**Assignment 3**

Branko Bajic; Matt Hylton

Our filesystem was designed using contiguous file allocation. We used an array to mimic memory. Upon initialization (when readModules2() is ran) the files are copied from memory into the array so that we can work with them.

Read API:

Parameters: (char \*buf, size\_t nbyte)

Return Value: size\_t nbyte

Read(), beginning at the initial offset (aka the head index of the file being read), copies the character at that index into memory and increments the offset by 1. Thus when it is called again it writes the second character in memory and so on. If an end of file character is detected, it returns 0. Otherwise it returns the length of the bytes read.

Write API:

Parameters: (char a, size\_t nbyte)

Return Value: size\_t nbyte

Write() has 2 cases:

**Case 1:** The original file hasn’t been wiped from the array/memory yet. This case occurs the first time write() is called on a file. Here we set every block that belongs to the file to null and write null to the virtual memory. Then we write the first character to the very last available free block in the array and set the start pointer for the ram file to point to that block.

**Case 2:** The original file has already been wiped from the array/memory. In this case we simply write the character into the next available free block to the right of the ram file’s head pointer.

As an example:

