Linux cut command

Updated: 04/26/2017 by Computer Hope

About cut

Remove or "cut out" sections of each line of a file or files.

cut syntax

```
cut OPTION... [FILE]...
```

Options

-b, **--bytes**=*LIST*	Select only the bytes from each line as specified in *LIST*. *LIST* specifies a byte, a set of bytes, or a range of bytes; see Specifying *LIST* below.
-c, **--characters**=*LIST*	Select only the characters from each line as specified in *LIST*. *LIST* specifies a character, a set of characters, or a range of characters; see Specifying *LIST* below.
-d, **--delimiter**=*DELIM*	use character *DELIM* instead of a tab for the field delimiter.
-f, **--fields**=*LIST*	select only these fields on each line; also print any line that contains no delimiter character, unless the **-s** option is specified. *LIST* specifies a field, a set of fields, or a range of fields; see Specifying *LIST* below.
-n	This option is ignored, but is included for compatibility reasons.
--complement	complement the set of selected bytes, characters or fields.
-s, **--only-delimited**	do not print lines not containing delimiters.

--output-delimiter=*STRING*	use *STRING* as the output delimiter string. The default is to use the input delimiter.
--help	Display a help message and exit.
--version	output version information and exit.

Usage Notes

When invoking **cut**, use the **-b**, **-c**, or **-f** option, but only one of them.

If no *FILE* is specified, **cut** reads from the standard input.

Specifying *LIST*

Each *LIST* is made up of an integer, a range of integers, or multiple integer ranges separated by commas. Selected input is written in the same order that it is read, and is written to output exactly once. A range consists of:

N	the *N*th byte, character, or field, counted from **1**.
*N***-**	from the *N*th byte, character, or field, to the end of the line.
*N***-***M*	from the *N*th to the *M*th byte, character, or field (inclusive).
-*M*	from the first to the *M*th byte, character, or field.

For example, let's say you have a file named **data.txt** which contains the following text:

```
one     two     three   four    five
alpha   beta    gamma   delta   epsilon
```

In this example, each of these words is separated by a tab character, not spaces. The tab character is the default delimiter of **cut**, so it will by default consider a field to be anything delimited by a tab.

To "cut" only the third field of each line, use the command:

```
cut -f 3 data.txt
```

...which will output the following:

```
three
gamma
```

If instead you want to "cut" only the second-through-fourth field of each line, use the command:

```
cut -f 2-4 data.txt
```

...which will output the following:

```
two     three   four
beta    gamma   delta
```

If you want to "cut" only the first-through-second and fourth-through-fifth field of each line (omitting the third field), use the command:

```
cut -f 1-2,4-5 data.txt
```

...which will output the following:

```
one     two     four    five
alpha   beta    delta   epsilon
```

Or, let's say you want the third field and every field after it, omitting the first two fields. In this case, you could use the command:

```
cut -f 3- data.txt
```

...which will output the following:

```
three	four	five
gamma	delta	epsilon
```

Specifying a range with *LIST* also applies to **cut**ting characters (**-c**) or bytes (**-b**) from a line. For example, to output only the third-through-twelfth character of every line of **data.txt**, use the command:

```
cut -c 3-12 data.txt
```

...which will output the following:

```
e	two	thre
pha	beta	g
```

Remember that the "space" in between each word is actually a single tab character, so both lines of output are displaying ten characters: eight alphanumeric characters and two tab characters. In other words, **cut** is omitting the first two characters of each line, counting tabs as one character each; outputting characters three through twelve, counting tabs as one character each; and omitting any characters after the twelfth.

Counting bytes instead of characters will result in the same output in this case, because in an ASCII-encoded text file, each character is represented by a single byte (eight bits) of data. So the command:

```
cut -b 3-12 data.txt
```

...will, for our file **data.txt**, produce exactly the same output:

```
e       two     thre
pha     beta    g
```

Specifying A Delimiter Other Than Tab

The tab character is the default delimiter that **cut** uses to determine what constitutes a field. So, if your file's fields are already delimited by tabs, you don't need to specify a different delimiter character.

You can specify any character as the delimiter, however. For instance, the file **/etc/passwd** contains information about each user on the system, one user per line, and each information field is delimited by a colon ("**:**"). For example, the line of **/etc/passwd** for the **root** user may look like this:

```
root:x:0:0:root:/root:/bin/bash
```

These fields contain the following information, in the following order, separated by a colon character:

1. Username
2. Password (shown as **x** if encrypted)
3. User ID number (UID)
4. Group ID number (GID)
5. Comment field (used by the finger command)
6. Home Directory
7. Shell

The username is the first field on the line, so to display each username on the system, use the command:

```
cut -f 1 -d ':' /etc/passwd
```

...which will output, for example:

```
root
daemon
bin
sys
chope
```

(There are many more user accounts on a typical system, including many accounts specific to system services, but for this example we will pretend there are only five users.)

The third field of each line in the **/etc/passwd** file is the UID (user ID number), so to display each username and user ID number, use the command:

```
cut -f 1,3 -d ':' /etc/passwd
```

...which will output the following, for example:

```
root:0
daemon:1
bin:2
sys:3
chope:1000
```

As you can see, the output will be delimited, by default, using the same delimiter character specified for the input. In this case, that's the colon character ("**:**"). You can specify a different delimiter for the input and output, however. So, if you wanted to run the previous command, but have the output delimited by a space, you could use the command:

```
cut -f 1,3 -d ':' --output-delimiter=' ' /etc/passwd
```

```
root 0
daemon 1
```

```
bin 2
sys 3
chope 1000
```

But what if you want the output to be delimited by a tab? Specifying a tab character on the command line is a bit more complicated, because it is an unprintable character. To specify it on the command line, you must "protect" it from the shell. This is done differently depending on which shell you're using, but in the Linux default shell (bash), you can specify the tab character with **$'\t'**. So the command:

```
cut -f 1,3 -d ':' --output-delimiter=$'\t' /etc/passwd
```

...will output the following, for example:

```
root    0
daemon  1
bin     2
sys     3
chope   1000
```

cut Examples

```
cut -c 3 file.txt
```

Outputs the third character of every line of the file **file.txt**, omitting the others.

```
cut -c 1-3 file.txt
```

Outputs the first three characters of every line of the file **file.txt**, omitting the rest.

```
cut -c 3- file.txt
```

Outputs the third through the last characters of each line of the file **file.txt**, omitting the first two characters.

```
cut -d ':' -f 1 /etc/passwd
```

Outputs the first field of the file **/etc/passwd**, where fields are delimited by a colon ('**:**'). The first field of **/etc/passwd** is the username, so this command will output every username in the **passwd** file.

```
grep '/bin/bash' /etc/passwd | cut -d ':' -f 1,6
```

Outputs the first and sixth fields, delimited by a colon, of any entry in the **/etc/passwd** file which specifies **/bin/bash** as the login shell. This command will output the username and home directory of any user whose login shell is **/bin/bash**.

Related commands

grep — Filter text which matches a regular expression.
paste — Merge corresponding lines of files.