**Bash script - File combine using first file as input**

1. Prompting the user for the first input file

read -p "Enter the path of the first input file: " first\_file

The user is prompted to enter the path of the first input file. The input provided by the user will be stored in the first\_file variable.

1. Obtaining the directory of the first input file:

directory=$(dirname "$first\_file")

The dirname command is used to extract the directory path from the first input file. The directory path is stored in the directory variable.

1. Creating an array to store the paths of each file:

file\_paths=()

An empty array called file\_paths is created to store the paths of each input file.

1. Looping through each file in the same directory as the first input file

for file in "$directory"/\*; do

# Check if the file exists

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

continue

Fi

This loop iterates over each file in the same directory as the first input file.

1. Checking if the file exists and appending the file path to the array

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

continue

fi

file\_paths+=("$file")

Within the loop, it first checks if the current file exists and is a regular file. If it's not, the loop skips to the next iteration using the continue statement. If the file is valid, its path is appended to the file\_paths array.

1. Sorting the input files numerically based on filenames:

sorted\_files=($(printf "%s\n" "${file\_paths[@]}" | sort -V))

The printf command is used to print each file path from the file\_paths array on a separate line. This list of file paths is then piped to the sort -V command, which performs a numerical sort based on filenames. The sorted file paths are stored in the sorted\_files array.

1. Checking if any input files were found:

if [ ${#sorted\_files[@]} -eq 0 ]; then

echo "No input files found in the directory."

exit 1

Fi

This condition checks if the sorted\_files array is empty. If no input files were found, an error message is displayed, and the script exits with a status code of 1

1. Prompting the user for the path of the output file:

read -p "Enter the path of the output file: " output\_file

The user is prompted to enter the path of the output file. The input provided by the user will be stored in the output\_file variable.

1. Finding the length of the longest string among the input files:

max\_len=0

for file in "${sorted\_files[@]}"; do

chars=$(cat "$file")

len=${#chars}

if [ $len -gt $max\_len ]; then

max\_len=$len

fi

done

This loop iterates over each file path in the sorted\_files array. It reads the contents of each file using cat and assigns them to the chars variable. The length of the characters in each file is then compared to the current maximum length (max\_len). If the length is greater, it updates max\_len to the new value.

1. Combining the characters from the input files:

combine=""

for ((i = 0; i < max\_len; i++)); do

for file in "${sorted\_files[@]}"; do

chars=$(cat "$file")

if [ $i -lt ${#chars} ]; then

combine+="${chars:i:1}"

fi

done

done

This loop iterates from i = 0 to max\_len - 1, representing the index position of the characters being combined. Within this loop, another loop iterates over each file path in the sorted\_files array. It reads the contents of each file using cat and assigns them to the chars variable. If the current index i is within the length of the characters in the file, it extracts the character at index i and appends it to the combine variable

1. Writing the combined characters to the output file:

echo "$combine" > "$output\_file"

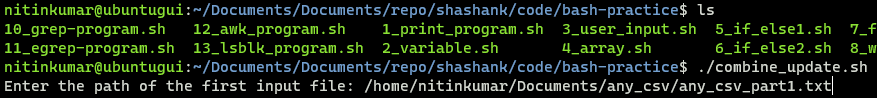
The combined characters are written to the output file specified by the user.

1. Displaying a completion message:

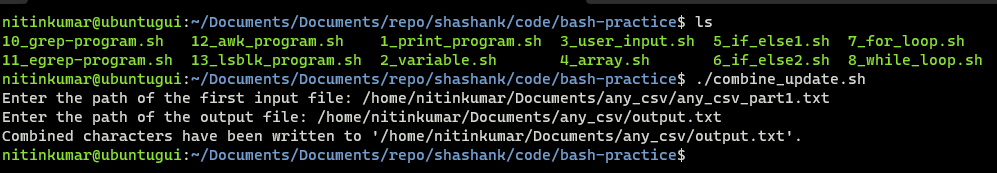
echo "Combined characters have been written to '$output\_file'."

Once the combining process is complete, a message is displayed to indicate that the characters have been written to the output file.

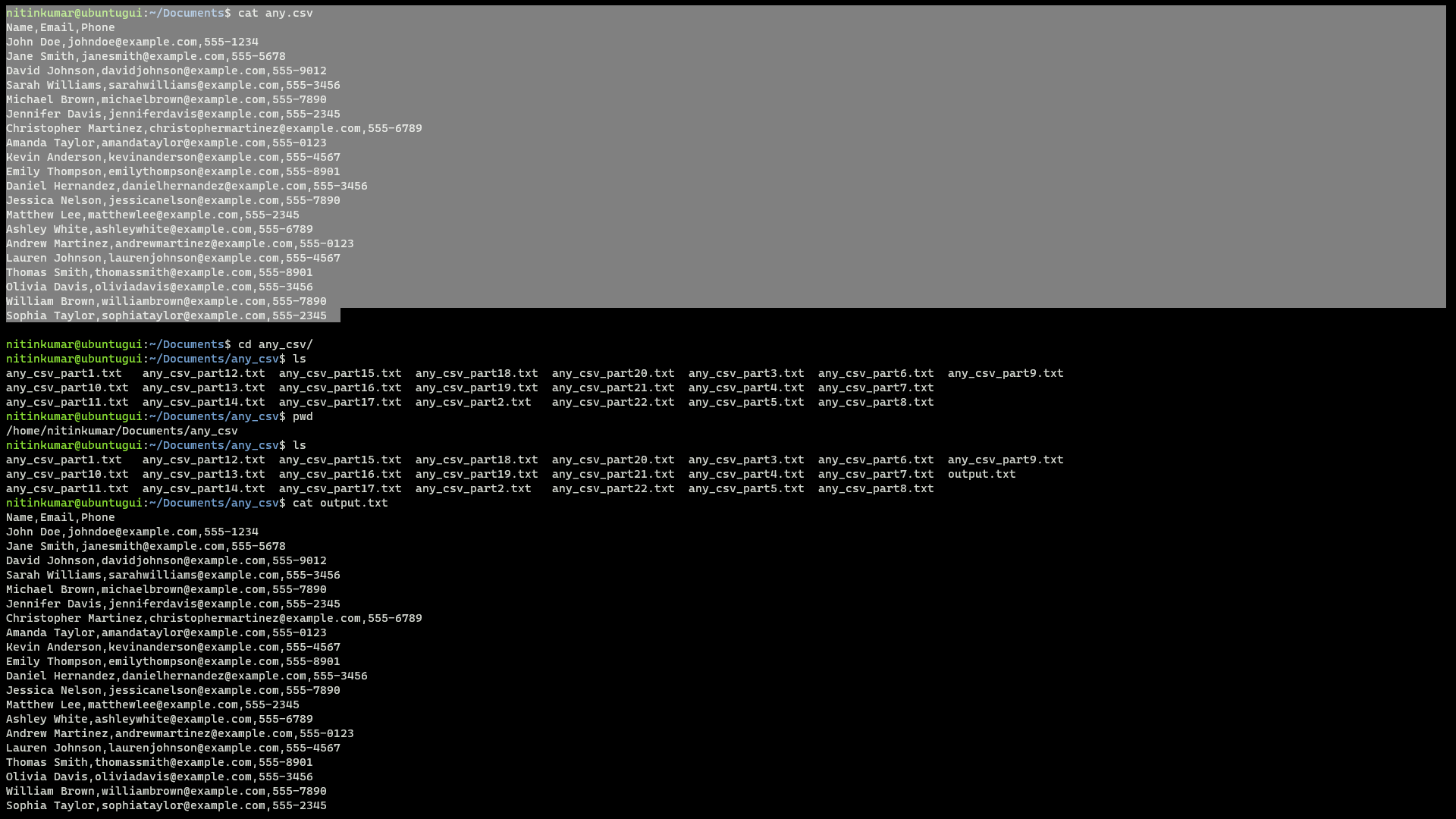
File output:



In this script Ask for file path with first part of file



Next step ask the path of where the output file is stored with name and extension

 final output of file is identical