Name: Ritika Mohanty

Roll Number: 44

ADVANCED OPERATING SYSTEM ASSIGNMENT

**Documentation for the Code:**

This code is a C program that provides several functionalities related to file operations using system calls. It includes functions to create a file, read data from a file, write data to a file, copy the content of a source file to a destination file using a pipe, display file statistics, and create a named pipe for communication between two processes.

**Functions:**

1. createFile:

- Description: Creates a file with specified permissions.

- Function Signature: void createFile(char\* fileName, int permissions)

- Inputs:

- fileName: Name of the file to be created.

- permissions: Permissions for the file in octal format (e.g., 0666).

2. createPipe:

- Description: Creates a named pipe with specified permissions.

- Function Signature: void createPipe(char\* filename, int permissions)

- Inputs:

- filename: Name of the named pipe to be created.

- permissions: Permissions for the named pipe in octal format (e.g., 0666).

3. readFileData:

- Description: Reads data from a file starting from a specified offset and of a specified size.

- Function Signature: void readFileData(char\* fileName, int offset, int size)

- Inputs:

- fileName: Name of the file to read data from.

- offset: Offset in bytes from where to start reading.

- size: Number of bytes to read.

4. writeFileData:

- Description: Writes data to a file starting from a specified offset.

- Function Signature: void writeFileData(char\* fileName, int offset, char\* data)

- Inputs:

- fileName: Name of the file to write data to.

- offset: Offset in bytes from where to start writing.

- data: Data to be written to the file.

5. copyUsingPipe:

- Description: Copies the content of a source file to a destination file using a pipe.

- Function Signature: void copyUsingPipe(char\* sourceFile, char\* destinationFile)

- Inputs:

- sourceFile: Name of the source file to be copied.

- destinationFile: Name of the destination file to copy the content to.

6. displayFileStats:

- Description: Displays file statistics including owner, permissions, inode, and timestamps.

- Function Signature: void displayFileStats(char\* fileName)

- Inputs:

- fileName: Name of the file to display statistics for.

7. createNamedPipeForComm:

- Description: Creates a named pipe for communication between two processes.

- Function Signature: void createNamedPipeForComm(char\* pipeName, int mode, char\* msg)

- Inputs:

- pipeName: Name of the named pipe for communication.

- mode: Mode of the pipe (O\_RDONLY or O\_WRONLY).

- msg: Message to be written to the pipe (only for writing mode).

**Libraries Used**

The code uses several library functions to perform file operations and inter-process communication:

* stdio.h: It is used for input and output operations, such as printf and scanf.
* stdlib.h: It provides general utilities, including exit and atoi.
* unistd.h: It includes API functions like fork, pipe, open, read, write, `lseek.
* lseek: It is used to move the file offset to a specified position within a file.
* fcntl.h: It provides various file control options, such as O\_RDONLY and O\_WRONLY, used for opening pipes in read or write mode.
* sys/types.h: It includes definitions for various types used in system calls, such as ssize\_t (signed size type).
* sys/stat.h: It contains structures and functions used for file status and information, such as stat for retrieving file statistics.

**Library Functions and their Usage:**

1. printf:

- Usage: printf("format string", arguments)

- Description: Prints formatted output to the standard output (console).

2. scanf:

- Usage: scanf("format string", arguments)

- Description: Reads formatted input from the standard input (console).

3. exit:

- Usage: exit(status)

- Description: Terminates the program execution with the specified status code.

4. mknod:

- Usage: mknod(filename, mode, dev)

- Description: Creates a special file (in this case, a named pipe) with the specified mode and device number.

5. open:

- Usage: open(filename, flags, mode)

- Description: Opens the specified file with the given flags and mode. Returns a file descriptor that can be used for further file operations.

6. read:

- Usage: read(fd, buffer, count)

- Description: Reads data from the file associated with the file descriptor (fd) into the buffer. It reads a maximum of count bytes.

7. write:

- Usage: write(fd, buffer, count)

- Description: Writes data from the buffer to the file associated with the file descriptor (fd). It writes a maximum of count bytes.

8. lseek:

- Usage: lseek(fd, offset, whence)

- Description: Sets the file offset for the file descriptor (fd) based on the specified offset and whence. It allows moving the file pointer to a specific position in the file.

9. close:

- Usage: close(fd)

- Description: Closes the file associated with the file descriptor (fd).

10. stat:

- Usage: stat(filename, stat\_buffer)

- Description: Retrieves information about the specified file and stores it in the stat structure pointed to by stat\_buffer. The stat structure contains details such as file size, permissions, owner, timestamps, etc.

**Input and Output Formats:**

// To create a file with permissions 0666

Input: ./a.out createFile ~/myfile.txt 0666

Output: File created successfully.

// To create a pipe with permissions 0666

Input: ./a.out copyUsingPipe file1.txt file2.txt

Output: File data from file1.txt copied to file file2.txt sucessfully!!

// Read data from the file

Input: ./a.out read file1.txt 5 100

Output: "File data read from file: <data read>"

// Write data to the file

Input Format: ./program write file1.txt 4 "Lorem Ipsum"

Data is now written in file file1.txt

// Display file statistics of a file

Input Format: ./a.out displayStats file1.txt

Output: File Statistics Information for file 'file1.txt':

Owner: 1000

Permissions: 100777

Inode: 178173660257910177

Last Access Time: Tue Jul 4 00:24:15 2023

Last Modification Time: Tue Jul 4 00:24:15 2023

Last Status Change Time: Tue Jul 4 00:24:15 2023

// Copy data from source file to destination file using a pipe

Input: ./a.out copyUsingPipe file1.txt file2.txt

Output: File data from file1.txt copied to file file2.txt sucessfully!!

// Communicate between processes using the named pipe

Input (Reading Process): ./a.out commUsingNamedPipe ~/myPipe 0

Output: Data is now written to the named pipe!

Input (Writing Process): ./a.out commUsingNamedPipe ~/myPipe 1 "This is a message for reading process"

Output: Data read from the pipe: This is a message for reading process