How to merge every two lines into one from the command line?

Ask Question



I have a text file with the following format. The first line is the "KEY" and the second line is the "VALUE".

119



```
KEY 4048:1736 string
3
KEY 0:1772 string
1
KEY 4192:1349 string
1
KEY 7329:2407 string
2
KEY 0:1774 string
```

I need the value in the same line as of the key. So the output should look like this...

```
KEY 4048:1736 string 3
KEY 0:1772 string 1
KEY 4192:1349 string 1
KEY 7329:2407 string 2
KEY 0:1774 string 1
```

It will be better if I could use some delimiter like \$ or , :

```
KEY 4048:1736 string , 3
```

How do I merge two lines into one?

edited Nov 15 '17 at 0:31



Benjamin W.

22.5k 13 56 59

asked Mar 7 '12 at 16:19



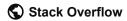
shantanuo

12k 59 160 267

There is a lot of way for doing this! I've done a <u>little bench with pr , paste , awk , xargs , sed and pure bash</u>! (xargs is the slower, slower than <u>bash</u>!) – F. Hauri Nov 6 '18 at 15:18

Home

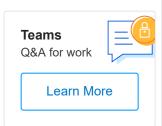
PUBLIC



Tags

Users

Jobs



20 Answers



awk:

143

awk 'NR%2{printf "%s ",\$0;next;}1' yourFile



note, there is an empty line at the end of output.



sed:

sed 'N;s/\n/ /' yourFile

edited Jul 4 '16 at 8:17

answered Mar 7 '12 at 16:39



Ken

48k 28 164 22

Tested on Ubuntu 13.04 – Leo Gallucci Dec 9 '13 at 22:54

- @elgalu: Because ANSI colors are just a bunch of escape character combinations. Do a hexedit on such an output, to see what you have. – not2qubit Feb 5 '14 at 19:02
- 7 This awk solution can break if printf expansion strings like %s are found within \$0. That failure can be avoided like this: 'NR%2{printf "%s ",\$0;next;}1' - ghoti Mar 11 '14 at 13:21
- 9 Because it's really hard to google, what does the 1 after the closing brace mean? erikbwork Jul 3 '15 at 13:19
- 6 @erikb85 Here you go stackoverflow.com/questions/24643240/... – Viraj Oct 4 '15 at 20:34



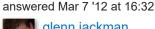
paste is good for this job:

204

paste -d " " - - < filename



+200



glenn jackman 173k 26 155 249

- 7 I think this is the best solution presented, despite using neither sed nor awk. On input that is an odd number of lines, Kent's awk solution skips the final newline, his sed solution skips the final line in its entirty, and my solution repeats the last line. paste, on the other hand, behaves perfectly. +1. – ghoti Mar 11 '14 at 13:42 /*
- I often use cut but always forget about paste. It rocks for this problem. I needed to combine all lines from stdin and did it easily with paste -sd ' ' - . - Clint Pachl Aug 8 '14 at 6:52

read from stdin, you can stack as many of them as you want I expect. – ThorSummoner Dec 8 '16 at 0:12 /

1 Yes, @ThorSummoner ... I had to paste every three lines into a single line and did paste - - - and it worked perfectly. – Daniel Goldfarb Jan 31 '17 at 23:02



Alternative to sed, awk, grep:

28

xargs -n2 -d'\n'



This is best when you want to join N lines and you only need space delimited output.

My original answer was xargs -n2 which separates on words rather than lines. -d can be used to split the input by any single character.

edited Oct 19 '16 at 10:54

answered Oct 27 '15 at 16:15



nnog

927 11 1

3 This is a nice method, but it works on words, not lines. To make it work on lines, could add -d '\n' - Don Hatch Oct 14 '16 at 8:34

Good point, Don. I've edited the answer. Thanks! – nnog Oct 19 '16 at 10:55 ✓

1 Wow, I'm a regular xargs user but didn't know this. Great tip.
– Sridhar-Sarnobat May 7 '17 at 21:47



There are more ways to kill a dog than hanging. [1]

24

awk '{key=\$0; getline; print key ", " \$0;}'



Put whatever delimiter you like inside the quotes.

References:

1. Originally "Plenty of ways to skin the cat", reverted to an older, potentially originating expression that also has nothing to do with pets.

edited Apr 13 '17 at 12:38

Community

1 1

answered Mar 7 '12 at 17:36



ghoti

36.1k 7 43 85

I love this solution. - luis.espinal May 23 '13 at 15:01

this is easier to understand thx! – Aquarius Power Oct 14 '14 at 2:43

- 4 As a cat owner I do not appreciate this kind of humor. witkacy26 Oct 28 '15 at 11:03
- 4 @witkacy26, Adjusted expression per your concern. ghoti Nov 15 '15 at 23:55

I love this awk solution but I don't understand how it works :S – Rubendob Nov 3 '17 at 11:15



Here is my solution in bash:

10

while read line1; do read line2; echo "\$line1, \$line2"; do



answered Mar 7 '12 at 18:21



Hai Vu **24k** 6 42 75



Although it seems the previous solutions would work, if a single anomaly occurs in the document the output would go to pieces. Below is a bit safer.



sed -n '/KEY/{
N
s/\n/ /p
}' somefile.txt

answered Mar 8 '12 at 0:16



7.**0**. 7**36** 6 13

Why is it safer? What does /KEY/ do? What does the p do at the end? – Stewart May 31 '16 at 11:16 ✓



Here is another way with awk:

10

awk 'ORS=NR%2?FS:RS' file



```
3
KEY 0:1772 string
1
KEY 4192:1349 string
1
KEY 7329:2407 string
2
KEY 0:1774 string
1

$ awk 'ORS=NR%2?FS:RS' file
KEY 4048:1736 string 3
KEY 0:1772 string 1
KEY 4192:1349 string 1
KEY 7329:2407 string 2
KEY 0:1774 string 1
```

As indicated by <u>Ed Morton</u> in the comments, it is better to add braces for safety and parens for portability.

```
awk '{ ORS = (NR%2 ? FS : RS) } 1' file
```

ORS stands for Output Record Separator. What we are doing here is testing a condition using the NR which stores the line number. If the modulo of NR is a true value (>0) then we set the Output Field Separator to the value of FS (Field Separator) which by default is space, else we assign the value of RS (Record Separator) which is newline.

If you wish to add , as the separator then use the following:

```
awk '{ ORS = (NR%2 ? "," : RS) } 1' file

edited May 23 '17 at 12:10

Community •
```

59.3k 15 86 124



jaypal singh

Definitely the right approach so +1 but I wonder what the condition is that's being evaluated to invoke the default action of printing the record. Is it that the assignment succeeded? Is it simply ORS and that's being treated as true since ORS gets a value thats not zero or a null string and awks guessing correctly that it should be a sting instead of numeric comparison? Is it something else? I'm really not sure and so I'd have written it as awk '{ORS=(NR%2?FS:RS)}1' file. I parenthesized the ternary expression to ensure portability too. — Ed Morton Aug 21 '14 at 17:27

- 1 @EdMorton Yeah, I just saw couple of upvotes on this answer was about to update it to include the braces for safety. Will add parens as well. – jaypal singh Aug 21 '14 at 17:29
- 1 @EdMorton Good point. Added some explanation too. :) jaypal singh Aug 21 '14 at 17:34 ✓



"ex" is a scriptable line editor that is in the same family as sed, awk, grep, etc. I think it might be what you are looking for. Many modern vi clone/successors also have a vi mode.



ex -c "%g/KEY/j" -c "wq" data.txt

This says for each line, if it matches "KEY" perform a **j** oin of the following line. After that command completes (against all lines), issue a **w** rite and **q** uit.

answered Mar 27 '14 at 20:57



167 3 1



If Perl is an option, you can try:

4

perl -0pe 's/(.*)\n(.*)\n/\$1 \$2\n/g' file.txt



answered Mar 7 '12 at 16:25



andrefs

181 9

Does the -0 tell perl to set the record separator (\$/) to null, so that we can span multiple lines in our matching pattern. The manpages are a bit too technical for me to figure out what it means in practice. — Sridhar-Sarnobat May 7 '17 at 21:52



You can use awk like this to combine ever 2 pair of lines:

4

awk '{ if (NR%2 != 0) line=\$0; else {printf("%s %s\n", line END {if (length(line)) print line;}' flle



edited Mar 7 '12 at 17:55

answered Mar 7 '12 at 16:24



anubhava

540k 48 340 417



You can also use the following vi command:



answered Aug 26 '14 at 8:57



Or even :%g//j since all you need is a match for the *join* to be executed, and a null string is still a valid regex. – ghoti Sep 18 '14 at 13:35

- 1 @ghoti, In Vim, when using just //, the previous search pattern will be used instead. If there is no previous pattern, Vim simply reports an error and do nothing. Jdamian's solution works all the time. Tzunghsing David Wong Sep 20 '16 at 16:54
- 1 @TzunghsingDavidWong that's a good pointer for vim users. Handily for me, neither the question nor this answer mentioned vim. – ghoti Sep 20 '16 at 17:44



A slight variation on <u>glenn jackman's answer</u> using paste: if the value for the -d delimiter option contains more than one character, paste cycles through the characters one by one, and combined with the -s options keeps doing that while processing the same input file.



This means that we can use whatever we want to have as the separator plus the escape sequence \n to merge two lines at a time.

Using a comma:

```
$ paste -s -d ',\n' infile
KEY 4048:1736 string,3
KEY 0:1772 string,1
KEY 4192:1349 string,1
KEY 7329:2407 string,2
KEY 0:1774 string,1
```

```
$ paste -s -d '$\n' infile
KEY 4048:1736 string$3
KEY 0:1772 string$1
KEY 4192:1349 string$1
KEY 7329:2407 string$2
KEY 0:1774 string$1
```

What this *cannot* do is use a separator consisting of multiple characters.

As a bonus, if the paste is POSIX compliant, this won't modify the newline of the last line in the file, so for an input file with an odd number of lines like

```
KEY 4048:1736 string
3
KEY 0:1772 string
```

paste won't tack on the separation character on the last line:

```
$ paste -s -d ',\n' infile
KEY 4048:1736 string,3
KEY 0:1772 string
```

answered Nov 15 '17 at 0:39





nawk '\$0 ~ /string\$/ {printf "%s ",\$0; getline; printf "%s'



This reads as

```
getline ## get the next line
printf "%s\n" ## print the whole line and carriage reture
```

edited Oct 31 '13 at 22:27



answered Oct 31 '13 at 22:06





Another solutions using vim (just for reference).

1

Solution 1:



Open file in vim vim filename, then execute command:% normal Jj

This command is quit easy to understand:

- % : for all the lines,
- normal: execute normal command
- Jj : execute Join command, then jump to below line

After that, save the file and exit with :wq

Solution 2:

Execute the command in shell, vim -c ":% normal Jj" filename, then save the file and exit with :wq.

answered Jan 8 '16 at 2:08



```
remapped. +1 for vim solution. – qeatzy Jul 21 '17 at 11:57
```

@qeatzy Thank you for teaching me that. Very glad to know it. ^_^ – Jensen Aug 16 '17 at 12:41



In the case where I needed to combine two lines (for easier processing), but allow the data past the specific, I found this to be useful



data.txt

```
string1=x
string2=y
string3
string4

cat data.txt | nawk '$0 ~ /string1=/ { printf "%s ", $0; go getline } { print }' > converted_data.txt
```

output then looks like:

converted_data.txt

```
string1=x string2=y
string3
string4
```

edited Apr 21 at 0:53



answered Oct 22 '14 at 16:27





Simplest way is here:



- 1. Remove even lines and write it in some temp file 1.
- 2. Remove odd lines and write it in some temp file 2.
- 3. Combine two files in one by using paste command with -d (means delete space)

sed '0~2d' file > 1 && sed '1~2d' file > 2 && paste -d " "

edited Aug 21 '14 at 17:37



jaypal singh

59.3k 15 86 124

answered Apr 10 '14 at 21:43



Serg



perl -0pE 's{^KEY.*?\K\s+(\d+)\$}{ \$1}msg;' data.txt > data





-0 gobbles the whole file instead of reading it line-by-line; pE wraps code with loop and prints the output, see details in http://perldoc.perl.org/perlrun.html;

^KEY match "KEY" in the beginning of line, followed by non-greedy match of anything (.*?) before sequence of

- 1. one or more spaces \s+ of any kind including line breaks;
- 2. one or more digit (\d+) which we capture and later re-insert as \$1;

followed by the end of line \$.

\k conveniently excludes everything on its left hand side from substitution so { \$1} replaces only 1-2 sequence, see http://perldoc.perl.org/perlre.html.

edited Aug 29 '14 at 0:03

answered Aug 28 '14 at 6:56



Onlyjob

4,109 1 26 30



A more-general solution (allows for more than one followup line to be joined) as a shell script. This adds a line between each, because I needed visibility, but that is easily remedied. This example is where the "key" line ended in : and no other lines did.



```
esac done

edited Sep 26 '14 at 21:45

answered Sep 26 '14 at 19:26

Jan Parcel
1 1
```

Try the following line:

-1 while read line1; do read line2; echo "\$line1 \$line2"; don

Put delimiter in-between

"\$line1 \$line2";

e.g. if the delimiter is | , then:

"\$line1|\$line2";

edited May 17 '16 at 19:04

coatless
12.7k 9 45 59

answered May 17 '16 at 18:22

Suman
209 2 4

16 '16 at 10:25

I agree partially, I try to add explanation and more generic It will not edit old file as well. Thanks for your suggestion – Suman Sep 24 '16 at 7:01 🖍



You can use xargs like this:



xargs -a file



edited Jun 16 '16 at 10:21



fedorqui

175k 55 363 403

answered May 4 '16 at 17:01



RSG

233 2 5

1 Hint. Just <u>edit</u> your answer with an explanation. – Mogsdad May 4 '16 at 20:29

This doesn't work at all – fedorqui Jun 16 '16 at 10:22

% cat > file a b c % xargs -a file a b c % Works for me – RSG Jul 19 '17 at 13:08 \nearrow

It does *something*, yes, but not what the OP asked for. Specifically, it joins as many lines as possible. You could actually get what you want with xargs -n 2 but this answer does not explain this at all. – tripleee Aug 30 '17 at 9:35