# Text File Extractor 1.1

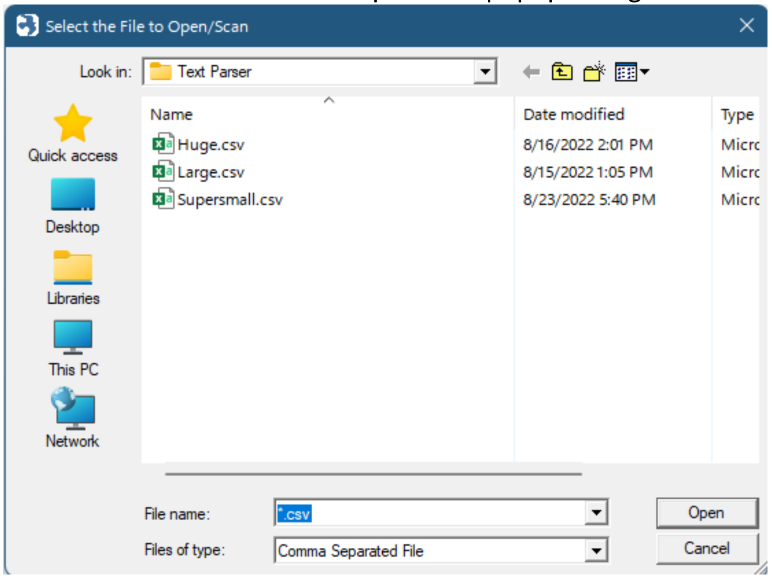
**Purpose:** Open a text-based file and create a second text-based file that does not contain lines which had an undesired character string.

**Example:** Open a router log file which contains lines of text that possess "United States"; output the file, minus lines that contained that text to a second unique file.

**Language Used:** Winbatch Compiler 2022

**Manual Process:** If no parameters are passed to the application (RemoveText.exe), the following screens are presented:

1. User is asked for a text file to open via a popup dialogue:



Select the File to Open/Scan

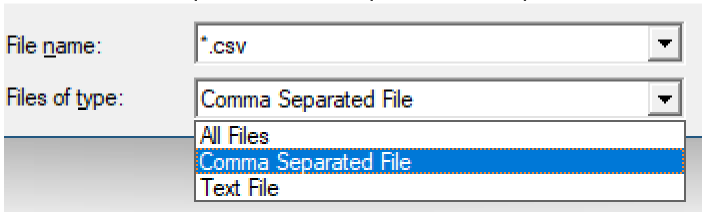
Look in: Text Parser

| Name | Date modified | Type |
|---|---|---|
| Huge.csv | 8/16/2022 2:01 PM | Micrc |
| Large.csv | 8/15/2022 1:05 PM | Micrc |
| Supersmall.csv | 8/23/2022 5:40 PM | Micrc |

File name: *.csv

Files of type: Comma Separated File

The bottom droplist has three predefined options:

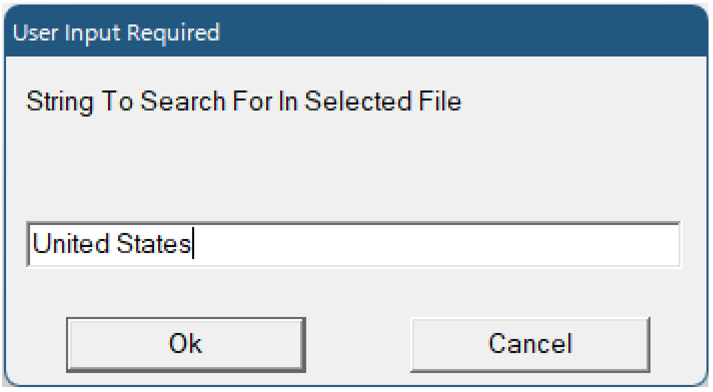


File name: *.csv

Files of type: Comma Separated File

Select the desired file by double left clicking the file or single left click the file and then click the Open button.

2. A popup entry window is displayed to enter the text string to search for within the selected file. If that text string is found within the line, the line is skipped and ***NOT*** written to the output file:



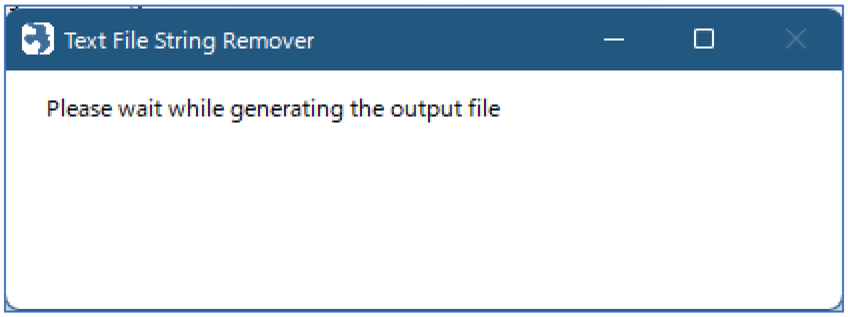
User Input Required

String To Search For In Selected File

United States

Note: the character string is CAsE SeNSiTIvE

3. A popup window is displayed as the input file is scanned (line by line); lines NOT containing the character string are written to the output file:

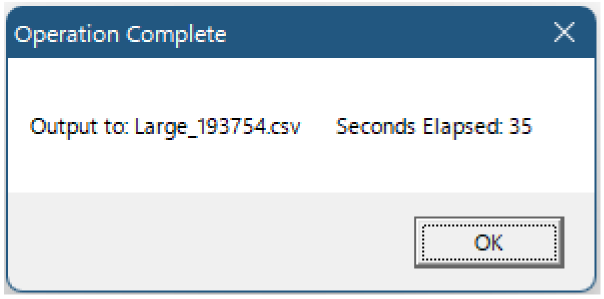


Please wait while generating the output file

Note: the output filename uses the same rootname as the input file, appended with an underline character and the 6-digit time (HHMMSS) to ensure it is a unique filename. The same file extension is used

Example: Large.Csv -> Large_193754.Csv

4. After the input file has been fully scanned a popup message is displayed that informs you what the output filename is and the elapsed time to create the output file:

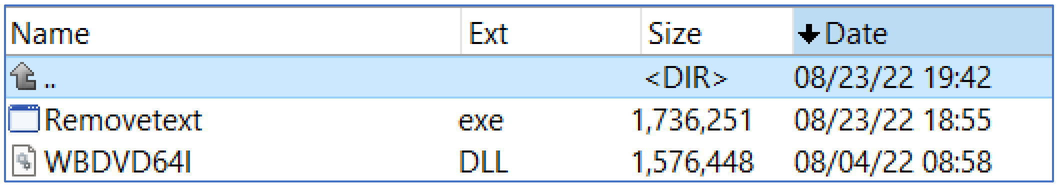


Output to: Large_193754.csv Seconds Elapsed: 35

a. Large File Support: the largest file processed with this utility (thus far) is 33.3GB
b. Sample Processing Statistics (removed “United States” from the .CSV file; the line length for the input file was 1,165 characters wide):

| Start # Of Lines | End # Of Lines | Time Elapsed |
|---|---|---|
| 200,298 | 4,824 | 35 seconds |
| 3,101,915 | 75,694 | 9:43 |

c. Files required for this application:



| Name | Ext | Size | Date |
|---|---|---|---|
| .. | | <DIR> | 08/23/22 19:42 |
| Removetext | exe | 1,736,251 | 08/23/22 18:55 |
| WBDVD64I | DLL | 1,576,448 | 08/04/22 08:58 |

**Batch Process:** Oftentimes, multiple passes against a source file are desired to remove lines that contain several text strings. This application can be controlled via a simple MsDos batch file (version 1.1 and higher) by passing parameters to the application:

RemoveText <inputfile> <search string to remove> <outputfile>

Two notable (and hard-fast) rules concerning parameters:

1. There are spaces in-between each of the string parameters.
2. Each parameter CANNOT have a space character embedded

Using notepad (or a text editor application), create a *.Bat file in the same folder that contains the file to be searched by this application. Build each command line with the desired outcome – for example:

Inputfile = testcsv.csv
Search string to remove = dodge
Outputfile = out_a.csv

would be entered into the batch file on a single line as:

```
removetext testcsv.csv dodge out_a.csv
```

When the batch file is executed, the 'testcsv.csv' file is opened; each line that contains 'dodge' is skipped and all other lines are written to the 'out_a.csv' file.

Note: When parameters are passed to this application, no message is displayed onscreen after the file is fully processed – this is by design, as it's assumed that 'silent operation' is desired with no user interaction desired (i.e. build the batch file, execute it and let it run unattended for a lengthy period of time).

This operation mode enables you to make multiple passes against a source file to continually 'whittle down' its content for multiple search strings. For example, the input file contains lines of text that contain 'dodge', 'plymouth' and 'fiat'; the batch file to remove lines that contain these three words looks like this:

```
removetext testcsv.csv dodge out_a.csv
removetext out_a.csv plymouth out_b.csv
removetext out_b.csv fiat out_c.csv
```

Notice the matching highlighting above (yellow and purple); each successive execution of the application is opening the output file from the previous execution to ensure the final output file (out_c.csv) does not contain all 3 of the search terms. This is a very important fact to keep in mind when building the batch file to process a file multiple times.

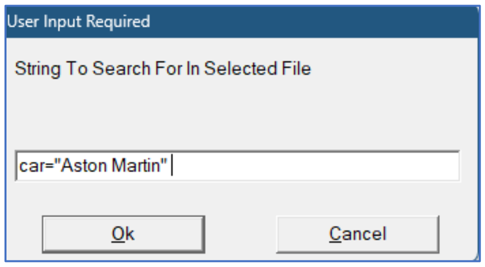
**MsDos Batch File Caveat:** A problem with MsDos batch files is passing a quotation symbol ("). It is not possible to pass a text string to search for within the source file that contains a quotation symbol to this application, as that symbol is stripped out of the parameter string. For example, the following parameter is passed:

```
removetext testcsv.csv car=”Aston Martin” output.csv
```

The parameter passed to this executable becomes:  car=Aston Martin

Naturally, that string won't exist in the testcsv.csv input file, so the desired text lines aren't found (and subsequently removed) from the input file.

The only workaround for this MsDos-created problem is to use the Manual Process method, which permits you to enter the search string with the quotation mark(s) and have it work as desired:



String To Search For In Selected File

car="Aston Martin"

```
; 1.0 Creation date: 24 August 2022
; 1.1 Creation date: 4 March 2023 (added exectable parameters to pass the input filename and output filename to
enable .BAT file usage
; Author: Scott Daughtry
; Purpose: Export a text file to a new text file, omitting lines that contain a user-defined string

; establish variables
cInputString = " "                  ; user-entered value used when scanning the input file
cOpenFile = " "                     ; contains the selected file to open
cFileName = " "                          ; file name for selected file
cFileExtension = " "                ; file extension for selected file
cWriteFile = " "                    ; will contain the output filename
cOpenPath = " "                     ; needed for AskFileName function?
cTypes = "All Files|*.*|Comma Separated File|*.csv|Text File|*.txt"  ; needed for AskFileName function
cOutputLine = " "                   ; line of text to write to the output file
cDateTime = " "                     ; store the computer's date & time
cDateTimeString = " "                   ; placeholder to manipulate the cDateTime variable
nCount = 0                          ; stores # of occurrences of the user-defined string
nContinue = 0                       ; used to determine if line should be written to new file
nHandleRead = 0                        ; handle for the file to open in read mode
nHandleWrite = 0                       ; handle for the file to open in write mode
tTimeStart = 0                      ; starting computer time
tTimeEnd = 0                        ; ending computer time
tTimeDiff = 0                       ; variable to calculate elapsed time

; generate unique number based upon computer time for output file name
cDateTime = TimeYmdHms()                ; YYYY:MM:DD:HH:MM:SS
cDateTimeString = StrSub(cDateTime,12,2) : StrSub(cDateTime,15,2) : StrSub(cDateTime,18,2)

; PARAM1 (passed input filename)
if param0 == 0
  cOpenFile = AskFileName("Select the File to Open/Scan", cOpenPath, cTypes, "*.csv", 1)
else
  cOpenFile = "%param1%"
endif

; PARAM2 (text string to search for)
if param0 > 1
  cInputString = "%param2%"
else
  cInputString = AskLine("User Input Required", "String To Search For In Selected File", "", 0)
endif

; trim string of right-trailing characters from the input box that might exist
cInputString = StrTrim(cInputString,2)

; User input a string shorter than 2 characters; tell them and terminate the application
if StrLen(cInputString) < 2
  Message("ERROR","No search string entered. Ending program")
  Exit
endif

; open the selected file
```

```
nHandleRead = FileOpen( cOpenFile, "READ" )

; parse the filename from its file extension
cFileName = FileRoot(cOpenFile)
cFileExtension = FileExtension(cOpenFile)

; PARAM3 (build the unique output filename)
if param0 == 3
  ; the output filename was passed as a 2nd parameter
  cWriteFile = "%param3%"
else
  ; No output filename passed as a parameter
  cWriteFile = cFilename : "_" : cDateTimeString : "." : cFileExtension
endif

; create & open the new output file
nHandleWrite = FileOpen( cWriteFile, "WRITE")

; display a please wait dialog window to entertain the user
BoxOpen("Text File String Remover","Please wait while generating the output file")

; start the clock
tTimeStart = TimeYmdHms()

; loop through the input file and output lines that do NOT contain the user-defined text string
While @TRUE
  cOutputLine = FileRead(nHandleRead)
  ; if we reached EOF then break the loop
  IF cOutputLine == "*EOF*" Then Break
  ; reset the search counter
  nCount = 0

  ; search the string for the user-defined text
  nCount = StrCnt(cOutputLine, cInputString, 1, -1, 0)

  ; write the line if the user-defined string was NOT found in the line of text
  If nCount == 0
    FileWrite( nHandleWrite, cOutputLine)
  EndIf
EndWhile

; close the two files
FileClose(nHandleRead)
FileClose(nHandleWrite)

; end the clock
tTimeEnd = TimeYmdHms()

; calculate the elapsed time and build the string for the final message()
tTimeDiff = TimeDiffSecs( tTimeEnd, tTimeStart)

; close the dialog window
BoxShut()
```

```
; inform the user we're done if no parameters passed
if param0 == 0
  Message("Operation Complete","Output to: " : cWriteFile : "     " : " Seconds Elapsed: " : tTimeDiff)
endif

; exit the program
Exit
```