**Bash Programme - File split in a multiple part**

1. Prompting for File Path:

read -p "Enter the file path: " file\_path

This line prompts the user to enter the file path and stores it in the file\_path variable.

1. Checking File Existence:

if [ ! -f "$file\_path" ]; then

echo "File not found."

exit 1

fi

This block checks if the file specified by file\_path does not exist (! -f "$file\_path"). If the file is not found, it prints an error message and exits the script with an exit code of 1.

1. Checking Newline Character in File

contains\_newline=$(grep -q $'\n' "$file\_path" && echo "true" || echo "false")

This line uses the grep command to search for a newline character ($'\n') in the file specified by file\_path. If the newline character is found, it assigns the value "true" to the contains\_newline variable; otherwise, it assigns "false". The -q flag is used to suppress the output of grep.

1. Getting Directory Path and Creating Folder:

directory\_path=$(dirname "$file\_path")

folder\_name=$(basename "$file\_path")

folder\_name="${folder\_name//./\_}"

mkdir -p "$directory\_path/$folder\_name"

These lines extract the directory path of the file using dirname "$file\_path" and store it in the directory\_path variable. Then, the base name of the file is obtained using basename "$file\_path" and assigned to folder\_name. Any dots (.) in the folder\_name are replaced with underscores (\_) using ${folder\_name//./\_}. Finally, the script creates a folder with the derived folder name in the specified directory path using mkdir -p "$directory\_path/$folder\_name".

1. Reading File Content and Calculating Length:

content=$(cat "$file\_path")

length=${#content}

These lines read the content of the file using cat "$file\_path" and store it in the content variable. The length of the content is calculated using ${#content} and stored in the length variable.

1. Asking for Number of Files:

read -p "Enter the number of files (n): " n

This line prompts the user to enter the number of files (n) and stores the value in the n variable.

1. Calculating Step Size:

step=$((length / n))

This line calculates the step size by dividing the length of the content by the number of files (n). The result is stored in the step variable.

1. Initializing Index and Looping through Content:

for ((i = 0; i < length; i++)); do

char="${content:i:1}"

These lines initialize the index i to 0 and start a loop that iterates until i is less than the length of the content. In each iteration, the variable char is assigned the current character at index i from the content string.

1. Calculating File Index and Generating File Name:

file\_index=$(( (i / step) + 1 ))

file\_name="${folder\_name}\_part${file\_index}.txt"

These lines calculate the file index based on the current index i and the step size, and store it in the file\_index variable. The file name is generated by combining the folder\_name, \_part, and file\_index with the .txt extension.

1. Writing Character to File and Displaying:

echo -n "$char" >> "$directory\_path/$folder\_name/$file\_name"

echo -n "$char"

These lines write the character ($char) to the corresponding file within the created folder using echo -n "$char" >> "$directory\_path/$folder\_name/$file\_name". The -n flag is used to suppress the newline character in the output. Additionally, echo -n "$char" is used to display the character in the terminal without a newline.

1. Handling Newline Character:

if [[ "$char" == $'\n' ]] && [[ "$contains\_newline" == "true" ]]; then

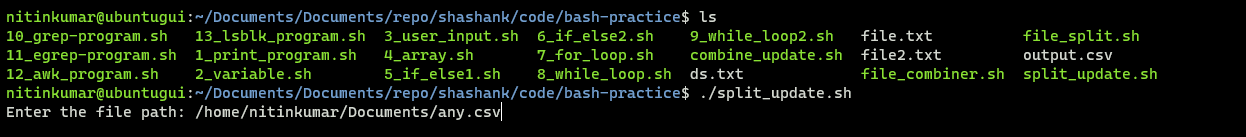
echo

i=$((i + step))

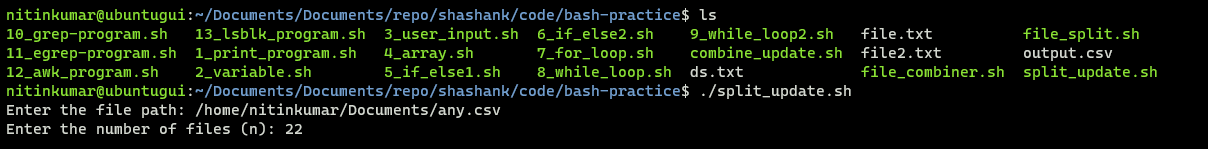
Fi

This block checks if the current character is a newline ($'\n') and if the file contains a newline character ($contains\_newline == "true"). If both conditions are true, it adds a newline character to the terminal output using echo and increments the index i by the step size to skip the subsequent characters until the next iteration.

Output : file -

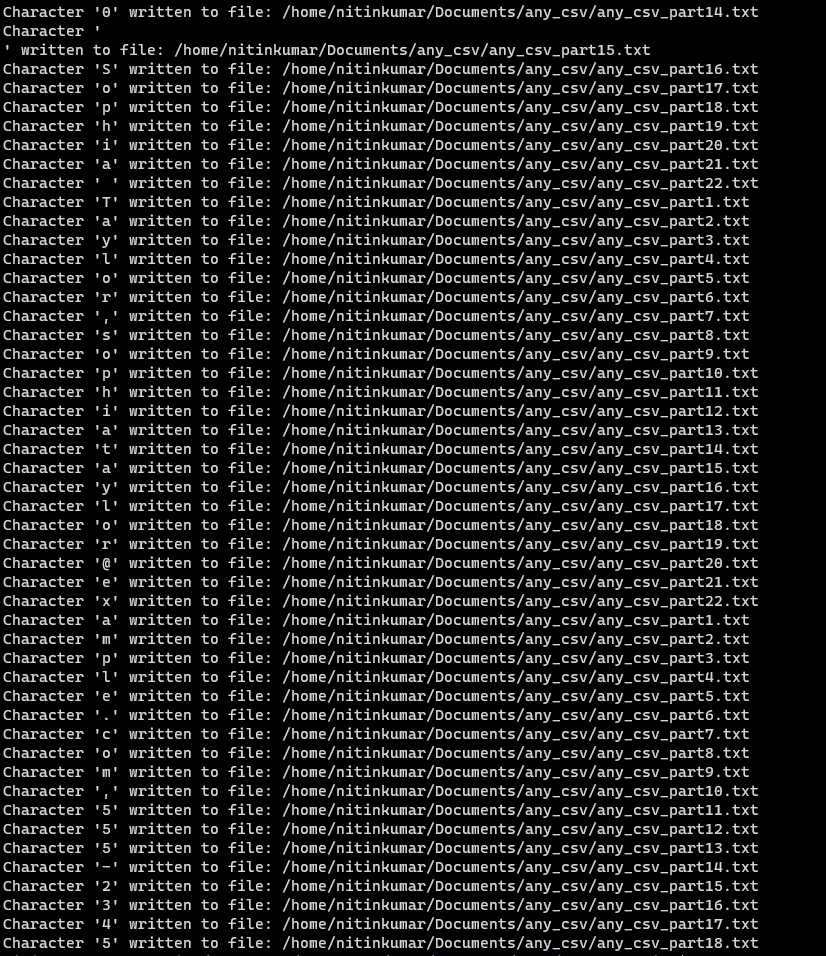


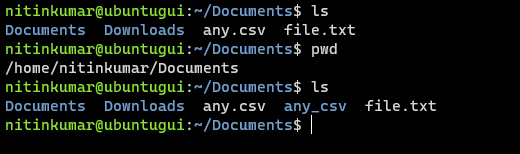
Here need enter path of file and file name which is going to split



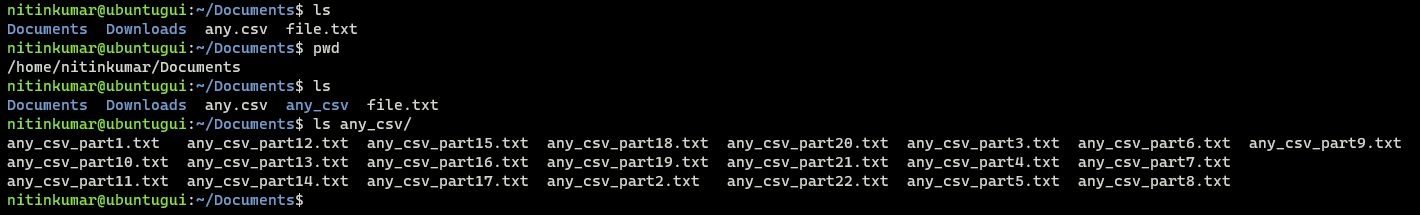
Next need to provide how many part user need to split the file

1. Here it create folder from file name (any.csv) as (any.csv)
2. Start splitting file character by character





Folder was created

Files were created