Team Blue Members:

Bhargava Kadiyala

Canxin Tang

Haolong Deng

Miguel Abalos

Mowtee Sailan

File System Design

**What our Volume Control Block looks like/contains:**

* Block number or index
* Volume ID for unique identifier for the volume
* Volume name
* Block size
* Volume size
* Total block count in the volume
* Count of available free blocks in the volume
* Count of allocated blocks in the volume

| #include <stdio.h> #include <stdlib.h> #include <time.h>  #define BLOCK\_SIZE 512  typedef struct VolumeControlBlock {  int \*bitmap;  int tBlock; // total number of blocks  int fBlock; // number of free blocks   int rootDirectory; // location of root directory  long signature; // signature for volume   time\_t time; } VCB;  VCB\* createVCB(int sizeOfVolume) {  // Create local variable to calculate total blocks  int calculatedBlocks = sizeOfVolume / BLOCK\_SIZE;    VCB \*vcb = (VCB\*)malloc(sizeof(VCB));   vcb->tBlock = calculatedBlocks; // Set total blocks in VCB  vcb->fBlock = calculatedBlocks; // Initially, all blocks are free  vcb->rootDirectory = 1; // Root directory starts from the first block  vcb->time = time(NULL);   vcb->signature = 0xABCDEF1234567890; // Set signature  // Calculate bitmap size  int bitmapSize = (calculatedBlocks + 31) / 32;  //Allocate bitmap memory by multiplying the bitmap size with the size  of an int  vcb->bitmap = (int\*)malloc(bitmapSize \* sizeof(int));  return vcb; }  int main() {  //Set the volume size and convert to bytes (in this case 256)  int volumeSize = 256 \* 1024;  VCB \*vcb = createVCB(volumeSize);   printf("Total Blocks: %d\n", vcb->tBlock);  printf("Free Blocks: %d\n", vcb->fBlock);  printf("Root Directory Location: %d\n", vcb->rootDirectory);  printf("Signature: 0x%lx\n", vcb->signature);  printf("Time Accessed: %s", ctime(&vcb->time));   // Free the memory  free(vcb->bitmap);  free(vcb);    return 0; } |
| --- |
|  |
|  |

**How we will track the free space:**

We will be using a bitmap to track free space, allowing us to efficiently find available blocks quickly while using minimal memory. In this system, bits in the bitmap are set to 1 when blocks are freed and to 0 when they are allocated. For larger file systems, we can incorporate grouping, where a fixed number of blocks are grouped together, enhancing both speed and efficiency. This approach ensures that determining the status of a block, whether it's free or allocated, will be faster and more straightforward.

**What our directory entry will look like:**

| **#include <stdio.h> #include <stdlib.h> #include <string.h> #include <stdbool.h> #include <time.h>  typedef struct Directory {**  **time\_t time;// date that file was created or accessed, 8 bytes  int size; // size  int location; // location  bool directory; // is it a directory 1 byte**  **char name[64]; // name  } DIR;  int main() {   DIR \*dir = malloc(sizeof(DIR));  strcpy(file->name, "name");  file->size = 256;  file->location = 0;  file->directory = false;  file->time = time(NULL);   printf("name: %s\n", file->name);  printf("size: %d\n", file->size);  printf("location: %d\n", file->location);  printf("Directory: %s\n", file->directory ? "true" : "false");  printf("time/date: %s\n", ctime(&file->time));   free(file);  return 0; }** |
| --- |

**Metadata that we want to have in the file system:**

* A character array to store the name of the file or directory
* Size of the file
* Location of the file or directory
* A boolean variable to determine whether this entry is a directory
* Timestamp of when the file was created or last accessed