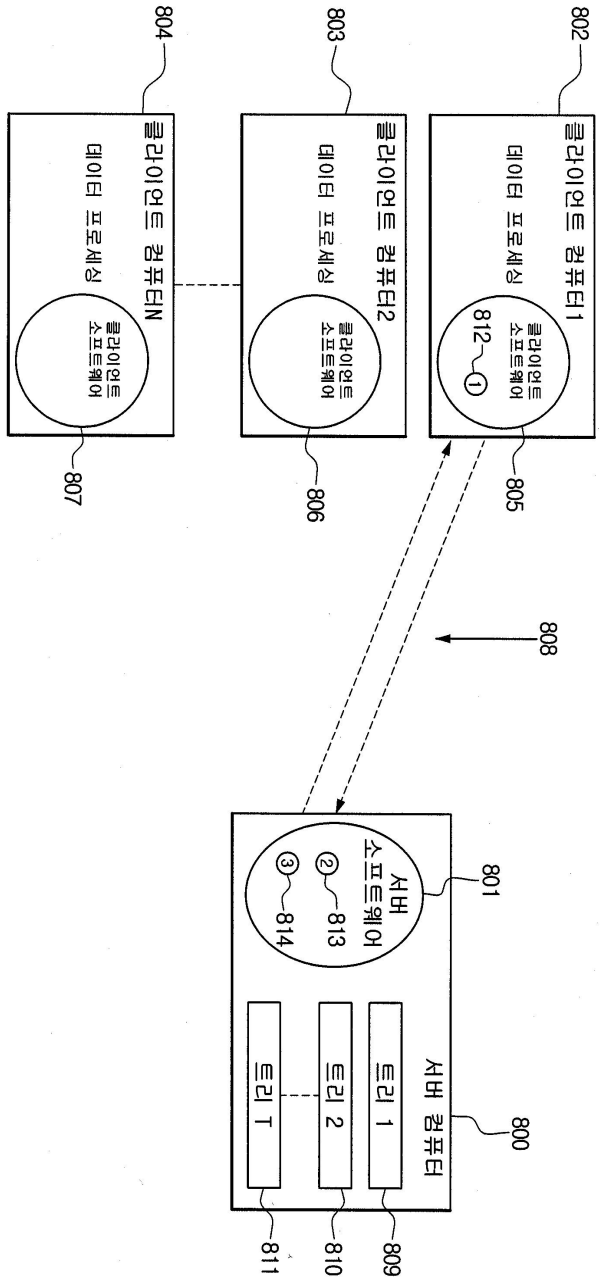
도면7


입력:
-파일의 이름
-획득 식별자
650
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠그는 시도
600
601
파일이 열리고 잠겼나?
예
아니오
602
시도의 시작에 파일이 존재했나?
예
아니오
603
에러: 파일이 존재하지 않음
604
최대 수에 도달 했거나 또는 타임아웃 기간이 경과되었는가?
예
아니오
605
에러: 시스템 에러
649
시간 릴레이
606
파일 헤더의 상태 파라미터 내의 파일의 사용 기록
607
파일이 사용 가능한가?
예
아니오
608
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 잠금 해제
609
에러: 파일 획득하지 않음
610
파일 헤더 내의 획득 식별자를 판독
611
식별자가 제출된 식별자와 같이 한가?
예
아니오
612
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 잠금 해제
613
에러: 획득 식별자 유효하지 않음
614
파일 헤더의 상태 파라미터 내에 식별자를 삭제
615
파일 헤더 내의 사용 가능 상태를 기록
616
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
651
출력:
- 파일의 릴리스의 확인
- 파일의 삭제의 확인

도면8


입력:
- 파일의 이름
750
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 열기 및 잠금을 시도
700
에러: 파일이 존재하지 않음
703
파일이 열리고 잠겼나?
701
예
아니오
시도의 시점에 파일이 존재했나?
702
아니오
예
시간 경과
749
최대 수에 도달 했는가?
704
아니오
예
에러: 시스템 에러
705
에러: 파일 여전히 획득 상태
710
파일 헤더의 상태 파라미터 내의 파일의 사용 기능
706
사용 가능한가?
707
예
아니오
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 잠금 해제
708
최대 수에 도달 했는가?
709
예
아니오
파일의 바디를 판독
711
파일 헤더의 상태 파라미터에 따라 파일의 바디를 처리 (압축 해독, 등)
712
운영 시스템의 파일 시스템의 레벨에서 파일의 동시적인 닫기 및 잠금 해제
713
출력:
- 파일의 획득 있는 파일의 단순 판독의 확인
- 파일의 수정 가능한 내용
751

도면9


802
클라이언트 컴퓨터1
데이터 프로세싱
812
①
클라이언트 소프트웨어
805
803
클라이언트 컴퓨터2
데이터 프로세싱
클라이언트 소프트웨어
806
804
클라이언트 컴퓨터N
데이터 프로세싱
클라이언트 소프트웨어
807
808
801
서버 소프트웨어
②
813
③
814
서버 컴퓨터
800
트리 1
809
트리 2
810
트리 T
811