CS 1550

Week 13

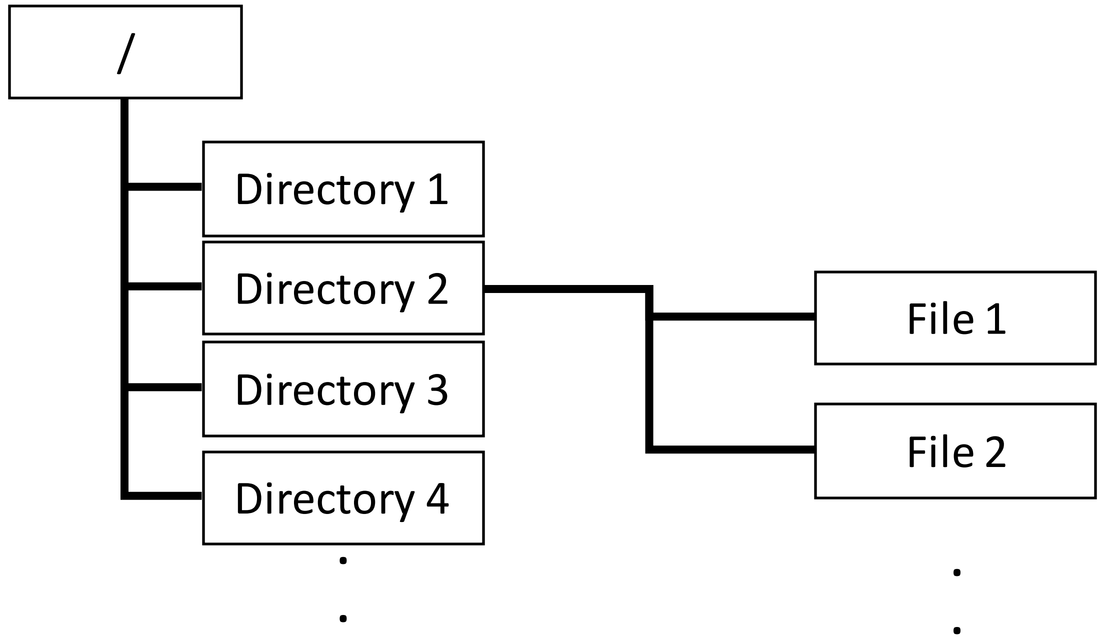
–

Project 4 cont.

TA: Henrique Potter

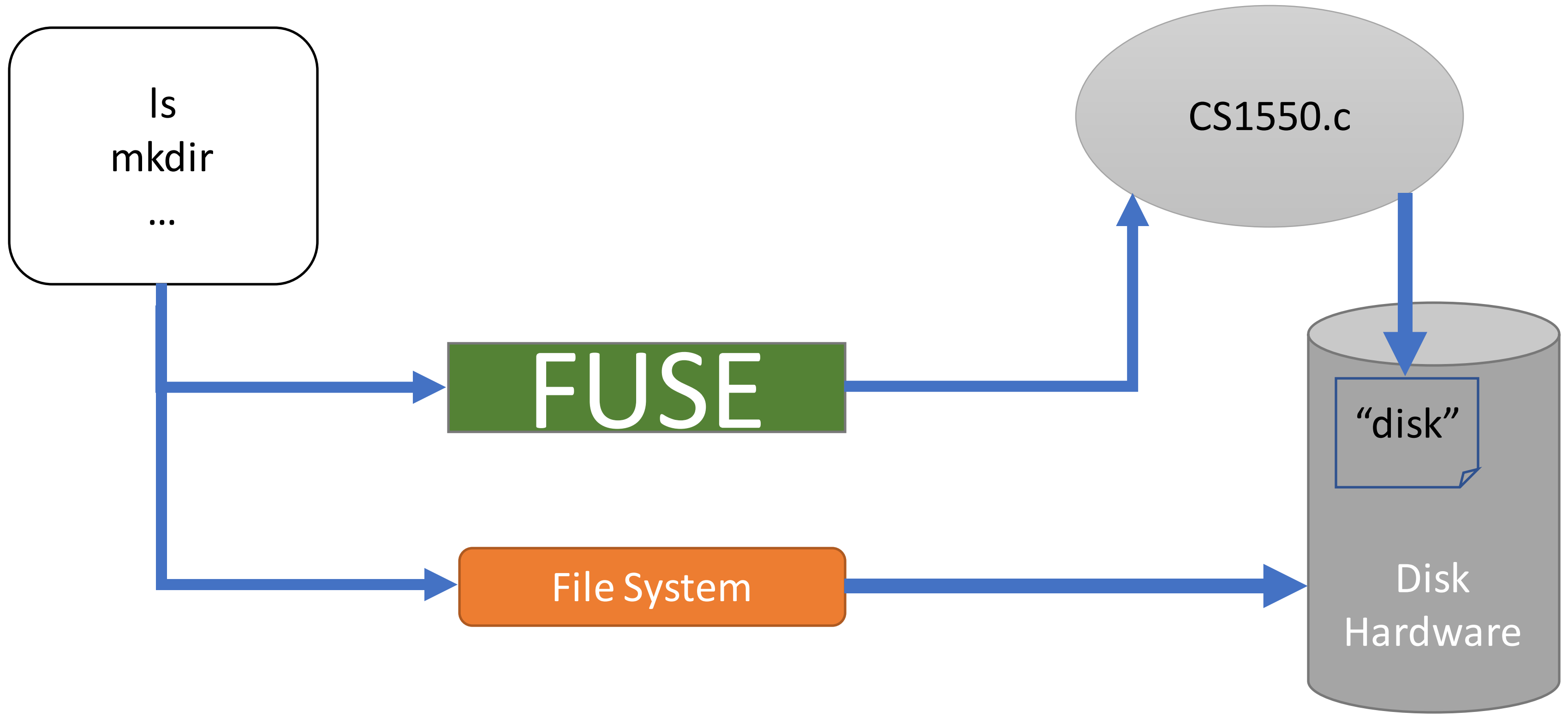
Project 4: File System

- Two-level directory system
  - The root directory “/” will only contain other subdirectories, and no regular files.
  - The subdirectories will only contain regular files, and no subdirectories of their own.



Create a file as “disk”

- We need to create a file as the “disk” for our file system. All metadata and file data in our file system will be stored in this “disk”.
  - Create a 5MB file: dd bs=1K count=5K if=/dev/zero of=.disk

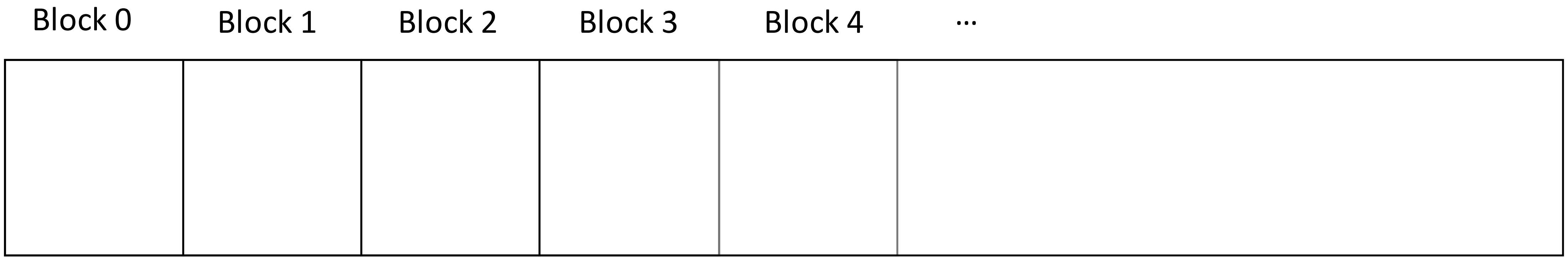


Syscalls

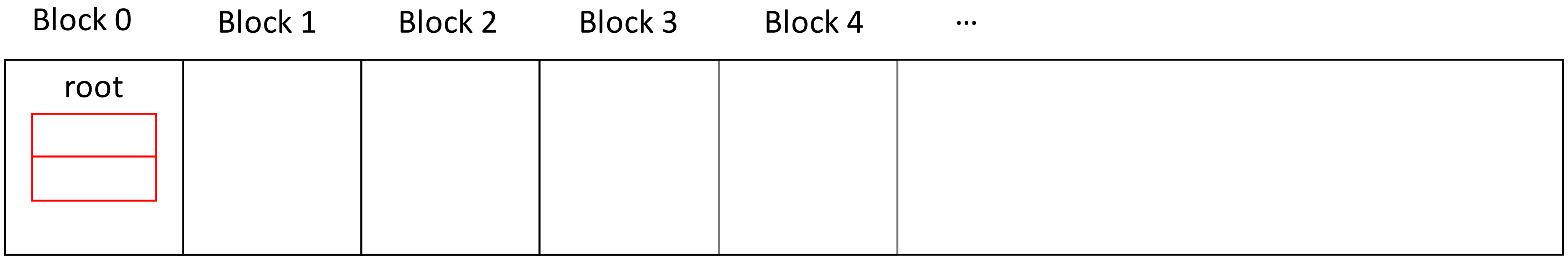
This project requires:

- **cs1550_getattr**
  - This function should look up the input path to determine if it is a directory or a file. If it is a directory, return the appropriate permissions. If it is a file, return the appropriate permissions as well as the actual size
- **cs1550_mkdir**
  - This function should add the new directory to the root level
- **cs1550_mknod**
  - This function should add a new file to a subdirectory

Initially, the ".disk" created contains all zeros



The block 0 is for root.

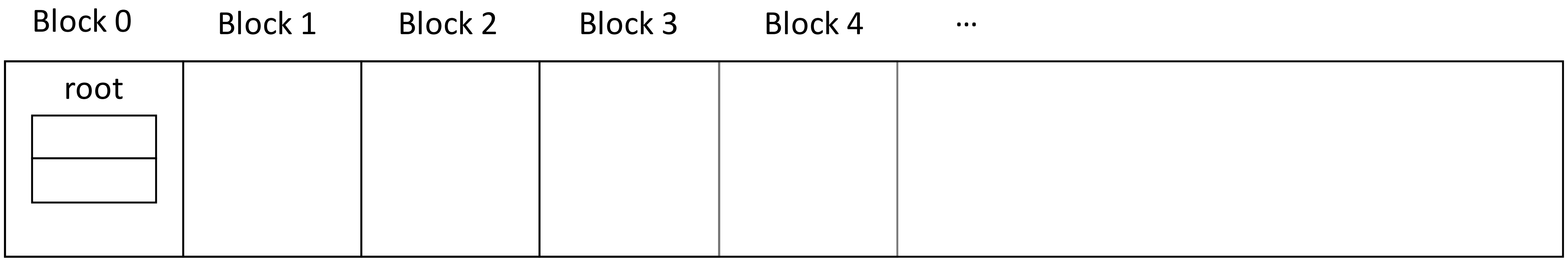


```
struct cs1550_root_directory
{
      long lastAllocatedBlock;   //The number of the last allocated block  = 0

      int nDirectories;    //How many subdirectories are in the root    = 0
                           //Needs to be less than MAX_DIRS_IN_ROOT
      struct cs1550_directory
      {
            char dname[MAX_FILENAME + 1];      //directory name (plus space for nul)
            long nStartBlock;          //where the directory block is on disk
      } directories[MAX_DIRS_IN_ROOT]; //There is an array of these
} ;
```

dname	nStartBlock
\0\0…	0
\0\0…	0

Now let's create a directory DirA:

We need to parse and validate the path "/DirA"



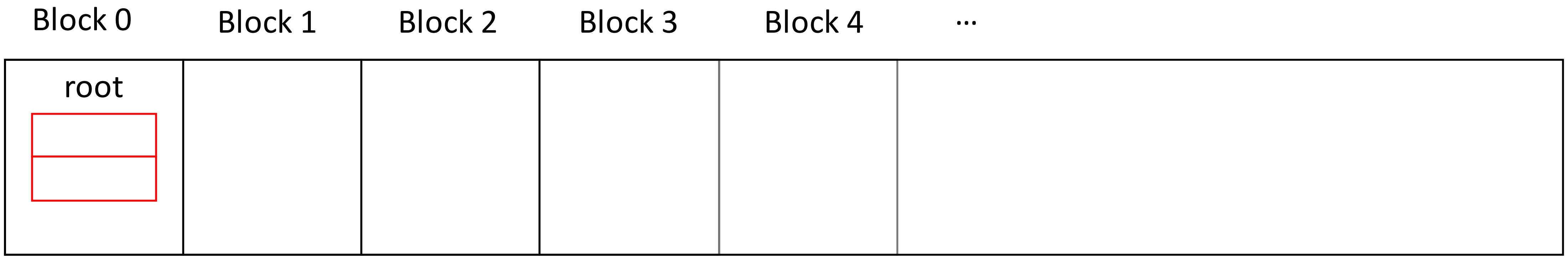
/DirA:

"/" : root directory of our file system

"DirA" : the subdirectory to be created inside root directory

Now let’s create a directory DirA:

We need to parse and validate the path “/DirA”



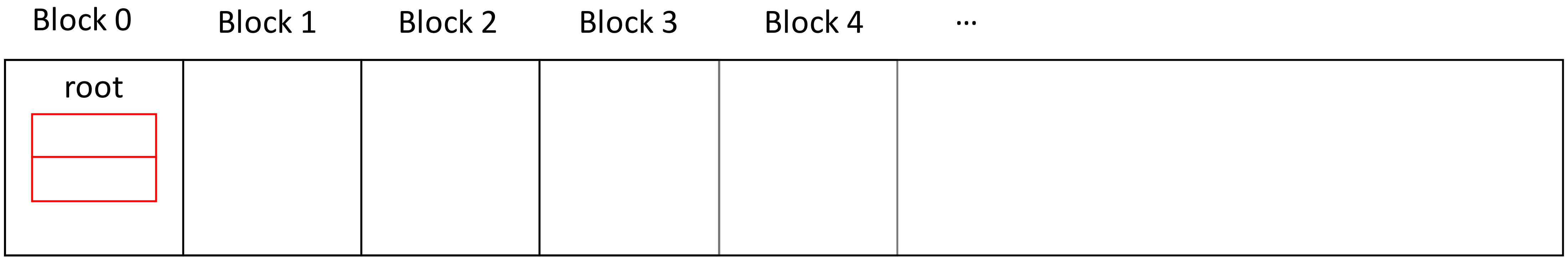
```
struct cs1550_root_directory
{
	long lastAllocatedBlock;   //The number of the last allocated block  = 0

	int nDirectories;   //How many subdirectories are in the root   = 0
	                    //Needs to be less than MAX_DIRS_IN_ROOT
	struct cs1550_directory
	{
		char dname[MAX_FILENAME + 1];      //directory name (plus space for nul)
		long nStartBlock;           //where the directory block is on disk
	} directories[MAX_DIRS_IN_ROOT]; //There is an array of these
} ;
```

dname	nStartBlock
\0\0…	0
\0\0…	0

DirA does not exist.
We can proceed.

Now let's create a directory DirA:

We need to allocate a block for directory metadata

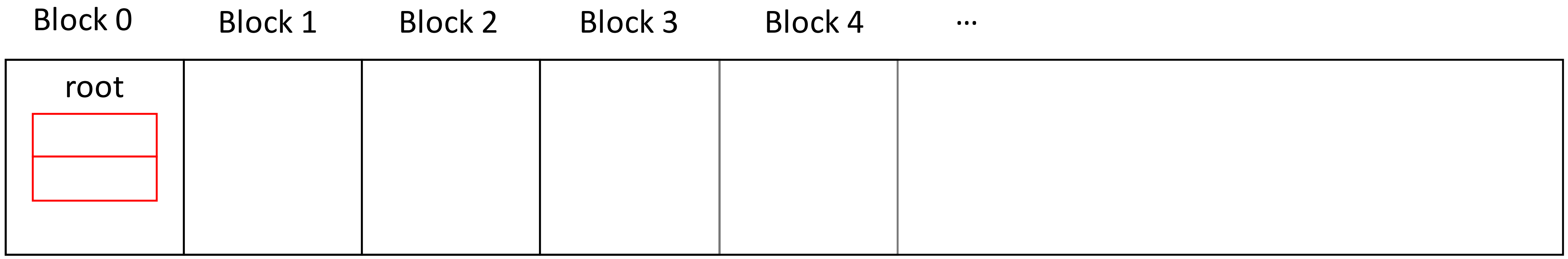


```
struct cs1550_root_directory
{
	long lastAllocatedBlock;   //The number of the last allocated block  = 0

	int nDirectories;   //How many subdirectories are in the root     = 0
	                    //Needs to be less than MAX_DIRS_IN_ROOT
	struct cs1550_directory
	{
		char dname[MAX_FILENAME + 1];      //directory name (plus space for nul)
		long nStartBlock;         //where the directory block is on disk
	} directories[MAX_DIRS_IN_ROOT]; //There is an array of these
} ;
```

dname	nStartBlock
\0\0…	0
\0\0…	0

Now let's create a directory DirA:

We need to allocate a block for directory metadata



```
struct cs1550_root_directory
{
	long lastAllocatedBlock;   //The number of the last allocated block

	int nDirectories;   //How many subdirectories are in the root
	                    //Needs to be less than MAX_DIRS_IN_ROOT
	struct cs1550_directory
	{
		char dname[MAX_FILENAME + 1];     //directory name (plus space for nul)
		long nStartBlock;            //where the directory block is on disk
	} directories[MAX_DIRS_IN_ROOT]; //There is an array of these
} ;
```

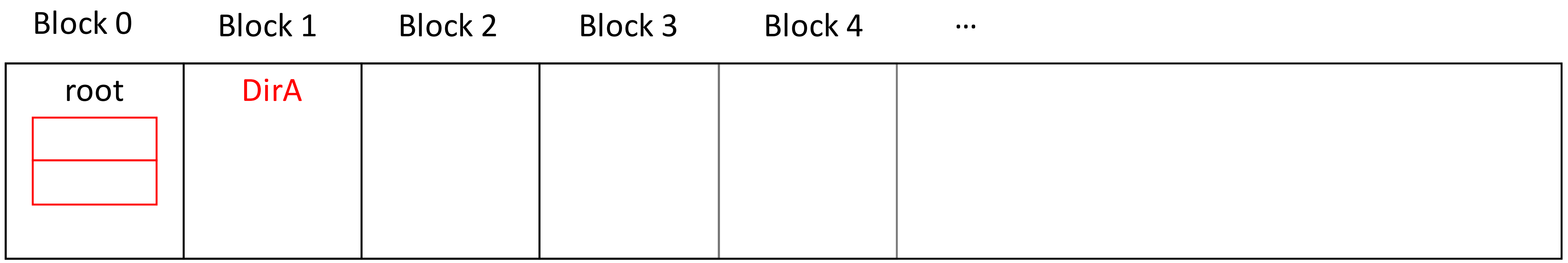
= 0, so we'll use block 1, and increase this number by 1

= 0

dname	nStartBlock
\0\0…	0
\0\0…	0

Now let's create a directory DirA:

Update the root metadata

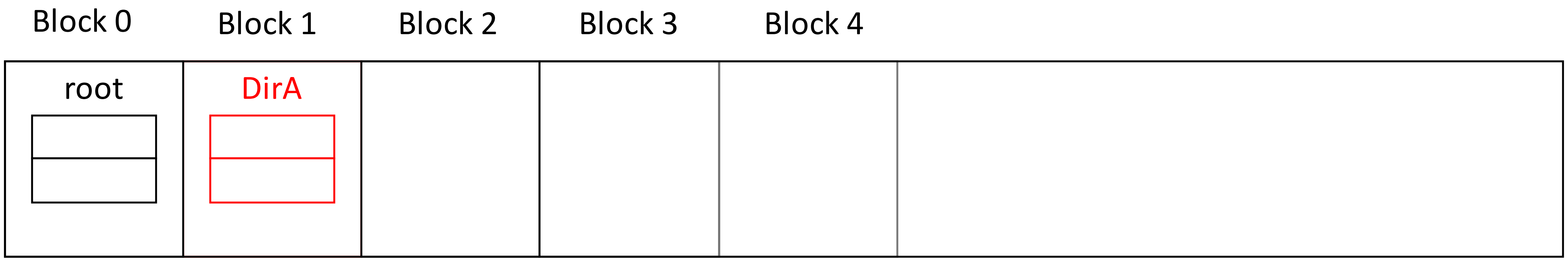


```
struct cs1550_root_directory
{
	long lastAllocatedBlock;   //The number of the last allocated block   = 0 ➔ 1

	int nDirectories;   //How many subdirectories are in the root   = 0 ➔ 1
	                    //Needs to be less than MAX_DIRS_IN_ROOT
	struct cs1550_directory
	{
		char dname[MAX_FILENAME + 1];      //directory name (plus space for nul)
		long nStartBlock;            //where the directory block is on disk
	} directories[MAX_DIRS_IN_ROOT]; //There is an array of these
} ;
```

dname	nStartBlock
DirA	**1**
\0\0…	0

Now let’s create a directory DirA:

<u>Update the directory metadata</u>

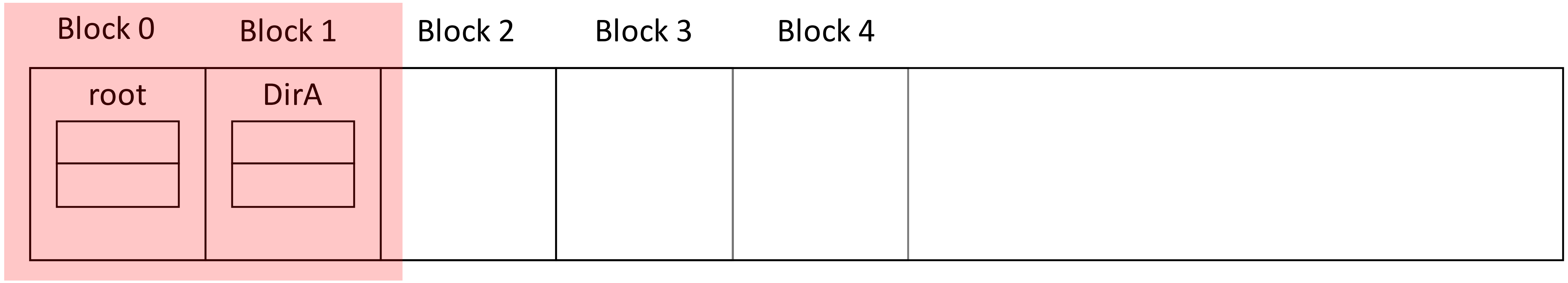


```
struct cs1550_directory_entry
{
	int nFiles;			//How many files are in this directory.
					//Needs to be less than MAX_FILES_IN_DIR

	struct cs1550_file_directory
	{
		char fname[MAX_FILENAME + 1];	//filename (plus space for nul)
		char fext[MAX_EXTENSION + 1];	//extension (plus space for nul)
		size_t fsize;			//file size
		long nIndexBlock;		//where the index block is on disk
	} files[MAX_FILES_IN_DIR];		//There is an array of these
};
```

= 0

Fname	nIndexBlock
\0\0..	0
\0\0...	0

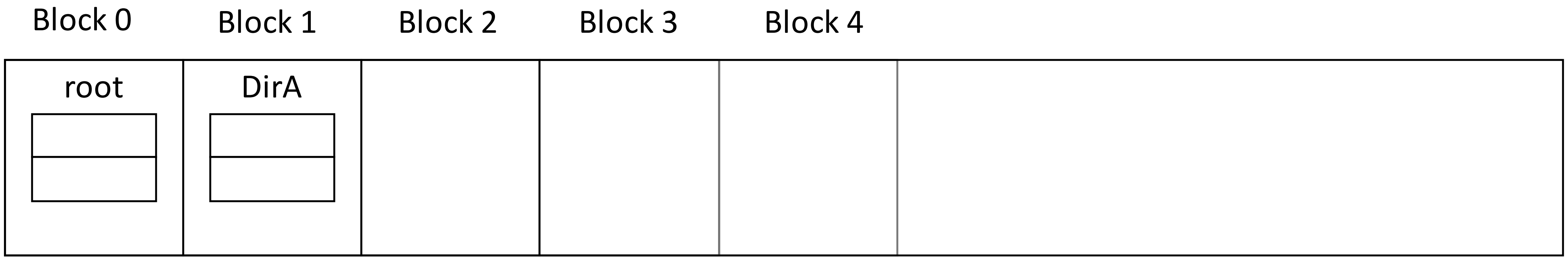
Now let's create a directory DirA: <u>Save changes to "disk"</u>



The creation of "DirA" modifies data inside block 0 and block 1, so we need to save these changes into "disk".

Now let’s create a file “FileA.txt” inside DirA:

Parse and validate the path “/DirA/FileA.txt”

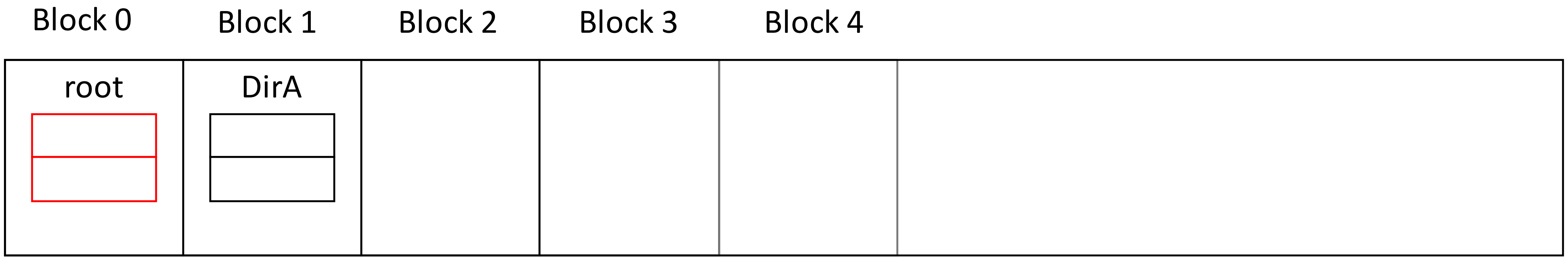


/DirA/FileA.txt:

“/”	: root directory of our file system
“DirA”	: the subdirectory that contains our target file
“FileA”	: the target file name
“txt”	: the target file extension

Now let's create a file "FileA.txt" inside DirA:

Parse and validate the path "/DirA/FileA.txt"



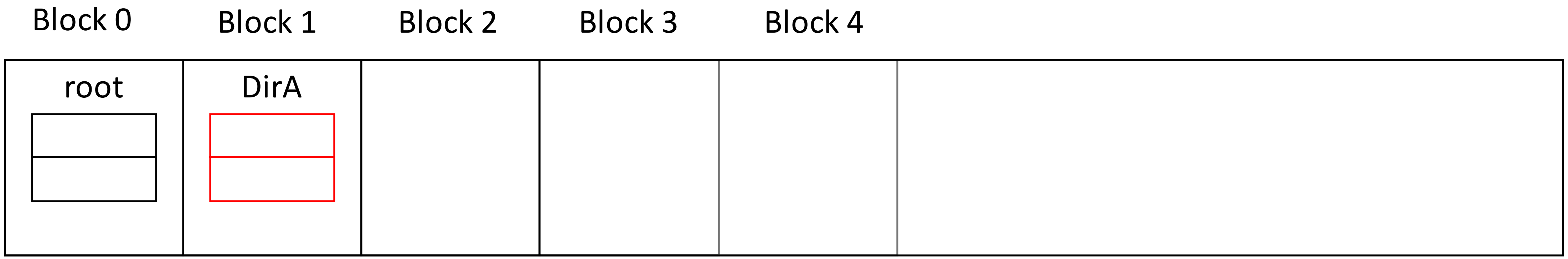
```
struct cs1550_root_directory
{
      long lastAllocatedBlock;   //The number of the last allocated block  = 1

      int nDirectories;   //How many subdirectories are in the root      = 1
                          //Needs to be less than MAX_DIRS_IN_ROOT
      struct cs1550_directory
      {
            char dname[MAX_FILENAME + 1];     //directory name (plus space for nul)
            long nStartBlock;           //where the directory block is on disk
      } directories[MAX_DIRS_IN_ROOT]; //There is an array of these
} ;
```

dname	nStartBlock
DirA	**1**
\0\0…	0

/DirA exists

Now let’s create a file “FileA.txt” inside DirA:

Parse and validate the path “/DirA/FileA.txt”



```
struct cs1550_directory_entry
{
	int nFiles;                     //How many files are in this directory.
	                                //Needs to be less than MAX_FILES_IN_DIR

	struct cs1550_file_directory
	{
		char fname[MAX_FILENAME + 1];      //filename (plus space for nul)
		char fext[MAX_EXTENSION + 1];      //extension (plus space for nul)
		size_t fsize;                   //file size
		long nIndexBlock;               //where the index block is on disk
	} files[MAX_FILES_IN_DIR];              //There is an array of these
};
```

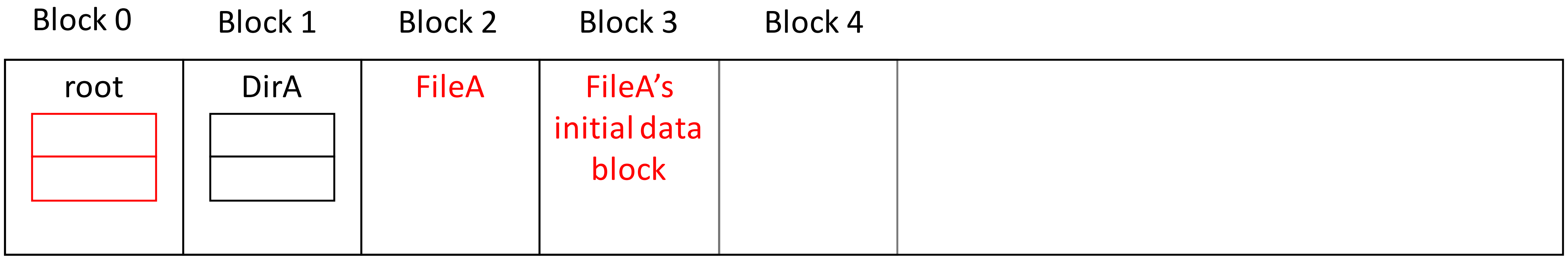
= 0

Fname	nIndexBlock
\0\0..	0
\0\0...	0

/DirA has no file named FileA.txt,
So we can proceed.

Now let's create a file "FileA.txt" inside DirA:

Allocate two blocks for FileA, and update root & dirA & FileA metadata

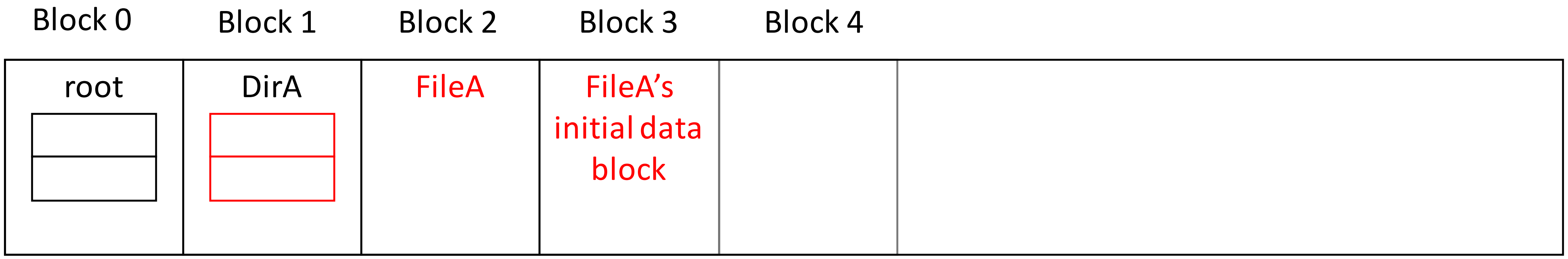


```
struct cs1550_root_directory
{
        long lastAllocatedBlock;   //The number of the last allocated block   = 1 ➔ 3, use block 2 & 3 for FileA

        int nDirectories;   //How many subdirectories are in the root   = 1
                            //Needs to be less than MAX_DIRS_IN_ROOT
        struct cs1550_directory
        {
                char dname[MAX_FILENAME + 1];      //directory name (plus space for nul)
                long nStartBlock;          //where the directory block is on disk
        } directories[MAX_DIRS_IN_ROOT]; //There is an array of these
} ;
```

dname	nStartBlock
DirA	**1**
\0\0…	0

Now let's create a file "FileA.txt" inside DirA:

Allocate two blocks for FileA, and update root & dirA & FileA metadata



```
struct cs1550_directory_entry
{
      int nFiles;                        //How many files are in this directory.
                                         //Needs to be less than MAX_FILES_IN_DIR

      struct cs1550_file_directory
      {
            char fname[MAX_FILENAME + 1];       //filename (plus space for nul)
            char fext[MAX_EXTENSION + 1];       //extension (plus space for nul)
            size_t fsize;                    //file size
            long nIndexBlock;                //where the index block is on disk
      } files[MAX_FILES_IN_DIR];             //There is an array of these
};
```

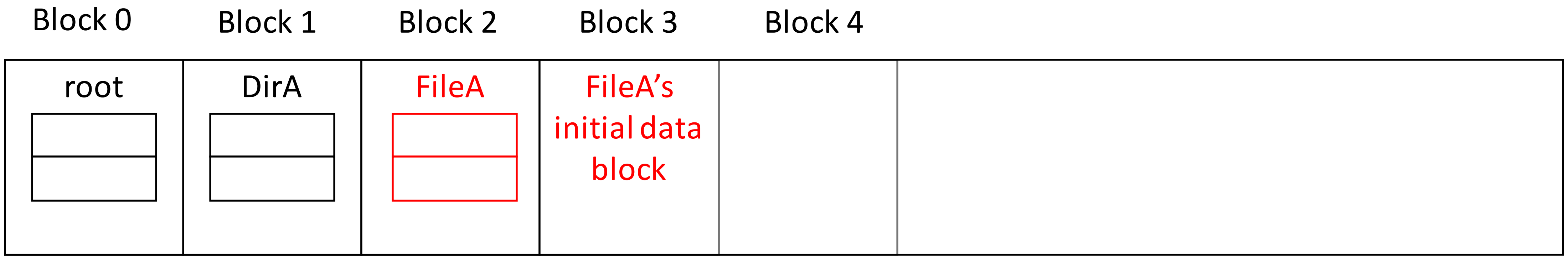
= 0 → 1

Fname	nIndexBlock
FileA	**2**
\0\0…	0

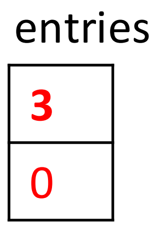
Set fext, fsize as well

Now let’s create a file “FileA.txt” inside DirA:

Allocate two blocks for FileA, and update root & dirA & FileA metadata

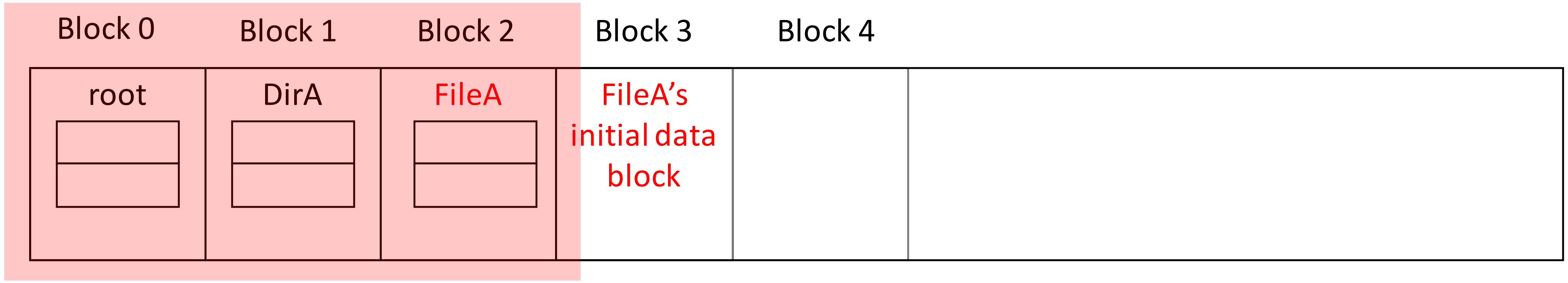


```
struct cs1550_index_block
{
      //All the space in the index block can be used for index entries. Each index
      //entry is a data block number.
      long entries[MAX_ENTRIES_IN_INDEX_BLOCK];
};
```



Now let's create a directory DirA:

Save changes to "disk"



The creation of "/DirA/FileA.txt" modifies data inside block 0, 1, and 2.
We don't have to modify the initial data block for fileA.
So we need to save these changes (block 0,1,2) into "disk".

Hints

- In each function, you'll open and read metadata from ".disk", remember to save changes and close ".disk" before return

- Example of reading root metadata using fopen/fread

```
struct cs1550_root_directory       root_dir;
FILE * fp;
// open .disk for binary read/wrirte
fp = fopen (".disk",  "rb+");
// read 1 item whose size is sizeof(struct  cs1550_root_directory)  to root_dir.
fread  (&root_dir,  sizeof(struct  cs1550_root_directory),   1,  fp);
```

  - You may use fseek to set the position indicator for reading/writing

Hints

- Test commands:
  - ls (with/without –al option), e.g., ls –al testmount
  - mkdir, e.g., mkdir testmount/DirA
  - echo, e.g., echo “” > testmount/DirA/FileA.txt
- Run you program with –d option for debugging
  - You’ll need two terminal windows, one for running you program and showing debug messages, the other for running test commands
  - You’ll see the triggered syscalls in the debug messages