FileTools C Program:

The operations available in this program are as follows:

Create File "-n <filename>…": Creates a new file with a specified name
Copy File "-cpy <src> <dest>": Creates a new file and copies the content from the source to it
Delete file "-df <filename>…": Deletes the file by filename
Show file "-o <filename>": Displays the content of the file
Append Line "-a <filename> <text>": Creates a new line containing the specified text at the end of a file
Insert Line "-i <filename> <line> <text>: Inserts a line of text at a specified line of a file
Delete Line "-dl <filename> <line>: Deletes the specified line of a file
Show Change Log "-log": Displays the change log (stored in log.txt)
Show Number of Lines "cl <filename>…": Shows the number of lines in each specified file

The additional operations are:

Show Help "-h": Displays the help for the program
Rename "-r <src> <dest>": Copies a file to a new file and removes the original file
Compare Files "-cmp <file1> <file 2> …": Compares two files and displays the lines where there are differences if there are any.
Encrypt "-enc <filename>": Performs an XOR encryption on a file using a password
Decrypt "-dec <filename>": Performs the refers XOR decryption on a file using a password.

Rename Operation:

The rename operation is the simplest of the additional operations. It compounds the Copy File and Delete File operation to effectively rename a file. While this operation is simple, it is also just as likely to be useful as the previous operations.

Compare Files Operation:

This operation reads through two files line by line and compares the content of each line. It will print out each line number with a difference, as well as printing out the total number of lines with differences on completion.

Encryption/ Decryption:

File encryption starts by taking in the user's password with scanf(). Using the PBKDF2[1] function and a randomly generated 128-bit salt, a 128-bit key is generated for encrypting the file using the sha256 hashing function. This key is then repeatedly used to perform a bitwise XOR operation on the content of the file (where each bit of the file content is XORed with a corresponding bit of the 128-bit key)[2]. This is then printed out to the file with the salt prefixing the content:

[salt | 16 bytes] [Encrypted content]

This is required as the PBKDF2 function requires the same salt for a password to generate the same key. Therefore, for decryption the salt must be communicated with the file, as the XOR cipher is a symmetric encryption algorithm. Meaning that the same key must be used for both encryption and decryption. So, when decrypting the file, the first 16 bytes are read and used as the salt for generating the decryption key. The remaining data is decrypted by performing the XOR cipher using this generated key. In this program there is no error checking for the file. Decryption with the wrong password is destructive. The notion of prefixing a salt is not a new

---

[1] RSA Laboratories. *Password-Based Cryptography Specification Version 2.0 (RFC 2898)*. Section 5.2., [Online] https://www.ietf.org/rfc/rfc2898.txt [Published September 2000, Accessed November 25, 2024]
[2] *M. Lewin, "All About XOR" June 2012,* [Online] https://accu.org/journals/overload/20/109/lewin_1915/ [Published June 2012, Accessed 30/11/2024]

one. A similar approach was used in early versions of UNIX where hashed passwords were stored with a prefixed salt. This salt was a 12-bit, random number, generated by reading the real-time clock. This was used to compare the results of a user inputted password and the one stored in the password file. [3]

Compiling the Program:

The encryption/ decryption section of this program requires the OpenSSL[4] library. This is used for generating an encryption key from a password using the PBKDF2 function. To compile the software using gcc, run:

gcc –o fileTools Main.c -lcrypto

Using the Program:

The program is used almost entirely with the command line arguments above. When encrypting/ decrypting, a password is prompted and must be entered. Some operations can take multiple arguments (deleting files, creating files, counting lines, and comparing files). These all work relatively intuitively, where each can take multiple files as arguments. When comparing multiple files, each file is just compared to the first file entered.

[3] *R. Morris, K. Thompson. Password Security: A Case History 03-04-1978* https://web.archive.org/web/20130821093338/http://cm.bell-labs.com/cm/cs/who/dmr/passwd.ps [Archived February 15, 2015, Accessed 29/11/2024]

[4] The OpenSSL Project. (2024). *OpenSSL Toolkit* (Version 3.0.5) [Software] https://www.openssl.org. [Accessed November 25, 2024]