Bjarni Þór Sigurðsson (bths2@hi.is), Vor 2023

TÖL401G - Heimaverkefni 24

1

The 98 direct block pointers provide support for $98 \cdot 1\text{KB} = 98\text{KB}$ of space.

The one single indirect block provides support for $100 \cdot 1\text{KB} = 100\text{KB}$ of space.

The one double indirect block provides support for $100 \cdot 100 \cdot 1\text{KB} = 10000\text{KB}$ of space.

The max files size that can be managed is therefore $98\text{KB} + 100\text{KB} + 10000\text{KB} = 10198\text{KB}$.

2

If the root directory inode as well as the blocks it refers to are already in memory then we search through that data to find a reference to the `usr/` inode. This does not require disk access. We need to access the disk to load the `usr/` inode. When that is loaded we also need disk access to load the blocks it refers to. When those are loaded we search for a reference to the `bin/` inode. We need to access the disk to load the `bin/` inode and to load its blocks. Now we can search that data for a reference to the file `vi` to see if it exists.

3

a

A hard link is a reference that points to the inode of an existing file. In this case we need to add a new file entry to the ParentDirectory inode. This requires allocating a block/blocks for storing the length of the entry, the inode reference number, the file name length and the filename. We also need to add 1 to the #links metadata of the target file.

b

A symlink is a file containing the path to another file. This requires adding a new entry to the ParentDirectory inode like in part a. Also we need to allocate blocks for the new inode and for the path data.

c

Unlinking requires subtracting 1 from the target file inode. Also we need to remove the hardlink entry from the ParentDirectory inode and free its blocks.