File input

Ch 6

Recap: File I/O

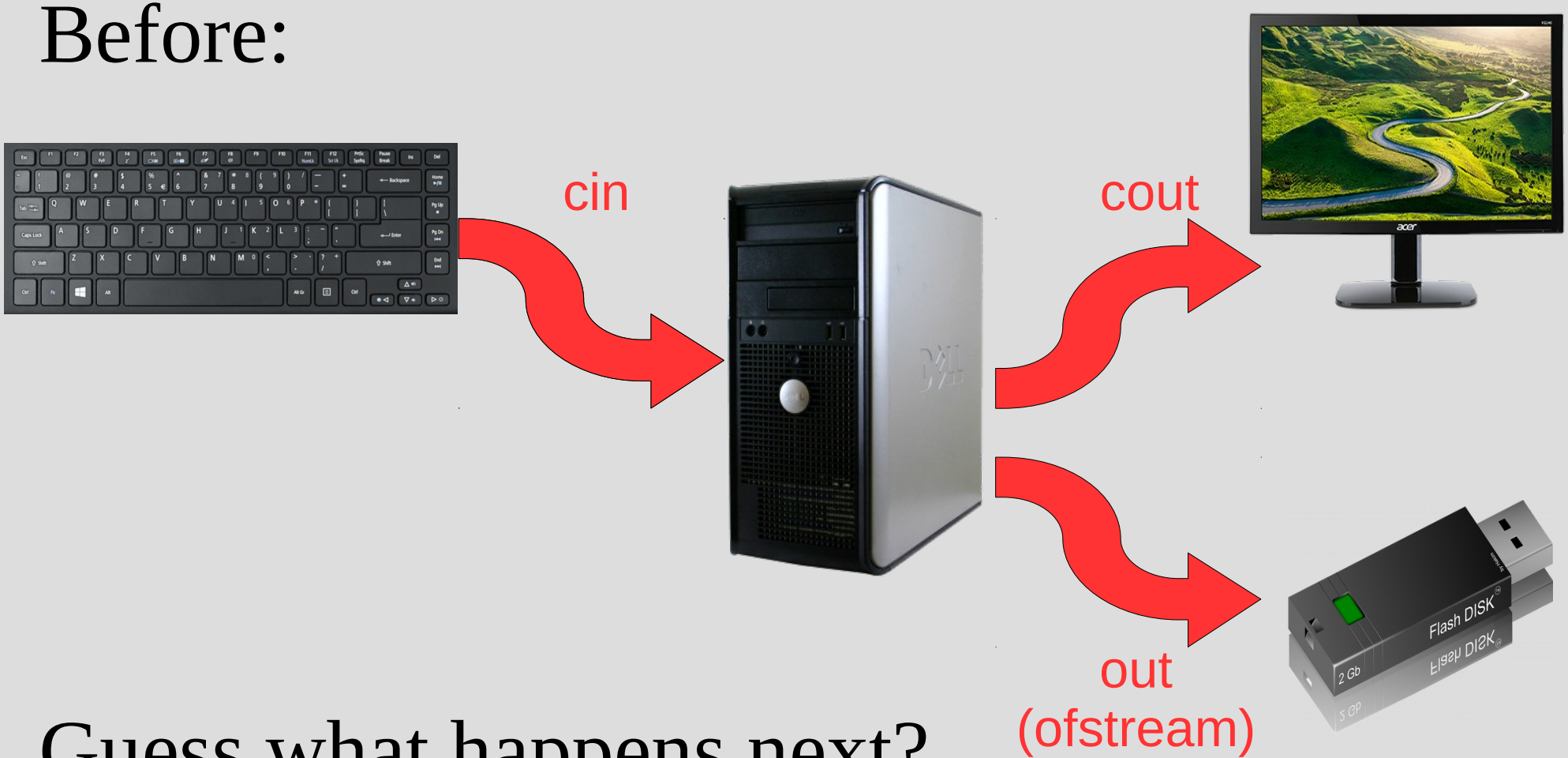
File I/O is basically the same as cout/cin except to a file and not terminal

Why use?

1. Files exist until deleted, terminal exists until program stops (i.e. grades no go away)
2. Easier to get info in/out of program (remember temperature lab problem?)
3. Can store hard computation for fast look-up

Writing to a file

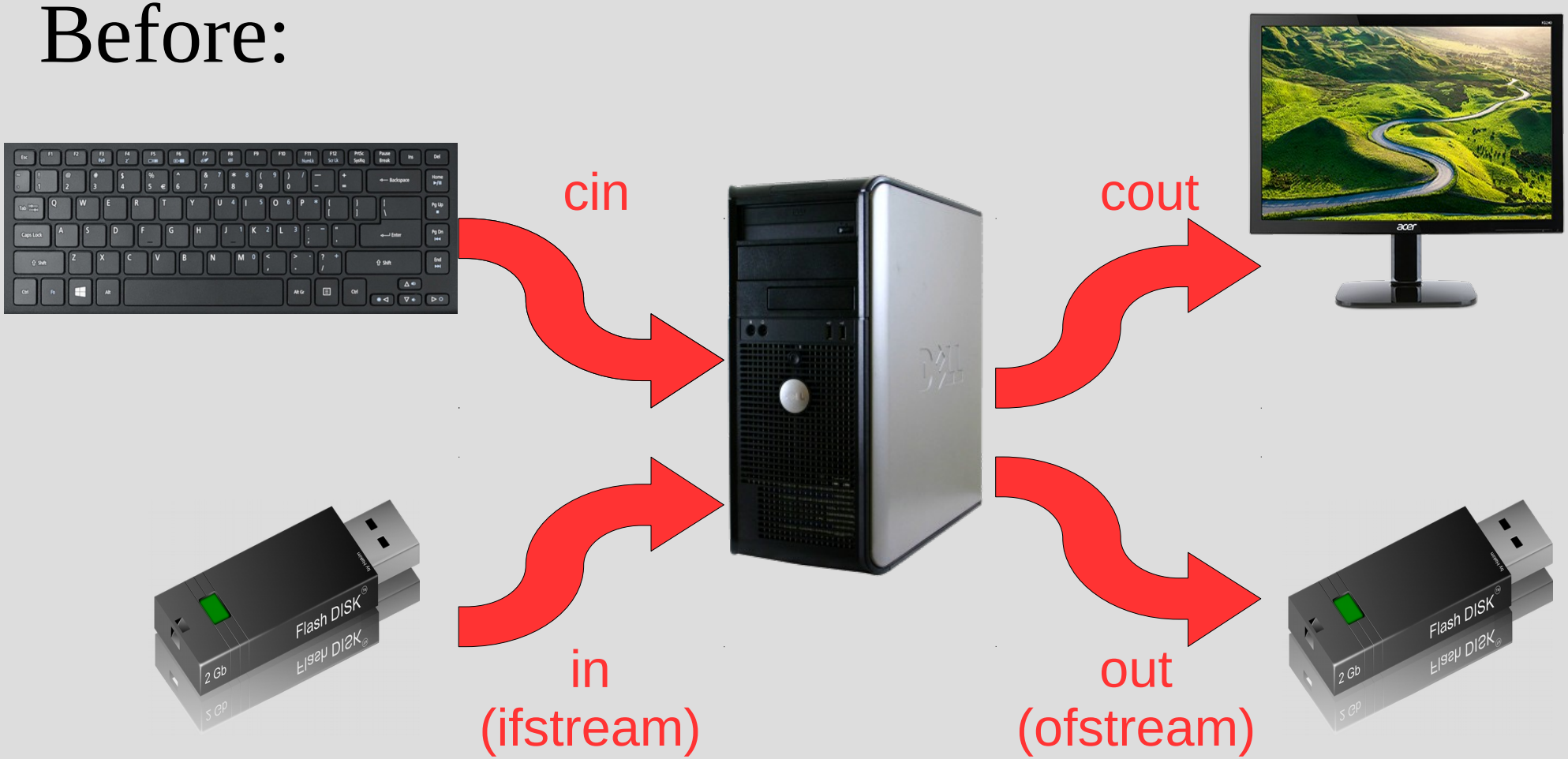
Before:



Guess what happens next?

Writing to a file

Before:



File input

Input is similar to output, we need to open a stream then use it similar to cin

```
string x;
ifstream in;
in.open("input.txt");
if(!in.fail())
{
    in >> x;
}
in.close();
```

(See: fileInBasics.cpp)

What is a major difference between reading and writing to a file?

End of file (EOF)

When there is nothing left in a file to read, we call it end of file

C++ is fairly nice about handling EOF, and you can detect it in 3 ways:

```
while(getline(in,x))
while(in >> x)
while(!in.eof())
```

reads from file

does not read from file (just tells if at end)

End of file (EOF)

Reading from file can be a bit tricky as the end of file is not detected until you try to read but **then fail** because there is nothing there

This can cause issues with counting the last input twice

To avoid this, make sure you right after “in >> var” you check if EOF
(see: fileInput.cpp)

Formatting

You can use also use setf() on your streams, but you can also use setw() and setprecision() from <iomanip>

setw(x) reserves x spaces (right justify)

```
ofstream outFile;
outFile.open("fancyOutput.txt");
outFile.setf(ios::showpoint);
outFile.setf(ios::fixed);
//outFile.unsetf(ios::fixed); //undoes above
outFile << "$"<< setprecision(2)<<setw(8) << 23.61 << endl;
```

(See readTable.cpp)