I/O Redirection

Most of us enjoy using a graphical user interface in a windowed environment. However, command interpreters, such as those provided by MS-DOS and UNIX allow some powerful techniques. One of these is I/O redirection. Both input and output can be redirected.

Redirection of Output

If at the MS-DOS prompt, you enter "`DIR A:`", the names of files in the `A:` drive are written to the console (monitor). However, this output can be redirected to a file via the command

```
DIR A: > MyFile.txt
```

This creates a file named "`MyFile.txt`" in the current working directory.

Now suppose you wish to print the list of filenames on `A:`. You could use the command

```
DIR A: > PRN
```

Note that ordinary output redirection overwrites files. For example, "`DIR A: > MyFile.txt`" replaces any previous file named "`MyFile.txt`". There is a form of redirection that preserves the old file, appending the new output. For example,

```
DIR C: >> MyFile.txt
```

will append the directory listing of `C:` to any previously existing file, "`MyFile.txt`".

Redirection of Input

Many DOS and UNIX programs are written so that by default, they take their input from "standard input", which is usually the keyboard. Input can be redirected via "<" to come from a file. As an example of how this is useful, consider the apparently worthless program below.

```
#include <iostream>
using namespace std;
void main()
{
  char Ch;
  cin.get(Ch);        //read first char from input stream
  while (cin.good())  //stop when EOF detected or error occurs
  {
    cout << Ch;       //display the last character read
    cin.get(Ch);      //read next char from input stream
  }
}
```

By default, the program simply reads chars from the keyboard and echoes them to the screen. However, with redirection, the program can be used to view files or even make a copy of a file! Suppose that the code above is compiled to create an executable named "`redir.exe`". Then to view a file, for example a file called "`words.1`", which is a text file, you use at command line:

```
redir < words.1
```

The program will run and the operating system will automatically feed the characters in the file "`words.1`" to the program. The statement "`cout << Ch;`" will then output the chars to the screen.