Exercise 7 – Data Streams

1. Review questions
2. What is the file descriptor number for standard error?

2

1. What is the difference between the redirection symbols **>** and **>>**?

A single chevron overwrites (clobbers) the file, a double chevron appends the file.

1. What is the difference between the redirection symbols **<** and **<<**?

A single chevron to the left overwrites standard input stream. A double chevron are followed by a label which terminate the document.

1. Why **cmd 1>file 2>file** is not a good idea ?

The standard error is not synchronised with standard output. In this case the two methods will overwrite each other.

1. What are available in BASH to merge standard output and standard error streams ?

**ls /usr/src /boot > out.txt 2>&1**

1. How do you protect files from being overwritten through redirection ?

Turn on noclobber. 🡪 set –o noclobber

1. Experiment with redirection

This exercise will help to clarify the concept of data streams: where the commands take their input from, and what happens to the data they generate as output.

The **cat** program normally opens and reads an existing file, whose name is provided as an argument, and writes the contents to stdout.

a) Use **cat** to read the contents of the **/etc/passwd** file and store it in a file **result0** in your home directory. Display the **result0** file to confirm it worked.

b) If no argument is supplied to the **cat** program it will read from stdin.

Use **cat** to create a new file called **result1** and interactively put some data into it, using **^D** (<CTRL>D) to signify End-Of-File from the keyboard.

1. We will now use the same method as above to create and edit several files. We will then see how we can use the **cat** command to perform a merge function.

Create three files (file1, file2 and file3) with:

$ **cat > file1**

**This is file1**

**^D**

$ **cat > file2**

**This is file2**

**^D**

$ **cat > file3**

**This is file3**

**^D**

Now try the following commands:

$ **cat file\***

$ **cat file\* > all\_files**

$ **cat all\_files**

1. More experiments with redirection

a) This is another “who does what and when” exercise. Try the following examples and consider principles behind them. Who opens the file, when and with what consequences?

The following two commands have the same effect. However, what was the difference at execution time?

$ **cat file1**

$ **cat < file1**

**The second command looks through file1 first and then uses cat. The first command reads through cat and then finds what needs to use cat.**

1. Can you see why the second of the following two commands does not work?

$ **cat file\***

$ **cat < file\***

Too many inputs in the second command. Linux cannot decide which file to put through the command.

1. More about redirection and sequence of events.
2. Try the following commands:

$ **cat file1**

$ **cat file1 file2 > file1**

$ **cat file1**

After all three commands have been applied, can you explain the contents of **file1**?

The second command overwrites the contents of file1 with the contents of file2.

1. How would you add the contents of **file2** onto the end of the existing data in **file3**?

cat file2 >> file3