Cut Command

Cut command in unix (or linux) is used to select sections of text from each line of files. You can use the cut command to select fields or columns from a line by specifying a delimiter or you can select a portion of text by specifying the range or characters. Basically the cut command slices a line and extracts the text.

Unix Cut Command Example

We will see the usage of cut command by considering the below text file as an example

```
> cat file.txt
unix or linux os
is unix good os
is linux good os
```

1. Write a unix/linux cut command to print characters by position?

The cut command can be used to print characters in a line by specifying the position of the characters. To print the characters in a line, use the -c option in cut command

```
cut -c4 file.txt
x
u
l
```

The above cut command prints the fourth character in each line of the file. You can print more than one character at a time by specifying the character positions in a comma separated list as shown in the below example

```
cut -c4,6 file.txt
xo
ui
ln
```

This command prints the fourth and sixth character in each line.

2.Write a unix/linux cut command to print characters by range?

You can print a range of characters in a line by specifying the start and end position of the characters.

```
cut -c4-7 file.txt
x or
unix
linu
```

The above cut command prints the characters from fourth position to the seventh position in each line. To print the first six characters in a line, omit the start position and specify only the end position.

```
cut -c-6 file.txt
unix o
is uni
is lin
```

To print the characters from tenth position to the end, specify only the start position and omit the end position.

```
cut -c10- file.txt
inux os
ood os
good os
```

If you omit the start and end positions, then the cut command prints the entire line.

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

3.Write a unix/linux cut command to print the fields using the delimiter?

You can use the cut command just as awk command to extract the fields in a file using a delimiter. The -d option in cut command can be used to specify the delimiter and -f option is used to specify the field position.

```
cut -d' ' -f2 file.txt
or
unix
linux
```

This command prints the second field in each line by treating the space as delimiter. You can print more than one field by specifying the position of the fields in a comma delimited list.

```
cut -d' ' -f2,3 file.txt
or linux
unix good
linux good
```

The above command prints the second and third field in each line.

Note: If the delimiter you specified is not exists in the line, then the cut command prints the entire line. To suppress these lines use the -s option in cut command.

4. Write a unix/linux cut command to display range of fields?

You can print a range of fields by specifying the start and end position.

```
cut -d' ' -f1-3 file.txt
```

The above command prints the first, second and third fields. To print the first three fields, you can ignore the start position and specify only the end position.

```
cut -d' ' -f-3 file.txt
```

To print the fields from second fields to last field, you can omit the last field position.

```
cut -d' ' -f2- file.txt
```

5. Write a unix/linux cut command to display the first field from /etc/passwd file?

The /etc/passwd is a delimited file and the delimiter is a colon (:). The cut command to display the first field in /etc/passwd file is

```
cut -d':' -f1 /etc/passwd
```

6. The input file contains the below text

```
> cat filenames.txt
logfile.dat
sum.pl
add_int.sh
```

Using the cut command extract the portion after the dot.

First reverse the text in each line and then apply the command on it.

```
rev filenames.txt | cut -d'.' -f1
```

For example, let's say we 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.