Linux Redirection

1 Part One

You will need to be able redirect output during program execution, to the screen. This can be done using the Linux operators '`>`' and '`>>`'.

- '`>`' Redirect the output creating a new file. Will overwrite an existing file.
- '`>>`' Redirect the output to a file. Will append output to an existing file else will create a new file.

For example.

```
steveb exercise3> f90 -o test test.f90
steveb exercise3> test > testout.txt
steveb exercise3> f90 -o test2 test2.f90
steveb exercise3> test2 >> testout.txt
```

1. First we will compile '`test.f90`' to create the executable '`test`'
2. Then we run the executable '`test`' and instead of sending the output to the screen, it is instead sent to the file '`testout.txt`' in the same directory.
3. Then we will compile '`test2.f90`' to create the executable '`test2`'.
4. Then we run the executable '`test2`' and instead of sending the output to the screen, it is instead **appended** to the file '`testout.txt`'.

You can inspect the file '`testout.txt`' by simply opening it with the '`nano`' editor.

```
steveb exercise3> nano testout.txt
```

2 Part Two

Note that when using the redirection operators '`>`' and '`>>`' to redirect output from your codes you will not actually see any output to the screen. This is because all the output from the code is being redirected to a file. This can be a problem if during the execution of the code you need to enter information (ie. a '`READ`' statement in the code), as you will not see any prompts asking the user to input the information.

Remember that when using the redirection operators '`>`' and '`>>`' you are redirecting output from the screen to a file. The '`PRINT*`' statements to prompt the user for input are being redirected to that file so you do not see them on the screen. You can however just carry on as normal and simply type in the information as usual after starting the executable without seeing the prompts on the screen.
As an alternative to the '`>`' and '`>>`' operators you can make use of the Linux '`tee`' command which does send the code output to both the screen and the file. You can use it as follows.

```
steveb exercise3> f90 -o test test.f90
steveb exercise3> test | tee testout.txt
steveb exercise3> f90 -o test2 test2.f90
steveb exercise3> test2 | tee -a  testout.txt
```

The '`|`' operator in linux is called a pipe and is simply a line of communication. It pipes the output from the code '`test`' into the '`tee`' command and the '`tee`' command writes the output to the file and also to the screen at the same time.

The '`-a`' argument to '`tee`' in the '`test2`' code line tells the '`tee`' command to append the output to '`testout.txt`' instead of overwriting it.