Demolition Derby

* 삭제 뒤 Extent 구조

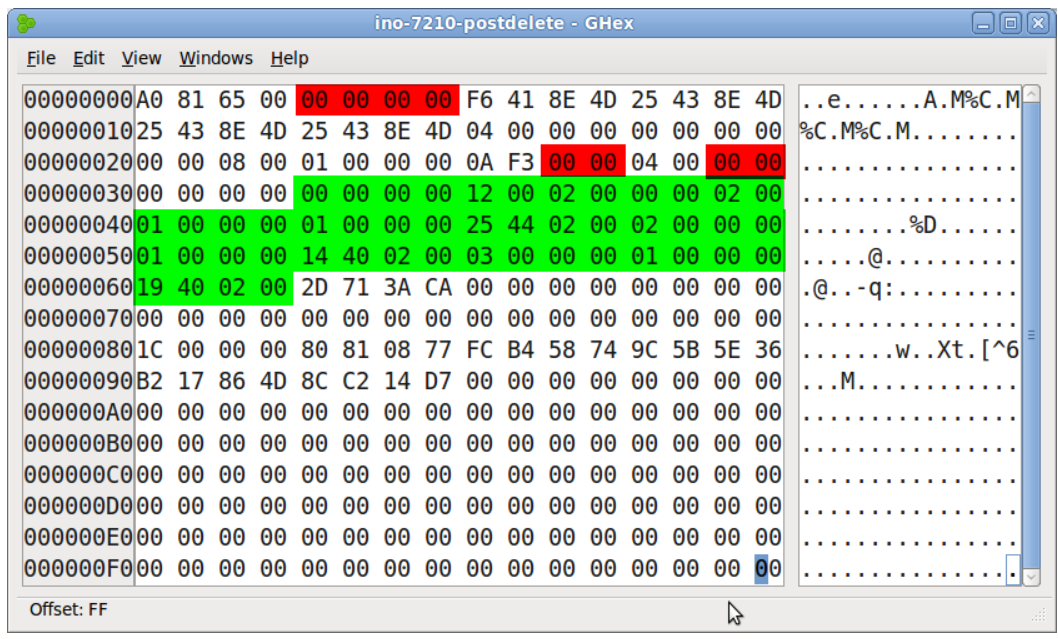
: /var/log/messages 파일을 삭제하면 어떻게 되나?

Q1. Inode 는 어떻게 변하는가?

Q2. Extent 트리 구조를 가지고 있는 131090 블록은 어떻게 되는가?

-

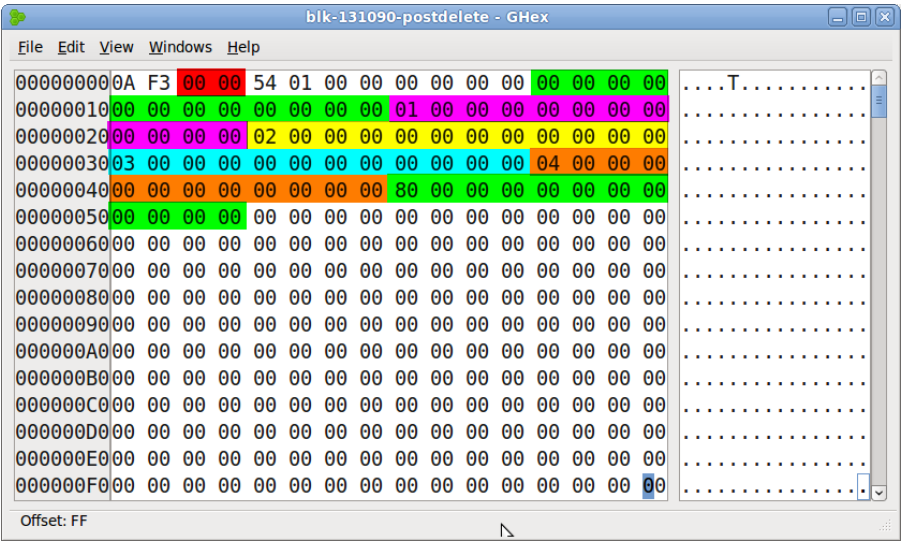
Ans ) 파일을 삭제했다. -> 로깅 데이터를 잃지 않았다. -> shred -u/var/log/messages 를 사용하여 데이터 블록을 널(null) 로 덮어썼다. 파일이 제거 되면 파일과 관련된 inode 와 131090 블록을 모두 덤프하고 16진 편집기에서 해당 파일을 살펴보았음.



* 바이트 4-7의 파일 크기가 0 으로 설정된다.
* 익스텐트 헤더구조의 익스텐트 수가 0으로 표시됨.
* 삭제 프로세스는 트리 깊이 필드를 다시 0 으로 설정함.
* 하지만 나머지 익스텐트 구조는 그대로이다.

: 익스텐트 구조가 남아있기 때문에 나머지 익스텐트 트리를 보유하는 데이터 블록의 블록 번호를 실제로 읽을 수가 있다. 또한 익스텐트 트리가 생성된 후에도 inode 에 남겨진 익스텐트의 기존 구조도 여전히 존재한다. -> 디스크에서 파일 청크를 찾을 수 있는 위치를 알려준다.

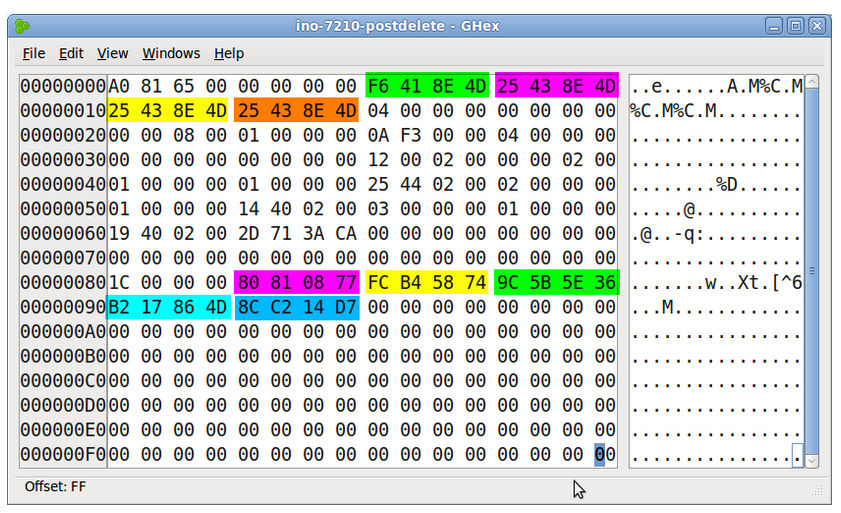
아래 사진은 131090 블록에 남아 있는 것들이다.



* 블록 시작부분에서 extent해더구조를 보면 extents의 수 필드가 0 으로 설정되어있는 것을 볼수 있다.
* 여기서 특이한점은 놀리 블록의 오프셋을 그대로 두었다.
* 이러한 점은 포렌식에서 힌트로 작용한다.

1. 원본 파일에 6개의 익스텐트가 있다고 말할 수 있다.
2. 처음 4개의 extent에 각각 하나의 블록이 포함되어 있음을 알수 있다.
3. 비슷하게 우리는 5번째 extent 가 124 블록을 포함하고 있다는 것을 알수 있다.
4. 마지막 extent 의 길이는 알지 못한다.
5. 시작 블록 주소와 그 extent 의 길이를 포함해서 Inode 자체에 변경되지 않은 extent 2~4 가 남아있다.

Ext4 에서 파일이 삭제될 때 타임스템프 값이 어떻게 변화하는지 알아보자.



Mtime seconds (16-19, yellow) 와 ctime seconds (12-15, purple) 을 보면 같은 값인 것을 알 수 있다. (파일이 삭제된 시간이다)

하지만 각 타임스템프에 대한 extra field 는 다르다.

Creation time 의 값은 완전히 변경 되지 않았으므로 삭제된 inode 는 해당 inode를 마지막으로 사용한 파일이 볼륨에서 생성된 시기를 알려준다.

Atime 값은 삭제 작업에 영향을 받지 않았지만 이 경우에는 시스템의 다른 작업이 atime 을 업데이트 해준다.

위에서 주황색으로 표시된 부분은 삭제시간으로 ext 파일 시스템에서는 이 삭제시간을 추적하지만 mtime 하고 ctime 이랑 같은 값이기 때문에 유용하지는 않다.