linux standard streams

- linux provides a standard stream for text input, referred to as stdin. (This stream is used by cin and scanf.)
- linux provides a standard stream for text output, referred to as stdout. (This stream is used by cout and printf.)
- linux also provides a second output stream for text output, stderr, that is conventionally used to display error messages. (This stream is used by cerr.)
- scripts, programs, and linux users are able to redirect the two output streams to different locations if they wish (perhaps storing standard output in a file and displaying error messages on screen)

Example using the two streams

- access to both is provided in iostream
  - cout << “blah blah blah”; // goes to stdout
  - cerr << “blah blah blah”; // goes to stderr
- by default, both streams go to the terminal window, with no visual distinction for the user
- output redirection, used from the linux command line, does allow us to pick/choose what goes where

One typical use in error handling

```
// get user to keep re-entering values until one is valid
int  quantity = -1;
do {
   cout << “Enter the number of items desired as an integer:” << endl;
   cin >> quantity;
   if (cin.fail()) {                    // cin failed: what they entered couldn't have been an integer
      cin.clear();                      // gets rid of an error setting cin has stored
      cin.ignore(LineLen, '\n');   // discards the garbage value still sitting in the input buffer
      cerr << “That was not an integer, please try again” << endl;
   }
   else if (quantity < 0) {         // it was an integer, but a negative one
      // (note: no need to do a cin.clear or ignore since it successfully read an int)
      cerr << “The value cannot be negative, please try again” << endl;
   }
} while (quantity < 0);
```

Redirecting I/O

- from the command line we can redirect standard output to a file using > (the stderr still goes to the screen)

  ./myprogram > somefile

- we can also redirect file content to standard input (so that cin reads from the file instead of the keyboard)

  ./myprogram < anotherfile

- we can also combine the two

  ./myprogram < anotherfile > somefile

Redirecting stderr

- we can redirect standard err (e.g. the cerr output) using 2>, in which case the standard output still goes to screen

  ./myprogram 2> somefile

- we can redirect both to a file using &>

  ./myprogram &> somefile

- we can also redirect the output of one program to be used as input to another using the pipe |

  ./firstprogram | secondprogram