listcompare

A file comparison tool

by Markus Gnam

Program version: 3.5.2.8
Document version: 000119

Table of contents

listcompare.exe - Compare two lists based on a key

```
listcompare.exe - Compare two lists based on a key
(c) Markus Gnam 2024. Version 3.5.2.8 20240401
```

USAGE: **listcompare <file1> <file2> [options]**

Compare two files based on a key given for each file.
They don't need to be sorted. There is no size limit.
Defaults, if no options are specified:
Key for comparisons for both files is the first field.
Default field separator is white space (spaces/tabs).

Result output (if keys for these files exist):
key in both files => File: **file1_and_file2.txt**
key only in file1 => File: **only_in_file1.txt**
key only in file2 => File: **only_in_file2.txt**

For additional output of the value files use "-v"
Value files contain the whole line where the key occurs.

For all options and more details type "listcompare --help"

OPTIONS: Details about all options and their defaults:
Mandatory arguments to long options are mandatory for short options too.

```
-f, --fs=FIELDSEP     The field separator [default: " "]
                      E.g. tabulator: --fs=\t or one space: --fs="[ ]"
    --fs1=FIELDSEP    The field separator of file 1 [default: " "]
    --fs2=FIELDSEP    The field separator of file 2 [default: " "]

-k, --key=FIELDNO     The field number to compare [default: 1]
                      Use --key=0 to get the whole line as key.
                      For Special cases see manual. E.g. part of field 8:
                      Field No[,Substr Pos][,Substr Length]: --key=8,1,4
                      A field name instead of a field number can be used:
                      E.g. --key=7 or with fname=: --key=fname=last_name
    --key1=FIELDNO    The field number of file 1 to compare [default: 1]
    --key2=FIELDNO    The field number of file 2 to compare [default: 1]

-v, --value={0,1}     Generate files with values (whole lines)
                      additional to the key files [default: 0]
    --value1=FL       Optional field list for value file 1 [default 0]
    --value2=FL       Optional field list for value file 2 [default 0]

-h, --header=N        Take care of N header lines [default: 0]
    --header1=N       Take care of N header lines of file 1 [default: 0]
    --header2=N       Take care of N header lines of file 2 [default: 0]

-j, --join={0,1}      Generate "inner join" files [default: 0]
    --join1=FL        Optional field list for join file 1 [default 0]
    --join2=FL        Optional field list for join file 2 [default 0]
    --joinleft={0,1}  Generate "left join" files [default: 0]
    --joinhint={0,1}  Separate join files by listcompare_hint [default: 1]
    --join_ofd={0,1}  Compare only first file 2 duplicate key [default: 0]

-c, --count={0,1}     Count the result files [default: 0]
-p, --print={0,1,2}   Print the result files: all 1, diffs 2, [default: 0]
    --csv1={0,1}      Parse CSV file 1 with RFC 4180 standard [default: 0]
    --csv2={0,1}      Parse CSV file 2 with RFC 4180 standard [default: 0]
```

```
    --lpadzero=N      Left pad the key with "0" to length N [default: 0]
    --tabcon={0,1}    Use tab as field concatenation suffix [default: 0]
    --conuni={0,1}    Union all keys of concatenated fields [default: 0]

-a, --difa={0,1}      Generate a difference file for fields [default: 0]
                      Use this option to compare all fields with same name
    --dife=FL         Optional field list to exclude for --difa comparison
    --difi=FL         Optional field list to include for --difa comparison
    --difa_info1=FL   file 1 info fields to add with the --difa difference
    --difa_info2=FL   file 2 info fields to add with the --difa difference
    --difa_seq={0,1}  Compare the same file sequentially [default: 0]
    --difa_ofd={0,1}  Compare only first file 2 duplicate key [default: 0]
    --difa_ci={0,1}   Case insensitive comparison of fields [default: 0]
    --difa_tr={0,1}   Trim all fields before the comparison [default: 0]
    --difa_fun=file   Use optional configuration file for difa functions
                      e.g. --difa_fun=difa_functions.txt
    --difa_level=N    N=0: Show only differences [default]
                      N=1: + no differences for all records           [0+1]
                      N=2: + no differences only for equal records    [0+2]
                      N=3: + only record keys for all records         [0+3]
                      N=4: + only record keys for equal records       [0+4]
                      N=5: + record keys only in file 1 or in file 2  [0+5]
                      N=6: = difa_level 1 + difa_level 5              [1+5]
                      N=7: = difa_level 2 + difa_level 5              [2+5]
                      N=8: = difa_level 3 + difa_level 5              [3+5]
                      N=9: = difa_level 4 + difa_level 5              [4+5]

-d, --diff={0,1}      Generate a difference file for values [default: 0]
    --dif1=FIELDNO    The field number of file 1 to compare [default: 0]
                      Use --dif1=0 for the whole line as value. See manual.
    --dif2=FIELDNO    The field number of file 2 to compare [default: 0]
                      Use --dif2=0 for the whole line as value. See manual.

    --condition1=RE   Condition for key1 to be fulfilled [default: 0]
                      RE=<Field No>:~:<regex in double quotes>. See manual.
    --condition2=RE   Condition for key2 to be fulfilled [default: 0]
                      RE=<Field No>:~:<regex in double quotes>. See manual.

    --trim={0,1}      Trim the key [default: 1]
    --ltrim={0,1}     Ltrim the key [default: 0]
    --rtrim={0,1}     Rtrim the key [default: 0]
    --upper={0,1}     Set the key to upper case [default: 0]
    --lower={0,1}     Set the key to lower case [default: 0]
    --utf81={0,1}     For substring use with UTF-8 file 1 [default: 0]
    --utf82={0,1}     For substring use with UTF-8 file 2 [default: 0]
    --debug={0,1}     Generate debug files [default: 0]
    --largefiles=N    Split file into pieces of N megabytes [default: 300]
-?, --help            Display this help and exit
    --version         Output version information and exit
```

Notes

```
You can use -, -- or / as first option character(s).
E.g. for the option -d, --diff={0,1} valid alternatives are
-diff=1, --diff=1, /diff=1
{0,1} means: Use "0" for No (False) or "1" for Yes (True).
Using the {0,1} or {0,1,2} options without values means "1".
For example, you can use either --diff=1 or simply --diff
For the short option you can use -d, -d=1 or -d 1
```

Simple Usage

Key files contain the keys.
Result output of the key files in sorted order (if keys exist):

```
key in both files => File: file1_and_file2.txt
key only in file1 => File: only_in_file1.txt
key only in file2 => File: only_in_file2.txt
```

E.g.
File numbers1.txt with content:
9
4

File numbers2.txt with content:
2
9

Example 1 **(example1.cmd):**

```
:: Most basic example: Compare keys of first field with FS white space:
listcompare numbers1.txt numbers2.txt
```

```
=> Result output of key files:
file1_and_file2.txt with content:
9
only_in_file1.txt with content:
4
only_in_file2.txt with content:
2
```

Venn diagram: file1_and_file2.txt shows the intersection of two lists

numbers1.txt: {9,4} numbers2.txt: {2,9}

Further, key files should normally not contain any duplicates so they are written (if any) to key_file1_duplicates.txt and key_file2_duplicates.txt

Value files contain the whole line where the key occurs.
Result output of the value files in original order (if values exist):

```
key in both files => Values of file1: file1_and_file2_values_file1.txt
key in both files => Values of file2: file1_and_file2_values_file2.txt
key only in file1 => Values of file1: only_in_file1_values.txt
key only in file2 => Values of file2: only_in_file2_values.txt
```

Value files are interesting if the files contain more than one field.

Example 2 (example2.cmd):

```
:: Field separator: Tab, Key: Field "last_name", values, one header line:
listcompare names1.txt names2.txt --fs=\t --key=fname=last_name -v -h=1
```

Goal: Get citizens with same last name who exist in file1 and in file2

file1: names1.txt

names1

	1	2	3	4	5	6	7
1	state	abbreviation	county	city	zip	first_name	last_name
2	Louisiana	LA	Orleans	New Orleans	70116	James	Butt
3	Alaska	AK	Anchorage	Anchorage	99501	Thomas	Mackay
4	California	CA	Santa Clara	San Jose	95111	Piers	Dowd
5	New York	NY	Suffolk	Middle Island	11953	Adam	Mackay
6	Ohio	OH	Geauga	Chagrin Falls	44023	Nathan	Anderson
7	Arizona	AZ	Maricopa	Phoenix	85013	Lisa	McLean
8	Tennessee	TN	Warren	Mc Minnville	37110	Matt	McLean
9	Michigan	MI	Wayne	Taylor	48180	Sam	Morgan

Key intersection of file1 and file2 is highlighted with a red rectangle.

file2: names2.txt

names2

	1	2	3	4	5	6	7
1	state	abbreviation	county	city	zip	first_name	last_name
2	Florida	FL	Seminole	Longwood	32750	Amy	Piper
3	New Jersey	NJ	Hudson	Kearny	07032	Trevor	Anderson
4	Pennsylvania	PA	Lackawanna	Old Forge	18518	Steven	Coleman
5	California	CA	San Diego	San Diego	92126	Blake	McLean
6	California	CA	Riverside	Thousand Palr	92276	Dylan	Coleman
7	Kansas	KS	Dickinson	Abilene	67410	Wendy	Dowd
8	California	CA	Marin	Novato	94945	Dominic	Wilkins
9	Idaho	ID	Ada	Boise	83709	Simon	Anderson

Key intersection of file1 and file2 is highlighted with a red rectangle.

⇨ **Results, key files (keys uniquely listed in sorted order):**

file1_and_file2.txt:
```
Anderson
Dowd
McLean
```

only_in_file1.txt:
```
Butt
Mackay
Morgan
```

only_in_file2.txt
```
Coleman
Piper
Wilkins
```

Further, key files about duplicate keys are written:

key_file1_duplicates.txt:
Mackay
McLean

key_file2_duplicates.txt
Anderson
Coleman

➪ **Results, value files (lines completely listed in original order):**
Opposite to key files these value files may contain duplicate key values.

file1_and_file2_values_file1.txt

file1_and_file2_values_file1

	1	2	3	4	5	6	7
1	state	abbreviation	county	city	zip	first_name	last_name
2	California	CA	Santa Clara	San Jose	95111	Piers	Dowd
3	Ohio	OH	Geauga	Chagrin Falls	44023	Nathan	Anderson
4	Arizona	AZ	Maricopa	Phoenix	85013	Lisa	McLean
5	Tennessee	TN	Warren	Mc Minnville	37110	Matt	McLean

*All lines of file1 **with** key intersection of file1 and file2 in original order.*

file1_and_file2_values_file2.txt

file1_and_file2_values_file2

	1	2	3	4	5	6	7
1	state	abbreviation	county	city	zip	first_name	last_name
2	New Jersey	NJ	Hudson	Kearny	07032	Trevor	Anderson
3	California	CA	San Diego	San Diego	92126	Blake	McLean
4	Kansas	KS	Dickinson	Abilene	67410	Wendy	Dowd
5	Idaho	ID	Ada	Boise	83709	Simon	Anderson

*All lines of file2 **with** key intersection of file1 and file2 in original order.*

only_in_file1_values.txt

only_in_file1_values

	1	2	3	4	5	6	7
1	state	abbreviation	county	city	zip	first_name	last_name
2	Louisiana	LA	Orleans	New Orleans	70116	James	Butt
3	Alaska	AK	Anchorage	Anchorage	99501	Thomas	Mackay
4	New York	NY	Suffolk	Middle Island	11953	Adam	Mackay
5	Michigan	MI	Wayne	Taylor	48180	Sam	Morgan

*All lines of file1 with **no** key intersection of file1 and file2 in original order.*

only_in_file2_values.txt

only_in_file2_values

	1	2	3	4	5	6	7
1	state	abbreviation	county	city	zip	first_name	last_name
2	Florida	FL	Seminole	Longwood	32750	Amy	Piper
3	Pennsylvania	PA	Lackawanna	Old Forge	18518	Steven	Coleman
4	California	CA	Riverside	Thousand Palr	92276	Dylan	Coleman
5	California	CA	Marin	Novato	94945	Dominic	Wilkins

*All lines of file2 with **no** key intersection of file1 and file2 in original order.*

Advanced Usage

Join files behave exactly like a database inner join on a key for two tables. **Result file: joined_key_both_in_file1_and_file2_values.txt**

Example 3 (example3.cmd):

```
listcompare fruits1.txt fruits2.txt --fs=\t --header=1 --
key=fname=fruit_id --join
```

Input: Content of "fruits1.txt":

id	fruit	fruit_id
1	Orange	5
2	Pear	2
3	Apple	9
4	Banana	3

Input: Content of "fruits2.txt":

id	fruit	fruit_id
1	Cranberry	18
2	Grapes	7
3	Orange	5
4	Apple	9

Result: Content of "joined_key_both_in_file1_and_file2_values.txt":

id	fruit	fruit_id	listcompare_hint	id	fruit	fruit_id
1	Orange	5	joined	3	Orange	5
3	Apple	9	joined	4	Apple	9

Explanation: This join is an inner join on the key „fruit_id". This corresponds to the SQL instruction: "SELECT a., b.* FROM fruits1 AS a JOIN fruits2 AS b ON (a.fruit_id = b.fruit_id)"*
The content of the left file is separated from the content of the right file by the „listcompare_hint" entry „joined".

Left join files behave exactly like a database left join on a key for two tables. **Result file: joined_key_file1_leftjoin_values.txt**

E.g. (example3.cmd):

```
listcompare fruits1.txt fruits2.txt --fs=\t --header=1 --
key=fname=fruit_id --joinleft
```

Result: Content of "joined_key_file1_leftjoin_values.txt":

Id	fruit	fruit_id	listcompare_hint	id	fruit	fruit_id
1	Orange	5	joined both	3	Orange	5
2	Pear	2	joined left			
3	Apple	9	joined both	4	Apple	9
4	Banana	3	joined left			

Explanation: This join is a left join on the key „fruit_id". This corresponds to the SQL instruction: "SELECT a., b.* FROM fruits1 AS a LEFT JOIN fruits2 AS b ON (a.fruit_id = b.fruit_id)"*
The content of the left file is separated from the content of the right file by the „listcompare_hint" entries „joined both" or „joined left".

Hints: 1. If for some reason you want to omit the listcompare_hint, add the option --joinhint=0
2. The option --join_ofd uses only the first row of duplicate keys from file 2 for the comparison
3. In case of different field separators for the two files, the separator and the csv setting and the file ending of the first (left) file are used for the join output also for the second (right) file

CSV file format

CSV files, as the name implies, are usually comma separated. However, they can also be semicolon separated or tab separated or use another separator. The parsing standard may be "RFC 4180" or "Delimiter Only".

Parsing standard "RFC 4180"

For comma separated and semicolon separated files the recommended parsing standard is **RFC 4180**: https://tools.ietf.org/html/rfc4180
In short (http://mcollado.z15.es/gawk-extras/csvmode/doc/csvmode.html):
A CSV file is a sequence of records separated by newline marks. A CSV record is a sequence of fields separated by commas. A field can contain almost any text. If a field contains commas, newlines or double quotes it must be enclosed in double quotes. Double quotes inside a field must be escaped by doubling them.

For RFC 4180 the options **--csv1** for file1 and **--csv2** for file2 **are needed**. The key and difference output is unquoted, the values output is original, f.i. --fs=, --key1=7 --key2=9 --csv1=1 --csv2=1 --lpadzero=13

Parsing standard "Delimiter only"

For tab separated files (separator "\t") the recommended parsing standard is **Delimiter only** if the file doesn't contain any newlines. In this case the tab is an unambiguous delimiter which doesn't need any quoting since a tab should not appear inside any field. "Delimiter only" simply means split the record by the delimiter into fields without further parsing.

This is the default for listcompare.

Difference files

General Difference method --diff (short option -d)

The Option --diff=1 generates the Difference file differences_values.txt (if results exist). Difference file means different values for the same key. Default for the value is the whole line. This can be adjusted to a specific field number or further specifications for dif1 and dif2. The "Special cases for keys: Parts and concatenation, functions" are valid for dif1, dif2, too. Especially the trim function is interesting.

```
E.g. ... --key=1 --diff --dif1=0:trim --dif2=0:trim
Or   ... --key=fname=GTIN --diff --dif1=fname=Title --dif2=fname=Title
```

Example 4 **(example4.cmd):**
:: Field separator: Tab, Key: Field 7, Difference file, one header line:
listcompare names1.txt names2.txt --fs=\t --key=7 --diff --header=1

```
=> Result file differences_values.txt with sample content:
*** Differences file1 and file2 for key: McLean
Tennessee   TN  Warren     Mc Minnville  37110  Matt   McLean
California  CA  San Diego  San Diego     92126  Blake  McLean
```

Hint: With the --diff option, additional files differences_file1.txt and differences_file2.txt are created with original content (whole line). They may be needed for database updates.

CSV file Difference method --difa (short option -a)

The -a option compares CSV files based on the key. Use it to simply compare all fields with same name. The file "differences_fields.txt" shows differences for fields with different values for the same key.

Example 5 **(example5.cmd):**

```
:: Field separator: Comma, RFC 4180 parsing, one header line, Key: field
name cust_no, CSV difference file:
listcompare customer1.csv customer2.csv --fs=, --csv1 --csv2 --header=1 -
-key=fname=cust_no -a
```

file 1: customer1.csv

	1	2	3	4	5	6	7
0	**cust_no**	**name**	**address**	**city**	**state**	**zip**	**last_invoice_date**
1	1560	Sam Witherspoon	15243 Underwater Fwy.	Marathon	FL	35003	09.04.2020 08:14:52
2	1563	Theresa Kunec	203 12th Ave. Box 746	Giribaldi	OR	91187	09.05.2020 12:05:42
3	1645	Michael Spurling	PO Box 5451-F	Sarasota	FL	32274	
4	1680	Desmond Ortega	6133 1/3 Stone Avenue	St Simons Isle	GA	32521	02.01.2020 09:33:40
5	2118	Harry Bathbone	63365 Nez Perce Street	Largo	FL	34684	08.06.2020 11:21:06
6	2135	Lloyd Fellows	1455 North 44th St.	Eugene	OR	90427	24.06.2020 10:29:46
7	2354	Rudolph Claus	42 Aqua Lane	Houston	TX	77079	08.02.2020 14:37:45
8	2975	Bill Wyers	#73 King Salmon Way	Lugoff	NC	29078	N/A
9	2984	Shirley Mathers	4734 Melinda St.	Hoover	AL	32145	05.05.2020 17:18:11
10	3041	Nancy Miller	634 Complex Ave.	Pelham	AL	32145	28.02.2020 19:29:34

file 2: customer2.csv

	1	2	3	4	5	6	7
0	**cust_no**	**last_invoice**	**name**	**address**	**city**	**state**	**zip**
1	1563	20.10.2020 00:00:28	Theresa Kunec	Box 264 Pleasure Point	Catalina Island	CA	90740
2	1560	25.10.2020 20:22:30	Sam Witherspoon	220 Elm Street	Venice	FL	39224
3	1680	02.01.2020 09:33:40	Desmond Ortega	3562 NW Bruce Street	Milwaukie	OR	96277
4	1645	N/A	Michael Spurling	PO Box 5459-B	Sarasota	FL	32274
5	2135	21.07.2020 08:40:05	Lloyd Fellows	1701-D N Broadway	Santa Maria	CA	95443
6	2118	02.12.2020 00:00:07	Harry T. Bathbone	6345 W. Shore Lane	Santa Monica	CA	90410
7	3041	11.10.2020 10:11:33	Nancy Bean	7865 NE Barber Ct.	Portland	OR	99271
8	2354	08.02.2020 14:37:45	Rudolph Claus	43 Aqua Lane	Houston	TX	77079
9	2975		Bill Wyers	1043 Broad Street	Camden	SC	29020
10	2984	05.05.2020 17:18:11	Shirley Mathers	4734 Melinda St.	Hoover	AL	32145

Result: Content of file differences_fields.txt:

```
CSV files differences report for
file 1: customer1.csv
file 2: customer2.csv


INFORMATION
--------------------------
fields: key field is cust_no
file 1 fields not in file 2: last_invoice_date
file 2 fields not in file 1: last_invoice
```

```
used option:
compare all fields with same name

compared fields
(file 1 field no. "field name" <-> file 2 field no. "field name"):
1 "cust_no" <-> 1 "cust_no",
2 "name" <-> 3 "name",
3 "address" <-> 4 "address",
4 "city" <-> 5 "city",
5 "state" <-> 6 "state",
6 "zip" <-> 7 "zip"


RESULTS
--------------------------
Record with key "1560" is different:
 file 1 row no. 1 compared with file 2 row no. 2
  difference in file 1 field no. 3 - file 2 field no. 4
   field name: address
    file 1 > 15243 Underwater Fwy. <
    file 2 > 220 Elm Street <

  difference in file 1 field no. 4 - file 2 field no. 5
   field name: city
    file 1 > Marathon <
    file 2 > Venice <

  difference in file 1 field no. 6 - file 2 field no. 7
   field name: zip
    file 1 > 35003 <
    file 2 > 39224 <

--------------------------------------------------------------------------

Record with key "1563" is different:
 file 1 row no. 2 compared with file 2 row no. 1
  difference in file 1 field no. 3 - file 2 field no. 4
   field name: address
    file 1 > 203 12th Ave. Box 746 <
    file 2 > Box 264 Pleasure Point <

  difference in file 1 field no. 4 - file 2 field no. 5
   field name: city
    file 1 > Giribaldi <
    file 2 > Catalina Island <

  difference in file 1 field no. 5 - file 2 field no. 6
   field name: state
    file 1 > OR <
    file 2 > CA <

  difference in file 1 field no. 6 - file 2 field no. 7
   field name: zip
    file 1 > 91187 <
    file 2 > 90740 <

--------------------------------------------------------------------------
…
Report hints: The header row is row no. 0, data rows start with row no. 1
The empty value is shown with two spaces with the difference format: >  <
```

Exclude fields with option --dife

--dife=FIELDLIST
If you want to exclude fields from the --difa comparison (compare all fields with same name), you can list them with the --dife option.
You can use either **a list of field names to skip**: --dife=**fnames=state,zip**
or you can list **the field numbers of file 1 to skip**: --dife=**5,6**
When using field numbers (instead field names listed after "fnames="), this means the field names of these field numbers of file 1 are excluded.

Examples for using this option:
1. Compare all fields with same name except the field name zip

```
listcompare customer1.csv customer2.csv --fs=, --csv1 --csv2 --header=1 --key=fname=cust_no -a --dife=fnames=zip
```

Alternative: *This is the same as using* ***--dife=6***

2. Compare all fields with same name except the field names state and zip

```
listcompare customer1.csv customer2.csv --fs=, --csv1 --csv2 --header=1 --key=fname=cust_no -a --dife=fnames=state,zip
```

Alternative: *This is the same as using* ***--dife=5,6***

Result: Content of file differences_fields.txt:

```
CSV files differences report for
file 1: customer1.csv
file 2: customer2.csv


INFORMATION
--------------------------
fields: key field is cust_no
file 1 fields not in file 2: last_invoice_date
file 2 fields not in file 1: last_invoice
excluded fields: state, zip

used option:
compare all fields with same name
except fields listed with option --dife=fnames=state,zip

compared fields
(file 1 field no. "field name" <-> file 2 field no. "field name"):
1 "cust_no" <-> 1 "cust_no",
2 "name" <-> 3 "name",
3 "address" <-> 4 "address",
4 "city" <-> 5 "city"


RESULTS
--------------------------
Record with key "1560" is different:
 file 1 row no. 1 compared with file 2 row no. 2
  difference in file 1 field no. 3 - file 2 field no. 4
   field name: address
    file 1 > 15243 Underwater Fwy. <
    file 2 > 220 Elm Street <

  difference in file 1 field no. 4 - file 2 field no. 5
   field name: city
    file 1 > Marathon <
    file 2 > Venice <
…
```

Include fields with option --difi

--difi=FIELDLIST
You may have noticed the previous reports listed:
file 1 fields not in file 2: last_invoice_date
file 2 fields not in file 1: last_invoice

So, these field names can't be compared with the --difa comparison unless you add an include field list with option --difi to the --difa comparison.
This list consists of comma separated pairs of the field number mappings:
The field numbers of file 1 and 2 inside a pair are separated by a colon.
General syntax: <**file 1 field no.**>**:**<**file 2 field no.**>[**,**…]
e.g. --difi=**7:2** or --difi=**2:3,3:4,4:5,5:6,6:7,7:2**
*Only **exactly** the fields of this mapping are compared, no other fields.*

Examples for using this option:
1. Compare file 1 field no. 7 (=> field name "last_invoice_date") with file 2 field no. 2 (=> field name "last_invoice")

```
listcompare customer1.csv customer2.csv --fs=, --csv1 --csv2 --header=1 --key=fname=cust_no -a --difi=7:2
```

Result: Content of file differences_fields.txt:

```
CSV files differences report for
file 1: customer1.csv
file 2: customer2.csv


INFORMATION
--------------------------
fields: key field is cust_no
file 1 fields not in file 2: last_invoice_date
file 2 fields not in file 1: last_invoice

used option:
compare only fields included in mapping
(file 1 field no.:file 2 field no.,...)
with field number pairs mapping option --difi=7:2

compared fields
(file 1 field no. "field name" <-> file 2 field no. "field name"):
7 "last_invoice_date" <-> 2 "last_invoice"


RESULTS
--------------------------
Record with key "1560" is different:
 file 1 row no. 1 compared with file 2 row no. 2
  difference in file 1 field no. 7 - file 2 field no. 2
   field name file 1: last_invoice_date - field name file 2: last_invoice
    file 1 > 09.04.2020 08:14:52 <
    file 2 > 25.10.2020 20:22:30 <

--------------------------------------------------------------------------------

Record with key "1563" is different:
 file 1 row no. 2 compared with file 2 row no. 1
  difference in file 1 field no. 7 - file 2 field no. 2
   field name file 1: last_invoice_date - field name file 2: last_invoice
    file 1 > 09.05.2020 12:05:42 <
    file 2 > 20.10.2020 00:00:28 <
```

```
--------------------------------------------------------------------------------

Record with key "1645" is different:
 file 1 row no. 3 compared with file 2 row no. 4
  difference in file 1 field no. 7 - file 2 field no. 2
   field name file 1: last_invoice_date - field name file 2: last_invoice
    file 1 >  <
    file 2 > N/A <
…
```

2. Complete field mapping (all fields) for the customer example files:

```
listcompare customer1.csv customer2.csv --fs=, --csv1 --csv2 --header=1 --key=fname=cust_no -a --difi=1:1,2:3,3:4,4:5,5:6,6:7,7:2
```

Hints: Actually, --difi=2:3,3:4,4:5,5:6,6:7,7:2 is sufficient because 1:1 is the key field cust_no which has same values because it is used as key.

Instead of field numbers ***--difi=7:2*** *field names with fname can be used:*
--difi=fname=last_invoice_date:fname=last_invoice
However, if one file contains duplicate field names --difi has to be used with field numbers. This allows an exact field mapping without any issues.
(If --difi is not used as described, only the rightmost duplicate field of each file is compared)
A further special case is **-difi=keys**. As an example see Use of --difi=keys.

Special cases for values

Skip values with skipvalues.txt

If you want special values to be treated as same values (e.g. N/A and empty value) and therefore not to appear as differences, you have to create a tab separated file named "skipvalues.txt" (or any other name, like "skipvalues.example.txt") in the same directory with header line

```
comment, fieldname1, function1, param1, value1, value2, param2, function2, fieldname2
```

(To reproduce you can use the file skipvalues.example.txt from the examples directory in the zip file shipped with this product):

file skipvalues.example.txt:

skipvalues.example.txt ×

	1	2	3	4	5	6	7	8	9
0	**comment**	**fieldname1**	**function1**	**param1**	**value1**	**value2**	**param2**	**function2**	**fieldname2**
1		last_invoice_date				N/A			last_invoice
2		last_invoice_date			N/A				last_invoice

Example (this is the same example as the previous example, but with a file skipvalues.example.txt in the directory and the option --difa_fun):

Compare file 1 field no. 7 (=> field name "last_invoice_date") with file 2 field no. 2 (=> field name "last_invoice")

```
listcompare customer1.csv customer2.csv --fs=, --csv1 --csv2 --header=1 --key=fname=cust_no -a --difi=7:2 --difa_fun=skipvalues.example.txt
```

```
=> compare only fields included in mapping
(file 1 field no.:file 2 field no.,...)
with field number pairs mapping option --difi=7:2
... applying functions file skipvalues.example.txt
```

This **omits** the output of this difference:

```
Record with key "1645" is different:
 file 1 row no. 3 compared with file 2 row no. 4
  difference in file 1 field no. 7 - file 2 field no. 2
   field name file 1: last_invoice_date - field name file 2: last_invoice
    file 1 >  <
    file 2 > N/A <
```

More functions with --difa_fun

On top of using --difa_fun for skipping values by filling value1 and value2, there are more sophisticated possibilities to use functions:

Example 6 **(example6.cmd):**

```
:: Field separator: Tab, one header line, Key: field name cust_no, CSV
difference file, functions configuration file:
listcompare clients1.txt clients2.txt --fs=\t --header=1 --
key=fname=cust_no -a --difa_fun=difa_functions.example.txt
```

file difa_functions.example.txt:

difa_functions.example.txt ×

	1	2	3	4	5	6	7	8	9
0	**comment**	**fieldname1**	**function1**	**param1**	**value1**	**value2**	**param2**	**function2**	**fieldname2**
1		last_invoice_date	skipvalues			N/A		skipvalues	last_invoice_date
2		last_invoice_date	skipvalues		N/A			skipvalues	last_invoice_date
3							`,,.	replace	tax_rate
4		tax_rate	skipvalues		6	6%		skipvalues	tax_rate
5		tax_rate	skipvalues		6%	6		skipvalues	tax_rate
6		tax_rate	leftpad	2,0			2,0	leftpad	tax_rate
7		state	match	^.{2}					
8		country	upper					upper	country
9	#	*	lower					lower	*
10		phone	replace	\+49,0			\+49,0	replace	phone
11		phone	replace	[^0-9]+,			[^0-9]+,	replace	phone
12	#	phone	replace	[/()]+,			[/()]+,	replace	phone
13							D-	prefix	zip
14							^Mr[.]? *,	replace	contact

Available functions are leftpad, rightpad, trim, ltrim, rtrim, lower, upper, prefix, suffix, replace, match, sort.

If values are to be skipped, they have to be entered as value1 and value2. As function name "skipvalues" can be used but this name is not evaluated.

The *1 header names are used for file1 and the *2 header names for file2.

The functions are explained in the chapter: "More about functions". However, this chapter describes the use of functions for keys so there are a few differences:

1. param1 and param2: All params for a function have to be entered comma separated without putting in double quotes in the field param1 for function1 and param2 for function2. If a param contains a comma, it has to be escaped with a backtick `.
2. The configuration table is evaluated in the given line order. Each line action is based upon each other from top to bottom.

3. fieldname1: * means: apply for all fields of file 1.
   fieldname2: * means: apply for all fields of file 2.
4. Comment # => This line is skipped.
5. The match function is useful for comparing only a part of a field, e.g. ^.{2} means that only the first 2 characters of the field are used for the comparison.
   Grouping examples: Use param1: ^.{2}(.*),1 for the substring from position 3 to the end. The second parameter 1 means: Use grouping (default 0=no grouping) and cut the first parentheses as result.
   More examples: - Cut numbers inside a regex: \"A?N[.]*([0-9]+)\",1 - Match second parentheses group, then (+) first parentheses group: Use param2: ^(.{4})(.{4}),2+1 => B123A123 gets matched as A123B123
6. For match and replace a preview is available when using --debug=1
7. The sort function for differences differs from the key function: function1: sort, param1: | splits the values by the param and sorts them, e.g. the value c|b|a gets changed to a|b|c

As result of example6 **all differences are omitted** so these aren't shown:

```
field name: country
 file 1 > USA <
 file 2 > usa <

field name: tax_rate
 file 1 > 8.5 <
 file 2 > 8,5 <

field name: last_invoice_date
 file 1 >  <
 file 2 > N/A <

field name: last_invoice_date
 file 1 > N/A <
 file 2 >  <

field name: zip
 file 1 > D-10117 <
 file 2 > 10117 <

field name: country
 file 1 > DEU <
 file 2 > deu <

field name: phone
 file 1 > +49 30 8344-21 <
 file 2 > 030 8344 21 <

field name: tax_rate
 file 1 > 6 <
 file 2 > 6% <

field name: tax_rate
 file 1 > 09 <
 file 2 > 9 <

field name: contact
 file 1 > Paul Still <
 file 2 > Mr Paul Still <

field name: state
 file 1 > FLORIDA <
 file 2 > FL <
```

Global functions --difa_ci and –difa_tr

The option **--difa_ci** enables case insensitive comparison for all fields.
The option **--difa_tr** trims all fields before the comparison.

These two functions also work in combination with the configuration file above, e.g. --difa_ci creates case insensitive use of param1 and param2.

Compare only the first duplicate row

The option **--difa_ofd** uses only the first row of duplicate keys from file 2 for the comparison.
Otherwise, all rows with duplicate keys from file2 are compared (default).

Comparing the same file sequentially

When comparing the same file, it is most useful to do this sequentially line by line to track changes from one line to the next line for the key. This can be achieved with the **--difa_seq=1** command.

To be more precise, the line with the key is compared with the line of the previous occurrence of the key. So, this command even works for keys that don't appear in pairs of successive rows.

orders.csv

	1	2	3	4	5
0	**customer_id**	**article_id**	**order_date**	**ship_date**	**product_name**
1	13	567	2015-11-25	2015-11-25	3-ring staple pack
2	13	567	2015-11-26	2015-11-27	3-ring staple pack white
3	13	567	2015-12-18	2015-12-20	3-ring staple pack white, plastic

Example 7 **(example7.cmd):**

```
:: Field separator: comma, one header line, Key: field name article_id,
CSV difference file comparing the same file sequentially:
listcompare orders.csv orders.csv --fs=, --csv1 --csv2 --header=1
--key=fname=article_id -a --difa_seq=1 --dife=fnames=order_date,ship_date
--difa_info1=fnames=customer_id,order_date,ship_date
--difa_info2=fnames=customer_id,order_date,ship_date
```

The **--difa_seq** command does the sequential comparison for the same file. With this command initial field values are shown, see result file below.
Note that the --difa_seq command can only be used with difa_level=0 up to difa_level=4.

Info fields

This example introduces two other options: **--difa_info1** and **--difa_info2**
Use field names (--difa_info1=fnames=customer_id,ship_date) or field numbers (--difa_info1=1,4)
For each difference record info fields show values for important fields.
e.g. **info 1 -> customer_id: 13, order_date: 2015-11-25, ship_date: 2015-11-25**
*There are two special cases --difa_info1=**all** to show key <-> value pairs like above for all fields and --difa_info1=**line** to show the original line as value for the key.*

Result: Content of file differences_fields.txt:

```
CSV files differences report for
file 1: orders.csv
file 2: orders.csv


INFORMATION
-------------------------
fields: key field is article_id
all field names of file 1 and file 2 are identical
excluded fields: order_date, ship_date

used option:
compare all fields with same name
except fields listed with option --dife=fnames=order_date,ship_date
```

```
compare sequentially with option --difa_seq

initial fields
(field no. "field name" > first field value <):
1 "customer_id" > 13 <
2 "article_id" > 567 <
5 "product_name" > 3-ring staple pack <


RESULTS
--------------------------
Record with key "567" is different:
 file 1 row no. 1 compared with file 2 row no. 2
 info 1 -> customer_id: 13, order_date: 2015-11-25, ship_date: 2015-11-25
 info 2 -> customer_id: 13, order_date: 2015-11-26, ship_date: 2015-11-27
  difference in file 1 field no. 5 - file 2 field no. 5
   field name: product_name
    file 1 > 3-ring staple pack <
    file 2 > 3-ring staple pack white <

--------------------------------------------------------------------------------

Record with key "567" is different:
 file 1 row no. 2 compared with file 2 row no. 3
 info 1 -> customer_id: 13, order_date: 2015-11-26, ship_date: 2015-11-27
 info 2 -> customer_id: 13, order_date: 2015-12-18, ship_date: 2015-12-20
  difference in file 1 field no. 5 - file 2 field no. 5
   field name: product_name
    file 1 > 3-ring staple pack white <
    file 2 > 3-ring staple pack white, plastic <

--------------------------------------------------------------------------------
```

Special comparison levels with --difa_level

```
--difa_level=N   N=0: Show only differences [default]
                 N=1: + no differences for all records           [0+1]
                 N=2: + no differences only for equal records    [0+2]
                 N=3: + only record keys for all records         [0+3]
                 N=4: + only record keys for equal records       [0+4]
                 N=5: + record keys only in file 1 or in file 2  [0+5]
                 N=6: = difa_level 1 + difa_level 5              [1+5]
                 N=7: = difa_level 2 + difa_level 5              [2+5]
                 N=8: = difa_level 3 + difa_level 5              [3+5]
                 N=9: = difa_level 4 + difa_level 5              [4+5]

The difa_level option enables full control about which and how
differences are printed. Keys are always sorted alphabetically.

Example files:
file 1: level_testfile1.csv
```

	1	2	3	4
0	**id**	**city**	**state_id**	**population**
1	1	New York	NY	18680025
2	2	Los Angeles	CA	12531334
3	4	Miami	FL	6076316

file 2: level_testfile2.csv

	1	2	3	4
0	**id**	**city**	**state_id**	**population**
1	2	Los Angeles	CA	12531334
2	3	Chicago	IL	8586888
3	4	Miami	FL	6077894

difa_level=0

N=0: Show only differences [default]

Terms:

For the same key in both files

field differences = different field values
different records = records with field differences
no differences = equal field values
equal records = records with no field differences

For keys in both files or only in file 1 or only in file 2

records keys = show only the keys and the row numbers

Explanation: The default when comparing is --difa_level=0 to show only field differences.
Scope: **Key in both files**

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=0
```

⇨ Common header:

```
CSV files differences report for
file 1: level_testfile1.csv
file 2: level_testfile2.csv


INFORMATION
---------------------------
fields: key field is id
all field names of file 1 and file 2 are identical

used option:
compare all fields with same name

compared fields
(file 1 field no. "field name" <-> file 2 field no. "field name"):
1 "id" <-> 1 "id",
2 "city" <-> 2 "city",
3 "state_id" <-> 3 "state_id",
4 "population" <-> 4 "population"
```

➔

```
RESULTS
---------------------------
Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

--------------------------------------------------------------------------------
```

difa_level=1

N=1: Show field differences + no differences for all records

Explanation: Additionally to the field differences, also values with no differences and so completely equal records can be shown.
Scope: **Key in both files**

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=1
```

➔

```
RESULTS
--------------------------
Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1
  no difference in file 1 field no. 1 - file 2 field no. 1
   field name: id
    file 1 > 2 <
    file 2 > 2 <

  no difference in file 1 field no. 2 - file 2 field no. 2
   field name: city
    file 1 > Los Angeles <
    file 2 > Los Angeles <

  no difference in file 1 field no. 3 - file 2 field no. 3
   field name: state_id
    file 1 > CA <
    file 2 > CA <

  no difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 12531334 <
    file 2 > 12531334 <

------------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
  no difference in file 1 field no. 1 - file 2 field no. 1
   field name: id
    file 1 > 4 <
    file 2 > 4 <

  no difference in file 1 field no. 2 - file 2 field no. 2
   field name: city
    file 1 > Miami <
    file 2 > Miami <

  no difference in file 1 field no. 3 - file 2 field no. 3
   field name: state_id
    file 1 > FL <
    file 2 > FL <

  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

------------------------------------------------------------------------
```

difa_level=2

N=2: Show field differences + no differences only for equal records

Explanation: Additionally to the field differences, show no differences only for equal records.
Scope: **Key in both files**

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=2
```

➔

```
RESULTS
-------------------------
Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1
  no difference in file 1 field no. 1 - file 2 field no. 1
   field name: id
    file 1 > 2 <
    file 2 > 2 <

  no difference in file 1 field no. 2 - file 2 field no. 2
   field name: city
    file 1 > Los Angeles <
    file 2 > Los Angeles <

  no difference in file 1 field no. 3 - file 2 field no. 3
   field name: state_id
    file 1 > CA <
    file 2 > CA <

  no difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 12531334 <
    file 2 > 12531334 <

------------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

------------------------------------------------------------------------
```

difa_level=3

N=3: Show field differences + only record keys for all records

Explanation: Show only record keys for both field differences and equal records.
Scope: **Key in both files**

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --key=fname=id --header=1 -a --difa_level=3
```

➔

```
RESULTS
-------------------------
Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1

------------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3

------------------------------------------------------------------------
```

difa_level=4

N=4: Show field differences + only record keys for equal records

Explanation: Show field differences plus only record keys for equal records.
Scope: **Key in both files**

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=4
```

➔

```
RESULTS
-------------------------
Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1

-----------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

-----------------------------------------------------------------------
```

difa_level=5

N=5: Show field differences + record keys only in file 1 or in file 2

Explanation: Show field differences plus record keys that exist only in file 1 or in file 2.
Scope: **Key in both files, key only in file 1, key only in file 2**

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=5
```

➔

```
RESULTS
-------------------------
Record with key "1" exists only in file 1 row no. 1

-----------------------------------------------------------------------

Record with key "3" exists only in file 2 row no. 2

-----------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

-----------------------------------------------------------------------
```

difa_level=6

N=6: Show field differences + no differences for all records + record keys only in file 1 or in file 2

Explanation: Show field differences and no differences both for different and equal records plus record keys that exist only in file 1 or in file 2.
Scope: **Key in both files, key only in file 1, key only in file 2**

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=6
```

➔

```
RESULTS
--------------------------
Record with key "1" exists only in file 1 row no. 1

------------------------------------------------------------------------

Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1
  no difference in file 1 field no. 1 - file 2 field no. 1
   field name: id
    file 1 > 2 <
    file 2 > 2 <

  no difference in file 1 field no. 2 - file 2 field no. 2
   field name: city
    file 1 > Los Angeles <
    file 2 > Los Angeles <

  no difference in file 1 field no. 3 - file 2 field no. 3
   field name: state_id
    file 1 > CA <
    file 2 > CA <

  no difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 12531334 <
    file 2 > 12531334 <

------------------------------------------------------------------------

Record with key "3" exists only in file 2 row no. 2

------------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
  no difference in file 1 field no. 1 - file 2 field no. 1
   field name: id
    file 1 > 4 <
    file 2 > 4 <

  no difference in file 1 field no. 2 - file 2 field no. 2
   field name: city
    file 1 > Miami <
    file 2 > Miami <

  no difference in file 1 field no. 3 - file 2 field no. 3
   field name: state_id
    file 1 > FL <
    file 2 > FL <

  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

------------------------------------------------------------------------
```

difa_level=7

N=7: Show field differences + no differences only for equal records + record keys only in file 1 or in file 2

Explanation: Show field differences plus no differences only for equal records plus record keys that exist only in file 1 or in file 2.
Scope: **Key in both files, key only in file 1, key only in file 2**

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=7
```

➔

```
RESULTS
--------------------------
Record with key "1" exists only in file 1 row no. 1

------------------------------------------------------------------------

Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1
  no difference in file 1 field no. 1 - file 2 field no. 1
   field name: id
    file 1 > 2 <
    file 2 > 2 <

  no difference in file 1 field no. 2 - file 2 field no. 2
   field name: city
    file 1 > Los Angeles <
    file 2 > Los Angeles <

  no difference in file 1 field no. 3 - file 2 field no. 3
   field name: state_id
    file 1 > CA <
    file 2 > CA <

  no difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 12531334 <
    file 2 > 12531334 <

------------------------------------------------------------------------

Record with key "3" exists only in file 2 row no. 2

------------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

------------------------------------------------------------------------
```

difa_level=8

N=8: Show field differences + only record keys for all records + record keys only in file 1 or in file 2

Explanation: Show only record keys for both different and equal records. Additionally, show record keys that exist only in file 1 or in file 2.
Scope: **Key in both files, key only in file 1, key only in file 2**

Example:

Example 8 **(example8.cmd):**

```
:: Field separator: comma, Key: field name id, one header line, CSV
difference file with difa_level=8
```

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=8
```

➔

```
RESULTS
--------------------------
Record with key "1" exists only in file 1 row no. 1

-------------------------------------------------------------------------

Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1

-------------------------------------------------------------------------

Record with key "3" exists only in file 2 row no. 2

-------------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3

-------------------------------------------------------------------------
```

Use of --difi=keys

If you want to compare only if the keys are present in file 1 and file 2, you can use the special difi scope "--difi=keys" with difa_level=8.

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=8 --difi=keys
```

Then, there are no field differences if the keys exist in both files because the keys always have same values since they are used as keys. Therefore these lines are shown as "exists both in file 1 and file 2".

```
RESULTS
--------------------------
Record with key "1" exists only in file 1 row no. 1

-------------------------------------------------------------------------

Record with key "2" exists both in file 1 and file 2
 file 1 row no. 2 compared with file 2 row no. 1

-------------------------------------------------------------------------

Record with key "3" exists only in file 2 row no. 2

-------------------------------------------------------------------------

Record with key "4" exists both in file 1 and file 2
 file 1 row no. 3 compared with file 2 row no. 3

-------------------------------------------------------------------------
```

difa_level=9

N=9: Show field differences + only record keys for equal records + record keys only in file 1 or in file 2

Explanation: Show field differences plus only record keys for equal records plus show record keys that exist only in file 1 or in file 2.
Scope: Key in both files, key only in file 1, key only in file 2

Example:

Example 9 **(example9.cmd):**

```
:: Field separator: comma, Key: field name id, one header line, CSV
difference file with difa_level=9
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=9
```

➔

```
RESULTS
-------------------------
Record with key "1" exists only in file 1 row no. 1

-----------------------------------------------------------------------

Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1

-----------------------------------------------------------------------

Record with key "3" exists only in file 2 row no. 2

-----------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

-----------------------------------------------------------------------
```

Use of --difa_info1=all

difa_level=9 with additional use of the info fields difa_info1=all and difa_info2=all shows all field names and their values in one single line.

Example:

```
listcompare level_testfile1.csv level_testfile2.csv -fs=, -csv1 -csv2 --
key=fname=id --header=1 -a --difa_level=9 --difa_info1=all --
difa_info2=all
```

➔

```
RESULTS
-------------------------
Record with key "1" exists only in file 1 row no. 1
 info 1 -> id: 1, city: New York, state_id: NY, population: 18680025

-----------------------------------------------------------------------

Record with key "2" is equal:
 file 1 row no. 2 compared with file 2 row no. 1
 info 1 -> id: 2, city: Los Angeles, state_id: CA, population: 12531334
 info 2 -> id: 2, city: Los Angeles, state_id: CA, population: 12531334

-----------------------------------------------------------------------

Record with key "3" exists only in file 2 row no. 2
 info 2 -> id: 3, city: Chicago, state_id: IL, population: 8586888

-----------------------------------------------------------------------

Record with key "4" is different:
 file 1 row no. 3 compared with file 2 row no. 3
 info 1 -> id: 4, city: Miami, state_id: FL, population: 6076316
 info 2 -> id: 4, city: Miami, state_id: FL, population: 6077894
```

```
  difference in file 1 field no. 4 - file 2 field no. 4
   field name: population
    file 1 > 6076316 <
    file 2 > 6077894 <

-------------------------------------------------------------------------
```

Special cases for keys: Parts and concatenation, functions

```
  Syntax: field      Field No[,Substr Pos][,Substr Length]
          function [:Function name 1][,Param1],[,Param2] ...
     next function [:Function name 2][,Param1],[,Param2] ...
  Repeat           [+Field No of next field to concatenate] ... ... ...
```

E.g. **listcompare file1.txt file2.txt --fs=\t --key1=18,1,4:trim+7:leftpad,"13","0" --key2=17,2:trim+20:trim:leftpad,"13","0"**

Explanation for --key1=18,1,4:trim+7:leftpad,"13","0"
From field 18 cut substring 1 to 4 and trim it. Concatenate (+) with the result of: Left pad field 7 with sign "0" to length 13.

Explanation for --key2=17,2:trim+20:trim:leftpad,"13","0"
From field 17 cut substring 2 to the end and trim it. Concatenate (+) with the result of: Trim field 20. Left pad this with sign "0" to length 13.

More about functions

Function arguments: The signs [:,+] have to be escaped with a backtick `.
E.g. replacing all dots with commas: --key=fname=price:replace,"\.","`,"
[BTW, the same applies to field names, e.g. --key1=fname=`+code,1,4
For use of spaces in field names put the fname value in double quotes.]
The first argument has to be omitted when calling the function.
The function parameters should always be put in double quotes.

Available functions (only lower case for function names allowed):

leftpad(InputString,**MyLength**,**MySign**)
call: leftpad,MyLength,MySign
e.g.: --key1=20:leftpad,"13","0"

rightpad(InputString,**MyLength**,**MySign**)
call: rightpad,MyLength,MySign
e.g.: -key=2:rightpad,"2","0"

trim(InputString)
call: trim
e.g.: --key2=17:trim+20:trim:leftpad,"13","0"

ltrim(InputString)
call: ltrim
e.g.: --key1=18,1,4:ltrim:rtrim:suffix," "+7:leftpad,"13","0"

rtrim(InputString)
call: rtrim
e.g.: --key1=20,2,5+19,1,6:rtrim

lower(InputString)
call: lower
e.g.: --key=fname=first_name:lower

upper(InputString)
call: upper
e.g.: --key=fname=last_name:upper

prefix(InputString,**MyPrefix**)
call: prefix,MyPrefix
e.g.: --key2=17:ltrim:rtrim:prefix,"supplier no "+20:prefix,"\\t"
Special case: "\\t" means \t literally. "\t" means a tab character (e.g. see next function)

suffix(InputString,**MySuffix**)
call: suffix,MySuffix
e.g.: --key=1:suffix,"\t"+2:suffix,"\t"+3

Tip: If you want to concatenate all fields with a tab ("\t") as default suffix to ensure an unambiguous key, use the ***option --tabcon*** *(--key=1+2+3 --tabcon is equivalent to the example)*

replace(InputString**,from** ***(*Global substitute regular expression*)*****,to** ***(*String*)*****)**

call: replace,from,to[,max_replacements]

e.g.: --key2=fname=GTIN:replace,".$","":leftpad,"12","0"

There is an optional parameter "max_replacements" (default 0=unlimited) to limit the number of replacements. To replace only the first occurrence use "1": --key2=0:replace,"^[^]`+ ","","1"

Grouping can be achieved with ***pairs of parentheses*** *in* ***from*** *and the corresponding* ***$number*** *in* ***to***

e.g. remove leading zeros in numbers (0089 => 89, 00 => 0): --key=1:replace,"^0`+([0-9])","$1"

match(InputString**,matchvalue** ***(*First matching regular expression*)*****)**

call: match,matchvalue[,grouping: 1 or 1+2 or 2+1 ...]

e.g.: --key2=fname="Article no":match,"^[0-9]`+"

There is an optional parameter "grouping" (default 0=no grouping) to cut groups in parentheses,

e.g. to swap the first 4 chars with the next 4: --key1=fname=col_2:match,"^(.{4})(.{4}),2`+1"

split(InputString**,MySign)**

call: split,MySign

e.g.: --key2=0,63,979:split,"|"

Explanation: If the key string to split is "101|102|103|104|105…", this string is split by "|" and the keys 101, 102, 103, 104, 105… are generated as if they were each entered line by line.

sort(PartKeys**)**

call: sort

e.g.: --key1=1+2:sort

Special cases: Conditions

They are used for the corresponding key:

```
  Syntax: field      Field No[,Substr Pos][,Substr Length]
          operator  :(~ or !~ or == or != or gt or st or bw)
          value     :<("regex" or "string")>
  Repeat [:AND or OR:Field No of next condition to check] ... ... ...
```

E.g. --condition2=0:~:"^0102030405"
This means: For key2 the --condition2 value has to be fulfilled:
If the whole line (0) doesn't start with 0102030405, key2 is skipped.

First parameter, **field**: Field No: 0 => whole line, 1 => field 1 etc.
The field number may be followed by a substring and a length value.
E.g. 1,2 or 1,2,10

Second parameter, **operator**: **~** means contains, **!~** does not contain, **==** equals to, **!=** does not equal to, **gt** greater than, **st** smaller than, **bw** between. The last three mentioned operators are numerical comparisons.

Third parameter, **value**: Regular expression or string in double quotes.
The regular expression or string applies to the field.
E.g. All non-empty entries in field code: --condition1=fname=code:!=:""
E.g. All values greater than 5 in field 1: --condition1=1:gt:"5"

The value for the "bw" operator is a comma separated list for min and max:
E.g. --condition1=1:bw:"5,10"
The logical operators AND or OR can be used for a condition, so conditions are similar to a database SQL WHERE clause:

AND example:
--condition1=fname=code_no:~:"^20[78]$":**AND**:fname=is_valid,2,1:==:"1"
This means: For key1 the -condition1 value has to be fulfilled:
Field code_no has to start and end with 207 or 208 and for field is_valid substring 2 with length 1 has to be equal to "1". If not, key1 is skipped.

OR example:
--condition1=fname=category_name:==:"Games":AND:fname=code_name:~:"Game Boy":**OR**:fname=code_name:~:"[bB][lL][uU]-[rR][aA][yY]"

Hints:
- A colon in the third parameter has to be escaped: --condition1=2:~:"`:"
- :AND: can be omitted (this old syntax prior to version 2.3 still works)
- If the --debug option is added, debug_values_file1_skipped.txt and debug_values_file2_skipped.txt show the lines skipped by the condition.
- Value regex metacharacters meant as string have to be escaped with "\".
- There are two special field variables **line_no1 and line_no2** which means the line number of file 1 or file 2, e.g. --condition1=line_no1:bw:"3,11"

Here is an advanced example of using the line_no1 condition in combination with the split and match functions:

```
listcompare testfile_split_a.txt testfile_split_b.txt --fs=\t --
key1=0:split,"\t":match,".{10}" --key2=1 --header1=0 --header2=1 --
condition1=line_no1:==:"1"
```

testfile_split_a.txt

	1	2	3
1	**0000000001barcode**	**0000000002product_name**	**0000000003price**
2	4013788000014	T-shirt	14,99
3	4013788001554	pullover	16,76
4	4013788000632	jacket	20,88

testfile_split_b.txt

	1	2
1	**tag_field**	**tag_name**
2	0000000001	barcode
3	0000000002	product_name
4	0000000003	price
5	0000000004	stock_balance
6	0000000005	record_date

The goal is to get the missing tag fields of file 2 (testfile_split_b.txt) that don't exist in the first line of file 1 (testfile_split_a.txt). To do this, the first line of testfile_split_a.txt is split by "\t" into the keys 0000000001barcode, 0000000002product_name, and 0000000003price and then the first 10 characters are extracted (=> 0000000001, 0000000002, 0000000003). These keys are compared with field 1 of testfile_split_b.txt.

Result:

only_in_file2_values.txt

	1	2
1	**tag_field**	**tag_name**
2	0000000004	stock_balance
3	0000000005	record_date

Another useful example of using the line_no1 and line_no2 conditions is comparing the header fields of two files if they are equal or contain different fields.

Header fields comparison tab ("\t") separator example (txt files):

```
listcompare file1.txt file2.txt --key=0:split,"\t" -fs=\t --header=0 --
condition1=line_no1:==:"1" --condition2=line_no2:==:"1"
```

Header fields comparison comma (",") separator example (csv files):

```
listcompare file1.csv file2.csv --key=0:split,"`," -fs=, --header=0 --
condition1=line_no1:==:"1" --condition2=line_no2:==:"1" --csv1 --csv2
```

Special cases: --tabcon

If you need a concatenated key from various fields (e.g. from the fields distributor_number and ean_upc), the option --tabcon is very convenient. All fields with "+" in the key are concatenated with a tab ("\t") (instead of an empty string) where the "+" exists to ensure an unambiguous key with this option.
Example:

```
listcompare testfilezo1.txt testfilezo2.txt --fs=\t --header=1 --
key=fname=distributor_number+fname=ean_upc --tabcon --difa
```

Hint: Instead of **--key=**fname=distributor_number+fname=ean_upc it is also possible to use the shortened form **--key=**fname=distributor_number+ean_upc

Note that when using fname and + to concatenate fields only field names can be used and no mixture of field names and field numbers. The same applies to the next option.

Special cases: --conuni

With the help of this option, it is possible to get keys from several fields. E.g. --conuni --key1=fname=term --key2=fname=term1+term2+term3

- ⇨ It can be compared if the keys from one field (key1: field name term) exist in any of the three fields of key2 (field names: term1, term2, term3).

e.g.

file1.csv:

```
id,term
1,clarinet
2,bassoon
3,saxophone
```

file2.csv:

```
id,term1,term2,term3
1,bassoon,saxophone,clarinet
```

```
listcompare file1.csv file2.csv --fs=, --csv1 --csv2 --header=1 --lower=1
--conuni --key1=fname=term --key2=fname=term1+term2+term3
```

- ⇨ **file1_and_file2.txt:**

```
bassoon
clarinet
saxophone
```

Note that the conuni option is used for keys with "+" in the field list and cannot be used in combination with the tabcon option since --tabcon combines keys (-> LOGIGAL AND) while --conuni union keys (-> LOGICAL OR).

Hint: Using --conuni with --value or --join or --joinleft works correctly, but at first glance the result lines might look a bit strange because keys for both files and keys only for one file may appear on the same result line. Further, keys only in file 1 can also appear in the file with keys only in file 2 and the other way round.

Special cases: --key=line_no

In this case line numbers of both files are keys for the comparison.

Special cases: --key=0

Use --key=0 to get the whole line as key. In this case "\t" is the recommended field separator for additional use of --value and --join.
listcompare testfileA.txt testfileB.txt **--fs=\t --key=0** --value --join

Special cases: Field separator

- Default: --fs=" " or no field separator: White space (spaces or tabs)
- Pipe: If you want to use "|" as field separator you cannot use --fs=|
-> Use --fs=^| or --fs="|" instead.
- Other: Tabulator: --fs=\t, **One space:** -> Use --fs="[ ]"

Special cases: UTF-8

This program works correctly with ANSI and UTF-8 encoded text files.
A BOM for a UTF-8 file is removed to ensure proper data processing.
Options -utf81 for file 1 and -utf82 for file 2:
These cases are rarely needed, only if you use substrings in keys and the substring contains multibyte characters with a UTF-8 encoded file, f.i. --fs=\t --key1=19,4,5 --key2=fname="Column B",2,5 --utf81 --utf82

Special cases: Large file support

This means: If the given megabytes limit N (default: --largefiles=300) is reached, the file is split into pieces of temporary files with maximal N megabytes and is later merged together.
This applies only if needed. So, there is no size limit for this program!

Special cases: Debug files

These files are for debug purposes. They can be interesting, e.g.
debug_key_file1.txt shows the whole concatenated sorted key of file1.
debug_file1_unique_keys.txt shows the whole line sorted by unique keys.
e.g. it is even possible to compare the same file using its name twice:
listcompare test.csv test.csv -key1=1+2:sort -key2=1+2:sort -fs=, --debug
debug_file1_unique_keys.txt shows unique results independent of the order of field 1 and field 2 by using the :sort function for concatenated keys.

debug_file1_or_file2.txt shows sorted union keys instead of intersection.
debug_file1_xor_file2.txt shows the sorted union only of the differences.

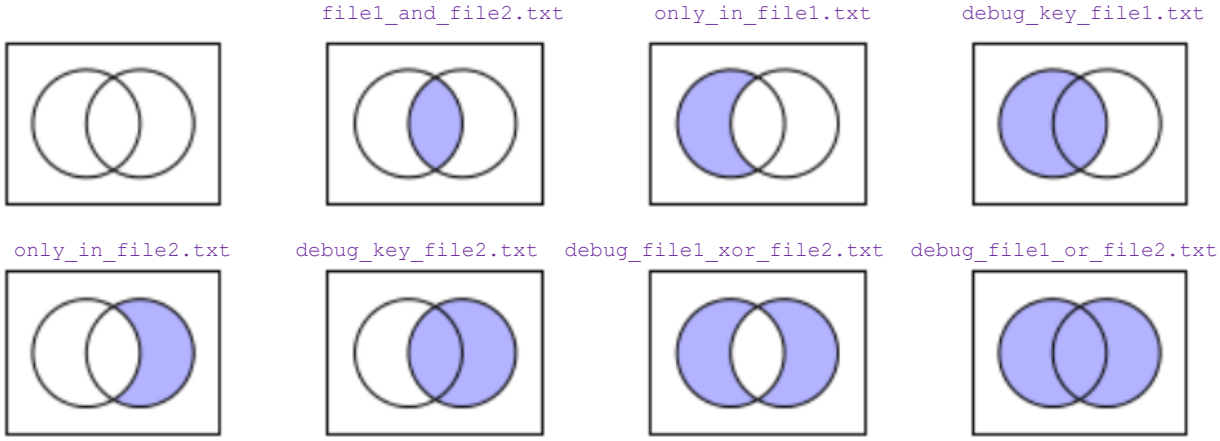
Special cases: headers fields

As a very special case, even **multiline headers** can be used by escaping \n (which means a newline) with a backtick `, e.g. --key=fname=001`**\n**barcode
This also applies to the Special cases: output options --value1, --value2, --join1, --join2 described below, e.g. --value2=fnames=001`**\n**barcode,cpno

As a further special case for keys, a regular expression for the file name can be used with the "~" operator after fname instead of "=". So, the multiline header field above could be written simpler as follows:
--key=fname~"^001"
Important: The **regex has to be enclosed in double quotes** and it must be ensured that the result is unique. The search is case insensitive except character classes ([...]) are used somewhere with the regular expression.

Short excursus on Set Operations

The graphic below shows all possible kinds of set operations (unique sorted keys): **Intersection** (file1_and_file2.txt), **difference file1 - file2** (only_in_file1.txt), **difference file2 - file1** (only_in_file2.txt), **Symmetric Difference** (debug_file1_xor_file2.txt), **Union** (debug_file1_or_file2.txt).



Special cases: Output options

The default output for value files and join files is the whole line:
--value1=0, --value2=0, --join1=0, --join2=0
However, with an optional field list only selected fields can be printed.
Either field numbers or field names after "fnames=" can be used.
Examples:
--value1=1 or with field names **--value1=fnames=id**
--value1=5,6 or with field names **--value1=fnames=state,zip**
--join2=4,1 or with field names **--join2=fnames="Column D","Column A"**

Output files

These output files will be created and deleted with program start:
Note that any existing files will be overwritten, without warning.

```
Key files (if results exist):
     file1_and_file2.txt, only_in_file1.txt, only_in_file2.txt
     key_file1_duplicates.txt, key_file2_duplicates.txt
Value files (if requested and if results exist):
     file1_and_file2_values_file1.txt, file1_and_file2_values_file2.txt
     only_in_file1_values.txt, only_in_file2_values.txt
Join files (if requested and if results exist):
     joined_key_both_in_file1_and_file2_values.txt
     joined_key_file1_leftjoin_values.txt
Difference files (if requested and if results exist):
     differences_values.txt, differences_fields.txt
     differences_file1.txt, differences_file2.txt
Debug files (if requested and if results exist):
     debug_key_file1.txt, debug_key_file2.txt
     debug_file1_or_file2.txt, debug_file1_xor_file2.txt
     debug_values_file1_skipped.txt, debug_values_file2_skipped.txt
     debug_file1_unique_keys.txt, debug_file2_unique_keys.txt
```

Hints: These output files are created in the current directory (the directory from where listcompare is called). There should be only one call of listcompare in the same directory at the same time.

Also, for very big files temporary files are created in the Windows TEMP folder so do not delete the %TEMP% folder while this program is running.

*For value files and join files the file extensions of file1 and file2 remain unchanged. E.g. file1 name: data1.**csv***
- ⇨ *file1_and_file2_values_file1.**csv** etc. (see above)*
- ⇨ *joined_key_both_in_file1_and_file2_values.**csv** etc. (see above)*

*If no file extension is given, the default extension "**txt**" is used.*
*All files (e.g. only_in_file1_values.*****) are deleted with program start.*

*If **only** **file1_and_file2.txt** exists and **not** **only_in_file1.txt** **or** **only_in_file2.txt**, **errorlevel 0** is returned, else **errorlevel 1**.*
After calling listcompare you can check: if errorlevel 1 echo NOT EQUAL

The items of these three key files can be counted with the option
--count (or simply -c) => Count the result files
e.g. file1_and_file2.txt (86 items #Identical)
If duplicate keys exist, also the duplicate key items are counted.
Also, the records of all other result files (except debug files) are counted: record count of value files and join files (excluding header) and for difference files the number of different (modified) records.

```
Example:
*** Statistics ***
file1 (ARGV[1]: 4 rows) and file2 (ARGV[2]: 4 rows) successfully processed.
-> Result files:
Key files:
  file1_and_file2.txt (2 items #Identical)
  only_in_file1.txt (1 items #Deletions)
  only_in_file2.txt (1 items #Additions)
Value files:
  file1_and_file2_values_file1.csv (2 records)
  file1_and_file2_values_file2.csv (2 records)
  only_in_file1_values.csv (1 records)
  only_in_file2_values.csv (1 records)
Difference files:
  differences_fields.txt (1 records #Modifications)
```

Explanation:
"**rows**" means processed lines **including** header. If there is a multiline CSV row, this multiline row is counted as one row.

Note that the "rows" (e.g. file1 (ARGV[1]: 4 rows) are always printed for the two input files, even without the count command. Normally, the complete input file is processed except for some line number restriction conditions, e.g. --condition1=line_no1:==:"1"

"**records**" means processed lines **excluding** header. If there is a multiline CSV row, this multiline row is counted as one row.

#Deletions, **#Additions** and **#Modifications** are terms related to the first file as base file.

Further, the content of the result files can be printed to STDOUT with:
--print=1 (or --print or simply -p) prints all files (except debug files), **--print=2** prints only diffs files (only_in_file1.txt, only_in_file2.txt, differences_values.txt, differences_fields.txt).

*Special cases: **--print=file1** prints file1 and exits and **--print=file2** prints file2 and exits. The print option should only be used for small files and correct display is only guaranteed with ASCII characters. If a file is bigger than 50 MB, only the 15 first lines are printed.*

listcompareGI: A graphical user interface for listcompare

The command line interface (CLI) on MS Windows can easily be accessed. To access the command prompt using the Run window press the Win + R keys on your keyboard. Then, type **cmd** and press Enter or click/tap OK.

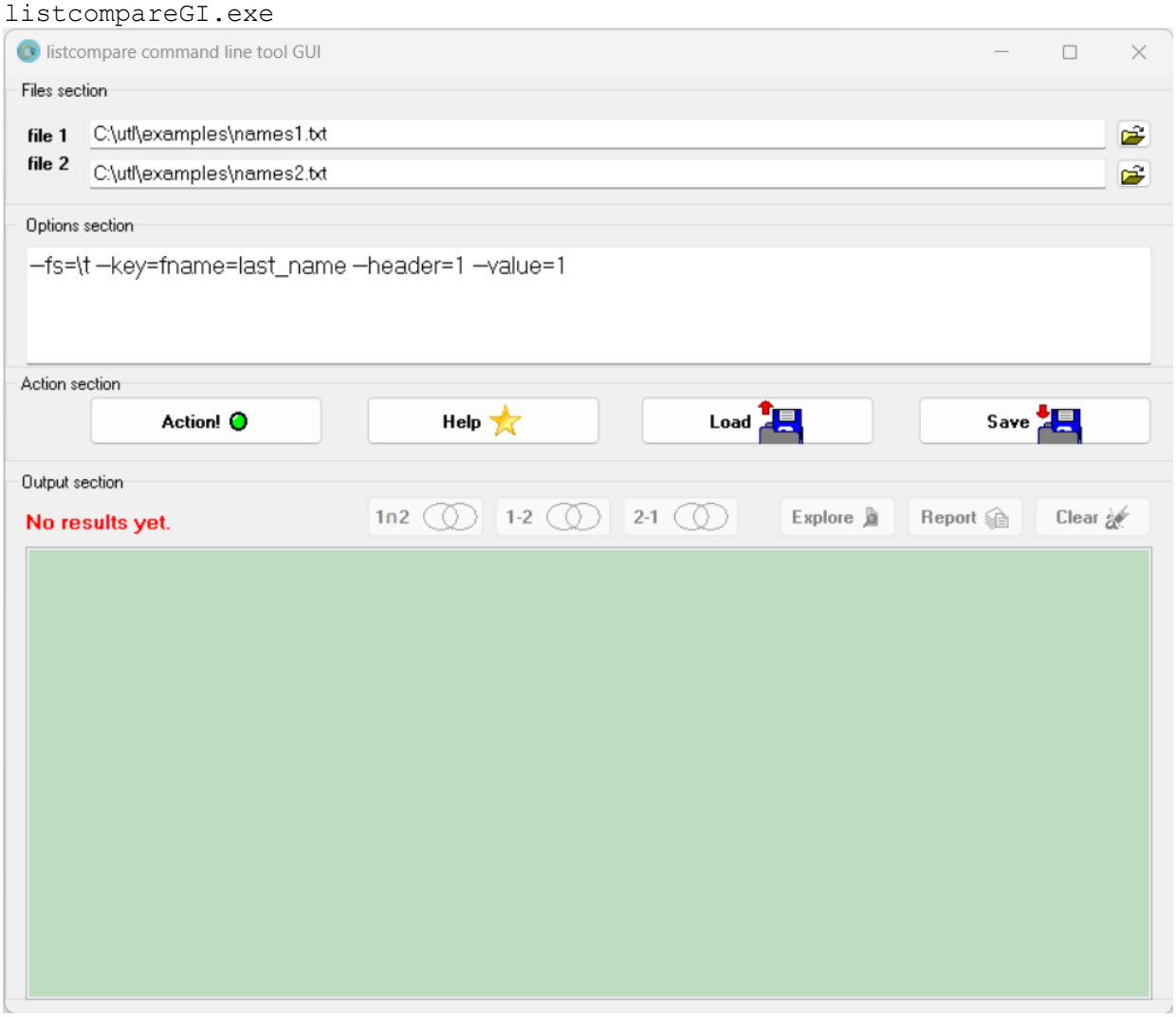
However, many people prefer a graphical user interface (GUI). This is the reason why **listcompareGI.exe** was created: It is a GUI wrapper for calling listcompare.exe without the need to use the command line. Note that the command line is still the preferred way to use listcompare, though. For very big files the GUI looks like it was frozen, but it works correctly.

To use listcompareGI.exe, extract the zip file listcompare.zip with the original directory structure (including the Examples directory) to a path from where you wish to use listcompareGI.exe, e.g. the directory c:\utl\.

c:\utl\listcompareGI.exe
c:\utl\listcompare.exe
c:\utl\listcompare_readme.pdf
c:\utl\Examples\...

Then start listcompareGI.exe. If you get a message "Windows protected your PC – Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk", don't worry. Click on "More info" and then "Run anyway".

listcompareGI.exe



The program starts with a ready-to-use example (example 2 of this manual). There is a Files section (preselected with the files "names1.txt" and "names2.txt" of the examples subdirectory), an Options section, an Action section, and an output section.

Files section

file 1
file 2

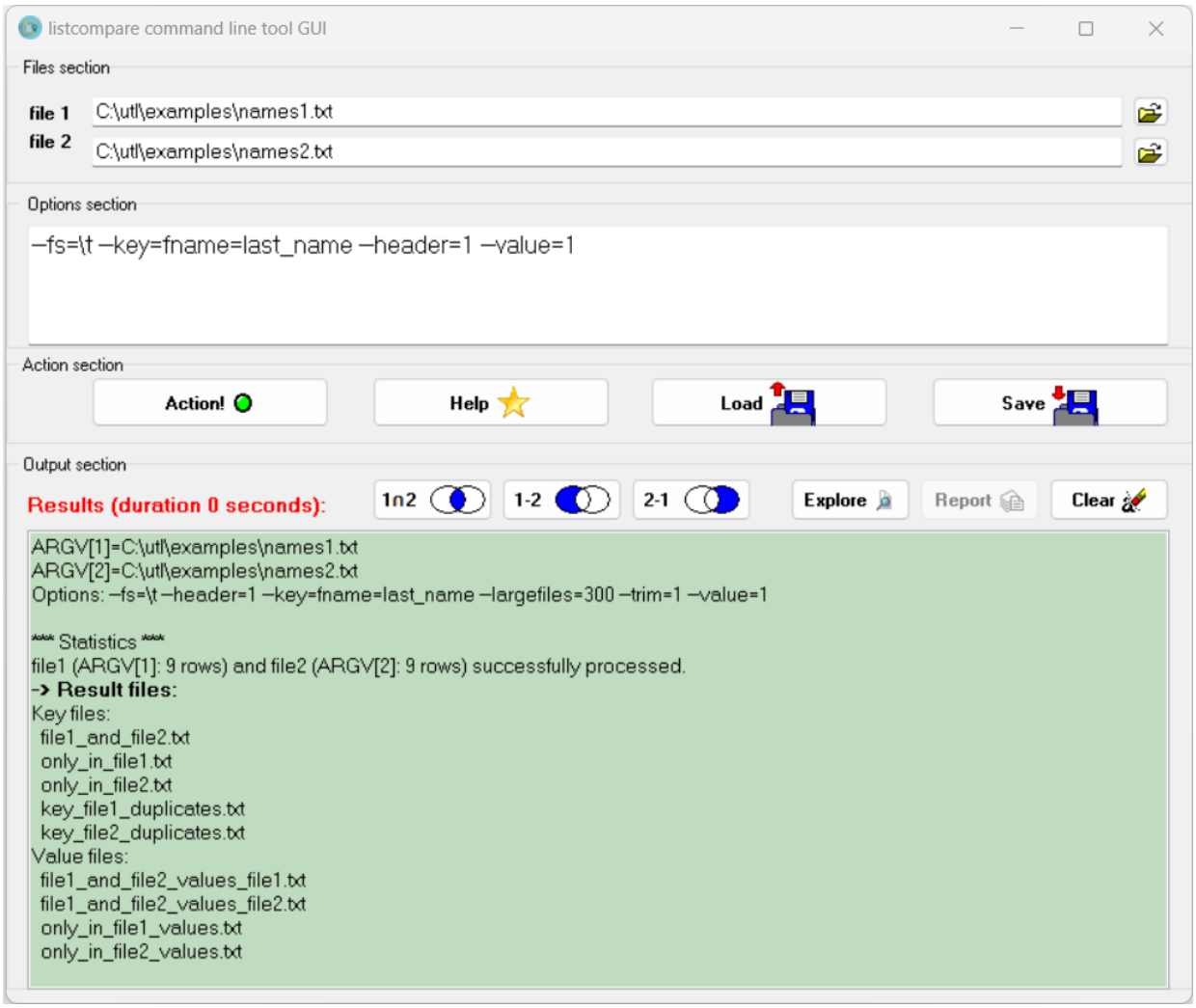
The files can be entered directly into the field or chosen with the button on the right or entered with drag and drop from the Explorer. Please prefer ASCII file names. ASCII names work, ANSI may or may not work. The same applies to the Options section.

Options section

All options described above in this manual can be entered here, e.g. --fs=\t --key=fname=last_name --header=1 --value=1

Action section

When you click on **Action!** (or use the shortcut **F9**), listcompare is started and the program looks like the following picture.



Clicking on **Help** opens this manual.

Clicking on **Load** loads a batch file (file name with ending .bat or .cmd, e.g. c:\utl\Examples\example2.cmd) into this program (using drag & drop from the Explorer is possible, too). The Files section and the Options section get updated with the values from the batch file and the program can be started for these values with the Action! button.

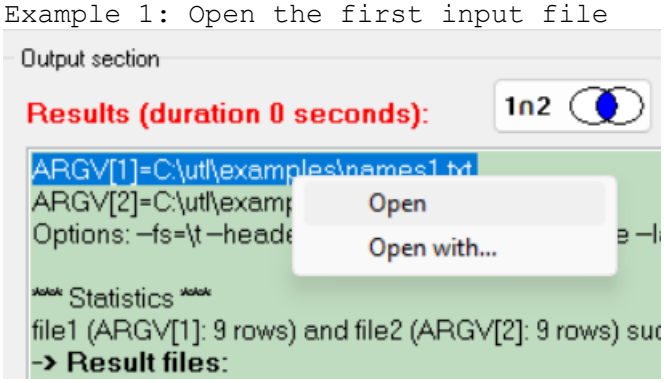
Clicking on **Save** saves the values from the Files section and the Options sections into a newly created batch file. The directory and name of the batch file can be chosen. Later, this batch file can be loaded again into this program with the Load button.

Output section

The output section shows the results text in **green**. If something went wrong ("ERROR"), the text is shown in **red**. If there is a "WARNING", the text gets turned into **yellow**. The result files directory can be accessed in the Explorer with the **Explore** button. The listcompareGI result files are located in the listcompareGI.exe directory, in this example c:\utl\. The file differences_fields.txt can be opened with the **Report** button and the results window text can be cleared with the Clear button.
CSV Report example: Please load c:\utl\Examples\example9.cmd.

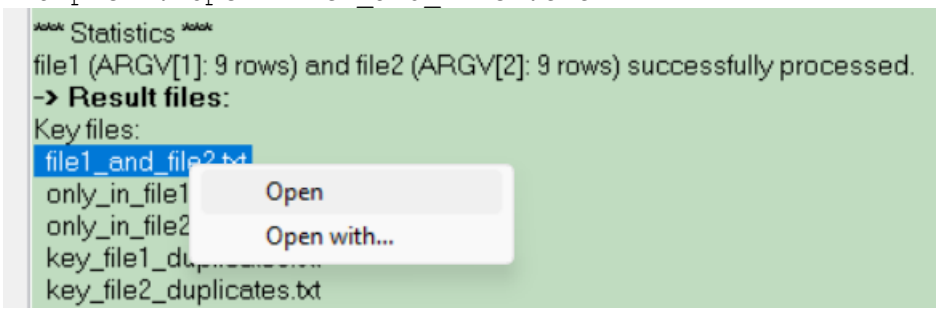
Any file in the output section can be opened by a triple-click in the results window text which selects the entire line. Then, with a context menu click on "Open" the file can be opened with its default application. A click on "Open with..." opens the MS Windows File Open with dialog box.

Example 1: Open the first input file



So, the first input file can be selected by a triple-click in the first line and then be opened with a context menu click on Open or Open with….

Example 2: Open file1_and_file2.txt



Note that the three Key files (file1_and_file2.txt, only_in_file1.txt, only_in_file2.txt) can also be opened with the corresponding buttons:

Version history

```
Version 3.5.2.8 20240401
- New option print added for printing the result files to STDOUT.
- For keys with a field name a regular expression can now be used.
  e.g. --key=fname~"^012345" matches the field name 012345tomatoes.

Version 3.5.1.8 20231126
- New option count added for counting the result files.
  e.g. file1_and_file2.txt (86 items #Identical)
- New option version added.

Version 3.5.0.7 20230312
- New option conuni added enabling getting keys from several fields.
- Conditions: New operator "bw" and scopes line_no1, line_no2 added.
- difa_info1 and difa_info2 special scopes line and all added.
- difi special scope keys added.
- function split added.

Version 3.4.1.4 20220702
- New option in combination with the difa option added: --difa_level
- File debug_file1_xor_file2.txt added to enable all set operations.

Version 3.4.0.5 20220205
- New output options for value files and join files added:
  value1, value2, join1, join2
- It is now possible to use joins with different field separators.

Version 3.3.0.6 20211117
- New options in combination with the difa and join options added:
  difa_fun, difa_ci, difa_tr, difa_ofd, join_ofd

Version 3.2.0.3 20210828
- Merging engine for big files improved by largefiles option changes:
  --largefiles=N: Use N megabytes as maximal size for temporary files
  (default N=300). (The old option used N lines (default N=1000000)).

Version 3.1.0.3 20210621
- New options in combination with the difa option added:
  difa_info1, difa_info2, difa_seq

Version 3.0.0.2 20210209
- difa option added. The new generation 3 of this software is able to
  compare CSV files on field level as a full featured comparison tool.

Version 2.3.0.0 20200609
- conditions completely reworked:
  Logical operators AND and OR introduced.
  Conditions are now similar to a SQL WHERE clause.

Version 2.2.0.0 20200113
- joinhint option added.
- function match added.
- tabcon option added.

Version 2.1.0.3 20190614
- joinleft option added. Since version 2.1 of this software it is
  possible to generate left join files exactly like a database
  left join on a key for two tables.
```

```
Version 2.0.1.9 20181018
- Arguments limits (no use of the special signs [:,+]) overcome:
  Backtick (`) can now be used to escape a plus sign, comma or colon:
  This applies to functions, field names (fname), conditions (colon).

Version 2.0.0.1 20180121
- join option added. The new generation 2 of this software is able to
  behave exactly like a database inner join on a key for two tables.
- function replace added.

Version 1.0.1.3 20171004
- function rightpad added.
- function arguments check improved.
- Diffs files differences_file1.txt and differences_file2.txt added.

Version 1.0.0.0 20170608
- Initial version.
```