I/O redirection

- Most of the time, programs display their output (stdout) to the monitor and take their input (stdin) from the keyboard
- There is also an error stream (stderr) that some programs take advantage of
- From the command line, you are able to seperately redirect where a program’s stdin comes from, and/or where its stdout goes to, and/or where its stderr goes to

Redirecting stdin with <

- If we want a program to read its input from a file instead of the keyboard, we can use “program < filename”
- The program will read sequentially through the file, treating whitespace in the file in the exact same way as if the user had entered it
- Of course, the content of the file needs to be consistent with what the program expects/can handle

Redirecting stdout with >

- Similarly, we can send a program output to a file instead of the monitor, e.g. “progname > filename”
- If the file doesn’t exist yet then this will create it, otherwise it will replace the old version of the file
- Of course, if the program goes into an infinite loop while producing output then you’re going to run into quota issues...
- Combinations are also possible, e.g. “prog < infile > outfile” to read from the first file and write to the second

Piping output between programs

- We can use the output from one program as the input to another using the pipe (vertical bar), e.g. “prog1 | prog2”
- This can also be combined with > and <, e.g.

  prog1 < infile | prog2 > outfile

  runs prog1 using data from infile, sends its output to prog2 to be used as input there, then prog2’s output goes to outfile

Redirecting stderr

- The > is used to redirect stdout
- We can instead redirect stderr using 2> or redirect both of them using &>

```
prog > file # stdout goes to file, stderr goes to screen

prog 2> file # stderr goes to file, stdout goes to screen

prog > file # both go to file

prog # both go to screen
```