Linux comm command

Updated: 07/20/2017 by Computer Hope

About comm

Compare two sorted files line-by-line.

Description

Compare sorted files **FILE1** and **FILE2** line-by-line.

With no options, **comm** produces three-column output. Column one contains lines unique to **FILE1**, column two contains lines unique to **FILE2**, and column three contains lines common to both files. Each of these columns can be suppressed individually with options.

comm syntax

```
comm [OPTION]... FILE1 FILE2
```

Options

-1	suppress column 1 (lines unique to **FILE1**)
-2	suppress column 2 (lines unique to **FILE2**)
-3	suppress column 3 (lines that appear in both files)
--check-order	check that the input is correctly sorted, even if all input lines are pairable
--nocheck-order	do not check that the input is correctly sorted
--output-	separate columns with string *STR*

delimiter=*STR*

--help	display a help message, and exit.
--version	output version information, and exit.

Examples

Let's say you have two text files, **recipe.txt** and **shopping-list.txt**.

recipe.txt contains these lines:

```
All-Purpose Flour
Baking Soda
Bread
Brown Sugar
Chocolate Chips
Eggs
Milk
Salt
Vanilla Extract
White Sugar
```

And **shopping-list.txt** contains these lines:

```
All-Purpose Flour
Bread
Brown Sugar
Chicken Salad
Chocolate Chips
Eggs
Milk
Onions
Pickles
Potato Chips
Soda Pop
```

```
Tomatoes
White Sugar
```

As you can see, the two files are different, but many of the lines are the same. Not all of the recipe ingredients are on the shopping list, and not everything on the shopping list is part of the recipe.

If we run the **comm** command on the two files, it will read both files and give us three columns of output:

```
comm recipe.txt shopping-list.txt
```

```
All-Purpose Flour
Baking Soda
                Bread
                Brown Sugar
        Chicken Salad
                Chocolate Chips
                Eggs
                Milk
        Onions
        Pickles
        Potato Chips
Salt
        Soda Pop
        Tomatoes
Vanilla Extract
                White Sugar
```

Here, each line of output has either zero, one, or two tabs at the beginning, separating the output into three columns:

1. The first column (zero tabs) is lines that only appear in the first file.
2. The second column (one tab) is lines that only appear in the second file.
3. The third column (two tabs) is lines that appear in both files.

(The columns overlap visually because in this case, our terminal prints a tab as eight spaces. It might look different on your screen.)

Next, let's look at how we can bring our separated data into a spreadsheet.

Creating a CSV File For Spreadsheets

One useful way to use **comm** is to output to a CSV file, which can then be read by a spreadsheet program. CSV files are just text files that use a certain character, usually a comma, tab, or semicolon, to delimit data in a way that can be read as a spreadsheet. By convention, CSV filenames have the extension **.csv**.

For instance, let's run the same command, but this time let's redirect the output to a file called **output.csv** by using the > operator:

```
comm recipe.txt shopping-list.txt > output.csv
```

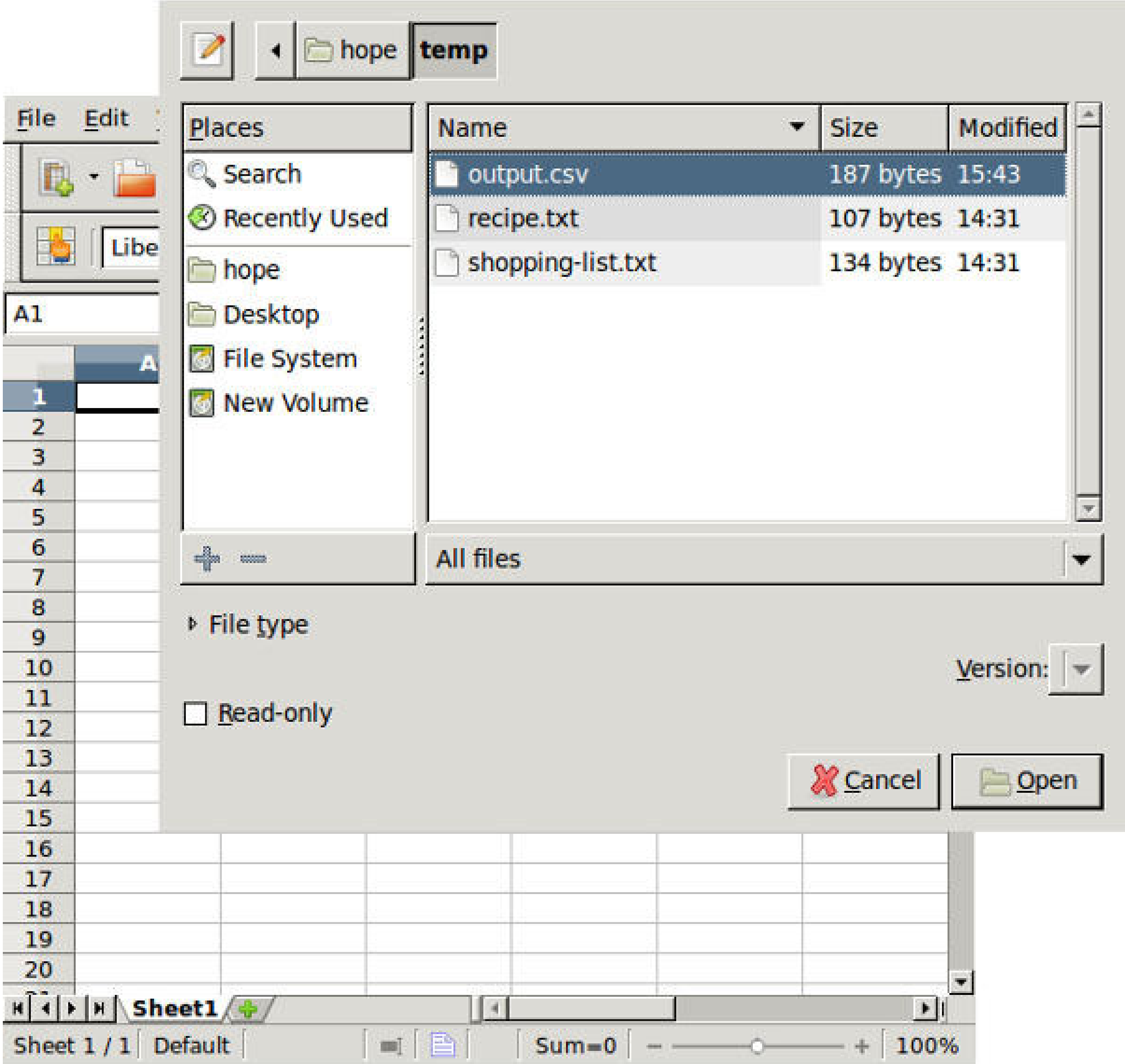
This time there is no output on the screen. Instead, output is sent to a file called **output.csv**. To check that it worked correctly, we can cat the contents of **output.csv**:

```
cat output.csv
```

```
All-Purpose Flour
Baking Soda
                Bread
                Brown Sugar
        Chicken Salad
                Chocolate Chips
                Eggs
                Milk
        Onions
        Pickles
        Potato Chips
Salt
```

```
        Soda Pop
        Tomatoes
Vanilla Extract
                White Sugar
```

To bring this data into a spreadsheet, we can open it in LibreOffice Calc:

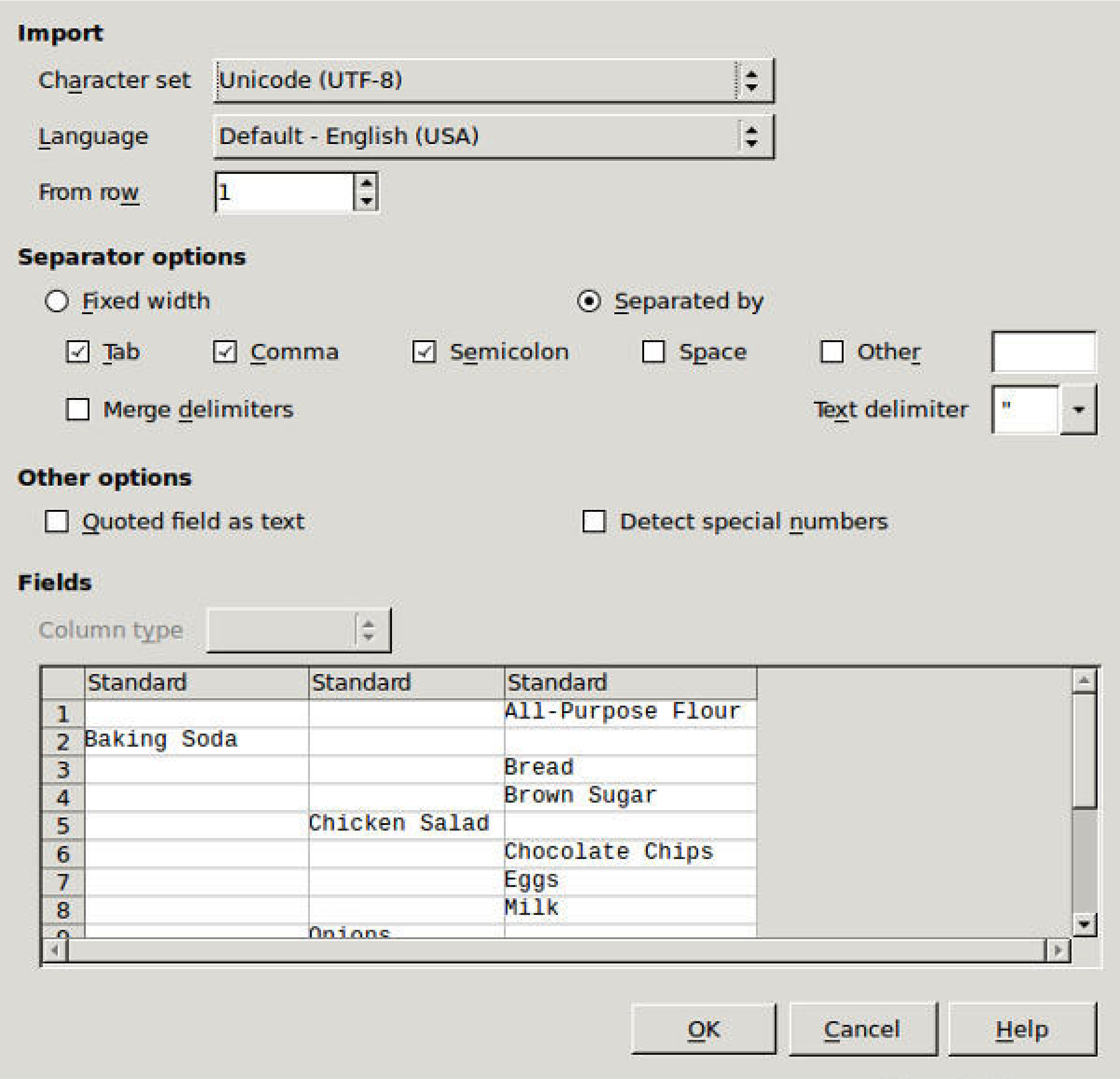


Name	Size	Modified
output.csv	187 bytes	15:43
recipe.txt	107 bytes	14:31
shopping-list.txt	134 bytes	14:31

ComputerHope.com

Before it opens the file, LibreOffice asks us how to interpret the file data.

We want the column delimiter to be tab characters, which is already checked by default. (There are no commas or semicolons in our data, so we don't have to worry about the other checkboxes.) It also gives us a preview of how the data will look, given the options we selected.



Import

Character set: Unicode (UTF-8)

Language: Default - English (USA)

From row: 1

Separator options

○ Fixed width ● Separated by

☑ Tab ☑ Comma ☑ Semicolon □ Space □ Other

□ Merge delimiters Text delimiter: "

Other options

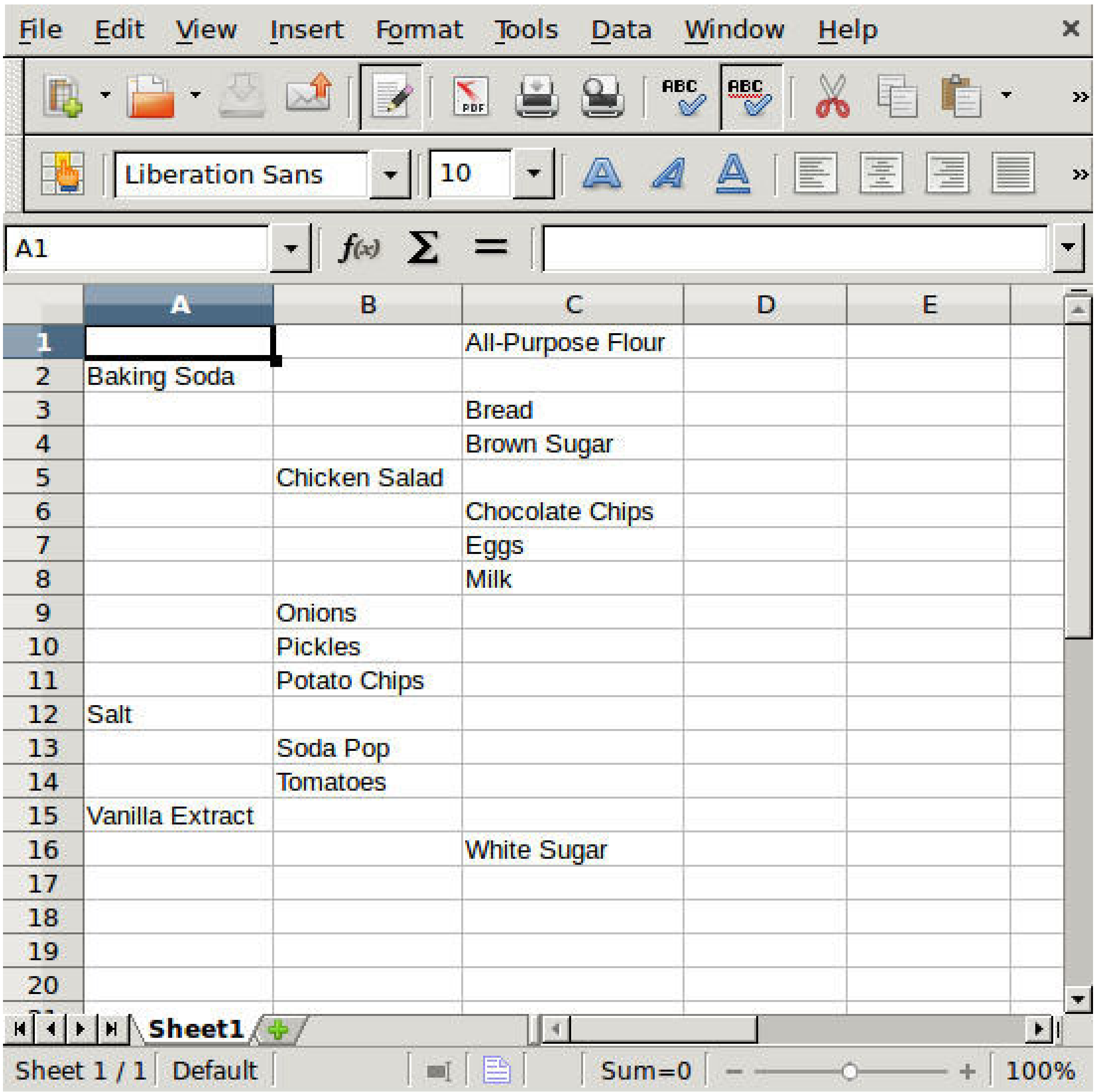
□ Quoted field as text □ Detect special numbers

Fields

Column type

	Standard	Standard	Standard
1			All-Purpose Flour
2	Baking Soda		
3			Bread
4			Brown Sugar
5		Chicken Salad	
6			Chocolate Chips
7			Eggs
8			Milk
9		Onions	

Everything looks good, so we can click OK, and LibreOffice will import our data into a spreadsheet.



	A	B	C	D	E
1			All-Purpose Flour		
2	Baking Soda				
3			Bread		
4			Brown Sugar		
5		Chicken Salad			
6			Chocolate Chips		
7			Eggs		
8			Milk		
9		Onions			
10		Pickles			
11		Potato Chips			
12	Salt				
13		Soda Pop			
14		Tomatoes			
15	Vanilla Extract				
16			White Sugar		
17					
18					
19					
20					

Now if we wanted to, we could save the spreadsheet in another format such as a Microsoft Excel file, or an XML file, or even HTML.

Suppressing Columns

If you only want to output specific columns, you can specify the column numbers to suppress in the command, preceded by a dash. For instance, this command will suppress columns 1 and 2, displaying only column 3 — lines shared by both files. This isolates the items on the shopping list that are also part of the recipe:

```
comm -12 recipe.txt shopping-list.txt
```

```
All-Purpose Flour
Bread
```

```
Brown Sugar
Chocolate Chips
Eggs
Milk
White Sugar
```

The next command will suppress columns 2 and 3, displaying only column 1 — lines in the recipe which are not in the shopping list. This shows us what ingredients we already have in our cupboard:

```
comm -23 recipe.txt shopping-list.txt
```

```
Baking Soda
Salt
Vanilla Extract
```

And the next command will suppress column 3, displaying only columns 1 and 2 — the items in the recipe that are not on the shopping list, and the items on the shopping list that are not in the recipe, each in their own column.

```
comm -3 recipe.txt shopping-list.txt
```

```
Baking Soda
        Chicken Salad
        Onions
        Pickles
        Potato Chips
Salt
        Soda Pop
        Tomatoes
Vanilla Extract
```

Related commands

cmp — Compare two files byte by byte.

diff — Identify the differences between two files.

join — Join the lines of two files which share a common field of data.

sort — Sort the lines in a text file.

uniq — Identify, and optionally filter out, repeated lines in a file.