Create Your Own Commands

Sometimes, you may be having a hard time remembering one command. Other times, you will find yourself running a very long command over and over again, and that drives you insane. In this chapter, you will learn how you can make your *own* commands, because you are the real boss.

Your first alias

Let's assume that you always forget that the command [free -h] displays the memory information of your system:

```
elliot@ubuntu-linux:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           3.9G        939M        2.2G        6.6M        752M        2.7G
Swap:          947M          0B        947M
```

You may be asking yourself: "Why can't I just type [memory] to display the memory information instead of [free -h]?". Well, you certainly can do that by creating an [alias].

The [alias] command instructs the shell to replace one string (word) with another. Well, how is this useful? Let me show you; if you run the following command:

```
elliot@ubuntu-linux:~$ alias memory="free -h"
```

Then every time you enter [memory], your shell will replace it with [free -h]:

```
elliot@ubuntu-linux:~$ memory
              total        used        free      shared  buff/cache   available
Mem:           3.9G        936M        2.2G        6.6M        756M        2.7G
Swap:          947M          0B        947M
```

Wow! So now you have achieved your dream! You can create an alias for any Linux command that you are having trouble remembering. Notice that the general format of the [alias] command is as follows:

```
alias alias_name="command(s)_to_run"
```

One alias for multiple commands

You can use a semicolon to run multiple commands on the same line. For example, to create a new directory named [newdir] and change to [newdir] all at once, you can run the following command:

```
elliot@ubuntu-linux:~$ mkdir newdir; cd newdir
elliot@ubuntu-linux:~/newdir$
```

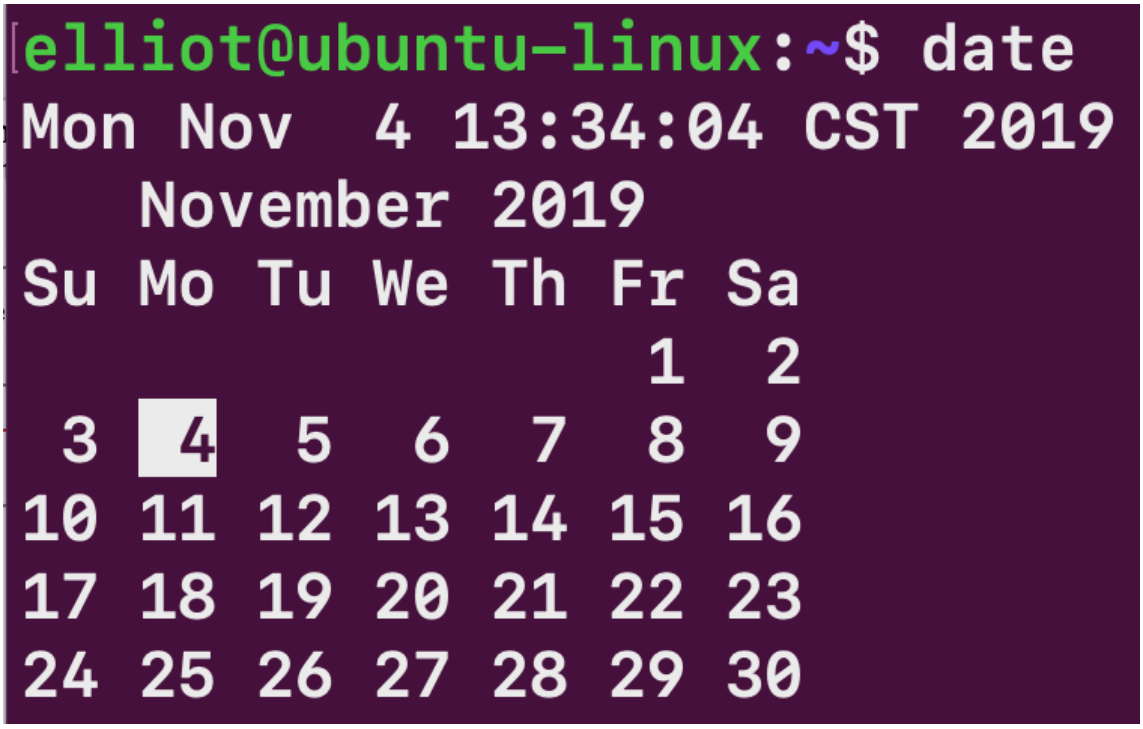
So you use a semicolon to separate each command. In general, the syntax for running multiple commands on the same line is as follows:

```
command1; command2; command3; command4; ....
```

We often like to check the calendar and the date at the same time, right? For that, we will create an alias named [date] so that every time we run [date], it will run both the [date] and [calendar] commands:

```
elliot@ubuntu-linux:~$ alias date="date;cal"
```

Now let's run [date] and see what's up:



Notice here that we used the alias name [date], which is already the name of an existing command; this is completely fine with aliases.

Listing all aliases

You should also know that aliases are user-specific. So the aliases created by [elliot] will not work for user [smurf]; take a look:

```
elliot@ubuntu-linux:~$ su - smurf
Password:
smurf@ubuntu-linux:~$ date
Mon Nov 4 13:33:36 CST 2019
smurf@ubuntu-linux:~$ memory
Command 'memory' not found, did you mean:
    command 'lmemory' from deb lmemory
Try: apt install <deb name>
```

As you can see, [smurf] can't use the aliases of user [elliot]. So every user has their own set of aliases. Now, let's exit back to user [elliot]:

```
smurf@ubuntu-linux:~$ exit
logout
elliot@ubuntu-linux:~$ memory
              total        used        free      shared  buff/cache   available
Mem:           3.9G        937M        2.0G        6.6M        990M        2.7G
Swap:          947M          0B        947M
```

You can run the [alias] command to list all the aliases that can be used by the currently logged-in user:

```
elliot@ubuntu-linux:~$ alias
alias date='date;cal'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
alias memory='free -h'
```

Creating a permanent alias

So far, we have been creating temporary aliases; that is, the two aliases of [memory] and [date] that we created are temporarily and only valid for the current Terminal session. These two aliases will vanish as soon as you close your Terminal.

Open a new Terminal session, then try and run the two aliases we have created:

```
elliot@ubuntu-linux:~$ date
Mon Nov 4 13:43:46 CST 2019
elliot@ubuntu-linux:~$ memory

Command 'memory' not found, did you mean:
    command 'lmemory' from deb lmemory
Try: sudo apt install <deb name>
```

As you can see, they are gone! They are not even in your list of aliases anymore:

```
elliot@ubuntu-linux:~$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

To create a permanent alias for a user, you need to include it in the hidden [.bashrc] file in the user's home directory. So to permanently add our two aliases back, you have to add the following two lines at the very end of the [/home/el- liot/.bashrc] file:

```
alias memory = "free -h"
alias date = "date;cal"
```

You can do it by running the following two [echo] commands:

```
elliot@ubuntu-linux:~$ echo 'alias memory="free -h"' >> /home/elliot/.bashrc
elliot@ubuntu-linux:~$ echo 'alias date="date;cal"' >> /home/elliot/.bashrc
```

After you add both aliases to the [/home/elliot/.bashrc] file, you need to run the [source] command on the [/home/elliot/.bashrc] file for the change to take effect in the current session:

```
elliot@ubuntu-linux:~$ source /home/elliot/.bashrc
```

Now you can use your two aliases, [memory] and [date], forever without worrying that they will disappear after you close your current Terminal session:

```
elliot@ubuntu-linux:~$ memory
              total        used        free      shared  buff/cache   available
Mem:           3.8G        233M        3.3G        672K        282M        3.4G
Swap:            0B          0B          0B
elliot@ubuntu-linux:~$ date
Mon Nov  4 13:35:59 CST 2019
   November 2019
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Removing an alias

Let's create another temporary alias named [lastline] that will display the last line in a file:

```
elliot@ubuntu-linux:~$ alias lastline="tail -n 1"
```

Now let's try our new alias on the [/home/elliot/.bashrc] file:

```
elliot@ubuntu-linux:~$ lastline /home/elliot/.bashrc
alias date="date;cal"
```

Alright! It works well. Now, if you wish to delete the alias, then you can run the [unalias] command followed by the alias name:

```
elliot@ubuntu-linux:~$ unalias lastline
```

So now the [lastline] alias has been deleted:

```
elliot@ubuntu-linux:~$ lastline /home/elliot/.bashrc
lastline: command not found
```

You can also use the [unalias] command to temporarily deactivate a permanent alias. For example, if you run the following command:

```
elliot@ubuntu-linux:~$ unalias memory
```

Now, the permanent alias [memory] will not work in the current Terminal session:

```
elliot@ubuntu-linux:~$ memory

Command 'memory' not found, did you mean:
```

```
    command 'lmemory' from deb lmemory
Try: sudo apt install <deb name>
```

However, the alias [memory] will come back in a new Terminal session. To remove a permanent alias, you need to remove it from the [.bashrc] file.

Some useful aliases

Now let's create some useful aliases that will make our life much more enjoyable while working on the Linux command line.

A lot of people hate to remember all the [tar] command options, so let's make it easy for these people then. We will create an alias named [extract] that will extract files from an archive:

```
elliot@ubuntu-linux:~$ alias extract="tar -xvf"
```

You can try the alias on any archive, and it will work like a charm.

Similarly, you can create an alias named [compress_gzip] that will create a gzip-compressed archive:

```
elliot@ubuntu-linux:~$ alias compress_gzip="tar -czvf"
```

You may also want to create an alias named [soft] that will create soft links:

```
elliot@ubuntu-linux:~$ alias soft="ln -s"
```

You can use the soft alias to create a soft link named [logfiles] that points to the [/var/logs] directory:

```
elliot@ubuntu-linux:~$ soft /var/logs logfiles
elliot@ubuntu-linux:~$ ls -l logfiles
lrwxrwxrwx 1 elliot elliot 9 Nov 4 15:08 logfiles -> /var/logs
```

Now let's create an alias named [LISTEN] that will list all the listening ports on your system:

```
elliot@ubuntu-linux:~$ alias LISTEN="netstat -tulpen| grep -i listen"
```

Now let's try and run the [LISTEN] alias:

```
elliot@ubuntu-linux:~$ LISTEN
tcp      0      0 127.0.0.53:53     0.0.0.0:*     LISTEN
tcp      0      0 0.0.0.0:22        0.0.0.0:*     LISTEN
tcp      0      0 127.0.0.1:631     0.0.0.0:*     LISTEN
tcp      0      0 127.0.0.1:25      0.0.0.0:*     LISTEN
tcp6     0      0
tcp6     0      0 ::1:631
tcp6     0      0 ::1:25
```

This is pretty cool! Let's create one final alias, [sort_files], that will list all the files in the current directory sorted by size (in descending order):

```
alias sort_files="du -bs * | sort -rn"
```

Now let's try and run the [sort_files] alias:

```
elliot@ubuntu-linux:~$ sort_files
9628732 Downloads
2242937 Pictures
65080 minutes.txt
40393 load.txt
32768 dir1
20517 Desktop
20480 small
8192 hackers
476 game.sh
168 practise.txt
161 filetype.sh
142 noweb.sh
108 3x10.sh
92 rename.sh
92 numbers.sh
88 detect.sh
74 hello3.sh
66 fun1.sh
59 hello20.sh
37 hello2.sh
33 hello.sh
17 mydate.sh
16 honey
9 logs
6 softdir1
0 empty
```

As you can see, the files in the current directory are listed in descending order of size (that is, the biggest first). This will prove to be particularly useful when you are doing some cleaning on your system and you want to inspect which files are occupying the most space.

Adding safety nets

You can also use aliases to protect against dumb mistakes. For example, to protect against removing important files by mistake, you can add the following alias:

```
elliot@ubuntu-linux:~$ alias rm="rm -i"
```

Now you will be asked to confirm each time you attempt to remove a file:

```
elliot@ubuntu-linux:~$ rm *
rm: remove regular file '3x10.sh'?
```

Go crazy with aliases

You can also have some fun with aliases and make users go crazy; take a look at this alias:

```
elliot@ubuntu-linux:~$ alias nano="vi"
```

Now when user [elliot] tries to open the [nano] editor, the [vi] editor will open instead! User [elliot] can overcome this dilemma by typing in the full path of the [nano] editor. Here is another funny alias:

```
elliot@ubuntu-linux:~$ alias exit="echo No I am not exiting ..."
```

Now look what will happen when user [elliot] tries to exit the Terminal:

```
elliot@ubuntu-linux:~$ exit
No I am not exiting ...
elliot@ubuntu-linux:~$ exit
No I am not exiting ...
```

I will let you deal with this by yourself; I am evil like that! Haha.

Knowledge check

For the following exercises, open up your Terminal and try to solve the following tasks:

1. Create a temporary alias called [ins] for the [apt-get install] command.
2. Create a temporary alias called [packages] for the [dpkg -l] command.
3. Create a permanent alias called [clean] that will remove all the files in the [/tmp] directory.