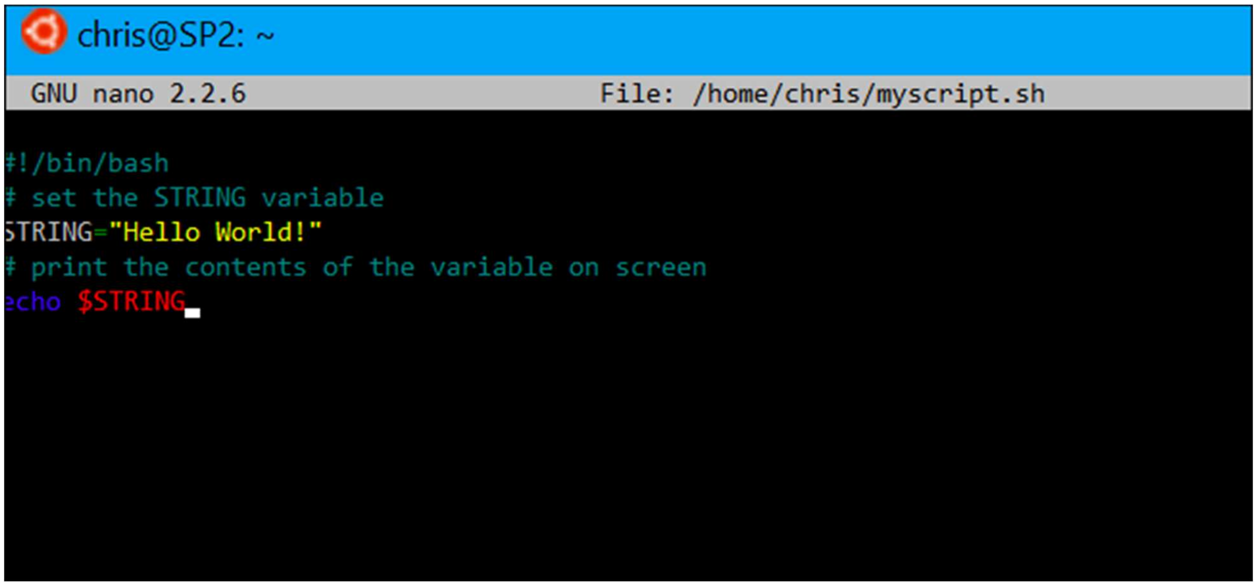
How to Create and Run Bash Shell Scripts on Windows 10



```
GNU nano 2.2.6                              File: /home/chris/myscript.sh

!/bin/bash
 set the STRING variable
TRING="Hello World!"
 print the contents of the variable on screen
cho $STRING
```

With the arrival of Windows 10's Bash shell, you can now create and run Bash shell scripts on Windows 10. You can also incorporate Bash commands into a Windows batch file or PowerShell script.

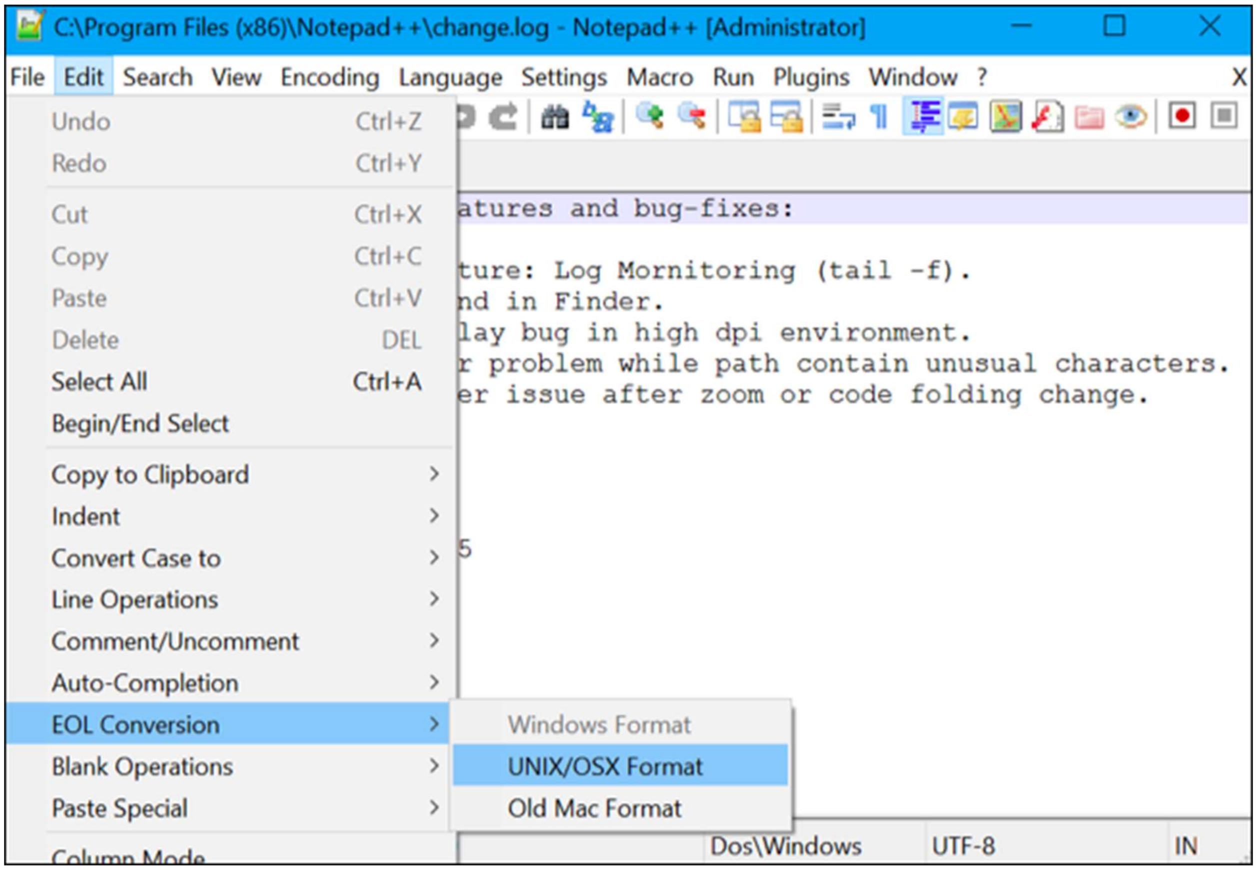
Even if you know what you're doing, this isn't necessarily as simple as it seems. Windows and UNIX use different end-of-line characters, and the Windows file system is accessible in a different location in the Bash environment.

How to Write a Bash Script on Windows 10

RELATED: ***How to Install and Use the Linux Bash Shell on Windows 10***

When writing shell scripts on Windows, bear in mind that Windows and UNIX-like systems like Linux use different "end of line" characters in text files in shell scripts.

In other words, this means that you can't simply write a shell script in Notepad. Save the file in Notepad and it won't be interpreted properly by Bash. However, you can use more advanced text editors–for example, Notepad++ allows you to give a file UNIX end-of-line characters by clicking Edit > EOL Conversion > UNIX/OSX Format.



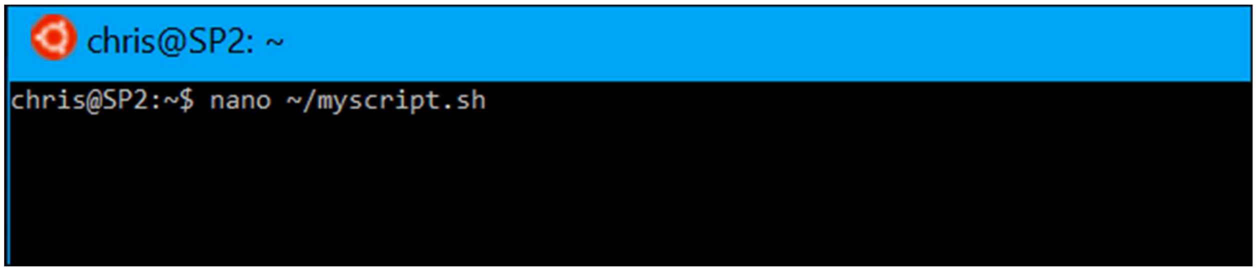
```
atures and bug-fixes:

ture: Log Mornitoring (tail -f).
nd in Finder.
lay bug in high dpi environment.
r problem while path contain unusual characters.
er issue after zoom or code folding change.
```

However, you're better off just writing the shell script in the Bash environment itself. The Ubuntu-based Bash environment comes with both the vi and nano text editors. The vi editor is more powerful, but if you've never used it before, you may want to start with nano. It's easier to use if you're new.

For example, to create a bash script in nano, you'd run the following command in bash:

```
nano ~/myscript.sh
```

This would open the Nano text editor pointed at a file named "myscript.sh" in your user account's home directory. (The "~" character represents your home directory, so the full path is /home/username/myscript.sh.)



```
chris@SP2:~$ nano ~/myscript.sh
```

Start your shell script with the line:

```
#!/bin/bash
```

RELATED: ***The Beginner's Guide to Shell Scripting: The Basics***

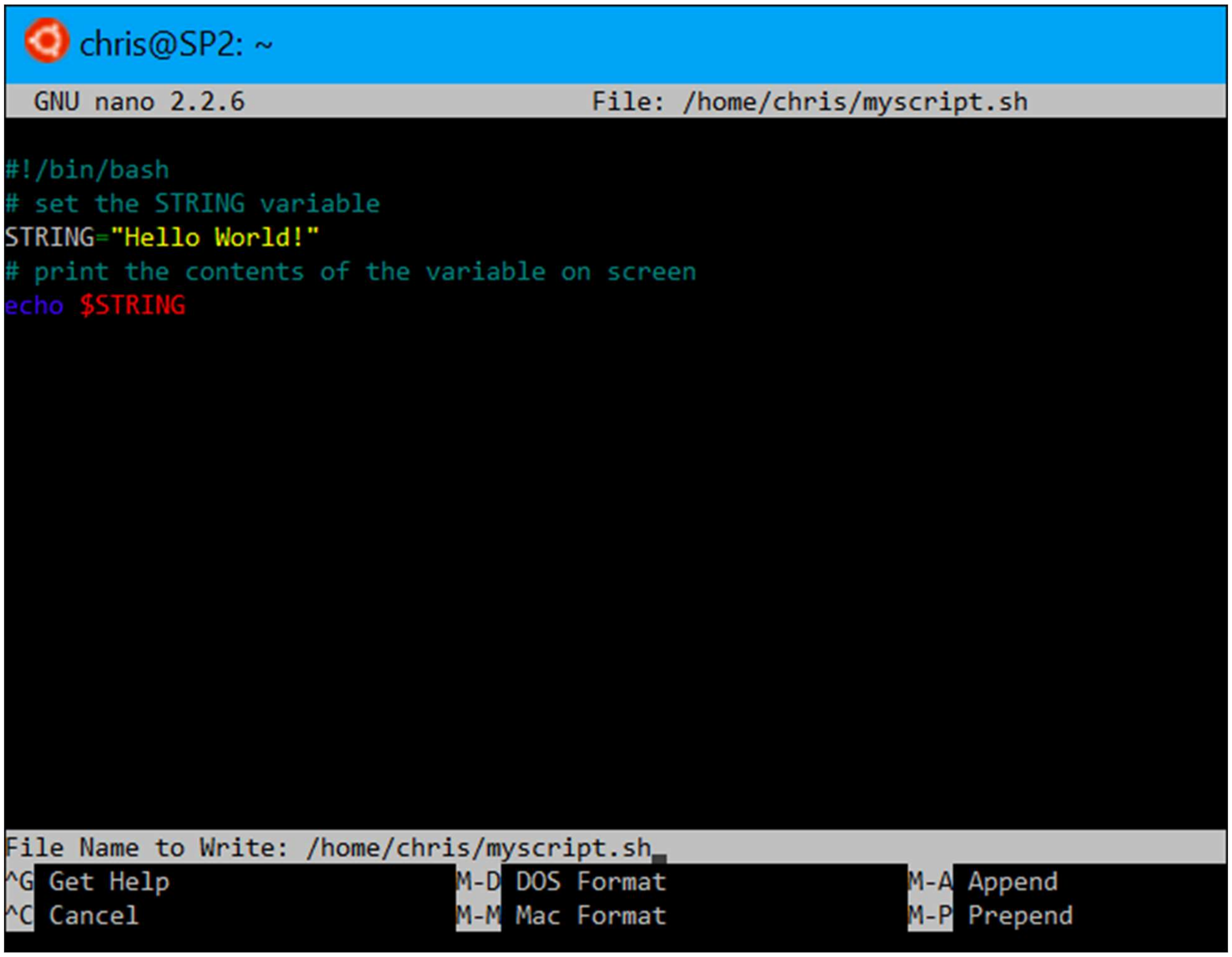
Enter the commands you want to run, each one on its own line. The script will run each command in turn. Add a "#" character before a line to treat it as a "comment", something which helps you and other people understand the script but which isn't run as a command. For more advanced tricks, consult a more detailed guide to Bash scripts on Linux. The same techniques will work in Bash on Ubuntu on Windows.

Note that there's no way to run Windows programs from within the Bash environment. You're restricted to Linux terminal commands and utilities, just as you would be on a typical Linux system.

For example, let's just use a basic "hello world" script as an example here:

```
#!/bin/bash
# set the STRING variable
STRING="Hello World!"
# print the contents of the variable on screen
echo $STRING
```

If you're using the Nano text editor, you can save the file by pressing Ctrl+O and then Enter. Close the editor by pressing Ctrl+X.

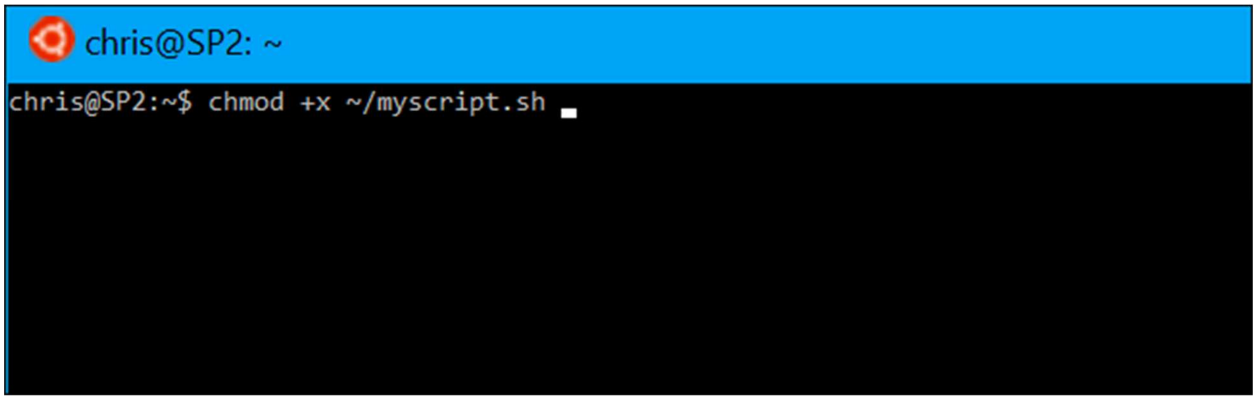


```
chris@SP2: ~
  GNU nano 2.2.6                     File: /home/chris/myscript.sh

#!/bin/bash
# set the STRING variable
STRING="Hello World!"
# print the contents of the variable on screen
echo $STRING











File Name to Write: /home/chris/myscript.sh
^G Get Help                    M-D DOS Format                 M-A Append
^C Cancel                      M-M Mac Format                 M-P Prepend
```

Make the Script Executable and then Run It

You’ll probably want the make the script executable so you can run it more easily. On Linux, that means you need to give the script file the executable permission. To do so, run the following command in the terminal, pointing it at your script:

```
chmod +x ~/myscript.sh
```

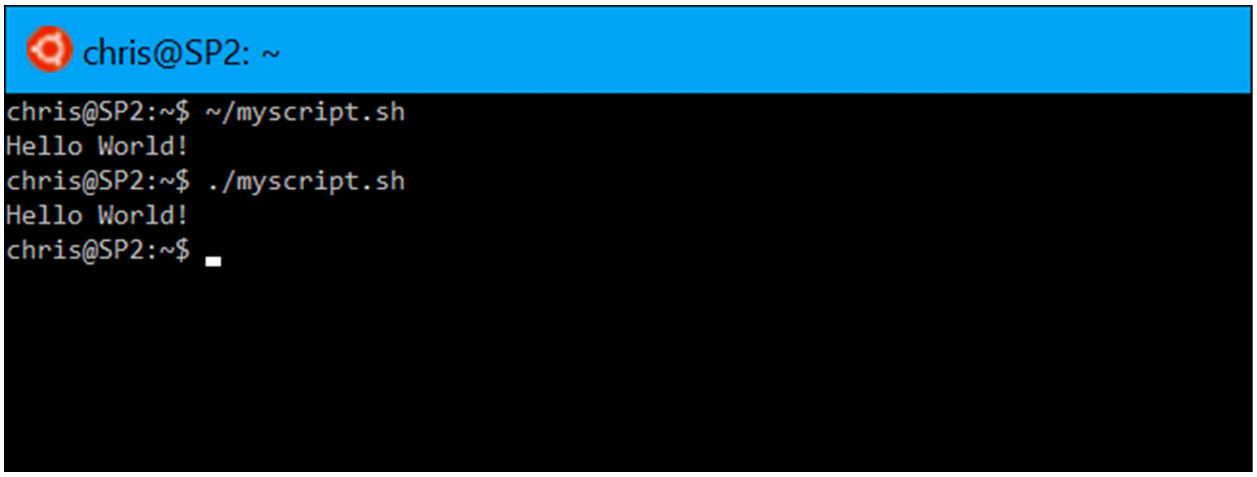


```
chris@SP2: ~
chris@SP2:~$ chmod +x ~/myscript.sh
```

To run the script, you can now just run it in the terminal by typing its path. Whenever you want to launch the script in the future, just open the Bash shell and type the path to the script.

```
~/myscript.sh
```

(If the script is in the current directory, you can run it with ./myscript.sh)



```
chris@SP2:~$ ~/myscript.sh
Hello World!
chris@SP2:~$ ./myscript.sh
Hello World!
chris@SP2:~$ _
```

How to Work With Windows Files in a Bash Script

RELATED: ***How to Access Your Ubuntu Bash Files in Windows (and Your Windows System Drive in Bash)***

To access Windows files in the script, you'll need to specify their path under /mnt/c, not their Windows path. For example, if you wanted to specify the C:\Users\Bob\Downloads\test.txt file, you'd need to specify the /mnt/c/Users/Bob/Downloads/test.txt path. Consult our guide to file locations in Windows 10's Bash shell for more details.

How to Incorporate Bash Commands into a Batch or PowerShell Script

RELATED: ***How to Set Your Default Linux Distribution on Windows 10***

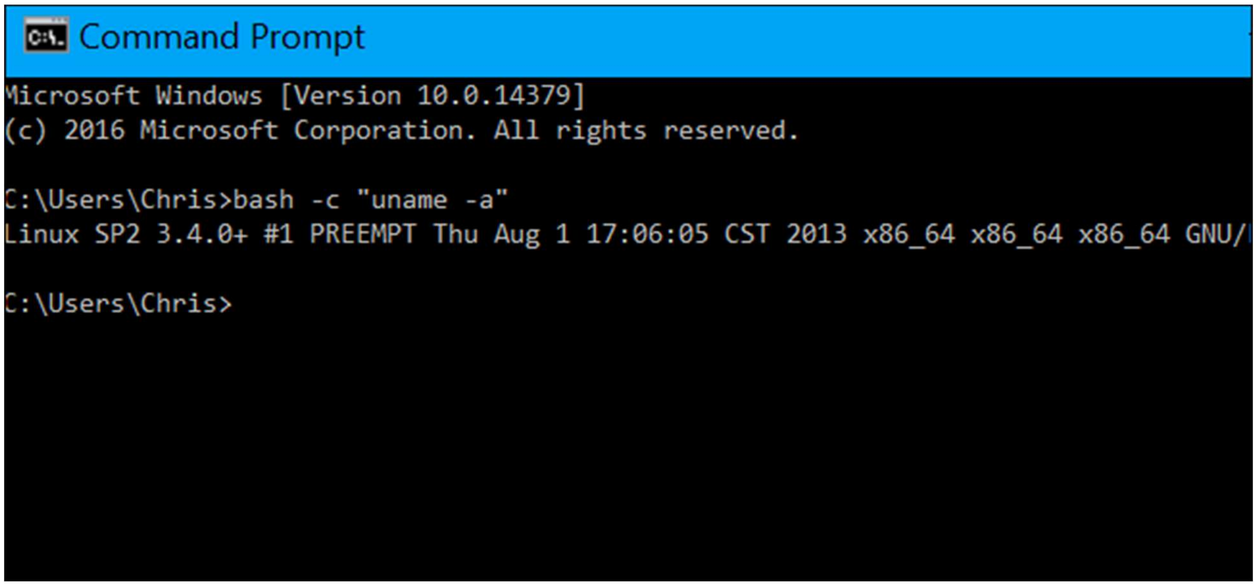
Lastly, if you have an existing batch file or PowerShell script you want to incorporate commands into, you can run Bash commands directly using the `bash -c` command.

For example, to run a Linux command in a Command Prompt or PowerShell window, you can run the following command:

```
bash -c "command"
```

This trick allows you to add Bash commands into batch files or PowerShell scripts. The Bash shell window will appear when a Bash command is running.

Update: If you have multiple Linux environments installed, you can use the wslconfig command to choose the default Linux environment used when you run the `bash -c` command.



```
Microsoft Windows [Version 10.0.14379]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Chris>bash -c "uname -a"
Linux SP2 3.4.0+ #1 PREEMPT Thu Aug 1 17:06:05 CST 2013 x86_64 x86_64 x86_64 GNU/

C:\Users\Chris>
```

To create a shortcut to a Bash script from within Windows, just create a shortcut like normal. For the shortcut's target, use the `bash -c` command we outlined above and point it at the Bash script you created.

For example, you'd point a shortcut at " `bash -c "~/myscript.sh"` " to run the example script above. You can also just run this command from a Command Prompt or PowerShell window, too.