**I/O Linux system calls - Creat()**

**creat() linux system call for file creation**

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <stdio.h>

int main() {

char filename[] = "example.txt";

mode\_t permissions = S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH;

int fileDescriptor;

// Create a new file with the specified permissions

fileDescriptor = creat(filename, permissions);

if (fileDescriptor == -1) {

perror("Error creating the file");

return 1;

}

// Write a message to the file

char message[] = "Hello, World!";

ssize\_t bytesWritten = write(fileDescriptor, message, strlen(message));

if (bytesWritten == -1) {

perror("Error writing to the file");

close(fileDescriptor);

return 1;

}

printf("Message written to the file: %s\n", message);

// Close the file

close(fileDescriptor);

return 0;

}

In this program, the creat() system call is used to create a new file named "example.txt" with the specified permissions (S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH). If the file is successfully created, the program writes the message "Hello, World!" to the file using the write() system call. Finally, the file descriptor is closed using the close() system call.

Please note that creat() is a legacy function, and it is recommended to use the open() system call with appropriate flags for file creation and permissions instead.

Program 1: Creating a File and Setting Permissions

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

int main() {

char filename[] = "example.txt";

mode\_t permissions = S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH;

int fileDescriptor;

// Create a new file with the specified permissions

fileDescriptor = creat(filename, permissions);

if (fileDescriptor == -1) {

perror("Error creating the file");

return 1;

}

printf("File created successfully: %s\n", filename);

// Close the file

close(fileDescriptor);

return 0;

}

Program 2: Creating a File and Writing Data

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <stdio.h>

int main() {

char filename[] = "example.txt";

mode\_t permissions = S\_IRUSR | S\_IWUSR;

int fileDescriptor;

// Create a new file with the specified permissions

fileDescriptor = creat(filename, permissions);

if (fileDescriptor == -1) {

perror("Error creating the file");

return 1;

}

// Write data to the file

char message[] = "Hello, World!";

ssize\_t bytesWritten = write(fileDescriptor, message, strlen(message));

if (bytesWritten == -1) {

perror("Error writing to the file");

close(fileDescriptor);

return 1;

}

printf("Message written to the file: %s\n", message);

// Close the file

close(fileDescriptor);

return 0;

}

Program 3: Creating a New File and Appending Data

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <stdio.h>

int main() {

char filename[] = "example.txt";

mode\_t permissions = S\_IRUSR | S\_IWUSR;

int fileDescriptor;

// Create a new file with the specified permissions

fileDescriptor = creat(filename, permissions);

if (fileDescriptor == -1) {

perror("Error creating the file");

return 1;

}

// Write initial data to the file

char initialMessage[] = "Initial Data\n";

ssize\_t initialBytesWritten = write(fileDescriptor, initialMessage, strlen(initialMessage));

if (initialBytesWritten == -1) {

perror("Error writing initial data to the file");

close(fileDescriptor);

return 1;

}

// Append more data to the file

char additionalMessage[] = "Additional Data\n";

ssize\_t additionalBytesWritten = write(fileDescriptor, additionalMessage, strlen(additionalMessage));

if (additionalBytesWritten == -1) {

perror("Error writing additional data to the file");

close(fileDescriptor);

return 1;

}

printf("Data written to the file: %s\n", filename);

// Close the file

close(fileDescriptor);

return 0;

}

Program 4: Creating a File and Verifying Existence

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

int main() {

char filename[] = "example.txt";

mode\_t permissions = S\_IRUSR | S\_IWUSR;

int fileDescriptor;

// Create a new file with the specified permissions

fileDescriptor = creat(filename, permissions);

if (fileDescriptor == -1) {

perror("Error