(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0130568
(43) 공개일자 2024년08월29일

(51) 국제특허분류(Int. Cl.)
G06F 11/07 (2006.01) *G06F 3/06* (2006.01)
(52) CPC특허분류
G06F 11/0793 (2013.01)
G06F 11/0727 (2013.01)
(21) 출원번호 10-2023-0046530
(22) 출원일자 2023년04월10일
심사청구일자 2023년04월10일
(30) 우선권주장
1020230023018 2023년02월21일 대한민국(KR)

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
박정흠
서울시 마포구 마포대로 195 408동 2205호
김준호
경기도 성남시 중원구 광명로323번길 20-10, 1동 301호 (금광동, 금광동 3573 공동주택 (김상윤))
(뒷면에 계속)
(74) 대리인
특허법인위솔

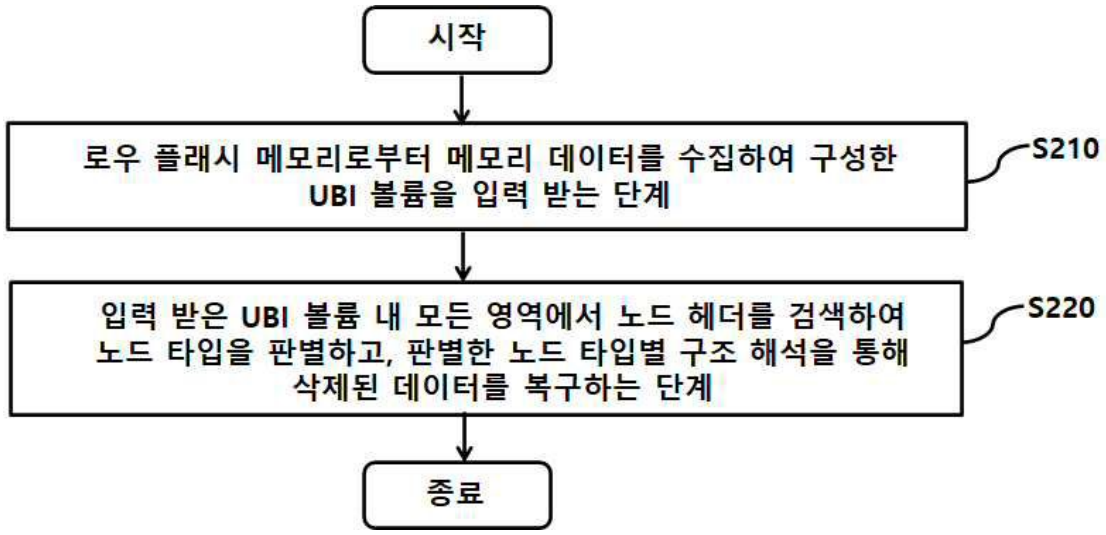
전체 청구항 수 : 총 11 항

(54) 발명의 명칭 **UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법 및 이를 위한 장치**

(57) 요 약

본 발명의 일 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법은 (a) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 단계 및 (b) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 단계를 포함하며, 상기 판별한 노드 타입은, B+ 트리 구조에 따른 리프 노드(Leaf Node)인 디렉터리 노드 및 데이터 노드 중 어느 하나이다.

대 표 도 - 도2



(52) CPC특허분류
G06F 3/0604 (2013.01)
G06F 3/0614 (2013.01)
G06F 3/0661 (2013.01)
G06F 2212/1016 (2013.01)
G06F 2212/1032 (2013.01)
(72) 발명자
정영기
서울특별시 삼양로 19길 113, 105-203
윤우성
서울특별시 강동구 천호대로 1231, 101동 1104호
(길동, 현대아파트)
이상진
서울시 종로구 사직로8길 20 파크팰리스 101동 1503호

이 발명을 지원한 국가연구개발사업

과제고유번호	1535000377
과제번호	201904682
부처명	해양경찰청
과제관리(전문)기관명	해양수산과학기술진흥원
연구사업명	불법선박대응을위한장비선진화기술개발(R&D)
연구과제명	해양사고 현장 디지털증거물 무결성 및 증거능력 확보를 위한 항해장비 디지털포렌식 기법 개발
기 여 율	1/1
과제수행기관명	한국전자통신연구원
연구기간	2022.01.01 ~ 2022.12.31

명 세 서

청구범위

청구항 1

프로세서 및 메모리를 포함하는 장치가 UBIFS(Unsorted Block Image File System)로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 있어서,

(a) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 단계; 및

(b) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 단계;

를 포함하며,

상기 판별한 노드 타입은,

B+ 트리 구조에 따른 리프 노드(Leaf Node)인 디렉터리 노드 및 데이터 노드 중 어느 하나인,

UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 2

제1항에 있어서,

상기 제1 단계는,

기기 내에 상기 로우 플래시 메모리가 존재하는지 판단하는 제1-1 단계;

상기 판단 결과, 로우 플래시 메모리가 존재한다면, 상기 로우 플래시 메모리로부터 메모리 데이터를 수집하는 제1-2 단계;

상기 수집한 메모리 데이터 내의 스페어 영역을 제거하는 제1-3 단계;

상기 스페어 영역을 제거한 메모리 데이터의 파티션 정보를 해석하여 파일 시스템 영역을 식별하는 제1-4 단계;

상기 식별한 파일 시스템 영역의 물리적인 구조를 파악하여 상기 UBI 볼륨을 구성하는 제1-5 단계; 및

상기 구성한 UBI 볼륨을 입력 받는 제1-6 단계;

중 어느 하나 이상을 포함하는 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 3

제2항에 있어서,

상기 제1-2 단계에서의 메모리 데이터의 수집은,

UART(Universal Asynchronous Receiver Transmitter) 통신 방식을 통해 상기 로우 플래시 메모리로부터 메모리 데이터를 수집하는 것인,

포함하는 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 4

제2항에 있어서,

상기 제1-5 단계는,

상기 식별한 파일 시스템 영역의 물리적인 구조 중, MTD(Memory Technology Device) 영역 상의 하나 이상의 PEB(Physical Erase Block)를 식별하는 제1-5-1 단계;

상기 식별한 하나 이상의 PEB 각각이 포함하는 EC 헤더에서 VID(Version Identifier) 헤더의 오프셋만큼 이동하

는 제1-5-2 단계;

상기 이동 결과 VID 헤더가 존재하는 경우, 상기 VID 헤더 내부에 기재된 LEB(Logical Erase Block) 번호를 판단하여 상기 PEB 와 매핑시키는 제1-5-3 단계; 및

상기 PEB와 매핑시킨 LEB를 나열하여 UBI 볼륨을 구성하는 제1-5-4 단계;

중 어느 하나 이상을 포함하는 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 5

제1항에 있어서,

상기 제2 단계는,

상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하는 제2-1 단계;

상기 검색한 노드 헤더의 구조 해석을 통해 노드 타입을 판별하고, 삭제된 데이터의 파일명과 실제 데이터를 획득하는 제2-2 단계;

상기 입력 받은 UBI 볼륨 내 모든 영역에서 모든 노드 헤더를 검색했는지 판단하는 제2-3 단계; 및

상기 판단 결과 모든 노드 헤더를 검색했다면, 삭제된 데이터에 대한 노드들의 관계를 해석하고 상기 삭제된 데이터를 복구하는 제2-4 단계;

중 어느 하나 이상을 포함하는 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 6

제5항에 있어서,

상기 제2-1 단계에서의 노드 헤더의 검색은,

노드 헤더의 고유 값인 0 x 31181006을 상기 UBI 볼륨 내 모든 영역에서 검색하는 것인,

UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 7

제5항에 있어서,

상기 제2-2 단계에서 판별한 노드 타입이 디렉터리 노드라면, 상기 제2-2 단계는,

상기 디렉터리 노드 내 Inum 값이 0인지 판단하는 제2-2-1 단계;

상기 판단 결과, Inum 값이 0이라면, 삭제된 데이터의 디렉터리 노드로 규정하고, 이로부터 삭제된 데이터의 파일명을 획득하는 제2-2-2 단계; 및

상기 삭제된 데이터의 디렉터리 노드 바로 다음에 연결되어 있는 동일한 데이터에 대한 아이 노드의 노드 키 값을 통해 데이터가 삭제되기 전의 상기 디렉터리 노드의 Inum 값을 유추하는 제2-2-3 단계;

중 어느 하나 이상을 포함하는 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 8

제5항에 있어서,

상기 제2-2 단계에서 판별한 노드 타입이 데이터 노드라면, 상기 제2-2 단계는,

상기 데이터 노드의 노드 키 값을 식별하는 제2-2-1´ 단계;

상기 데이터 노드에 저장된 삭제된 데이터의 길이를 식별하는 제2-2-2´ 단계; 및

데이터의 시작 위치부터 상기 식별한 데이터의 길이만큼 추출하여 삭제된 실제 데이터를 획득하는 제2-2-3´ 단계;

중 어느 하나 이상을 포함하는 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 9

제5항에 있어서,

상기 제2-4 단계에서의 삭제된 데이터에 대한 노드들의 관계 해석은,

삭제된 데이터의 디렉터리 노드에 대하여 이와 연결되어 있는 동일한 데이터에 대한 아이 노드의 노드 키 값을 통해 유추한 Inum 값과 동일한 노드 키 값을 갖는 데이터 노드를 동일한 데이터에 대한 디렉터리 노드와 데이터 노드로 해석하는,

UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법.

청구항 10

하나 이상의 프로세서;

네트워크 인터페이스;

상기 프로세서에 의해 수행되는 컴퓨터 프로그램을 로드(Load)하는 메모리; 및

대용량 네트워크 데이터 및 상기 컴퓨터 프로그램을 저장하는 스토리지를 포함하되,

상기 컴퓨터 프로그램은 상기 하나 이상의 프로세서에 의해,

(A) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 오퍼레이션; 및

(B) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 오퍼레이션;

을 실행하며,

상기 판별한 노드 타입은,

B+ 트리 구조에 따른 리프 노드(Leaf Node)인 디렉터리 노드 및 데이터 노드 중 어느 하나인,

UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치.

청구항 11

컴퓨팅 장치와 결합하여,

(AA) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 단계; 및

(BB) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 단계;

를 포함하며,

상기 판별한 노드 타입은,

B+ 트리 구조에 따른 리프 노드(Leaf Node)인 디렉터리 노드 및 데이터 노드 중 어느 하나인,

컴퓨터로 판독 가능한 매체에 저장된 컴퓨터 프로그램.

발명의 설명

기 술 분 야

[0001] 본 발명은 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법 및 이를 위한 장치에 관한 것이다. 보다 자세하게는 로우 플래시 메모리에 포맷된 UBIFS를 분석하여 미할당 영역에 남아 있는 삭제된 데이터를 복구하는 방법 및 이를 위한 장치에 관한 것이다.

배 경 기 술

[0002] 플래시 메모리(Flash Memory)는 전원이 꺼져도 데이터를 유지할 수 있고, 저전력으로 동작할 수 있어 크기가 큰

메모리를 내장하기 어려운 냉장고, 세탁기, CCTV 등과 같은 임베디드 시스템에 널리 활용되는 메모리이다.

[0003] 한편, UBIFS(Unsorted Block Image File System)는 플래시 메모리의 대표적인 파일 시스템으로서 많은 임베디드 시스템에서 이를 사용하고 있으며, 최근 UBIFS의 사용률이 높아짐에 따라 이에 대한 디지털 포렌식 분석의 수요 역시 함께 증가하고 있으나, UBIFS에 저장된 정상 데이터를 획득하는 방법은 잘 알려진 반면, 미할당 영역에 존재하는 삭제된 데이터를 복구하는 방법은 아직 개발되지 않은 현실이다.

선행기술문헌

특허문헌

[0004] (특허문헌 0001) 대한민국 등록특허공보 제 10-2400723호(2022.05.18)

발명의 내용

해결하려는 과제

[0005] 본 발명이 해결하고자 하는 기술적 과제는 UBIFS에 대한 디지털 포렌식 분석의 수요 증가에 발 맞추어 미할당 영역에 존재하는 삭제된 데이터를 복구할 수 있는 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법 및 이를 위한 장치를 제공하는 것이다.

[0006] 본 발명의 기술적 과제들은 이상에서 언급한 기술적 과제들로 제한되지 않으며, 언급되지 않은 또 다른 기술적 과제들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0007] 상기 기술적 과제를 달성하기 위한 본 발명의 일 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법은 (a) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 단계 및 (b) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 단계를 포함하며, 상기 판별한 노드 타입은, B+ 트리 구조에 따른 리프 노드(Leaf Node)인 디렉터리 노드 및 데이터 노드 중 어느 하나이다.

[0008] 일 실시 예에 따르면, 상기 제1 단계는, 기기 내에 상기 로우 플래시 메모리가 존재하는지 판단하는 제1-1 단계, 상기 판단 결과, 로우 플래시 메모리가 존재한다면, 상기 로우 플래시 메모리로부터 메모리 데이터를 수집하는 제1-2 단계, 상기 수집한 메모리 데이터 내의 스페어 영역을 제거하는 제1-3 단계, 상기 스페어 영역을 제거한 메모리 데이터의 파티션 정보를 해석하여 파일 시스템 영역을 식별하는 제1-4 단계, 상기 식별한 파일 시스템 영역의 물리적인 구조를 파악하여 상기 UBI 볼륨을 구성하는 제1-5 단계 및 상기 구성한 UBI 볼륨을 입력 받는 제1-6 단계중 어느 하나 이상을 포함할 수 있다.

[0009] 일 실시 예에 따르면, 상기 제1-2 단계에서의 메모리 데이터의 수집은, UART(Universal Asynchronous Receiver Transmitter) 통신 방식을 통해 상기 로우 플래시 메모리로부터 메모리 데이터를 수집하는 것일 수 있다.

[0010] 상기 제1-5 단계는, 상기 식별한 파일 시스템 영역의 물리적인 구조 중, MTD(Memory Technology Device) 영역상의 하나 이상의 PEB(Physical Erase Block)를 식별하는 제1-5-1 단계, 상기 식별한 하나 이상의 PEB 각각이 포함하는 EC 헤더에서 VID(Version Identifier) 헤더의 오프셋만큼 이동하는 제1-5-2 단계, 상기 이동 결과 VID 헤더가 존재하는 경우, 상기 VID 헤더 내부에 기재된 LEB(Logical Erase Block) 번호를 판단하여 상기 PEB와 매핑시키는 제1-5-3 단계 및 상기 PEB와 매핑시킨 LEB를 나열하여 UBI 볼륨을 구성하는 제1-5-4 단계 중 어느 하나 이상을 포함할 수 있다.

[0011] 일 실시 예에 따르면, 상기 제2 단계는, 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하는 제2-1 단계, 상기 검색한 노드 헤더의 구조 해석을 통해 노드 타입을 판별하고, 삭제된 데이터의 파일명과 실제 데이터를 획득하는 제2-2 단계, 상기 입력 받은 UBI 볼륨 내 모든 영역에서 모든 노드 헤더를 검색했는지 판단하는 제2-3 단계 및 상기 판단 결과 모든 노드 헤더를 검색했다면, 삭제된 데이터에 대한 노드들의 관계를 해석하고 상기 삭제된 데이터를 복구하는 제2-4 단계중 어느 하나 이상을 포함할 수 있다.

[0012] 일 실시 예에 따르면, 상기 제2-1 단계에서의 노드 헤더의 검색은, 노드 헤더의 고유 값인 0 x 31181006을 상기

UBI 볼륨 내 모든 영역에서 검색하는 것일 수 있다.

[0013] 일 실시 예에 따르면, 상기 제2-2 단계에서 판별한 노드 타입이 디렉터리 노드라면, 상기 제2-2 단계는, 상기 디렉터리 노드 내 Inum 값이 0인지 판단하는 제2-2-1 단계, 상기 판단 결과, Inum 값이 0이라면, 삭제된 데이터의 디렉터리 노드로 규정하고, 이로부터 삭제된 데이터의 파일명을 획득하는 제2-2-2 단계 및 상기 삭제된 데이터의 디렉터리 노드 바로 다음에 연결되어 있는 동일한 데이터에 대한 아이 노드의 노드 키 값을 통해 데이터가 삭제되기 전의 상기 디렉터리 노드의 Inum 값을 유추하는 제2-2-3 단계 중 어느 하나 이상을 포함할 수 있다.

[0014] 일 실시 예에 따르면, 상기 제2-2 단계에서 판별한 노드 타입이 데이터 노드라면, 상기 제2-2 단계는, 상기 데이터 노드의 노드 키 값을 식별하는 제2-2-1´ 단계, 상기 데이터 노드에 저장된 삭제된 데이터의 길이를 식별하는 제2-2-2´ 단계 및 데이터의 시작 위치부터 상기 식별한 데이터의 길이만큼 추출하여 삭제된 실제 데이터를 획득하는 제2-2-3´ 단계 중 어느 하나 이상을 포함할 수 있다.

[0015] 일 실시 예에 따르면, 상기 제2-4 단계에서의 삭제된 데이터에 대한 노드들의 관계 해석은, 삭제된 데이터의 디렉터리 노드에 대하여 이와 연결되어 있는 동일한 데이터에 대한 아이 노드의 노드 키 값을 통해 유추한 Inum 값과 동일한 노드 키 값을 갖는 데이터 노드를 동일한 데이터에 대한 디렉터리 노드와 데이터 노드로 해석하는 것일 수 있다.

[0016] 상기 기술적 과제를 달성하기 위한 본 발명의 또 다른 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치는 하나 이상의 프로세서, 네트워크 인터페이스, 상기 프로세서에 의해 수행되는 컴퓨터 프로그램을 로드(Load)하는 메모리 및 대용량 네트워크 데이터 및 상기 컴퓨터 프로그램을 저장하는 스토리지를 포함하되, 상기 컴퓨터 프로그램은 상기 하나 이상의 프로세서에 의해, (A) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 오퍼레이션 및 (B) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 오퍼레이션을 실행하며, 상기 판별한 노드 타입은, B+ 트리 구조에 따른 리프 노드(Leaf Node)인 디렉터리 노드 및 데이터 노드 중 어느 하나이다.

[0017] 상기 기술적 과제를 달성하기 위한 본 발명의 또 다른 실시 예에 따른 매체에 저장된 컴퓨터 프로그램은 컴퓨팅 장치와 결합하여, (AA) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 단계 및 (BB) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 단계를 포함하며, 상기 판별한 노드 타입은, B+ 트리 구조에 따른 리프 노드(Leaf Node)인 디렉터리 노드 및 데이터 노드 중 어느 하나이다.

발명의 효과

[0018] 상기와 같은 본 발명에 따르면, B+ 트리 구조 및 노드의 특성을 이용하여 UBIFS의 미할당 영역에 존재하는 삭제된 데이터를 복구할 수 있는바, UBIFS에 대한 디지털 포렌식 분석이 보다 용이하게 이루어질 수 있다는 효과가 있다.

[0019] 본 발명의 효과들은 이상에서 언급한 효과들로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해 될 수 있을 것이다.

도면의 간단한 설명

[0020] 도 1은 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치가 포함하는 전체 구성을 예시적으로 도시한 도면이다.

도 2는 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법의 대표적인 단계를 나타낸 순서도이다.

도 3은 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 있어서, UBI 볼륨을 구성하여 입력 받는 제1 단계인 S210 단계를 구체화한 순서도이다.

도 4는 파일 시스템 영역을 포함하고 스페어 영역이 제거된 메모리 영역을 예시적으로 도시한 도면이다.

도 5는 UBI 볼륨을 구성하는 모식도를 예시적으로 도시한 도면이다.

도 6은 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에

있어서, UBI 볼륨을 구성하는 S210-1-5 단계를 구체화한 순서도이다.

도 7은 EC 헤더의 내부 구조를 예시적으로 도시한 도면이다.

도 8은 VID 헤더의 내부 구조를 예시적으로 도시한 도면이다.

도 9는 구성한 UBI 볼륨의 구조를 예시적으로 도시한 도면이다.

도 10은 B+ 트리 구조를 예시적으로 도시한 도면이다.

도 11 은 아이 노드의 내부 구조를 예시적으로 도시한 도면이다.

도 12는은 데이터 노드의 내부 구조를 예시적으로 도시한 도면이다.

도 13 은 디렉터리 노드의 내부 구조를 예시적으로 도시한 도면이다.

도 14는 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 있어서, 삭제된 데이터를 복구하는 제2 단계인 S220 단계를 구체화한 순서도이다. .

도 15는 동일한 파일에 대한 디렉터리 노드, 아이 노드 및 데이터 노드를 예시적으로 도시한 도면이다.

도 16은 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 있어서, 삭제된 데이터의 파일명과 실제 데이터를 획득하는 S220-2 단계를 구체화한 순서도이다.

발명을 실시하기 위한 구체적인 내용

[0021] 본 발명의 목적과 기술적 구성 및 그에 따른 작용 효과에 관한 자세한 사항은 본 발명의 명세서에 첨부된 도면에 의거한 이하의 상세한 설명에 의해 보다 명확하게 이해될 것이다. 첨부된 도면을 참조하여 본 발명에 따른 실시 예를 상세하게 설명한다.

[0022] 본 명세서에서 개시되는 실시 예들은 본 발명의 범위를 한정하는 것으로 해석되거나 이용되지 않아야 할 것이다. 이 분야의 통상의 기술자에게 본 명세서의 실시 예를 포함한 설명은 다양한 응용을 갖는다는 것이 당연하다. 따라서, 본 발명의 상세한 설명에 기재된 임의의 실시 예들은 본 발명을 보다 잘 설명하기 위한 예시적인 것이며 본 발명의 범위가 실시 예들로 한정되는 것을 의도하지 않는다.

[0023] 도면에 표시되고 아래에 설명되는 기능 블록들은 가능한 구현의 예들일 뿐이다. 다른 구현들에서는 상세한 설명의 사상 및 범위를 벗어나지 않는 범위에서 다른 기능 블록들이 사용될 수 있다. 또한, 본 발명의 하나 이상의 기능 블록이 개별 블록들로 표시되지만, 본 발명의 기능 블록들 중 하나 이상은 동일 기능을 실행하는 다양한 하드웨어 및 소프트웨어 구성들의 조합일 수 있다.

[0024] 또한, 어떤 구성요소들을 포함한다는 표현은 "개방형"의 표현으로서 해당 구성요소들이 존재하는 것을 단순히 지칭할 뿐이며, 추가적인 구성요소들을 배제하는 것으로 이해되어서는 안 된다.

[0025] 나아가 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급될 때에는, 그 다른 구성요소에 직접적으로 연결 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 한다.

[0026] 이하에서는 도면들을 참조하여 본 발명의 세부적인 실시 예들에 대해 살펴보도록 한다.

[0027] 도 1은 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)가 포함하는 전체 구성을 예시적으로 도시한 도면이다.

[0028] 그러나 이는 본 발명의 목적을 달성하기 위한 바람직한 실시 예일 뿐이며, 필요에 따라 일부 구성이 추가되거나 삭제될 수 있고, 어느 한 구성이 수행하는 역할을 다른 구성이 함께 수행할 수도 있음은 물론이다.

[0029] 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)는 프로세서(10), 네트워크 인터페이스(20), 메모리(30), 스토리지(40) 및 이들을 연결하는 데이터 버스(50)를 포함할 수 있으며, 기타 본 발명의 목적을 달성함에 있어 요구되는 부가적인 구성들을 더 포함할 수 있음은 물론이라 할 것이다.

[0030] 프로세서(10)는 각 구성의 전반적인 동작을 제어한다. 프로세서(10)는 CPU(Central Processing Unit), MPU(Micro Processer Unit), MCU(Micro Controller Unit) 또는 본 발명이 속하는 기술 분야에서 널리 알려져 있는 형태의 인공지능 프로세서 중 어느 하나일 수 있다. 아울러, 프로세서(10)는 본 발명의 제2 실시 예에 따

른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법을 수행하기 위한 적어도 하나의 애플리케이션 또는 프로그램에 대한 연산을 수행할 수 있다.

[0031] 네트워크 인터페이스(20)는 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)의 유무선 인터넷 통신을 지원하며, 그 밖의 공지의 통신 방식을 지원할 수도 있다. 따라서 네트워크 인터페이스(20)는 그에 따른 통신 모듈을 포함하여 구성될 수 있다.

[0032] 메모리(30)는 각종 정보, 명령 및/또는 정보를 저장하며, 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법 을 수행하기 위해 스토리지(40)로부터 하나 이상의 컴퓨터 프로그램(41)을 로드할 수 있다. 도 1에서는 메모리(30)의 하나로 RAM을 도시하였으나 이와 더불어 다양한 저장 매체를 메모리(30)로 이용할 수 있음은 물론이다.

[0033] 스토리지(40)는 하나 이상의 컴퓨터 프로그램(41) 및 대용량 네트워크 정보(42)를 비임시적으로 저장할 수 있다. 이러한 스토리지(40)는 ROM(Read Only Memory), EPROM(Erasable Programmable ROM), EEPROM(Electrically Erasable Programmable ROM), 플래시 메모리 등과 같은 비휘발성 메모리, 하드 디스크(HDD), 보조 저장 매치(SSD), 착탈형 디스크, 또는 본 발명이 속하는 기술 분야에서 널리 알려져 있는 임의의 형태의 컴퓨터로 읽을 수 있는 기록 매체 중 어느 하나일 수 있다.

[0034] 컴퓨터 프로그램(41)은 메모리(30)에 로드되어, 하나 이상의 프로세서(10)에 의해, (A) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 오퍼레이션 및 (B) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 오퍼레이션을 실행할 수 있다.

[0035] 이상 간단하게 언급한 컴퓨터 프로그램(41)이 수행하는 오퍼레이션은 컴퓨터 프로그램(41)의 일 기능으로 볼 수 있으며, 보다 자세한 설명은 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법 에 대한 설명에서 후술하도록 한다.

[0036] 데이터 버스(50)는 이상 설명한 프로세서(10), 네트워크 인터페이스(20), 메모리(30) 및 스토리지(40) 사이의 명령 및/또는 정보의 이동 경로가 된다.

[0037] 이상 간단하게 설명한 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)는 독립된 디바이스의 형태, 예를 들어 전자 기기나 서버(클라우드 포함)의 형태일 수 있으며, 여기서 전자 기기는 한 장소에 고정 설치되어 사용하는 데스크톱 PC, 서버 디바이스 등과 같은 기기 뿐만 아니라, 스마트폰, 태블릿 PC, 노트북 PC, PDA, PMP 등과 같이 휴대가 용이한 포터블 기기 등이라도 무방한바, 프로세서(10)에 해당하는 CPU 등이 설치되고 네트워크 기능만 보유하고 있는 전자 기기라면 어떠한 것이라도 무방하다 할 것이다.

[0038] 이하, 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)가 독립된 디바이스인 전자 기기 형태 중, "서버"의 형태임을 전제로 삭제된 데이터를 복구하고자 하는 사용자의 사용자 단말(미도시)에 설치된 전용 어플리케이션을 통해 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법을 제공하는 과정에 대하여 도 2 내지 도 16을 참조하여 설명하도록 한다.

[0039] 도 2는 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법의 대표적인 단계를 나타낸 순서도이다.

[0040] 그러나 이는 본 발명의 목적을 달성함에 있어서 바람직한 실시 예일 뿐이며, 필요에 따라 일부 단계가 추가 또는 삭제될 수 있음은 물론이고, 어느 한 단계가 다른 단계에 포함되어 수행될 수도 있다.

[0041] 한편, 각 단계는 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)를 통해 이루어지는 것을 전제로 하며, 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)가 "서버"의 형태임을 전제로 하였기 때문에 사용자 단말(미도시)에 설치된 전용 어플리케이션을 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)와 동일한 의미로 볼 것이고, 설명의 편의를 위해 이들 모두를 "장치(100)"로 명명하도록 한다.

[0042] 또한, 설명에 따라"데이터", "정보", "파일" 등의 용어를 구별하여 사용할 것이나, 이들의 실제적인 의미는 동

일할 수 있음을 미리 밝혀두는 바이다.

[0043] 우선, 장치(100)가 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력받으며(S210), 이를 제1 단계라 한다.

[0044] 여기서 로우 플래시 메모리는 UBIFS라는 파일 시스템으로 포맷되어 있는 메모리이며, 특정 기기에 내재되어 있을 수 있다.

[0045] 한편, 제1 단계는 디지털 포렌식의 대상인 로우 플래시 메모리로부터 디지털 포렌식을 수행하기 위한 준비 단계에 해당하는바, 이하 도 3을 참조하여 자세히 설명하도록 한다.

[0046] 도 3은 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 있어서, UBI 볼륨을 구성하여 입력 받는 제1 단계인 S210 단계를 구체화한 순서도이다.

[0047] 그러나 이는 본 발명의 목적을 달성함에 있어서 바람직한 실시 예일 뿐이며, 필요에 따라 일부 단계가 추가 또는 삭제될 수 있음은 물론이고, 어느 한 단계가 다른 단계에 포함되어 수행될 수도 있다.

[0048] 우선, 기기 내에 로우 플래시 메모리가 존재하는지 판단하며(S210-1-1), 이를 제1-1 단계라 하고, 판단 결과 로우 플래시 메모리가 존재한다면, 로우 플래시 메모리로부터 메모리 데이터를 수집하며(S210-1-2), 이를 제1-2 단계라 한다.

[0049] 여기서 메모리 데이터의 수집은 메모리 데이터의 복제로 볼 수 있으며, UART(Universal Asynchronous Receiver Transmitter) 통신 방식을 통해 상기 로우 플래시 메모리로부터 메모리 데이터를 수집하거나, 칩오프를 통해 메모리 데이터를 수집할 수도 있다.

[0050] 이후, 수집한 메모리 데이터 내의 스페어 영역을 제거하며(S210-1-3), 이를 제1-3 단계라 한다.

[0051] 로우 플리시 메모리는 메모리 셀에 저장된 데이터에 대한 추가적인 정보를 메모리 내 스페어 영역에 저장하며, 보다 구체적으로 스페어 영역에는 ECC(Error Correction Code)나 웨어 레벨링(Wear leveling)과 같은 정보들이 저장되는바, 이와 같은 정보들은 파티션 해석에 방해가 될 수 있으므로 스페어 영역을 제거하는 것이다.

[0052] 스페어 영역을 제거했다면, 스페어 영역을 제거한 메모리 데이터의 파티션 정보를 해석하여 파일 시스템 영역을 식별하며(S210-1-4), 이를 제1-4 단계라 한다.

[0053] 로우 플래시 메모리는 용도 별로 여러 개의 파티션으로 분할되며, 플래시 메모리의 제조사에 따라 파티션을 분할하는 방법이 상이한바, 로우 플래시 메모리 내 파티션 정보를 해석하여 파일 시스템 영역을 식별할 필요성이 있으며, 도 4에는 파일 시스템 영역을 포함하고 스페어 영역이 제거된 메모리 영역을 예시적으로 도시해 놓았다. 도 4를 참조하면 가장 아래 파일 시스템 영역이 존재함을 확인할 수 있는바, S210-1-4 단계에서는 이와 같은 파일 시스템 영역의 위치와 크기를 식별하는 것으로 볼 수 있다.

[0054] 이후, 식별한 파일 시스템 영역의 물리적인 구조를 파악하여 UBI 볼륨을 구성하며(S210-1-5), 이를 제1-5 단계라 한다.

[0055] 파일 시스템 영역의 물리적인 구조의 파악은 메모리 데이터에서 정상적으로 저장되어 있는 파일들과 삭제된 데이터들을 식별하기 위함이며, 이하, UBI 볼륨을 구성하는 구체적인 방법을 도 5와 도6을 참조하여 설명하도록 한다.

[0056] 도 5는 UBI 볼륨을 구성하는 모식도를 예시적으로 도시한 도면이며, 도 6은 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 있어서, UBI 볼륨을 구성하는 S210-1-5 단계를 구체화한 순서도이다.

[0057] 그러나 이는 본 발명의 목적을 달성함에 있어서 바람직한 실시 예일 뿐이며, 필요에 따라 일부 단계가 추가 또는 삭제될 수 있음은 물론이고, 어느 한 단계가 다른 단계에 포함되어 수행될 수도 있다.

[0058] 우선, 식별한 파일 시스템의 물리적인 구조 중, MTD(Memory Technology Device) 영역 상의 하나 이상의 PEB(Physical Erase Block)를 식별하며(S210-1-5-1), 제1-5-1 단계라 한다.

[0059] 여기서 MTD 영역은 UBIFS와 같은 리눅스 커널 기반 임베디드 OS를 플래시 메모리에서 사용할 수 있도록 지원하며, 128KB의 PEB로 플래시 메모리 데이터를 관리한다. 이와 같은 MTD 영역은 복수 개의 PEB를 포함하는바, 도 5를 참조하면 각각의 PEB는 EC(Erase Counter) 헤더를 포함하고 있음을 확인할 수 있으며, EC 헤더는 VID(Volume Identifier) 헤더의 위치와 데이터 영역의 위치 정보를 가지고 있는 2048(0x800) 크기의 헤더이고,

그 내부 구조는 도 7에 예시적으로 도시해 놓았다.

[0060] PEB를 식별했다면, 식별한 하나 이상의 PEB 각각이 포함하는 EC 헤더에서 VID(Version Identifier) 헤더의 오프셋만큼 이동하며(S210-1-5-2), 이를 제1-5-2 단계라 한다.

[0061] VID 헤더는 UBI 볼륨 내 LEB(Logical Erase Block) 번호를 가지고 있는 2048(0x800) 크기의 헤더이며, 앞서 EC 헤더는 VID 헤더의 위치 정보를 가지고 있다고 한바, 해당 정보가 나타내는 위치에 VID 헤더가 존재하는지 여부를 확인하기 위해 VID 헤더의 오프셋 만큼 이동하는 것이다.

[0062] 앞서 EC 헤더의 내부 구조를 도시한 도 7을 다시 참조하면, EC 헤더 내부 구조에서 VID 헤더의 위치와 해당 블록에 저장된 데이터의 위치를 확인할 수 있는바, EC 헤더 구조는 0x55424923('UBI#')이라는 시그니처(EC header signature)를 가지고 있으며, 오프셋 16(0x10)부터 4바이트인 VID 헤더 오프셋 정보를 저장하고 있다. 예를 들어, 해당 값이 2048(0x800)이면 EC 헤더의 첫 번째 바이트에서 2048(0x800) 바이트만큼 이동한 곳이 VID 헤더가 위치한 곳이라는 의미이며, 이와 별개로 EC 헤더는 오프셋 20(0x14)부터 4바이트인 Data 오프셋 정보를 가지고 있는바, 값이 4096(0x1000)이면 EC 헤더 첫 번째 바이트에서 4096(0x1000) 바이트만큼 이동한 곳이 데이터가 저장된 영역이라는 의미이다.

[0063] 이와 별개로 도 5에는 PEB가 EC 헤더와 VID 헤더를 모두 포함하는 것으로 도시되어 있으나, 이는 예시적인 것이며, EC 헤더는 필수적으로 포함하지만 VID 헤더는 포함하지 않을 수 있다.

[0064] 이동 결과, VID 헤더가 존재하는 경우, VID 헤더 내부에 기재된 LEB 번호를 판단하여 PEB 와 매핑시키며(S210-1-5-3), 이를 제1-5-3 단계라 한다.

[0065] 도 8은 VID 헤더의 내부 구조를 예시적으로 도시한 도면이다. 도 8을 참조하면, VID 헤더 내부 구조에서 헤더의 시그니처(Vid header signature)인 0x55424921('UBI!')를 가지고 있으며, 오프셋 12(0x0C)에서 4바이트는 해당 블록의 UBI 볼륨 내 LEB 위치 번호를 의미하는바, 예를 들어, 해당 값이 11(0x0B)이면 해당 블록은 UBI 볼륨 내 11번째에 속한다는 것을 의미하는 것이며, 확인한 VID 헤더 내부에 기재된 LEB 번호를 통해 해당 번호를 가진 LEB와 PEB를 매핑시킬 수 있다.

[0066] 마지막으로 PEB와 매핑시킨 LEB를 나열하여 UBI 볼륨을 구성하며(S210-1-5-4), 이를 제1-5-4 단계라 한다.

[0067] UBI 볼륨 구성을 위해 나열하는 LEB는 PEB와 매핑시킨 모든 LEB를 나열하는 것이나, 앞서 설명한 바와 같이 LEB는 각각의 위치 번호를 보유하고 있기 때문에 모든 LEB의 나열이 위치 번호의 순서대로 이루어지는 것으로 이해할 수 있으나, 이는 필수적인 것은 아니며, 위치 번호 0번 LEB인 Superblock(UBIFS에 대한 메타데이터를 저장), 위치 번호 1번 LEB인 MasterBlock(루트 인덱스 노드에 대한 정보를 저장), 위치 번호 2번 LEB인 1번 LEB에 대한 복제본을 저장하는 MasterBlock 이렇게 3개의 LEB만 위치 번호의 순서대로 나열되면 충분하다 할 것이고, 이를 도 9에 예시적으로 도시해 놓았다.

[0068] 다시 도 3에 대한 설명으로 돌아가도록 한다.

[0069] UBI 볼륨을 구성했다면, 구성한 UBI 볼륨을 입력 받으며(S210-1-6), 이를 제1-6 단계라 한다.

[0070] 여기서 UBI 볼륨의 "입력"은 장치(100) 스스로 UBI 볼륨을 구성한 경우 "로딩"의 의미를 가진다 할 것이며, 타 장치가 구성한 UBI 볼륨을 받아 들이는 역할을 하는 경우 "수신"의 의미를 가진다 할 것인바, 어느 경우나 모두 S210-1-6 단계에 해당하는 것으로 볼 수 있다 할 것이다.

[0071] 지금까지 도 3 내지 도 9를 참조하여 UBI 볼륨을 구성하여 입력 받는 과정에 대하여 설명하였다. 이하 다시 도 2를 참조하여 삭제된 데이터를 복구하는 단계를 설명하기 전에 UBIFS에서 데이터를 저장할 때 활용하는 B+ 트리 구조 그리고 이를 이루고 있는 노드들에 대하여 먼저 설명하도록 한다.

[0072] 도 10은 B+ 트리 구조를 예시적으로 도시한 도면이다. 도 10을 참조하면 가장 위의 하나의 노드로부터 그 아래로 다른 노드가 연결되며, 다른 노드의 아래에 또 다른 노드가 연결된 모습이 마치 나무의 모습과 같기에 트리 구조라 명명된 것이며, 가장 위의 노드를 루트 인덱스(Root Index) 노드, 그 아래로 다른 노드를 인덱스(Index) 노드, 가장 아래단의 노드를 리프(Leaf) 노드라 한다.

[0073] 이와 같은 B+ 트리 구조는 모든 데이터를 가장 아래단의 리프 노드에 저장하고, 리프 노드를 찾아가기 위한 위치 정보들을 인덱스 노드에 저장하는바, 인덱스 노드는 다음 노드를 가리키는 주소(포인터)가 저장되어 있으며, 특정 데이터 검색 시 각 인덱스 노드가 가리키는 주소를 통해 데이터가 저장된 리프 노드까지 찾아갈 수 있는바, 대량의 데이터를 효율적으로 저장하고 검색할 수 있다는 장점이 있다.

[0074] 한편, B+ 트리 구조에서 실제 데이터를 저장하는 리프 노드는 파일 및 디렉터리에 대한 정보를 보유하고 있으며, 하나의 파일 및 디렉터리 당 세 가지 타입의 노드를 보유하는바, 노드 타입 값이 0인 아이 노드, 1인 데이터 노드 그리고 2인 디렉터리 노드가 그것이다.

[0075] 여기서 아이 노드는 데이터 노드의 위치와 파일의 생성, 수정 날짜 등의 메타 정보를 저장하고 있으며, 데이터 노드는 실질적인 파일 데이터가 저장되어 있어 데이터 위치와 길이 정보를 통해 저장된 데이터를 획득할 수 있고, 디렉터리 노드는 파일 이름 정보가 저장되어 있다.

[0076] 도 11 내지 도 13은 각각 아이 노드, 데이터 노드, 디렉터리 노드의 내부 구조를 예시적으로 도시한 도면인바, 도 11을 참조하면 아이 노드는 첫 바이트부터 오프셋 23(0x17)까지 공통된 노드 헤더를 가지며, 이때 오프셋 20(0x14)의 노드 타입(Node type) 값이 0(0x00)이면 아이 노드를 의미한다. 다음 오프셋 24(0x18) 부터 16바이트는 해당 노드의 키 값(Node key)을 저장하고 있으며, 노드 키의 상위 4바이트는 해당 노드와 관련 있는 데이터 노드와 아이 노드의 관계와 위치를 파악하는데 필요하다.

[0077] 도 12를 참조하면, 데이터 노드는 아이 노드와 마찬가지로 첫 바이트부터 오프셋 23(0x17)까지 공통 노드 헤더 구조를 지니며, 오프셋 20(0x14)의 노드 타입(Node type) 값은 1(0x01)이다. 오프셋 24(0x18) 부터 16바이트는 해당 노드의 키 값(Node key)을 의미하며, 키 값 16바이트의 상위 4바이트는 아이 노드의 키 값인 Inum의 상위 4바이트와 동일하고, 디렉터리 노드와 아이 노드 간의 위치를 파악하는데 활용된다. 40(0x28)부터 4바이트는 데이터의 길이를 뜻하며, 오프셋 50(0x32)부터 데이터 길이만큼 실제 데이터가 저장될 수 있다.

[0078] 도 13을 참조하면, 디렉터리 노드는 아이 노드, 데이터 노드와 마찬가지로 첫 바이트부터 오프셋 23(0x17)까지 공통 노드 헤더 구조를 지니며, 오프셋 20(0x14)의 노드 타입(Node type) 값은 2(0x02)이다. 오프셋 0x18 부터 16바이트는 해당 디렉터리 노드의 키 값(Node key)이 저장되며, 해당 값의 상위 4바이트는 상위 디렉터리의 Inum 값을 의미한다. 오프셋 0x28부터 8바이트는 Inum(I-node Number)의 값을 저장한다. 디렉터리 노드의 상위 4바이트 Inum과 아이 노드의 노드 키(Node key)의 상위 4바이트 그리고 데이터 노드의 노드 키(Node key)의 상위 4바이트는 같은 파일 및 디렉터리에 대하여 동일한 값을 가지고 있기 때문에 Inum 값을 통해 자신과 관련된 데이터 노드 및 아이 노드를 구분할 수 있다. 다음 오프셋 56(0x38)은 데이터의 파일 혹은 디렉터리 명(File/Directory name)을 의미한다.

[0079] 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법은 이상 설명한 B+ 트리 구조와 아이 노드, 데이터 노드 및 디렉터리 노드의 특성을 이용하여 삭제된 데이터를 복구할 수 있으며, 이는 제2 단계에 대한 것이다. 다시 도 2를 참조하여 설명하도록 한다.

[0080] UBI 볼륨을 입력 받았다면, 장치(100)가 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하며(S22), 이를 제2 단계라 한다.

[0081] 여기서 장치(100)가 판별한 노드 타입은 B+ 트리 구조에 따른 리프 노드인 디렉터리 노드 및 데이터 노드 중 어느 하나일 수 있으며, 앞에서 함께 설명한 아이 노드가 포함되지 않는 이유에 대해서는 후술하도록 한다.

[0082] 도 14는 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 있어서, 삭제된 데이터를 복구하는 제2 단계인 S220 단계를 구체화한 순서도이다. .

[0083] 그러나 이는 본 발명의 목적을 달성함에 있어서 바람직한 실시 예일 뿐이며, 필요에 따라 일부 단계가 추가 또는 삭제될 수 있음은 물론이고, 어느 한 단계가 다른 단계에 포함되어 수행될 수도 있다.

[0084] 우선, 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하며(S220-1), 이를 제2-1 단계라 한다.

[0085] 도 15는 동일한 파일에 대한 디렉터리 노드, 아이 노드 및 데이터 노드를 예시적으로 도시한 도면인바, 도 15를 참조하면 S220-1 단계 에서의 노드 헤더의 검색은 노드 헤더의 고유 값인 0x31181006(1500)을 UBI 볼륨 내 모든 영역에서 검색하는 것일 수 있다.

[0086] 노드 헤더를 검색했다면, 검색한 노드 헤더의 구조 해석을 통해 노드 타입을 판별하고, 삭제된 데이터의 파일명과 실제 데이터를 획득하며(S220-2), 이를 제2-2 단계라 한다.

[0087] 앞서 각 노드마다 노드 헤더의 구조 해석, 보다 구체적으로 노드 타입 값을 통해 노드 타입을 판별할 수 있다고 했던바, 노드 타입 값이 0(0x00), 1(0x01), 2(0x02) 각각이 아이 노드, 데이터 노드, 디렉터리 노드일 수 있다.

[0088] 장치(100)는 판별한 노드 타입이 디렉터리 노드인 경우와 데이터 노드인 경우에 삭제된 데이터의 파일명과 실제

데이터를 획득하기 위한 개별적인 프로세스를 진행할 수 있는바, 이하 도 16을 참조하여 설명하도록 한다.

[0089] 도 16은 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 있어서, 삭제된 데이터의 파일명과 실제 데이터를 획득하는 S220-2 단계를 구체화한 순서도이다.

[0090] 그러나 이는 본 발명의 목적을 달성함에 있어서 바람직한 실시 예일 뿐이며, 필요에 따라 일부 단계가 추가 또는 삭제될 수 있음은 물론이고, 어느 한 단계가 다른 단계에 포함되어 수행될 수도 있다.

[0091] 판별한 노드 타입이 디렉터리 노드라면, 디렉터리 노드 내 Inum 값이 0인지 판단하며(S220-2-1), 이를 제2-2-1 단계라 한다.

[0092] 디렉터리 노드의 Inum 값은 데이터의 삭제가 있기 전에는 특정 값을 나타내다가 데이터가 삭제된 후에는 0으로 변경되기에 디렉터리 노드 내 Inum 값이 0인지 판단하여 삭제된 데이터에 대한 디렉터리 노드인지 확인하는 것이다.

[0093] 판단 결과, Inum 값이 0이라면, 삭제된 데이터의 디렉터리 노드로 규정하고, 이로부터 삭제된 데이터의 파일명을 획득하며(S220-2-2), 이를 제2-2-2 단계라 한다.

[0094] 이를 앞선 도 15에 대입하여 설명하면, 도 15에서 노드 타입 값이 2(1501)인 노드가 디렉터리 노드인바, 오프셋 0x1E268800에 존재하는 노드가 디렉터리 노드이며, 해당 디렉터리 노드의 노드 헤더 고유 값이 0x31181006(1500)임은 앞서 확인하였다.

[0095] 한편, 해당 디렉터리 노드의 Inum 값을 확인하면 0x00000000(1503)임을 확인할 수 있는바, 해당 디렉터리 노드를 삭제된 데이터에 대한 디렉터리 노드로 규정할 수 있으며, 디렉터리 노드는 파일명을 저장하기에 파일명의 길이(1504)만큼 파일명(1505)을 획득하여 저장할 수 있다.

[0096] 이후, 삭제된 데이터의 디렉터리 노드 바로 다음에 연결되어 있는 동일한 데이터에 대한 아이 노드의 노드 키 값을 통해 데이터가 삭제되기 전의 디렉터리 노드의 Inum 값을 유추하며(S220-2-3), 이를 제2-2-3 단계라 한다.

[0097] 앞서, 장치(100)가 판별한 노드 타입은 B+ 트리 구조에 따른 리프 노드인 디렉터리 노드 및 데이터 노드 중 어느 하나이고, 아이 노드가 포함되지 않는 이유를 보류했던바, 디렉터리 노드의 바로 다음에 동일한 데이터에 대한 아이 노드가 무조건 연결되어 있기 때문에 검색하여 판별한 노드 타입에서 아이 노드가 포함되지 않는 것이다. 즉, 디렉터리 노드로 판별되면 아이를 이 노드에 관한 사항은 자연스럽게 확인할 수 있는 것이다.

[0098] 이를 앞선 도 15에 대입하여 설명하면, 디렉터리 노드 바로 다음에 연결되어 있는 오프셋 0x1E268848에 존재하는 노드가 노드 헤더의 고유 값이 0x31181006(1506)이고, 노드 타입 값이 0(1507)인 아이 노드이다.

[0099] 해당 아이 노드의 노드 키 값을 확인하면 0xAC000000(1508)임을 확인할 수 있는바, 동일한 삭제된 데이터에 대한 디렉터리 노드의 현재 Inum값인 0x00000000(1503)이 데이터 삭제 전에는 0xAC000000였을 것으로 유추할 수 있다.

[0100] 한편, 판별한 노드 타입이 데이터 노드라면, 데이터 노드의 노드 키 값을 식별하며(S220-2-1´), 이를 제2-2-1´ 단계라 한다.

[0101] 이를 앞선 도 15에 대입하여 설명하면, 오프셋 0x38755000에 존재하는 노드가 노드 헤더의 고유 값이 0x31181006(1509)이고, 노드 타입 값이 1(1510)인 데이터 노드이다.

[0102] 해당 데이터 노드의 노드 키 값을 확인하면 0xAC000000(1511)임을 확인할 수 있다.

[0103] 이후, 데이터 노드에 저장된 삭제된 데이터의 길이를 식별하며(S220-2-2´), 이를 제2-2-2´ 단계라 하고, 데이터의 시작 위치부터 식별한 데이터의 길이만큼 추출하여 삭제된 실제 데이터를 획득하며(S220-2-3´), 이를 제2-2-3´ 단계라 한다.

[0104] 이를 앞선 도 15에 대입하여 설명하면, 삭제된 데이터의 길이(1512)를 식별하고, 데이터의 시작 위치인 오프셋 0x38755030부터 식별한 길이만큼 실제 데이터(1513)를 획득할 수 있다.

[0105] 다시 도 14에 대한 설명으로 돌아가도록 한다.

[0106] 삭제된 데이의 파일명과 실제 데이터를 획득했다면, 입력 받은 UBI 볼륨 내 모든 영역에서 모든 노드 헤더를 검색했는지 판단하며(S220-2-3), 이를 제2-3 단계라 하고, 판단 결과 모든 노드 헤더를 검색했다면, 삭제된 데이터에 대한 노드들의 관계를 해석하고 삭제된 데이터를 복구하며(S220-2-4), 이를 제2-4 단계라 한다.

[0107] 여기서 삭제된 데이터에 대한 노드들의 관계 해석은 동일한 데이터에 대하여 아이 노드, 디렉터리 노드 및 데이터 노드가 존재한다고 했던바, 삭제된 데이터의 디렉터리 노드에 대하여 이와 연결되어 있는 동일한 데이터에 대한 아이 노드의 노드 키 값을 통해 유추한 Inum 값과 동일한 노드 키 값을 갖는 데이터 노드를 동일한 데이터에 대한 디렉터리 노드와 데이터 노드로 해석하는 것이다.

[0108] 이를 앞선 도 15에 대입하여 설명하면, 앞서 아이 노드의 노드 키 값이 0xAC000000(1508)임에 따라 디렉터리 노드의 데이터 삭제 전의 Inum 값이 0xAC000000임을 유추하였으며, 식별한 데이터 노드의 노드 키 값이 0xAC000000(1511)이었던바 동일하므로 해당 디렉터리 노드와 데이터 노드를 동일한 삭제된 데이터에 대한 디렉터리 노드 및 데이터 노드로 해석한다는 것이다.

[0109] 더 나아가, 디렉터리 노드를 통해 파일명을, 데이터 노드를 통해 실제 데이터를 획득하였으며, 두 노드가 동일한 삭제된 데이터에 대한 노드라고 해석되었기 때문에 획득한 파일명과 실제 데이터가 삭제된 데이터에 대한 파일명 및 실제 데이터로 복구될 수 있는 것이다.

[0110] 지금까지 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 대하여 설명하였다. 본 발명에 따르면 B+ 트리 구조 및 노드의 특성을 이용하여 UBIFS의 미할당 영역에 존재하는 삭제된 데이터를 복구할 수 있는바, UBIFS에 대한 디지털 포렌식 분석이 보다 용이하게 이루어질 수 있다.

[0111] 마지막으로, 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100)와 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법은 본 발명의 제3 실시 예에 따른 컴퓨터로 판독 가능한 매체에 저장된 컴퓨터 프로그램으로 구현할 수도 있는바, 이 경우 컴퓨팅 장치와 결합하여 (AA) 로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한 UBI 볼륨을 입력 받는 제1 단계 및 (BB) 상기 입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여 노드 타입을 판별하고, 상기 판별한 노드 타입별 구조 해석을 통해 삭제된 데이터를 복구하는 제2 단계를 실행할 수 있을 것이며, 중복 서술을 위해 자세히 기재하지는 않았지만 본 발명의 제1 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치(100) 및 본 발명의 제2 실시 예에 따른 UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 방법에 적용된 모든 기술적 특징은 본 발명의 제3 실시 예에 따른 컴퓨터로 판독 가능한 매체에 저장된 컴퓨터 프로그램에 모두 동일하게 적용될 수 있음은 물론이라 할 것이다.

[0112] 이상 첨부된 도면을 참조하여 본 발명의 실시 예들을 설명하였지만, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 그 기술적 사상이나 필수적인 특징을 변경하지 않고서 다른 구체적인 형태로 실시될 수 있다는 것을 이해할 수 있을 것이다. 그러므로 이상에서 기술한 실시 예들은 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다.

부호의 설명

[0113] 10: 프로세서

20: 네트워크 인터페이스

30: 메모리

40: 스토리지

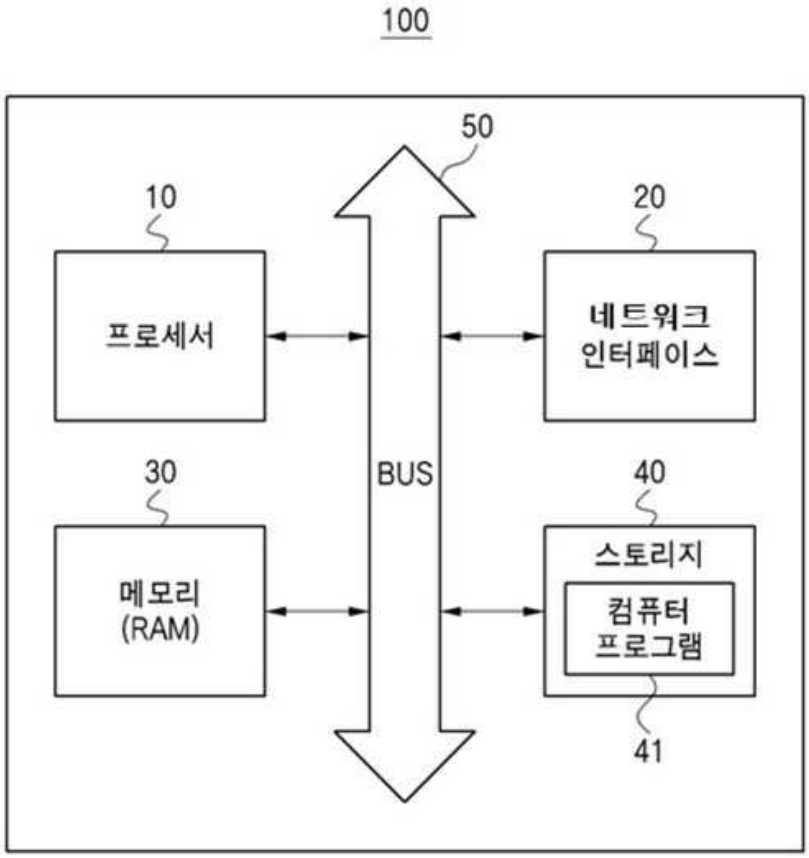
41: 컴퓨터 프로그램

50: 정보 버스

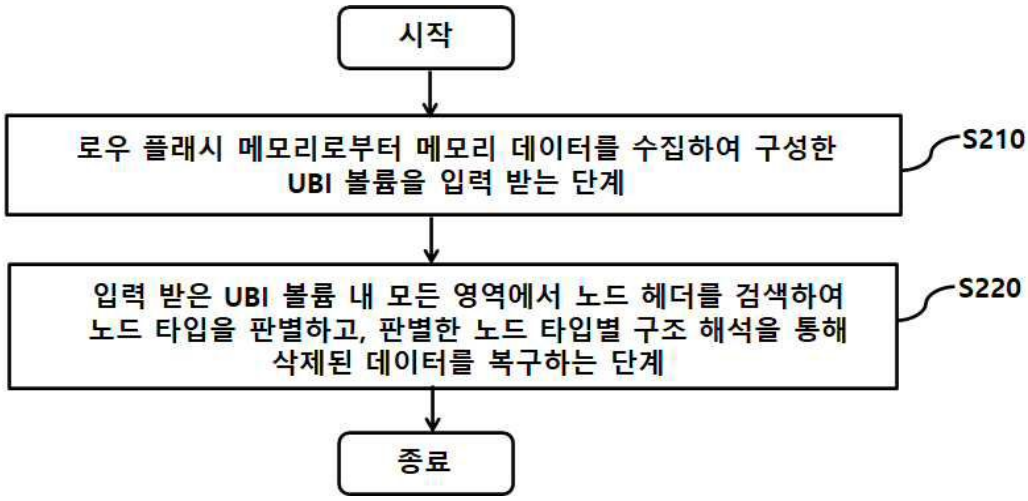
100: UBIFS로 포맷된 로우 플래시 메모리의 삭제된 데이터를 복구하는 장치

도면

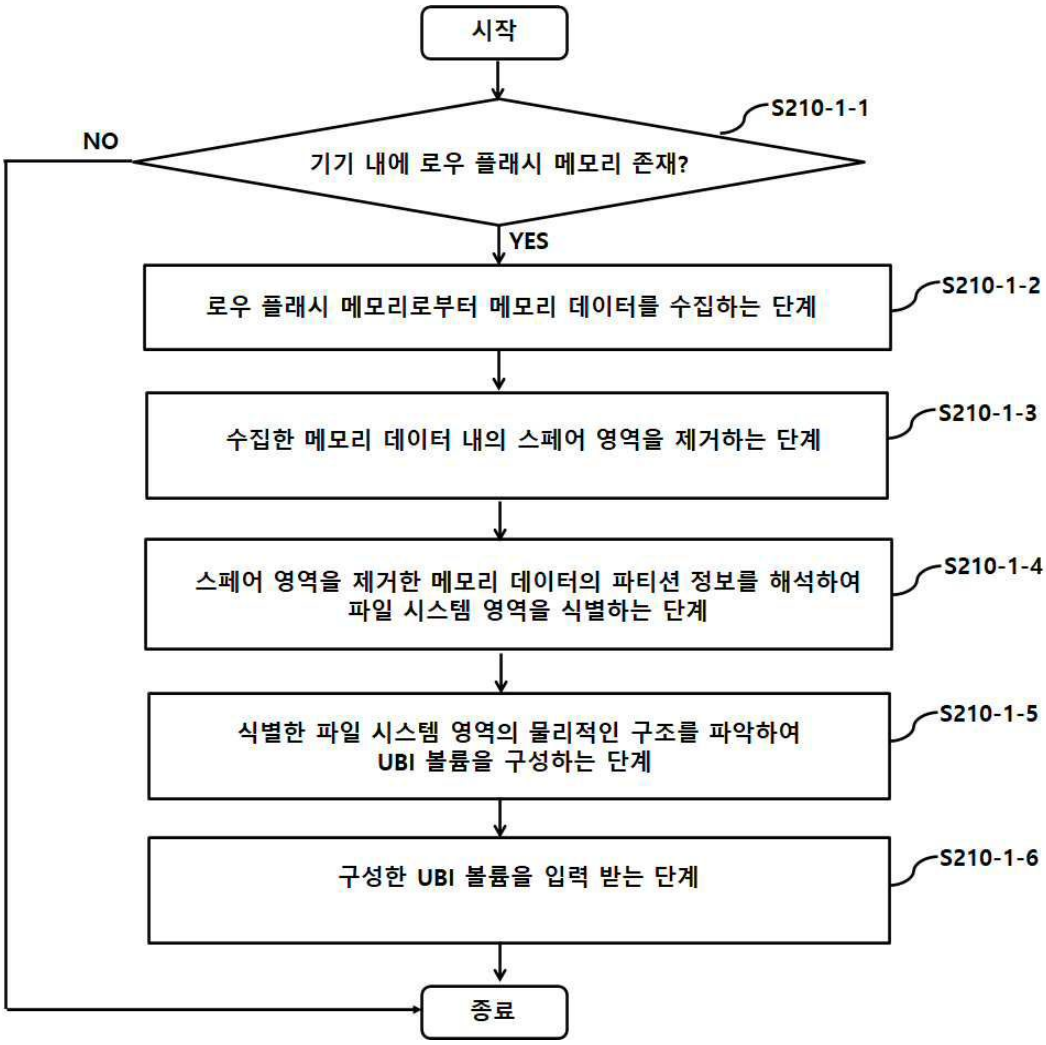
도면1


100
50
10
20
프로세서
네트워크
인터페이스
BUS
30
40
메모리
(RAM)
스토리지
컴퓨터
프로그램
41

도면2


시작
로우 플래시 메모리로부터 메모리 데이터를 수집하여 구성한
UBI 볼륨을 입력 받는 단계
S210
입력 받은 UBI 볼륨 내 모든 영역에서 노드 헤더를 검색하여
노드 타입을 판별하고, 판별한 노드 타입별 구조 해석을 통해
삭제된 데이터를 복구하는 단계
S220
종료

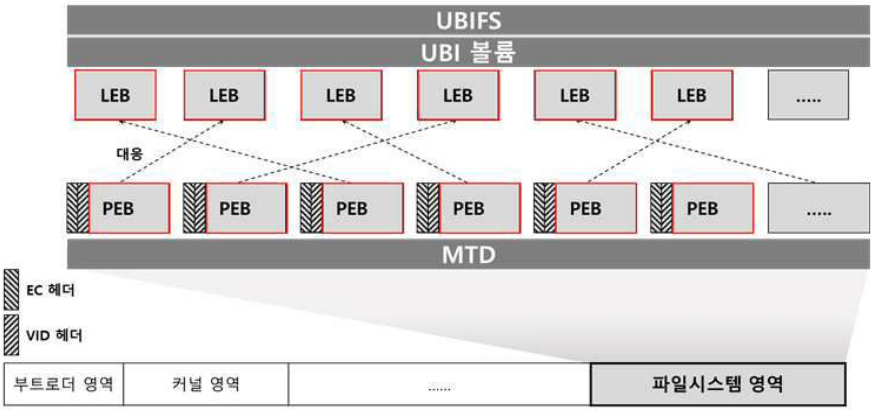
도면3


시작
S210-1-1
NO
기기 내에 로우 플래시 메모리 존재?
YES
로우 플래시 메모리로부터 메모리 데이터를 수집하는 단계
S210-1-2
수집한 메모리 데이터 내의 스페어 영역을 제거하는 단계
S210-1-3
스페어 영역을 제거한 메모리 데이터의 파티션 정보를 해석하여
파일 시스템 영역을 식별하는 단계
S210-1-4
식별한 파일 시스템 영역의 물리적인 구조를 파악하여
UBI 볼륨을 구성하는 단계
S210-1-5
구성한 UBI 볼륨을 입력 받는 단계
S210-1-6
종료

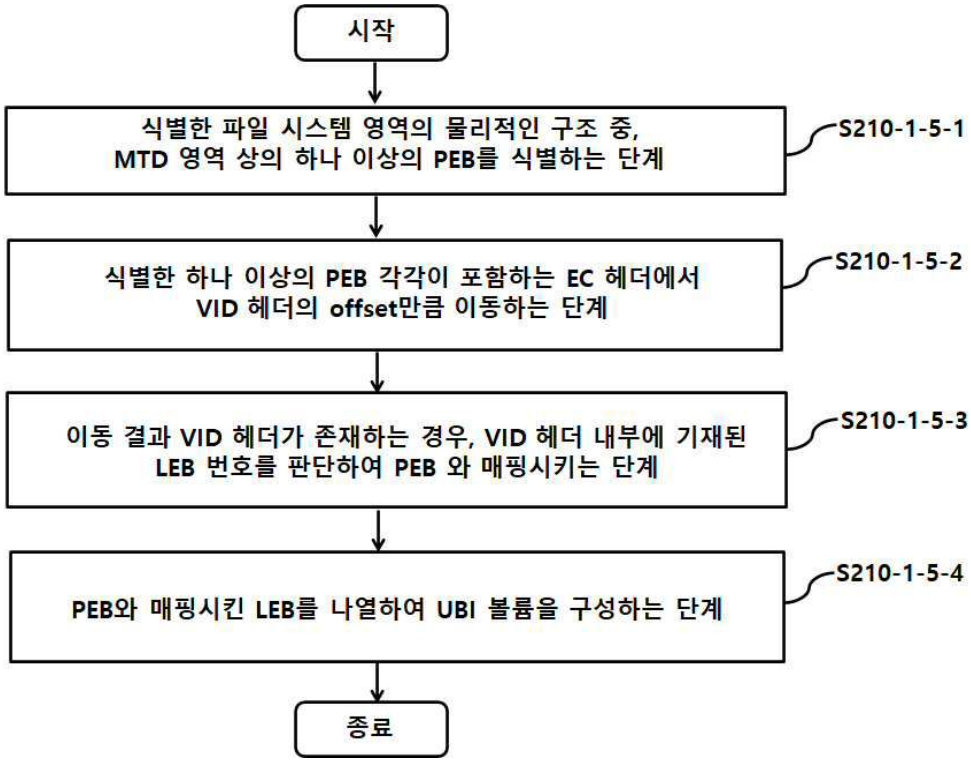
도면4


0x0000000
부트로더(Bootloader) 영역
0x0400000
커널(Kernel) 영역
0x0800000
이미지 스플래쉬(Splash) 영역
0x0C00000
램디스크(Ramdisk) 영역
0x2000000
파일시스템(UBIFS) 영역

도면5



도면6



도면7

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	EC header signature				UBI Version(0x04), Padding(0x05~0x07)				Erase counter							
0010	VID header offset				Data offset				Image seq				Padding			
0020	Padding															
0030	Padding												Header CRC			

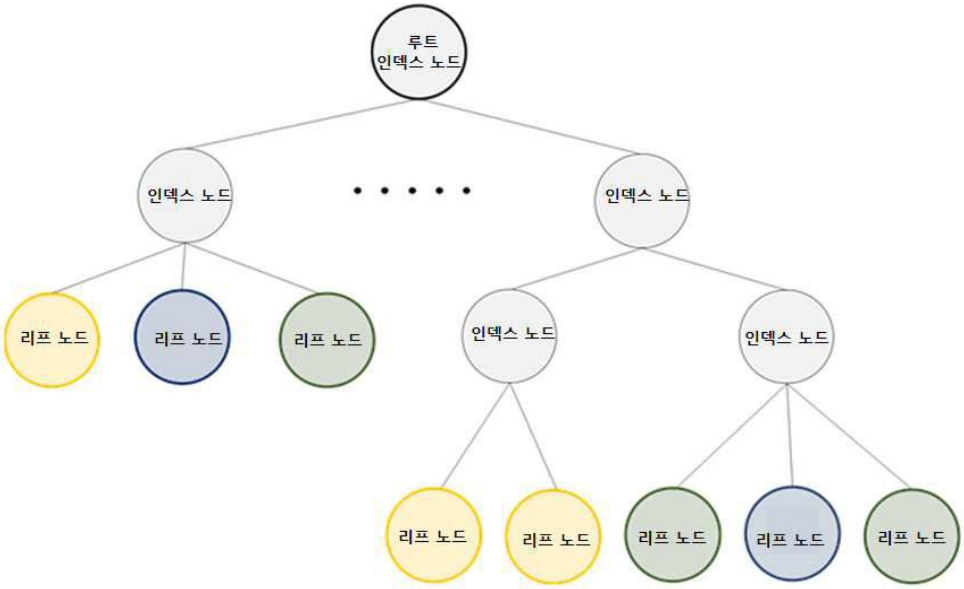
도면8

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	VID header signature				UBI Version(0x04), vol type(0x05), Copy flag(0x06), compact(0x07)				Volume id				LEB number			
0010	Padding				Data size				Used ebs				Data pad			
0020	Data CRC				Padding				Seq number							
0030	Padding												VID CRC			

도면9



도면10



도면11

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	Node signature				Node header CRC				Seq number							
0010	Node length				Node type	Group type	Padding		Node key							
0020	Node key								Create_sqnum							
0030	Access time seconds								Creation time seconds							
0040	Modification time seconds								Access time nanoseconds				Creation time nanoseconds			
0050	Modification time nanoseconds				.....											

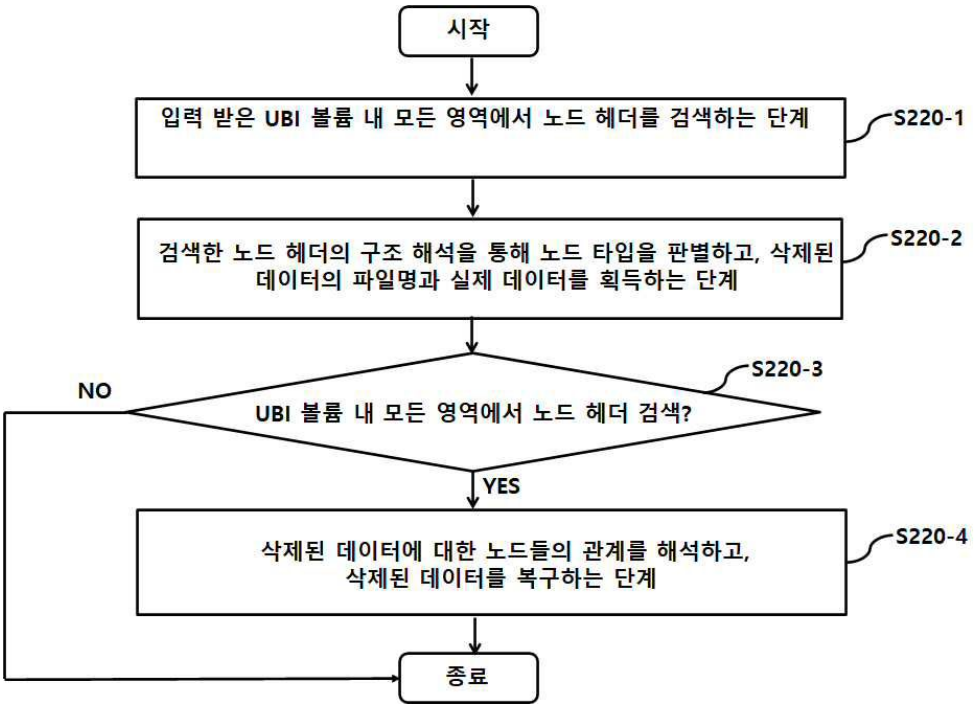
도면12

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	Node signature				Node header CRC				Seq number							
0010	Node length				Node type	Group type	Padding		Node key							
0020	Node key								Data length				Compr_type		Compr_size	
0030	Data															
.....	.....															

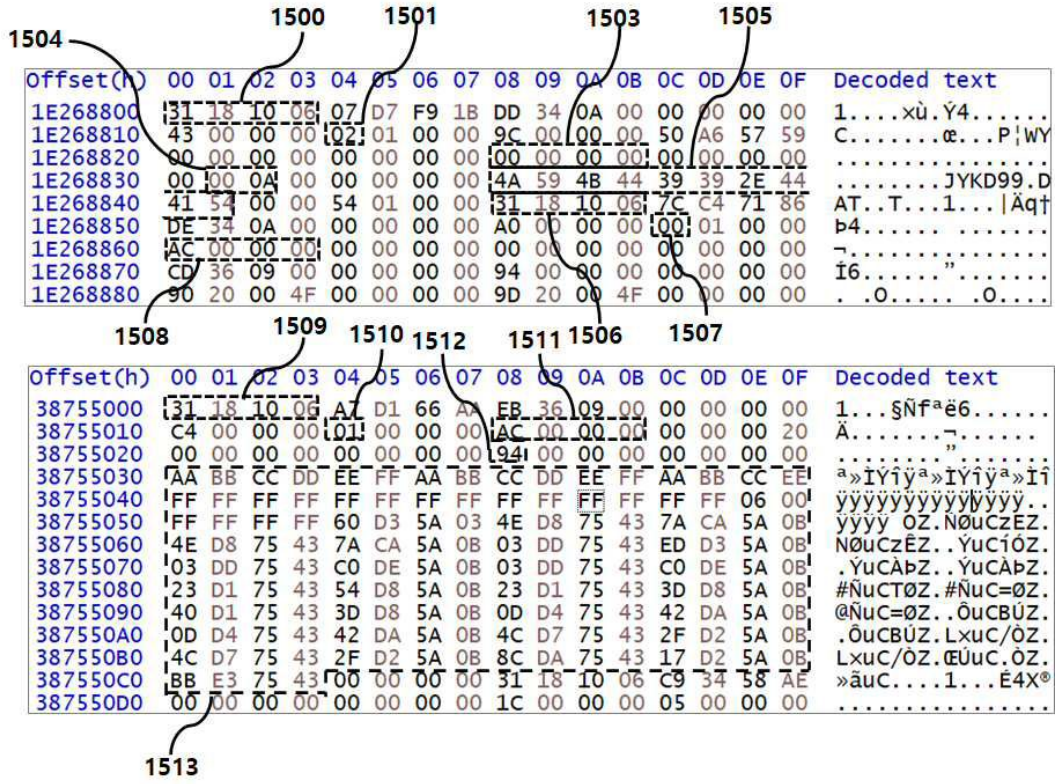
도면13

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	Node signature				Node header CRC				Seq number							
0010	Node length				Node type	Group type	Padding		Node key							
0020	Node key								Inum							
0030	Type	Length			Padding				File / Directory name							
0040	.....															

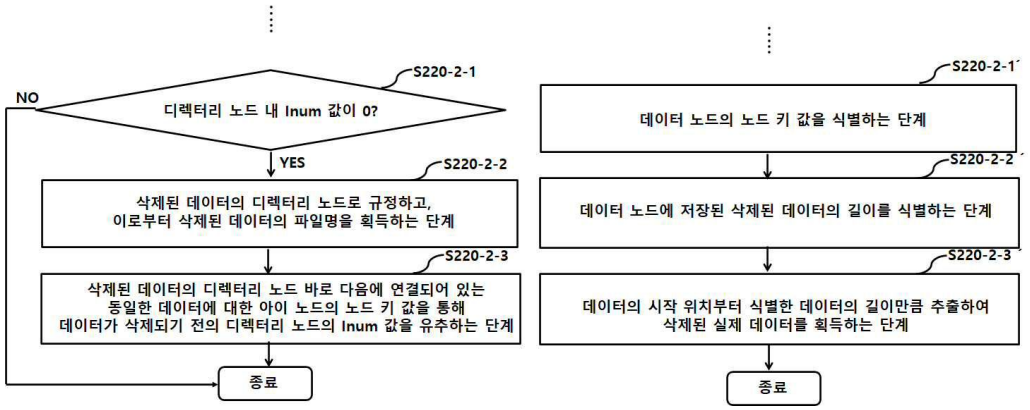
도면14



도면15


1500
1501
1503
1505
1504
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
1E268800 31 18 10 06 07 D7 F9 1B DD 34 0A 00 00 00 00 00 1....xù.Ý4.....
1E268810 43 00 00 00 02 01 00 00 9C 00 00 00 50 A6 57 59 C.......œ...P¦WY
1E268820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
1E268830 00 00 0A 00 00 00 00 00 4A 59 4B 44 39 39 2E 44 ....JYKD99.D
1E268840 41 54 00 00 54 01 00 00 31 18 10 06 7C C4 71 86 AT..T...1...|Äq†
1E268850 DE 34 0A 00 00 00 00 00 A0 00 00 00 00 01 00 00 Þ4...... ....
1E268860 AC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¬...............
1E268870 CD 36 09 00 00 00 00 00 94 00 00 00 00 00 00 00 Í6......”.......
1E268880 90 20 00 4F 00 00 00 00 9D 20 00 4F 00 00 00 00 . .O..... .O....
1508
1509
1510
1512
1511
1506
1507
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
38755000 31 18 10 06 A7 D1 66 AA EB 36 09 00 00 00 00 00 1...§Ñfªë6.....
38755010 C4 00 00 00 01 00 00 00 AC 00 00 00 00 00 00 20 Ä.......¬......
38755020 00 00 00 00 00 00 00 00 94 00 00 00 00 00 00 00 ....”.......
38755030 AA BB CC DD EE FF AA BB CC DD EE FF AA BB CC EE ª»ÌÝîÿª»ÌÝîÿª»Ìî
38755040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF 06 00 ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ..
38755050 FF FF FF FF 60 D3 5A 03 4E D8 75 43 7A CA 5A 0B ÿÿÿÿ`ÓZ.NØuCzÊZ.
38755060 4E D8 75 43 7A CA 5A 0B 03 DD 75 43 ED D3 5A 0B NØuCzÊZ..ÝuCíÓZ.
38755070 03 DD 75 43 C0 DE 5A 0B 03 DD 75 43 C0 DE 5A 0B .ÝuCÀÞZ..ÝuCÀÞZ.
38755080 23 D1 75 43 54 D8 5A 0B 23 D1 75 43 3D D8 5A 0B #ÑuCTØZ.#ÑuC=ØZ.
38755090 40 D1 75 43 3D D8 5A 0B 0D D4 75 43 42 DA 5A 0B @ÑuC=ØZ..ÔuCBÚZ.
387550A0 0D D4 75 43 42 DA 5A 0B 4C D7 75 43 2F D2 5A 0B .ÔuCBÚZ.L×uC/ÒZ.
387550B0 4C D7 75 43 2F D2 5A 0B 8C DA 75 43 17 D2 5A 0B L×uC/ÒZ.ŒÚuC.ÒZ.
387550C0 BB E3 75 43 00 00 00 00 31 18 10 06 C9 34 58 AE »ãuC....1...É4X®
387550D0 00 00 00 00 00 00 00 00 1C 00 00 00 05 00 00 00 ....
1513

도면16


S220-2-1
NO
디렉터리 노드 내 Inum 값이 0?
YES
S220-2-2
삭제된 데이터의 디렉터리 노드로 규정하고,
이로부터 삭제된 데이터의 파일명을 획득하는 단계
S220-2-3
삭제된 데이터의 디렉터리 노드 바로 다음에 연결되어 있는
동일한 데이터에 대한 아이 노드의 노드 키 값을 통해
데이터가 삭제되기 전의 디렉터리 노드의 Inum 값을 유추하는 단계
종료
S220-2-1'
데이터 노드의 노드 키 값을 식별하는 단계
S220-2-2'
데이터 노드에 저장된 삭제된 데이터의 길이를 식별하는 단계
S220-2-3'
데이터의 시작 위치부터 식별한 데이터의 길이만큼 추출하여
삭제된 실제 데이터를 획득하는 단계
종료