**Linux Shell Scripting: A project-based approach to learning**

15 May 2021

17:23

**EIGHTH SCRIPT**

1. Advanced Standard Input
2. Advanced Standard Output
3. Advanced Standard Error

* Writing text to a file:

echo "${UID}" > /tmp/textfile

OR,

FILE="/tmp/datafile"

head -n4 /etc/passwd > ${FILE}

* Reading from a file

Syntax:

read LINE < ${FILE}

echo "LINE"

* Changing the password of an existing user using the password stored in the file
* Echo "secret" > password
* Cat password
* Sudo passwd --stdin einstein < password --> This command will assign the line "secret" stored in password file to "einstein" user.
* Passwd --stdin : This command will indicate that passwd should read the new password from standard input, which can be a pipe.
  + Input coming from an output of the command - we use "|"
  + Input coming from a file - we use "<"
  + Reading from a file - "<"
  + Writing to a file - ">"
* Using single ">" will create a file if it doesn’t exist and will truncate and rewrite the file from the beginning if it does exist.
* Using double ">>" will create a file if it doesn’t exist and will append and add new text towards the end if it does exist.
* File Descriptor- A number that represents an open file. By default, every process begins with three open file descriptors represented by FD.
  + File Descriptors: FD0 - STDIN, FD1 - STDOUT, FD2- STDERR
  + One question that comes up, STDIN is keyboard but that is not a file, STDOUT and STDERR are for monitor and those are again not files. Then, why do we say every process begins with a FD0.
  + Reason:
    - Linux represents everything as a file.
    - FD : It’s a way that a program interacts with files or to other resources that work like files such as keyboard, monitor.
    - FD are like pointers to data or places where data can be written
    - This abstraction of using resources as file allows us to consider stdin of one resource as the stout to another resource.

* **FILE DESCRIPTOR INPUT OPERATION**
  + Implicit way of doing read operation:

Command:

* Read X < /etc/centos-release
* Echo ${X}
* Will display "CentOS Linux release 7.3.1611 (Core)"
* Explicit way of doing IO operation:

Command:

* Read X 0< /etc/centos-release
* Echo ${X}
* Will display "CentOS Linux release 7.3.1611 (Core)"
* By default, if we don’t specify the FD 0 in above command where "<" is used. it is 0 by default.
* Make sure it is exactly 0< and not 0[space]< otherwise, if we write "read X 0 < /etc/centos-release" instead of "read X 0< /etc/centos-release". If will throw an error "-bash: read `0': not a valid identifier". In other words, it will consider X and 0 as two arguments for the read command.

* **FILE DESCRIPTOR OUTPUT OPERATION**
  + Implicit way of doing output operation:

Command:

* Echo "${UID}" > uid
* Cat uid
* Output: 1000
* Explicit way of doing output operation:

Command:

* Echo "${UID}" 1> uid
* Cat uid
* Output: 1000

Or,

* Echo "${UID}" 1 > uid
* Cat uid
* Output: 1000 1
* By default, if we don’t specify the FD 1 in above command where ">" is used. it is 1 by default.
* Make sure it is exactly 1> and not 1[space]>otherwise, if we write "echo "${UID}" 1 > uid instead of "read X 1> uid. In other words, it will consider ${UID} and 1 as two arguments for the echo command.

[vagrant@localusers vagrant]$ **head -n1 /etc/passwd /etc/hosts**

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4[vagrant@localusers vagrant]$ head -n1 /etc/passwd /etc/fakefile

==> /etc/passwd <==

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

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

* In the above output standard output i.e. stdout as well as standard error i.e. stderr is generated.

[vagrant@localusers vagrant]$ **head -n1 /etc/passwd /etc/fakefile > head.out**

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

* In the above command standard output is written to the head.out file and standard error is displayed to the screen

[vagrant@localusers vagrant]$ cat head.out

==> /etc/passwd <==

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

[vagrant@localusers vagrant]$ **head -n1 /etc/passwd /etc/hosts /etc/fakefile 2>**

**head.out**

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

* In the above command standard output is diaplayed on screen whereas, standard error is written to the file head.out.

[vagrant@localusers vagrant]$ cat head.out

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

[vagrant@localusers vagrant]$

* **Writing standard output to one file and standard error to another file:**

[vagrant@localusers vagrant]$ **head -n1 /etc/passwd /etc/hosts /etc/fakefile 1>standard\_output.out 2> standard\_error.err**

[vagrant@localusers vagrant]$ cat standard\_output.out

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

[vagrant@localusers vagrant]$ cat standard\_error.err

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

[vagrant@localusers vagrant]$

* **Appending standard output to one file and standard error to another file**

[vagrant@localusers vagrant]$ **head -n1 /etc/passwd /etc/hosts /etc/fakefile 1>**

**>standard\_output.out 2>> standard\_error.err**

[vagrant@localusers vagrant]$ cat standard\_output.out

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

[vagrant@localusers vagrant]$ cat standard\_error.err

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

[vagrant@localusers vagrant]$

* **Writing standard output and standard error together to one single file using "&" - (OLD METHOD)**
  + For that, we use "2>&1". Usually, the syntax is --> FD> filename but in this case, since we are not using filename but file descriptor so, we separate them using the "&" operator and use FD>&FD. In this way, the standard output is already being redirected using the "1>" to the file standard\_output.out\_err but, standard\_error is getting redirected to standard\_output which is then inturn, getting redirected to be written to the file standard\_output.out\_err.
  + Syntax and Output shown below:

[vagrant@localusers vagrant]$ **head -n1 /etc/passwd /etc/hosts /etc/fakefile > standard\_output\_error.out\_err 2>&1**

[vagrant@localusers vagrant]$ cat standard\_output\_error.out\_err

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

* **Writing standard output and standard error together to one single file using "&" - (NEW METHOD)**

[vagrant@localusers vagrant]$ **head -n1 /etc/passwd /etc/hosts /etc/fakefile &> head.both**

[vagrant@localusers vagrant]$ cat head.both

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

[vagrant@localusers vagrant]$ head -n1 /etc/passwd /etc/hosts /etc/fakefile &>

> head.both

[vagrant@localusers vagrant]$ cat head.both

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

==> /etc/passwd <==

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

==> /etc/hosts <==

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

head: cannot open ‘/etc/fakefile’ for reading: No such file or directory

[vagrant@localusers vagrant]$

* Piping Standard Output to (stdin) cat command and "-n" switch to indexing the lines

Syntax:

[vagrant@localusers vagrant]$ **tail -4 /etc/passwd /fakefile | cat -n**

tail: option used in invalid context -- 4

[vagrant@localusers vagrant]$ tail -n4 /etc/passwd /fakefile | cat -n

tail: cannot open ‘/fakefile’ for reading: No such file or directory

1 ==> /etc/passwd <==

2 tedison:x:1005:1006:Thomas Edison:/home/tedison:/bin/bash

3 jlocke:x:1006:1007:John Locke:/home/jlocke:/bin/bash

4 brussell:x:1007:1008:Bertrand Russell:/home/brussell:/bin/bash

5 philapp:x:1008:1009:Philosophy Application User:/home/philapp:/bin/bash

* Piping Standard Output and error to cat command and "-n" switch to indexing the lines
* OLD METHOD:

[vagrant@localusers vagrant]$ **tail -n4 /etc/passwd /fakefile 2>&1| cat -n**

1 ==> /etc/passwd <==

2 tedison:x:1005:1006:Thomas Edison:/home/tedison:/bin/bash

3 jlocke:x:1006:1007:John Locke:/home/jlocke:/bin/bash

4 brussell:x:1007:1008:Bertrand Russell:/home/brussell:/bin/bash

5 philapp:x:1008:1009:Philosophy Application User:/home/philapp:/bin/bash

6 tail: cannot open ‘/fakefile’ for reading: No such file or directory

* NEW METHOD:

[vagrant@localusers vagrant]$ **tail -n4 /etc/passwd /fakefile |& cat -n**

1 ==> /etc/passwd <==

2 tedison:x:1005:1006:Thomas Edison:/home/tedison:/bin/bash

3 jlocke:x:1006:1007:John Locke:/home/jlocke:/bin/bash

4 brussell:x:1007:1008:Bertrand Russell:/home/brussell:/bin/bash

5 philapp:x:1008:1009:Philosophy Application User:/home/philapp:/bin/bash

6 tail: cannot open ‘/fakefile’ for reading: No such file or directory

* If we do not want to show the output/error on screen or save it to a file, we can store every output/error inside "/dev/null". This is used when we want the user to run the script but doesn’t want them to see the output or the error codes.
* To check if such a script ran successfully, we will just use the command echo "${?}". If it failed to run then it would return 1 exit status otherwise, if it ran successfully it will return 0 exit status.