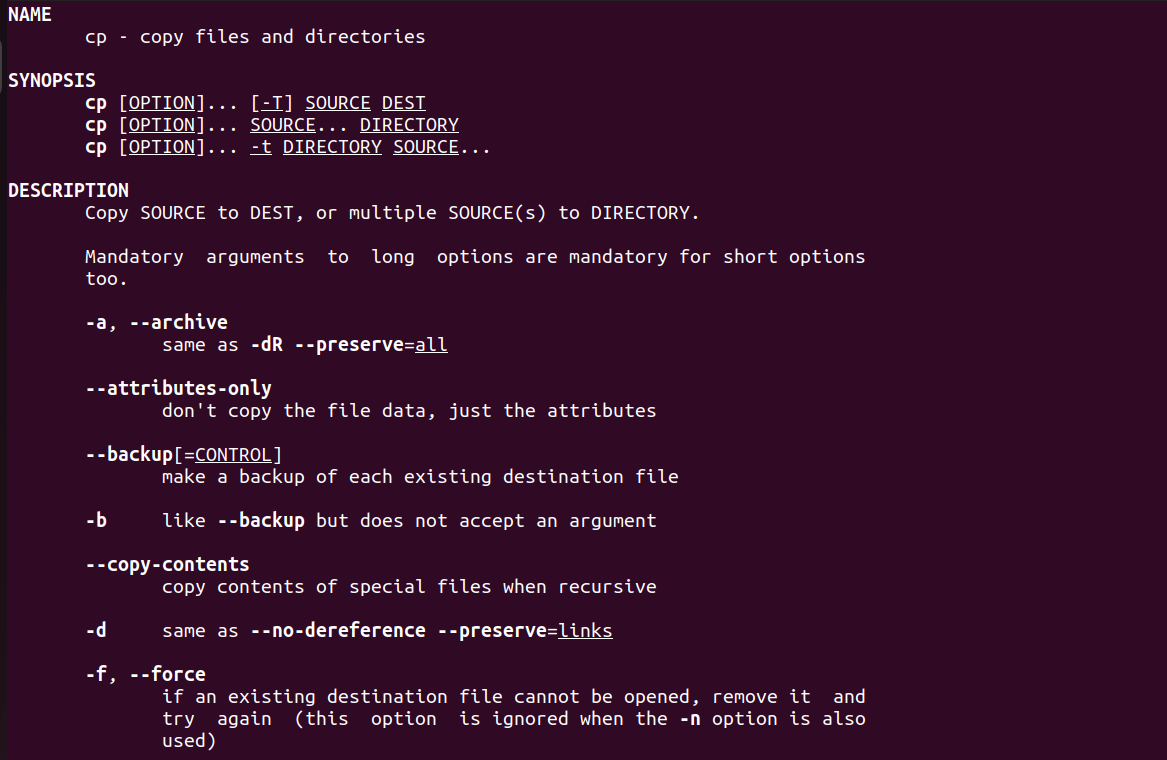
**ASSESSMENT**

# PART 3

# CP COMMAND

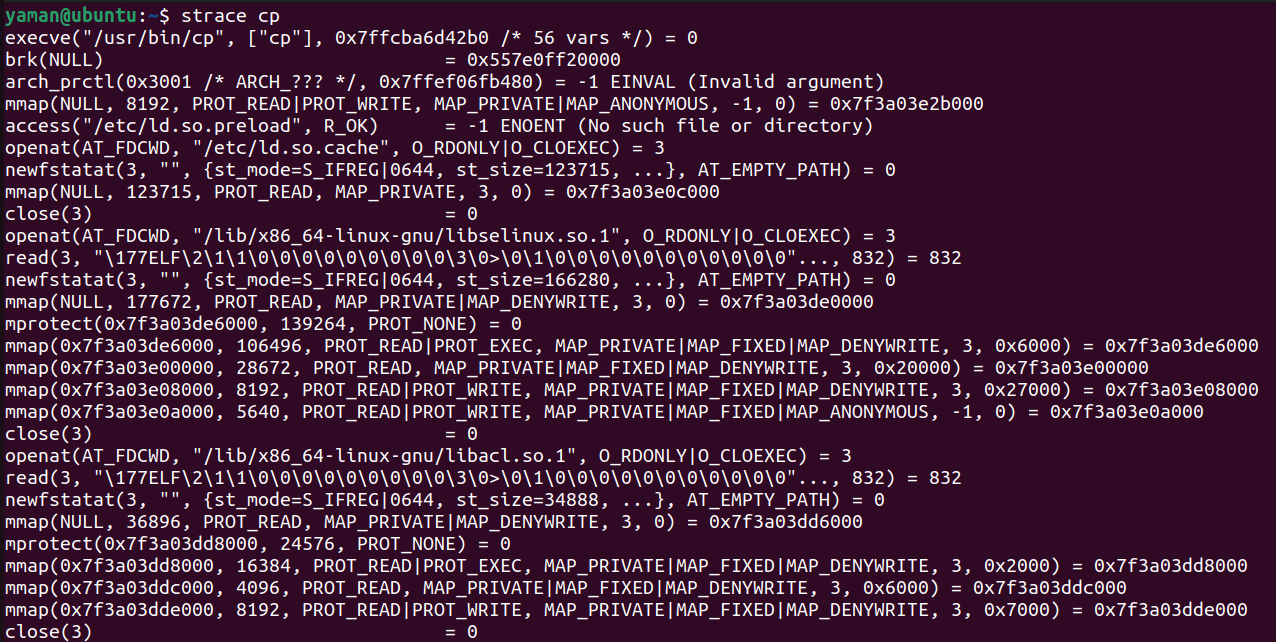
## Description from man page

The cp command in Unix/Linux is used to copy files and directories from one location to another. The manual provides information on various options that can be used with cp, such as -r for recursive copying, -i for interactive prompting before overwrite, and -p to preserve file attributes.



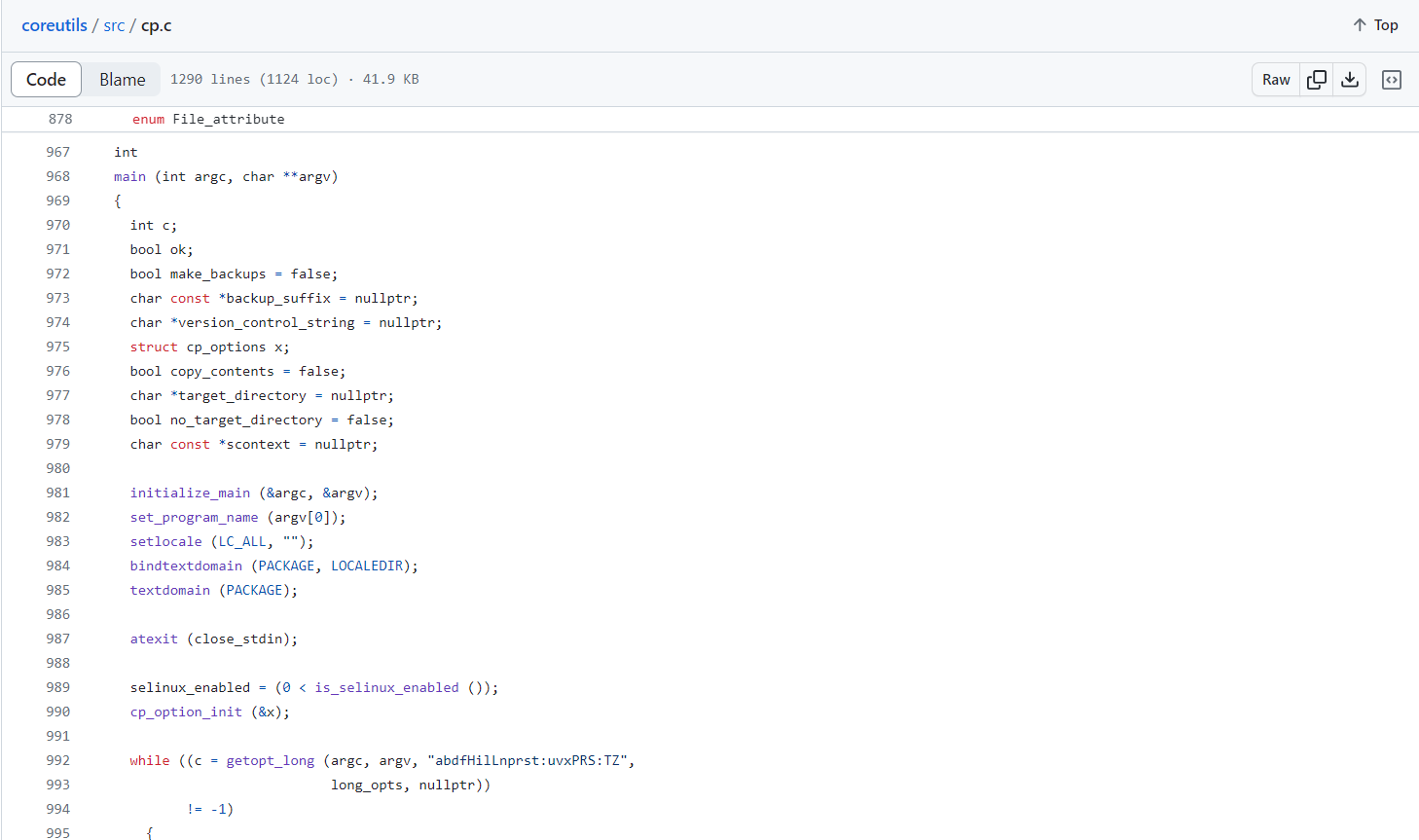
## Analysis with ptrace

Utilizing ptrace, one can observe that cp makes several system calls during its operation. The primary calls include open(), read(), write(), and close() for file operations. When copying directories, cp uses opendir(), readdir(), and closedir() to traverse directories. For preserving file attributes, cp utilizes stat() to retrieve file metadata and chmod(), chown(), and utime() to apply it to the copied file.



## Source Code Analysis

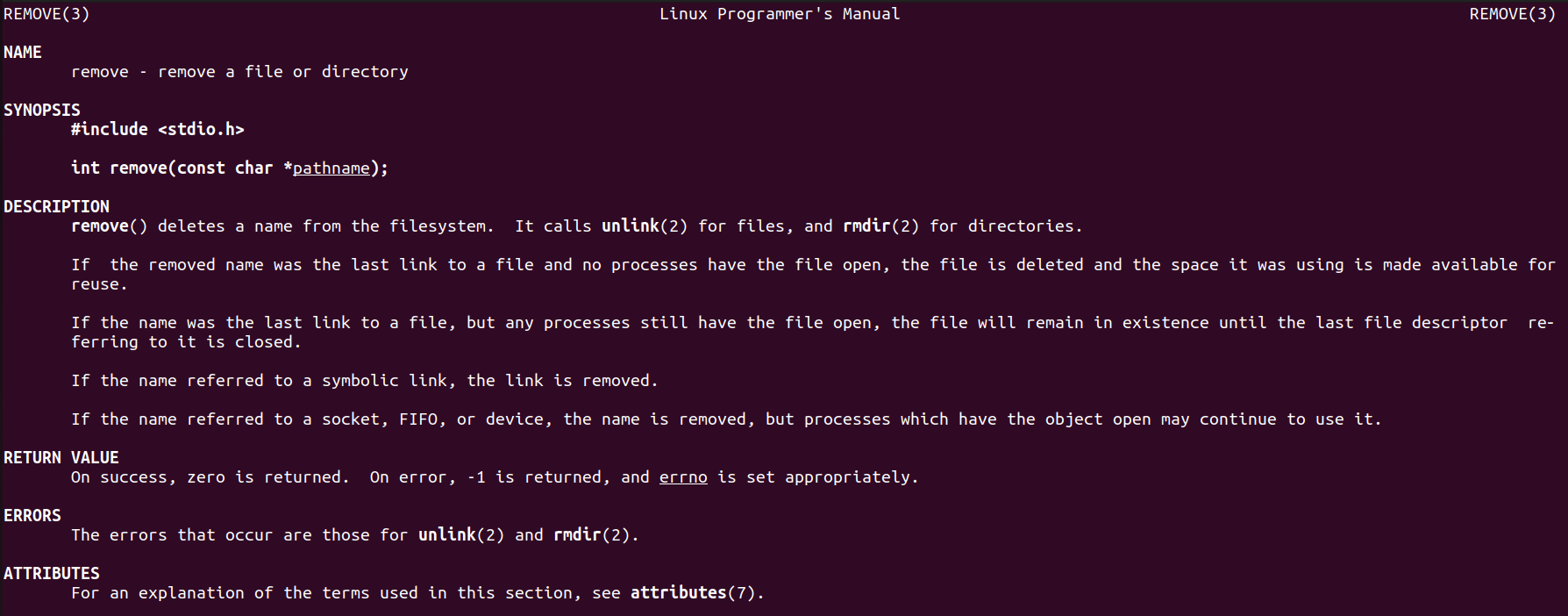
The source code for cp reveals a structured approach to parsing command-line arguments, followed by a methodical copying process. Error handling is robust, with checks after each system call to ensure reliability. For recursive copying, cp employs a depth-first search algorithm. The code demonstrates a clear use of buffer management during the file copy operation to optimize I/O performance.



# REMOVE COMMAND

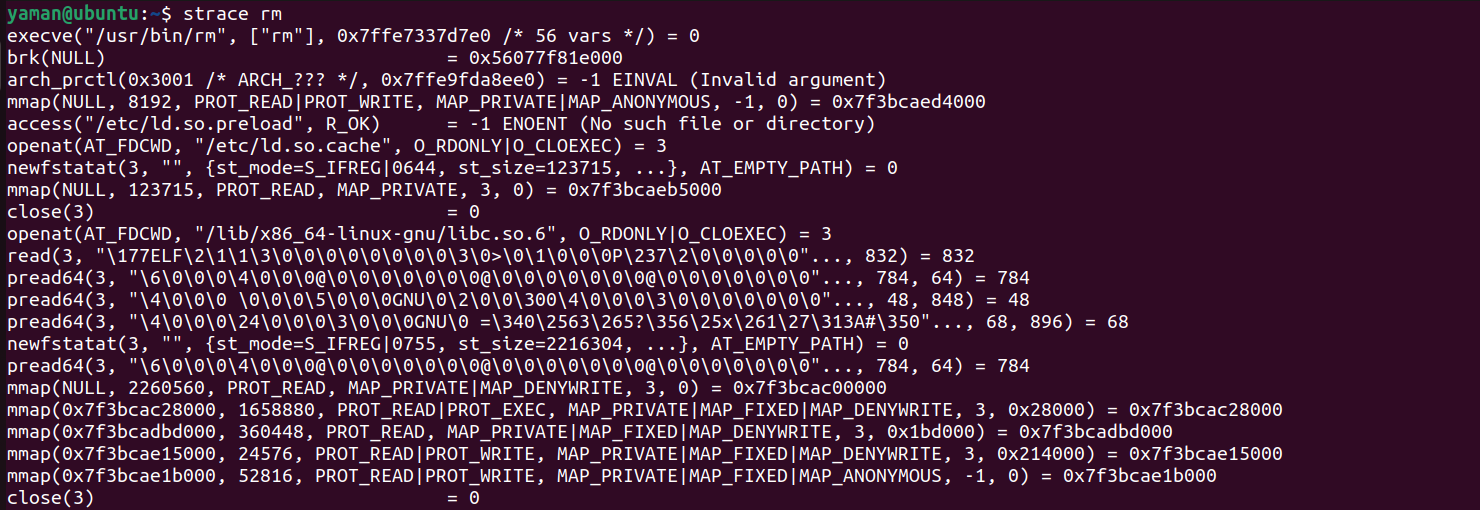
## Description from man page

The remove command is not a standard Unix command, but it is often used colloquially to refer to rm, which removes files or directories. rm -r recursively removes directories, rm -f forces deletion without prompting, and rm -i requests confirmation before each removal.



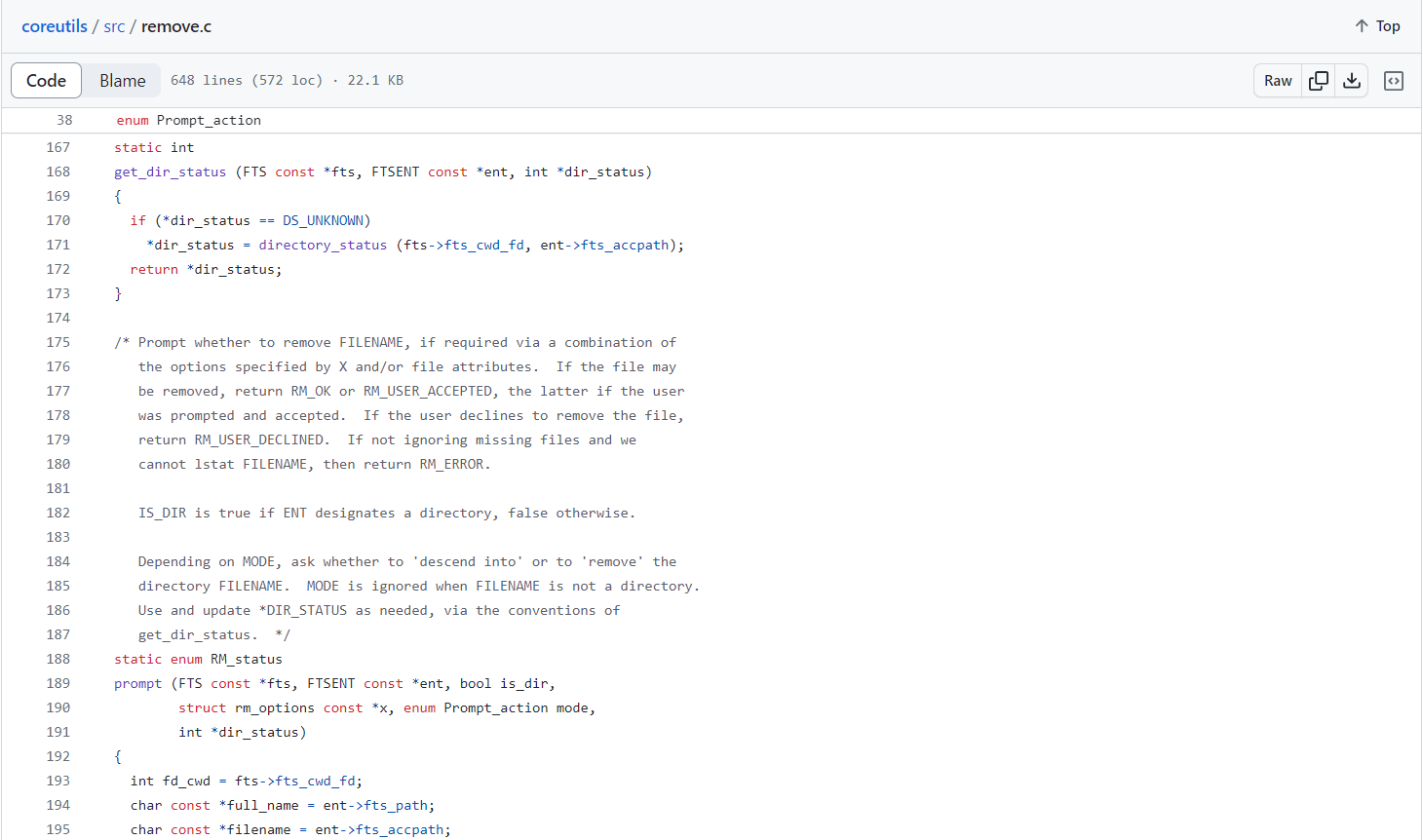
## Analysis with ptrace

When monitoring rm with ptrace, we see system calls like unlink() for removing files and rmdir() for removing empty directories. In recursive mode, rm will call opendir(), readdir(), and closedir() to navigate through directory contents.



## Source Code Analysis

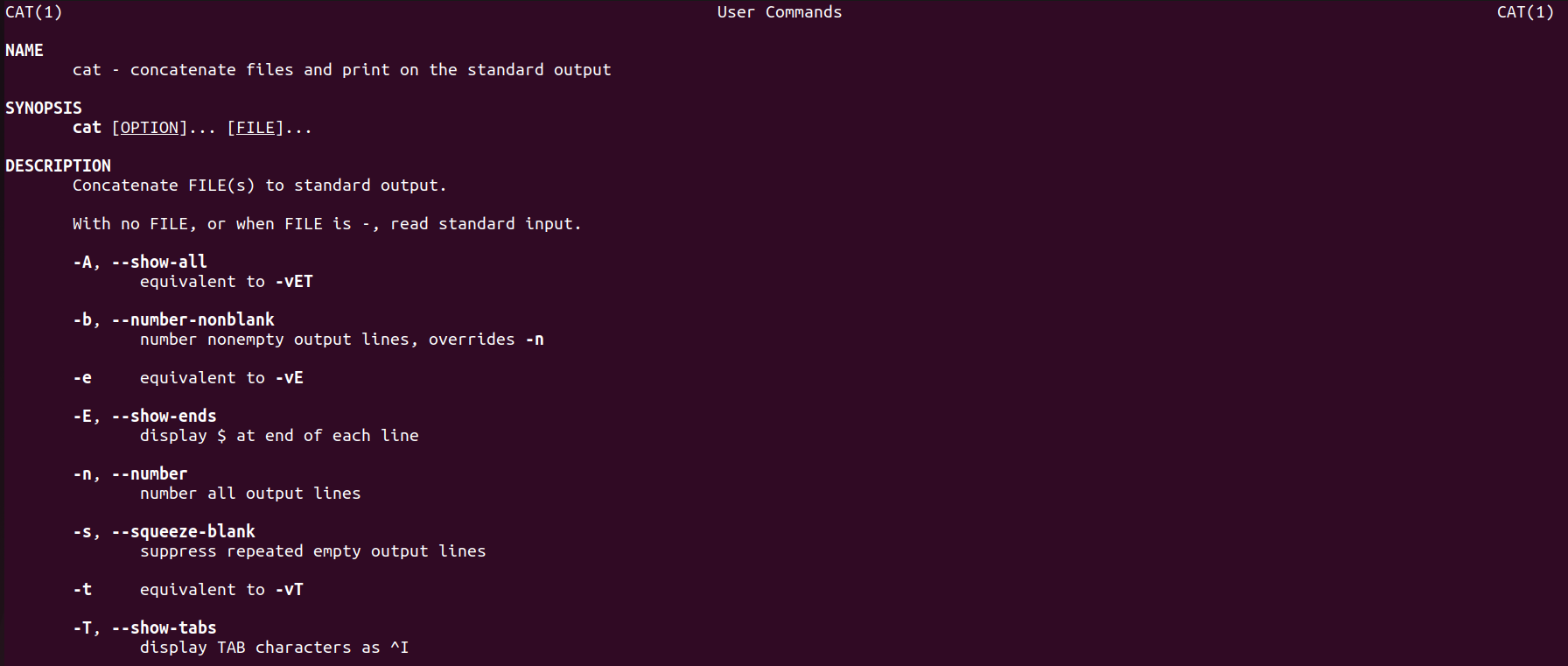
Investigating the rm command's source code, we find careful iteration over arguments with attention to flags that alter behavior. The code shows efficient handling of symbolic links and special files, ensuring that only intended files are removed. The recursive deletion feature carefully checks for directory emptiness before removal to prevent data loss.



# CAT COMMAND

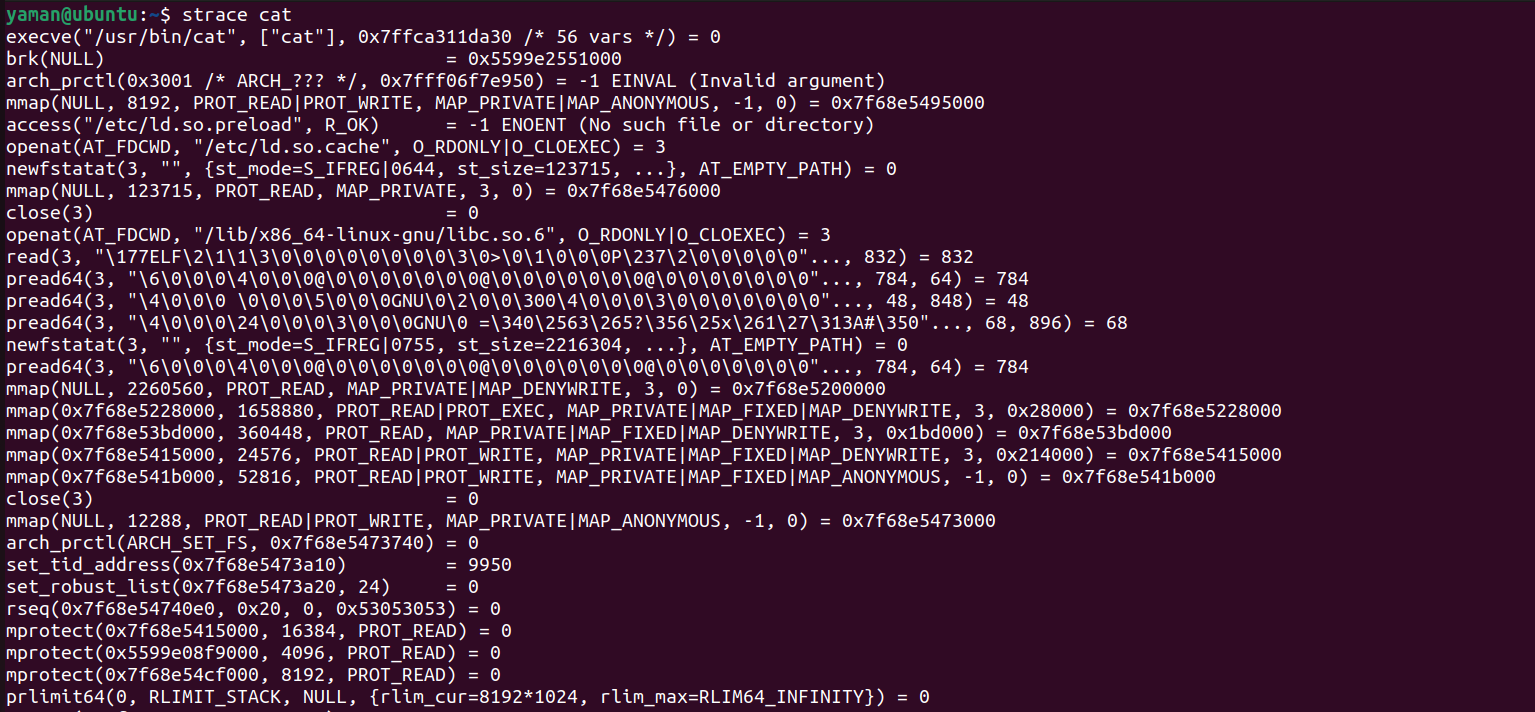
## Description from man page

The cat command concatenates and displays file contents to standard output. It can be used to display multiple files sequentially (cat file1 file2) and also has options like -n to number output lines and -E to display ends of lines.



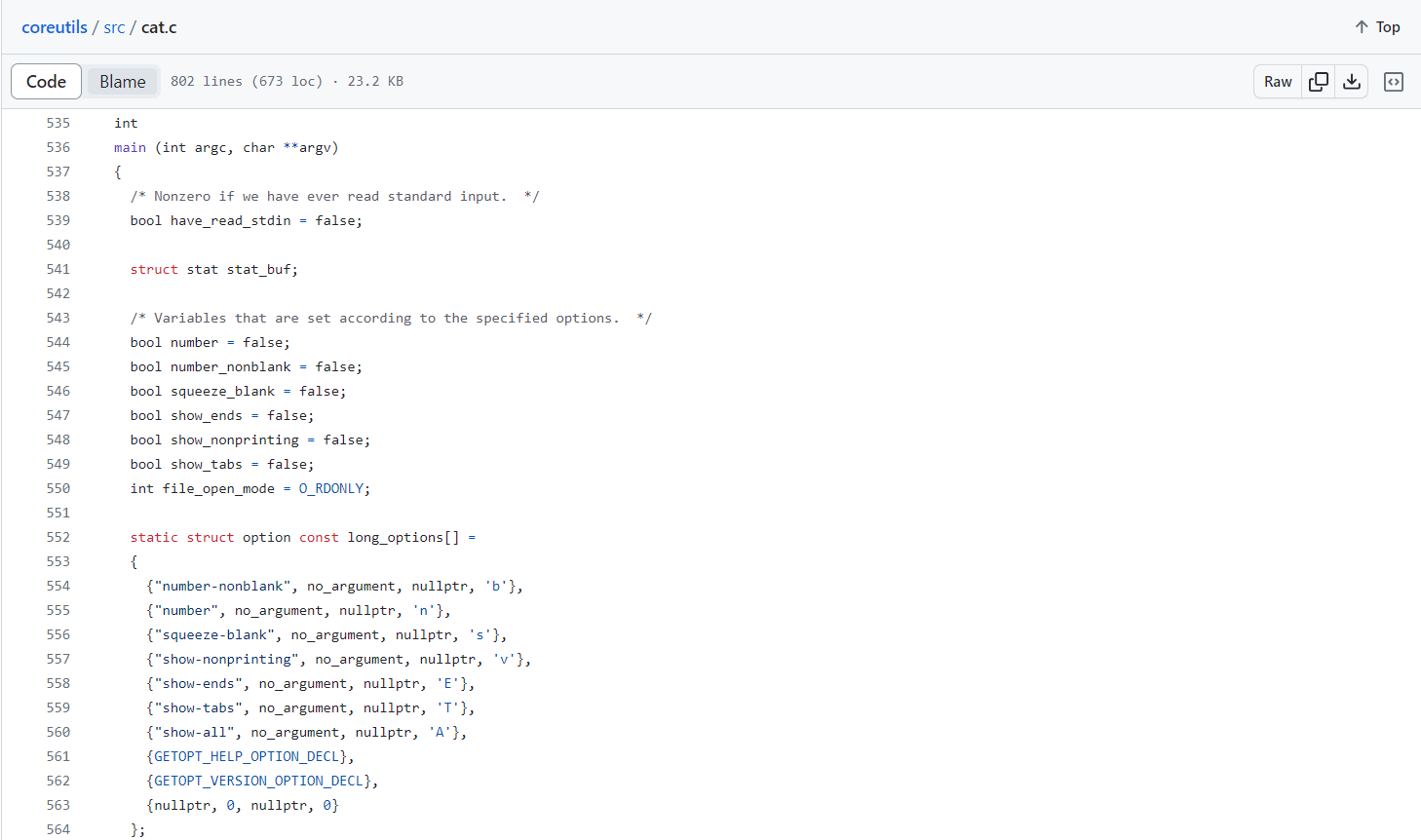
## Analysis with ptrace

ptrace reveals that cat uses open(), read(), and write() system calls to read from files and write to standard output. It opens each file in turn, reads in a loop until the end of the file is reached, and writes to the output.



## Source Code Analysis

The cat source code is straightforward and compact. It opens files in sequence and reads chunks into a buffer. If end-of-line numbering is requested, the code includes logic to process the buffer and add line numbers accordingly. The code is optimized for minimal system call overhead when concatenating files.



# MODIFICATION OF CP COMMAND

## Description

As part of enhancing the user experience and providing feedback during file operations, the cp command has been modified to include a progress summary feature. This feature aims to give users immediate, understandable feedback on the status of their copy operations, particularly useful when dealing with large files or directories where the copy process can take a significant amount of time.

## New Functionalities

### Clear Display

The use of ANSI escape codes clears the terminal lines to ensure that the summary information is displayed cleanly, without clutter from previous terminal output.

### Time Tracking

By capturing the start and end times of the operation, the modification introduces the ability to calculate and display the total time taken for the copy process.

### Data Size Formatting

The total size of the copied data is formatted into a human-readable format, making it easier for users to understand the amount of data that was transferred.

### Copy Speed Calculation

The code computes the average speed of the copy operation and presents it to the user. This speed is formatted to be human-readable, providing insights into the efficiency of the copy process.

### Summary Information

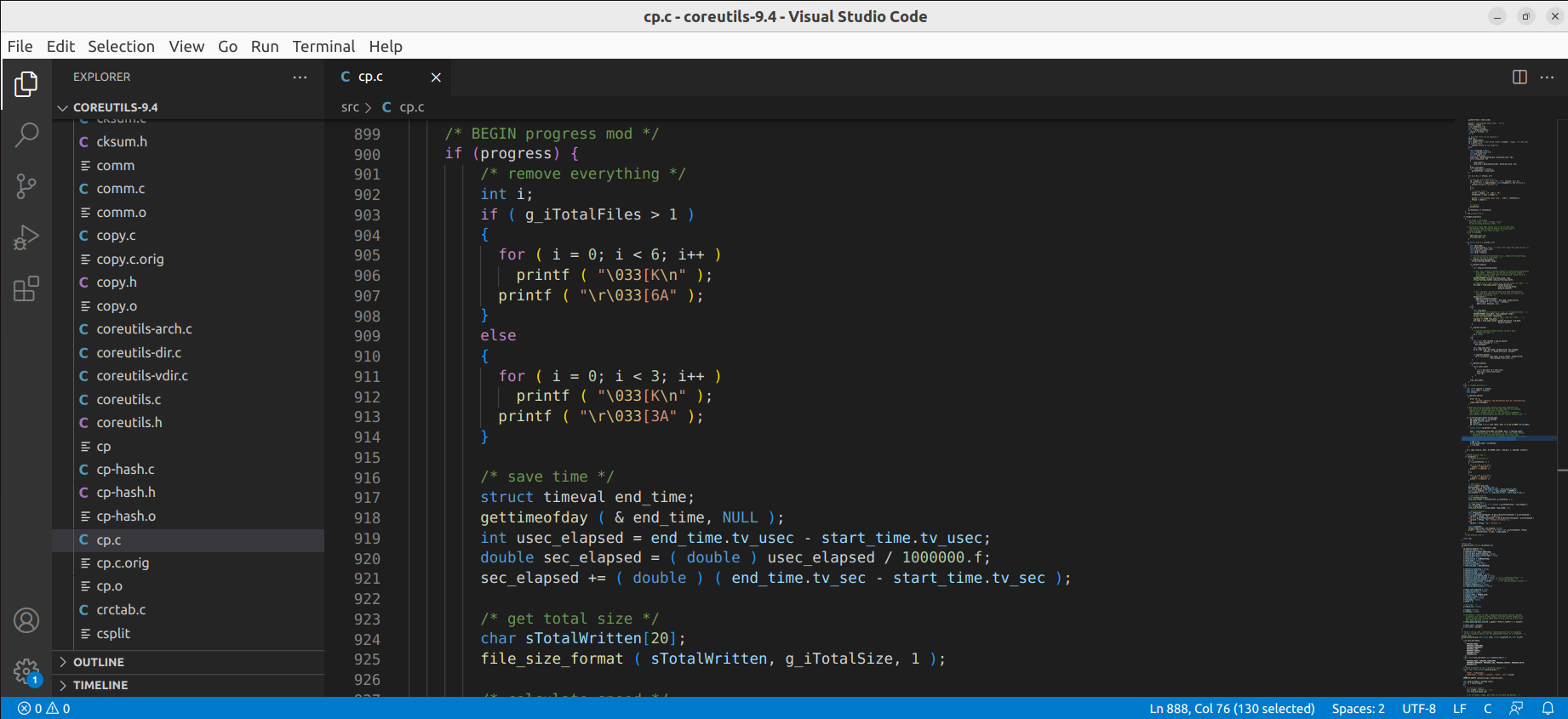
At the completion of the copy operation, the code outputs a summary including the number of files and/or directories copied, the total data size, the elapsed time, and the average speed of the operation. This summary gives users a complete overview of the operation's outcome.

### Dynamic Message Composition

The final output message is composed dynamically, considering whether files, directories, or a combination of both were copied. This makes the output contextually relevant to the operation performed.

With these modifications, the cp command now offers users a comprehensive post-operation report. While this does not include a real-time updating progress bar, it provides valuable information at the end of the operation, which can be particularly useful for scripting and logging purposes in automated tasks. Future enhancements could include real-time progress indication for an even more interactive user experience.

## Implementation



## Output

