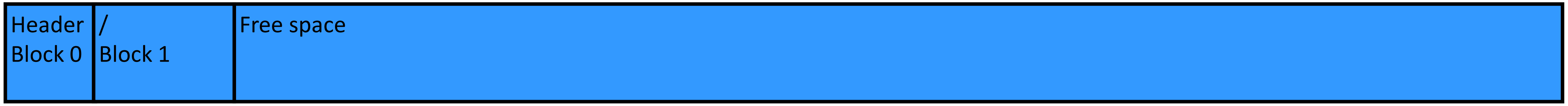


Header

Location: *fsptr

Header contains needed information about the filesystem and root directory attributes.
Will implement an integrity check to see if the filesystem has been initialized.

```
struct _fsHeader{
  size_t fsSize;
  size_t blockSize;
  int totalBlocks;
  int freeBlocks;
  fileAttr rtAttr;
  block_t firstFree;
  block_t lastFree;
```

Root Directory

Location: *fsptr + blockSize
Type: _dirBlock

Directory Block

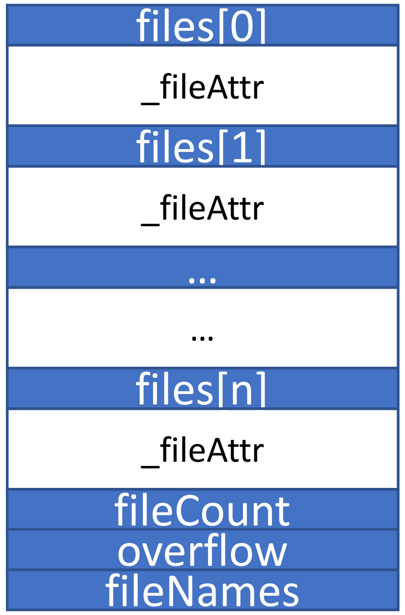
A directory block holds file attributes for files and directories within that directory. If a directory contains more than *n* files, a new block is allocated of struct _dirOverflowBlock, which looks just like the _dirBlock with only the files array and overflow pointer; creating a linked list of directory blocks.

Free Space

Free space will be managed using a linked list located on the free space itself. Each free block contains a struct _freeBlock, which has the number of the next free block.
Upon initialization, each block will simply point to its adjacent block.
Free space is allocated to files by block.
When a block is freed, a _freeBlock struct is created on that block and the linked list is updated accordingly.
Free blocks are zeroed out upon allocation.

```
struct _freeBlock{
  block_t next;
}
```

```
struct _fileAttr{
  offset_t name;
  block_t iNode;
  off_t size;
  nlink_t nlink;
  time_t atim;
  time_t mtim;
}
```

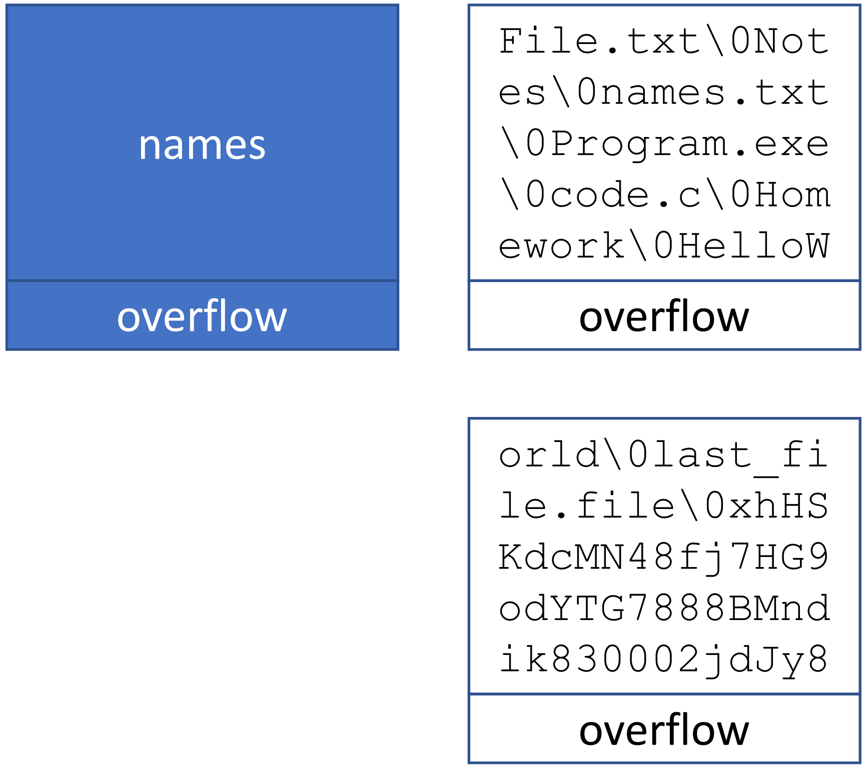


```
struct _dirBlock{
  struct _fileAttr files[(blockSize-(2*sizeof(block_t))-
           sizeof(int))/sizeof(_fileAttr)];
  int fileCount;
  block_t overflow = 0;
  block_t filenames = 0;
}
struct _dirOverflowBlock{
  struct _fileAttr files[(blockSize-sizeof(block_t))/sizeof(_fileAttr)];
  block_t overflow = 0;
}
```

Filenames

Each Directory Block contains an offset to a block containing a char array of names separated by \0 characters. The filesystem uses this block in conjunction with the offset in a file's attributes to find a file's name.
Some housekeeping must be done when files are renamed or removed.
The last bytes of the block gives an offset to a new block where the string is continued; creating a linked list of name blocks.

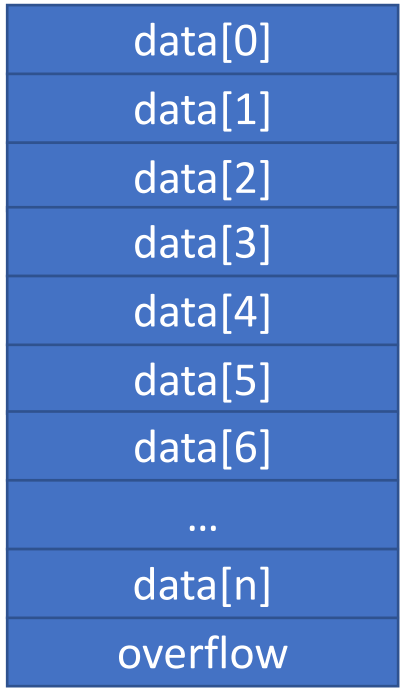
```
struct _filenameBlock{
  char names[blocksize-sizeof(block_t)];
  block_t overflow = 0;
```


names
overflow
File.txt\0Notes\0names.txt\0Program.exe\0code.c\0Homework\0HelloW
overflow
orld\0last_file.file\0xhHSKdcMN48fj7HG9odYTG7888BMndik830002jdJy8
overflow

Files

Directory blocks contain file attributes which contain the starting I-Node Block for each file. The file's I-Node Block has a data array containing offsets to all blocks the file uses (in sequential order). If the data array is full, the overflow offset designates a block where the array is continued; creating a linked list of i-nodes.

```
struct _fileBlock{
  offset_t data[(blkSize-sizeof(_fileBlock*))];
  block_t overflow = 0;
```



File System Attributes

Block Size = 4096 bytes = 4KB

Max filename length = N/A

Allocating free block = O(1)
Freeing used block = O(n)
Finding read/write offsets = O(n)
Locating filenames = O(n)