NAME

join — combine database tables by common field

SYNOPSIS

join [**-izH**] {[**-1** *field*] [**-2** *field*]}|[**-j** *field*]
[**-o** {**0**|*table.field*[**,0**|*table.field*]...}|**auto**] [**-e** *replacement*]
[**-t** *delimiter*] [**-**{**a**|**v**} **1**|**2**]... [**--**[**no**]**check-order**] *table1 table2*

DESCRIPTION

This is the relational inner join operation (⋈): given database records from both tables — ($field_{1,1}, field_{1,2}, field_{1,2}, \cdots$) and ($field_{2,1}, field_{2,2}, field_{2,2}, \cdots$) — finds rows where $field_{1,n_1}$ matches $field_{2,n_2}$ where n_1 and n_2 are the "join fields" (**-12**, default **1**) and produces combined records in the form ($field_{1,n_1}, field_{1,2}, field_{1,3}, \cdots, field_{2,2}, field_{2,3}, \cdots$) (but cf. **-o**). If more than one match is found for a given key, the result is the cross-product of matches from both sets.

-a facilitates the left/right/full outer join (⟕/⟖/⟗) by including unmatched rows from *table1*/*table2*/both.
-v facilitates the antijoin (▷) by *only* including unmatched rows and removing any matched ones.

The tables are read from files *table*[*12*] (standard input stream if "**-**"), rows are newline-delimited, and fields are whitespace-separated (but cf. **-t**). The rows must be sorted (according the current locale's collation sequence) by the join field (this can be achieved with **sort -bk** *field* (**sort -t** *delimiter* **-k** *field*)). Thus, for example:

```
$ cat bonds1
#Code   ISIN           value
AOW1225 PLO145900059   1000.00
INF0326 PLO234100074   750.00
INY0924 PLO234100041   700.00
PRF0125 PLGFPRE00297   100.00
$ cat bonds2
#Code   Maturity       Reference      Margin
AOW1225 2025-12-14     WIBOR 3M       6
INF0326 2026-03-16     WIBOR 3M       5.6
INY0924 2024-09-03     WIBOR 3M       5.25
PRF0125 2025-01-27     WIBOR 3M       6
$ cat rates
#Date          2024-04-12
WIBOR 3M       5.86
WIBOR 6M       5.87
WIBOR SW       5.82
$ join -t '       ' bonds?
#Code   ISIN           value   Maturity       Reference      Margin
AOW1225 PLO145900059   1000.00 2025-12-14     WIBOR 3M       6
INF0326 PLO234100074   750.00  2026-03-16     WIBOR 3M       5.6
INY0924 PLO234100041   700.00  2024-09-03     WIBOR 3M       5.25
PRF0125 PLGFPRE00297   100.00  2025-01-27     WIBOR 3M       6
$ sort -t '       ' -k 3 bonds2 | join -t '       ' -1 3 - rates
WIBOR 3M       INY0924 2024-09-03     5.25    5.86
WIBOR 3M       INF0326 2026-03-16     5.6     5.86
WIBOR 3M       AOW1225 2025-12-14     6       5.86
WIBOR 3M       PRF0125 2025-01-27     6       5.86
```

OPTIONS

-i, **--ignore-case** Compare the join fields case-insensitively. By default, input lines in each file must be sorted by the join field according to the current locale's collation sequence (with a fall-back to byte-wise comparison on locales with an `@`; cf. **HISTORY**, **Standards**), and the output is ordered with respect to the same

ordering.

-z, **--zero-terminated**
: Line separator is NUL, not newline.

-H, **--header**
: Don't try to join the first line, just output its fields in the right order. This is most useful outside the **C** (**POSIX**) locale (the samples above are actually unsorted in **en_GB**, since '**A**' < '**C**' < '**#C**' < '**D**' < '**I**').

-1 *field*
: Join *table1* by 1-indexed *field* (= n_1). Default: **1**.

-2 *field*
: Join *table2* by 1-indexed *field* (= n_2). Default: **1**.

-j *field*
: **-1** *field* **-2** *field*

-o **0**|*table.field*[**,0**|*table.field*]...
: By default, each output row takes the form $(field_{1,n_1})\cup((field_{1,1},\cdots)-field_{1,n_1})\cup((field_{2,1},\cdots)-field_{2,n_2})$ (that is, the key (as seen in *table1*), then all the non-key fields of *table1* and *table2*).

  -o instead dictates a strict output format, taking a comma- or whitespace-separated list of either "**0**" ($field_{1,n_1}$) or "*table.field*" ($field_{table,n_{field}}$), *table* naturally **1** or **2**; *field* 1-indexed.

-o auto
: Output rows consist of the key, then the non-key fields of *table1* and *table2* that are present in the first physical line (incl. the header if **-H**); this is equivalent to the default output format if both *table*s were trimmed to only have as many fields as on their first row.

-e *replacement*
: If a field is set to be output by **-o** but is not present in the *table*s, *replacement* is written instead. Defaults to the empty string.

-t *delimiter*
: Tokenise each row by simply splitting it on *delimiter* *and* separate fields by *delimiter* when writing the result. If '**\0**', use NUL (**0x0**).
  By default, fields are separated by any runs of whitespace (and indentation stripped), and the output is separated by a single space (**0x20**).

-a 1|2
: If no match is found for a row in the specified table, write it anyway. Both may be given. Excludes **-v**.

-v 1|2
: **-a 1|2** but *don't* write any rows that *do* match. Excludes **-a**.

--check-order
: Proactively verify the *table*s are sorted correctly. See **STANDARDS**.

--nocheck-order
: Ignore out-of-order rows entirely. See **STANDARDS**.

EXIT STATUS

1 if either *table* is detectably unsorted (unless **--nocheck-order**) or couldn't be opened, or if both *table*s are "**-**".

EXAMPLES

The second **join** above could more usefully be

```
$ sort -t '       ' -k 3 bonds2 |
  join -t '       ' -1 3 -H -o '1.1 1.2 2.2 1.4' - rates
#Code   Maturity       2024-04-12     Margin
INY0924 2024-09-03     5.86           5.25
INF0326 2026-03-16     5.86           5.6
AOW1225 2025-12-14     5.86           6
PRF0125 2025-01-27     5.86           6
```

Querying

```
$ echo AOW1225 | join - bonds1
AOW1225 PLO145900059 1000.00
$ echo PLO145900059 | join -2 2 - bonds1
PLO145900059 AOW1225 1000.00
join: bonds1: PLO234100074 > PLO234100041 (INF0326   PLO234100074   750.00 > INY092
join: input unsorted
```

```
$ echo PLO145900059 | join -2 2 - <(sort -b -k 2 bonds1)
PLO145900059 AOW1225 1000.00
```

-o auto

```
$ head auto?
==> auto1 <==
1 a
2 b c
3 d e f

==> auto2 <==
1 A A2
2 B C
3 D E F

$ join -o auto auto?
1 a A A2
2 b B C
3 d D E
```

SEE ALSO

comm(1), paste(1), sort(1), strcoll(3)

STANDARDS

Conforms to IEEE Std 1003.1-2024 (“POSIX.1”) — only **-12oetav** are standard: **-j** conforms to IEEE Std 1003.2 (“POSIX.2”); **-iz**, **--header**, **--**[**no**]**check-order**, and **-o auto** are extensions, originating from the GNU system; **-H** is an extension. Accepting any string (rather than one character exactly) in **-t** is an extension. "**-t \0**" is compatible with the GNU system. The GNU system refuses multiple **-12j** if they’re different; this implementation chooses which-ever one was given last.

IEEE Std 1003.1-2024 (“POSIX.1”) requires that the input be sorted, else behaviour is unspecified. This implementation allows any order, so long as it is *the same* order in both *table*s and *all* lines can be paired. Without **--check-order**, an error is produced if a row in one of the *table*s cannot be matched to the other *and* the next line sorts later.

HISTORY

Appeared, mostly-fully-formed, in Version 7 AT&T UNIX, as join(1) – "relational database operator":

join [options] file1 file2

with a usage string of

```
usage: join [-j1 x -j2 y] [-o list] file1 file2
```

but with a realistic **USAGE** of

join [**-j**[**1**|**2**] *field*] [**-o** [{**1**|**2**}.*field*]...] [**-e** *replacement*] [**-t***delimiter*] [**-a**[**1**|**2**]] *file1 file2*

(with plain **-j** (or any non-**1**|**2**) being equivalent to **-j1** *field* **-j2** *field*; this is undocumented) (with **-o** taking multiple arguments) (with plain **-t** making *delimiter* NUL; this is undocumented) (with plain **-a** (or any non-**1**|**2**) being equivalent to **-a 1 -a 2**; this is undocumented) (with only *file1* being allowed to be "**-**"; *file2* being a pipe yields an infinite loop), Lines may be up to BUFSIZ (**512**) bytes and up to 20 fields; this is undocumented. Without **-t**, an empty line in either input file is treated as EOF. The order is the byte order.

AT&T System III UNIX breaks **-t***nothing* (NUL).

X/Open Portability Guide Issue 2 (“XPG2”) carries a formalised-but-equivalent-to-Version 7 AT&T UNIX **join**.

X/Open Portability Guide Issue 3 (“XPG3”) invents the modern collation behaviour, shaded IN ("Internationalised functionality", defined as optional).

IEEE Std 1003.2 (“POSIX.2”), invents the present-day usage (**-v**, **-12**, no spacing requirements, &c.) to conform to the modern Utility Syntax Guidelines, shading **-j1** *field*, **-j2** *field*, *and* **-j** *field* OB(solete); notably, this is the first time **-j** is actually documented. **-o** is also a present-day-like single string (OB-shaded text allows multiple arguments); **0** is new. Both input files must be text files (LINE_MAX, no NULs), and either may be "**-**", as present-day.

4.4BSD invents a new **join** strictly to the letter of IEEE Std 1003.2 (“POSIX.2”) (with the obsolete bits) (though with dynamically-sized lines), which means that **-jQ** *field* is no longer equivalent to **-j** *field*.

IEEE Std 1003.1-2001 (“POSIX.1”) removes **-j1**, **-j2**, **-j**, and multi-argument **-o**.