OS PA5

2020-15607 김연재

1. New data structures

pwd: proc의 cwd가 아닌 pwd를 추가했다. 이 pwd를 이용해서 현재 작업중인 directory의 path name의 full name을 쉽게 알 수 있게 하였다. 이는 fork, reap할 때, 할당되거나 없어지며 chdir를 통해서 변할 수 있다. cwd를 없애고 pwd만 이용해도 되지만 그렇게 되면 pwd를 이용해서 매번 cwd를 구해야하기 때문에 overhead가 날 것이라 생각했다. 반대로 pwd를 만들지 않는다면 cwd를 이용해서 pwd를 구하기가 힘들거라 생각했다.

과제의 내용대로 모든 directory에서 .와 ..을 제거했다.

2. Algorithm design

모든 entry들은 root data block안에 존재할 것이기 때문에 dirlookup과 dirlink는 root inode에서만 탐색하도록 만들었다. 그리고 특정 path의 inode를 확인하는 것 뿐만 아니라 특정 path의 directory의 inode도 쉽게 확인할 수 있도록 함수를 구현하는 것이 편하기 때문에 기존 nameiparent를 바꿔서 namei(path)로는 해당 inode를, nameiparent(path)로는 directory의 inode를 구하도록 해주었다.

sysflie을 통해서 file을 조작하는 방법은 link, unlink, create 이다(open이나 다른 함수에서는 create을 이용한다.)

link: old가 존재하는지, old와 new가 같은지, new가 생성되는 곳이 올바른 directory인지를 확인하고 만들어준다(type이 dir이 맞는지, 이미 존재하는 dir인지를 확인한다).

unlink: 삭제하려는 inode를 먼저 구하고 이 inode의 link count를 줄이고 root data directory에서 해당 entry를 줄인다.

create: link와 비슷하게 구현된다. File이 새로 생기는 dir이 올바른 곳인지를 확인해주고 맞다면 create한다.

성능을 높여주기 위해 hash table을 이용한다. Double hashing을 이용하며 처음에 dirlookup이 이루어질 때 모든 file 내용들을 hash table에 담아두고 dirlink를 할 때는 hash_insert를 하고 sys_unlink를 할 때는 invalidate을 해준다. 이렇게 해서 빠르게 full path가 들어올 때 inum을 찾아준다.

log와 inode data에 접근을 할 때 lock을 사용한다. 이때, sleep lock을 사용하고(log는 직접적으로 sleep lock을 사용하지 않지만 sleep과 wake를 하는 것을 보면 sleep lock으로 볼 수 있다) 있다.

그런데 이때 wakeup을 할 때 모든 proc에 대해 acquire하고 sleeping인 경우에 깨운다. 이에 대한 overhead가 크기 때문에 waiting queue를 만들어서 하나의 proc만 깨우도록 만들어주었다.

3. Testing and validation

usertests -q를 이용해서 변형된 file system이 제대로 작동하는 지를 확인하였다. 가장 처음에는 path가 들어올 때 이를 parsing하는 문제가 발생하여 잘못된 처리를 했었다. 그리고 이를 고친 이후, link와 create에서 parent가 존재하는 지만 확인하고 type을 확인하지 않아 문제가 발생했었다. 그리고 이후에 한번은 ilock을 잘못 사용하여 panic이 발생하는 문제가 생겼다. 이를 알고 전체적인 ilock을 잡고 put을 하는 과정을 구체화하여 전체적인 코드를 수정했다. 이후 모든 usertests -q가 돌아가는 것을 확인하고 우연하게 작동한 것일수도 있기 때문에 qemu를 끄지 않은 상태로 usertests -q와 mkdir, cd, ls 등을 여러 번 불러보며 작동되는 것을 확인하였다.