Hello everyone, my name is **Sarah**, team leader of group 2, and it is my pleasure to introduce our implementation of a Unix-style file system.

CHANGE

Our goal was to design a simple yet effective program to meet the project requirements. So, we drafted a pseudocode for better planning and divided the workload equally.

CHANGE

To put it simply, the Unix file system is a logical method of organizing and storing large amounts of data.

CHANGE

Files in Unix systems are organized into a **multi-level hierarchy** structure called a directory tree, which represents the tree-like structure that forms the file system.

CHANGE

The **root directory** is identified by the slash symbol, and it encompasses all files and directories in the file system.

CHANGE

As a result, each file or directory is uniquely identified by its name, the directory that it resides in, and a unique identifier called an inode.

CHANGE

Our program implements an Inode-based file system through random access file and maintains 16MB of storage space.

CHANGE

The **random-access file** behaves like a large array of bytes stored in the file system. To perform functions in filesystem, we first seek to the specific point using the seek() method that sets the pointer position to the beginning point, followed by the operation such as read and write.

CHANGE

The storage space is further divided into blocks of 1KB, resulting in a maximum of 16,384 blocks.

CHANGE

Further, it is divided into five parts: Superblock, Inode , Block Bitmap, Inode space, and Data space.

CHANGE

First, the **Superblock** contains the filesystem meta data, which identifies the filesystem's structure and the amount of disk space it occupies.

CHANGE

As per a small file system standard, our program uses an inode of 128 bytes. The I-node table/space contains the metadata of the files and pointers to their respective data blocks.

CHANGE

Consequently, **each inode** contains ten direct pointers pointing to data blocks, and one indirect pointer pointing to an inode without data headers before pointing to data blocks.

CHANGE

The **I-node bitmaps** occupy a segment of memory, consisting of 0s and 1, with 0 indicating a free I-node and 1 indicating that it is already occupied. When modifying a file system, this technique helps find a free location on the inode table quickly. The same applies to **block bitmaps.**

CHANGE

Moreover, after file system allocation, the rest of the pending space is allocated to **data space**, which stores the raw data in blocks.

CHANGE

Lastly, the **main () function** provides a simple interface through which the user can interact with the filesystem and offers help and exit options. The help function lists down the supported commands, and the terminal reads the user input command and executes the corresponding functions to provide the desired output.

CHANGE

Finally, the exit command terminates the program. At termination, the data is stored in a file named ufs.sys

CHANGE

and can be loaded the next time the program is launched.

CHANGE

The users can check how much disk space is consumed by the system through the sum command, which displays the usage details of inode and blocks.

CHANGE

Moreover, the filesystem can be restored to its default state using the format command.

CHANGE

Hi, I'm **Ryan** and I dealt with functions that create and delete files or directories and the related functions that get addresses and format paths for the corresponding features to work.

CHANGE

1. To Create a file, the command is createFile

This feature **creates a new file** with fileName and fileSize as the parameter. It calls the addFile () function, which checks the file name, character length and duplicate names. Then the filesize is checked to see whether the size is smaller than what we allocated. If any of those conditions are wrong, it will output an error message.

CHANGE

When both conditions are met, the program checks if there is enough space in the disk and allocates it to a new block and inode.

CHANGE

Next, to **Create a directory** we call the addDir () function. The function takes file name as parameter and initializes the file size to zero. Before saving, the function checks if it is the first directory being created.

CHANGE

If it is not, it traces back to the parent directory. It then checks for errors similar to that of create file and creates it.

CHANGE

To **delete a file,** the DeleteFile command calls the rmFile () function which takes the name of the file to be deleted as a path parameter.

CHANGE

It searches for its inode address from the current directory and deletes it.

CHANGE

Similar to DeleteFile, the **deleteDir command** calls the rmDir () function which takes the name of the directory to be deleted.

CHANGE

However, this function can only be executed when the directory is currently not in use.

CHANGE

If the user tries to delete the current working directory it will run into an error and print an error message that it can't delete the current working path. after all functions,

CHANGE

the superblock, inodebmp, and blockbmp are updated respectively.

CHANGE

In addition, we defined three functions to retrieve the addresses, namely getDirAddr (), getPathInodeAddr () and getBlockAddr ().

The getDirAddr () retrieves the directory item address,

the getPathInodeAddr () function returns the inode path address based on the path and the current Address

and the getBlockAddr () maps the data block address for the current inode.

CHANGE

lastly, we do note that we successfully followed the requirement of creating nested directories as shown below.

I will now leave the floor to von to present her part. Thank you

CHANGE

Hello my name is **Vondrea** and i will be explaining the update and read functions.

CHANGE

The ChangeDir command is used to **change a directory** by specifying a path. This path is then used to get the inode address of the directory.

CHANGE

If the path is valid, it successfully changes the working directory otherwise prints an error msg

CHANGE

To **copy a file**, the cp command is use which takes the source and the destination path as inputs. It first checks if the path and inode address for the paths are valid.

CHANGE

In case the inode address for source is invalid, an error msg is printed on terminal.

CHANGE

Whereas, if the addresses are valid, the source file is copied to the destination path

CHANGE

In order to display **a list of files** in the current working directory, the dir command is used, which calls the ls () function. It first gets the total number of items in the working directory and then checks the file type.

CHANGE

If the dir item is of type file, the file name along with its size in bytes is printed. If it is a dir, just its name is printed.

CHANGE

To **print out file contents**, the command cat is used which calls the cat () function. It takes the file name as input and prints the random string on the terminal based on the file size.

CHANGE

If the path is invalid, it prints an error on the terminal

CHANGE

Working directory functions pwd () and ispwd () return and check if the pointer points to the current working directory. Furthermore, the **check functions** isvalid () and check () are defined to verify the format of the path and the name and format of the path at creation.

Thankyou for your attention.