File output

Ch 6

Highlights

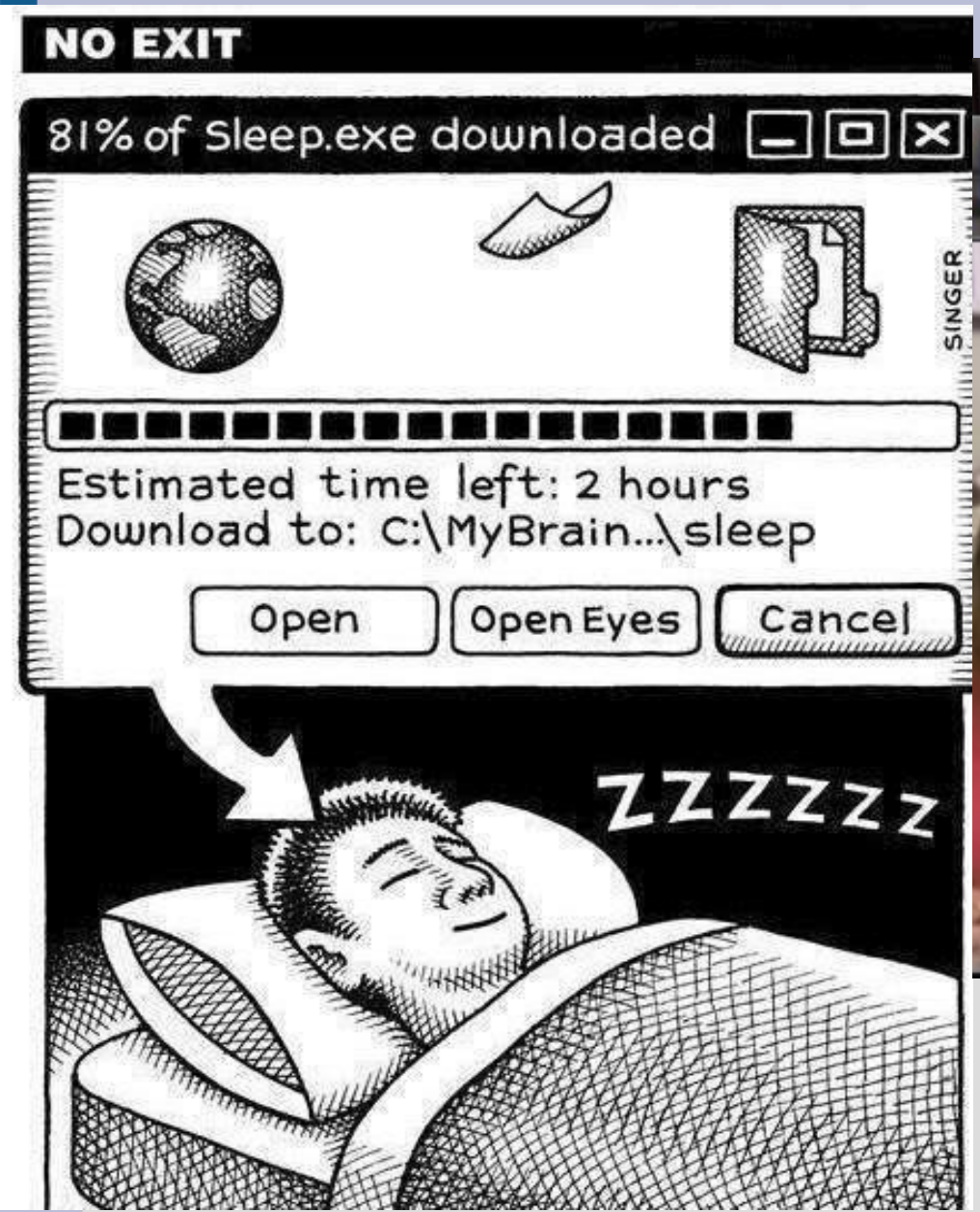
- text file output

```
ofstream out;
out.open("output.txt");
```

- text file input

```
ifstream in;
in.open("input.txt");
```

Download vs stream



Streams

A “stream” is information flow that is immediately processed

For example:

- Streaming video is watch as data arrives
- Downloading video stores it for later

For file input/output (file I/O), we will have to create a stream between file and code

Data persistence

The temperature decay problem from last lab had multiple inputs (annoying to re-enter)

What if you had a large amount to input to your program?

100 inputs?

1,000,000 data points for predicting weather?

Data persistence

Files are also nice, as you can look them up at a later time

After your program output ends,
the text disappears (unless you re-run it)

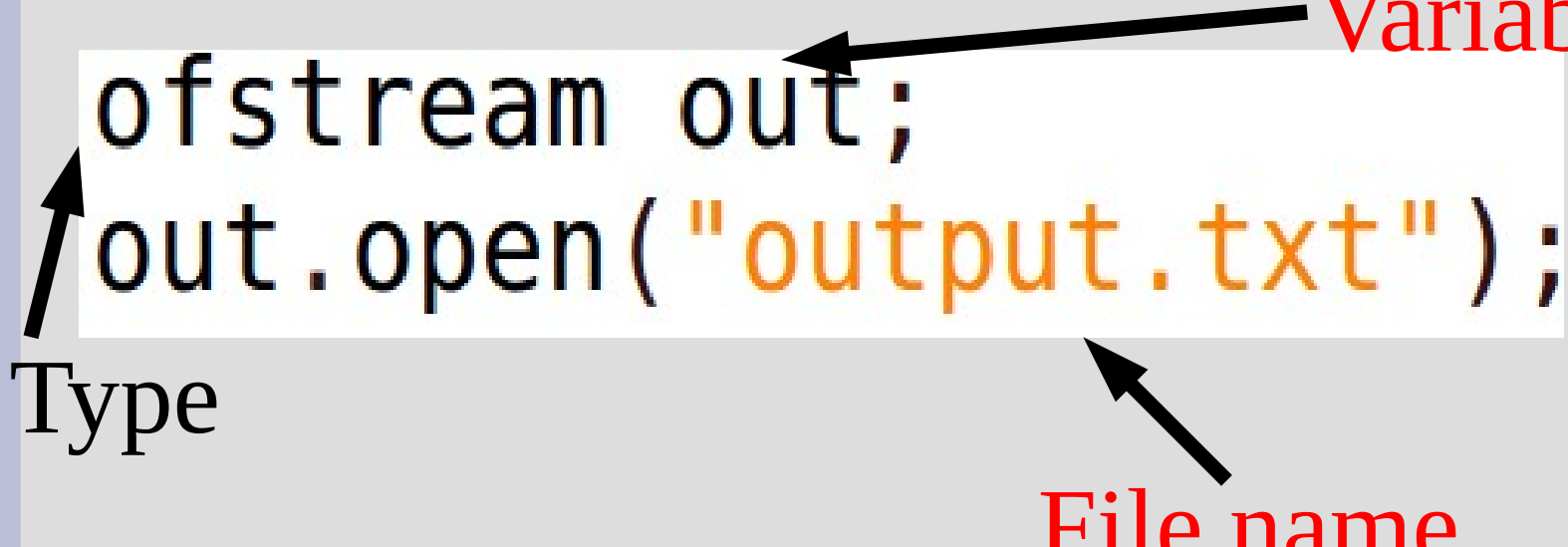
Files stay on your computer forever
(until comp dies)

“Opening” a file

File output is very similar to terminal output, except we have to open and close files

To create a stream between a variable name and file name:

```
ofstream out;
out.open("output.txt");
```

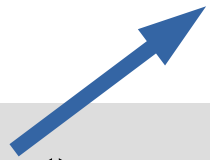


“Opening” a file

Sometime you cannot open a file (don't have permission)

You can check if the file actually opened by calling fail() (returns true if did NOT open):

```
if(out.fail())
{
    exit(1); // non-zero for an error state
}
```



exit() in <cstdlib>, causes program to terminate

Writing to a file

After you have opened a file (stream), you can then write to it

This is done in an almost as cout, except you use the your variable name for the file

Terminal: `cout << "Hello!\n";`

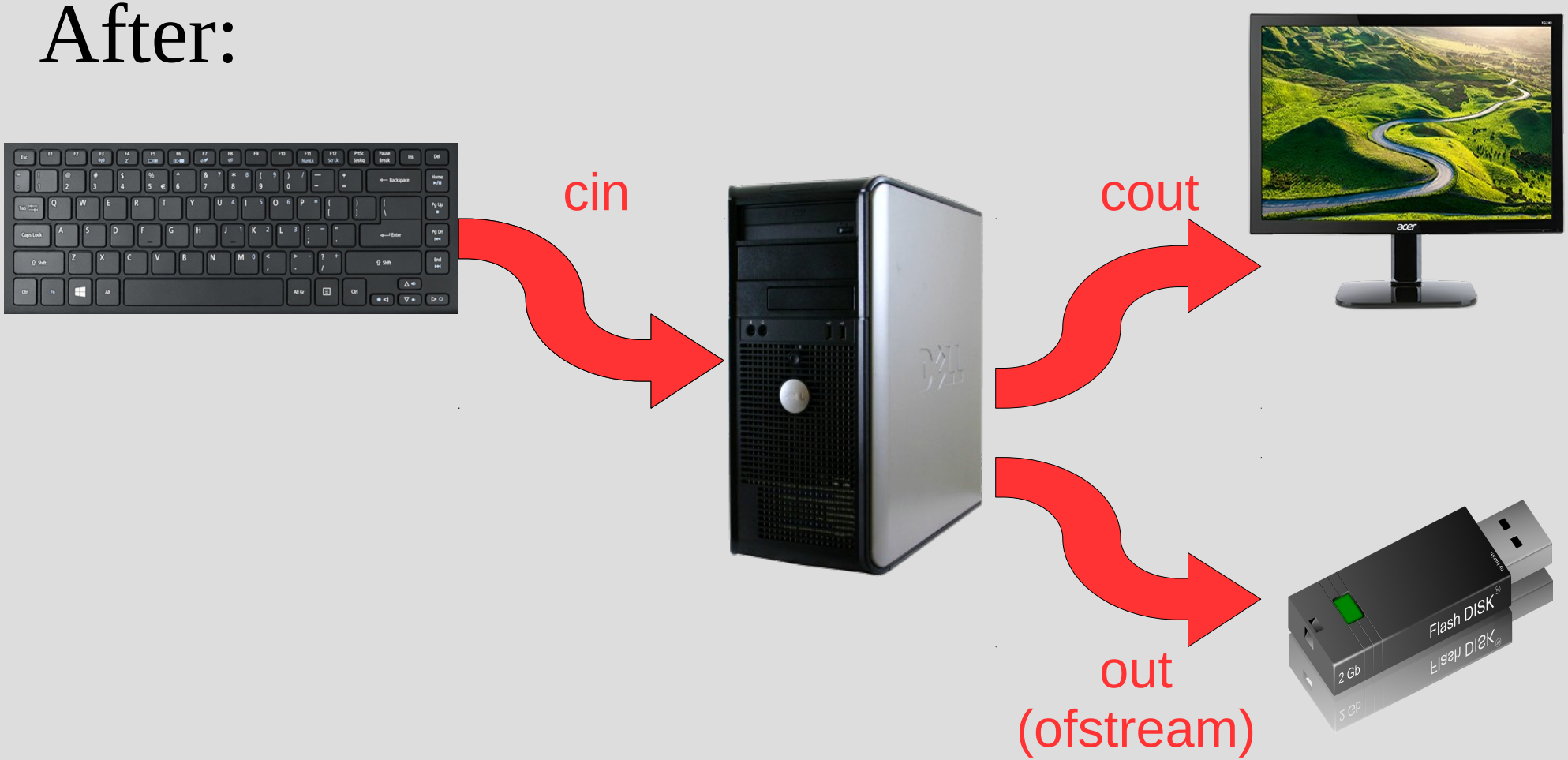
File: `out << "Hello!\n";`

Writing to a file

Before:

Writing to a file

After:



File output imports

To use ofstream type, you need to include <fstream>

```
#include <fstream>
```

This gives you ofstream (output file stream) and ifstream (input file stream), which we will see next

(See: helloWorldFile.cpp)

Closing a file

Once we are done writing to a file, we should close the stream

This is an extremely complicated process:

```
out.close();
```

Variable name

If you don't close your stream, something might be left in the buffer

Clo

(See: needClose.cpp)

```
                id=43571
                type=50
        }
}
-166={

        name="Waldst\E4tte"
        state=yes
        owner="SAV"
        controller="SAV"
        previous_controller="ADU"
        estate=1
        last_estate_grant=1578.6.20
        core="SAV"
        trade="rheinland"
        culture=swiss
        religion=catholic
        original_religion=catholic
        capital="Schwyz"
        is_city=yes
        base_tax=1.000
        original_tax=3.000
        base_production=1.000
        base_manpower=2.000
        likely_rebels="nationalist_rebels"
        trade_goods=iron
        local_autonomy=0.000
        min_autonomy=25.000
        max_autonomy=100.000
        marketplace=yes
        regimental_camp=yes
        history={
                owner="SWI"
                controller={
                        tag="SWI"
                }
                culture=swiss
                religion=catholic
                capital="Schwyz"
                trade_goods=iron
                hre=yes
                base_tax=4.000
                base_production=4.000
                base_manpower=2.000
```

Make sure I own...

Remove this line

Where did this file go?

The default “path” for a file is where your cpp file is located

You can specify the path when you open the file:

```
out.open("/home/park0580/PutItHere.txt");
```

You can also use relation operations:

```
out.open("../PutItHere.txt");
```

Appending to files

What happens if I run HelloWorldFile multiple times?

Open file and override:

```
out.open("output.txt");
```

Open file and append:

```
out.open("output.txt", ios::app);
```

(See: helloWorldFileAppend.cpp)

File writing overview

- You need to open a file before writing to it
- You should close the file when you are done
- You can either override or append to files
- Use .fail() to see if file actually opened

- You cannot go backwards and “replace” or “undo”
- You cannot “preppend” to a file (must either append from end or override)

Caution!

Be careful about writing an infinite loop while outputting to a file

You will very quickly run out of hard drive space

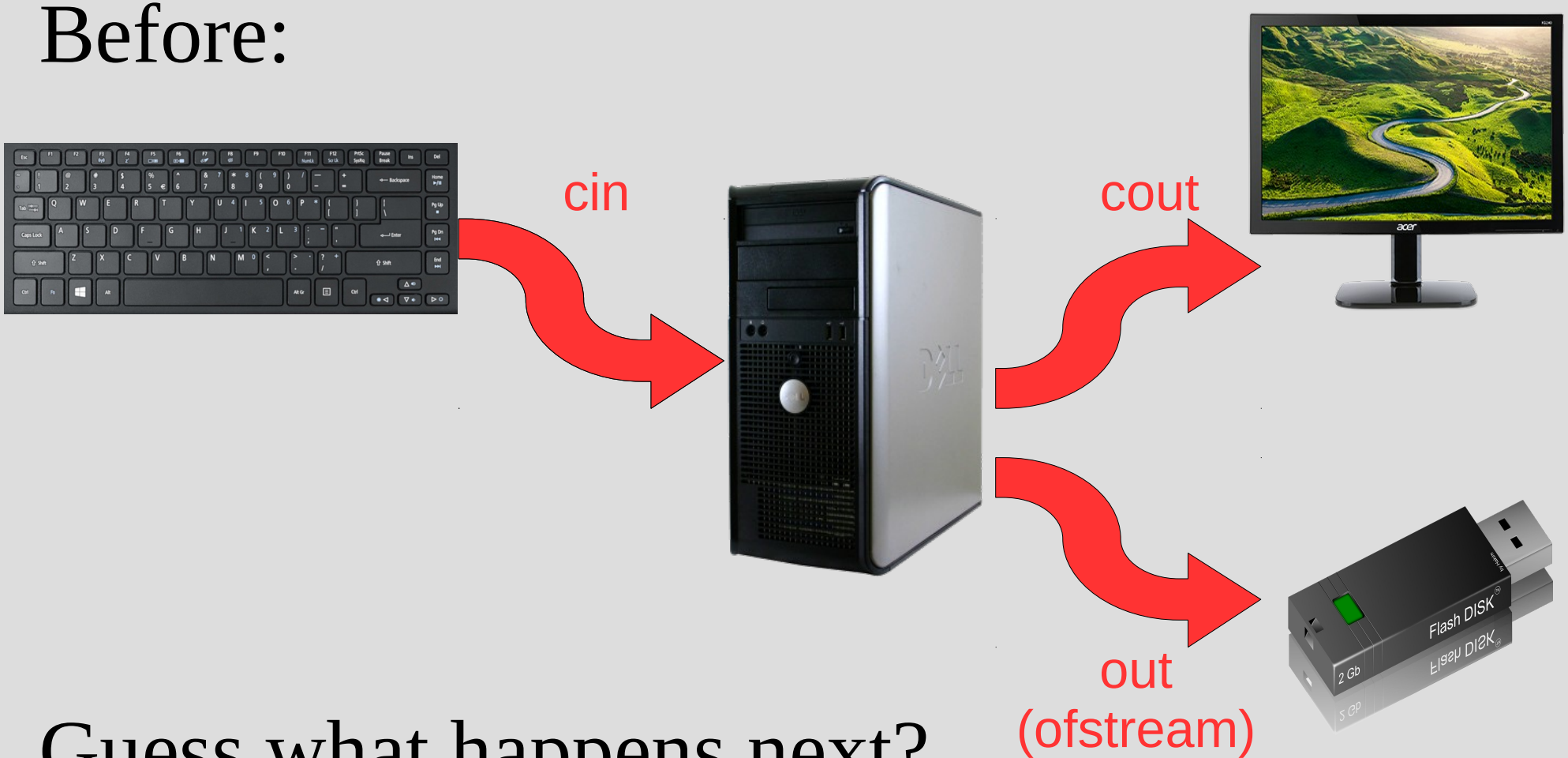
If you think it is stuck in an infinite loop, press ctrl+c to kill the program (from the window)
(see: nomNomHD.cpp)
https://www.youtube.com/watch?v=_95I_1rZiIs

File input

Ch 6

Writing to a file

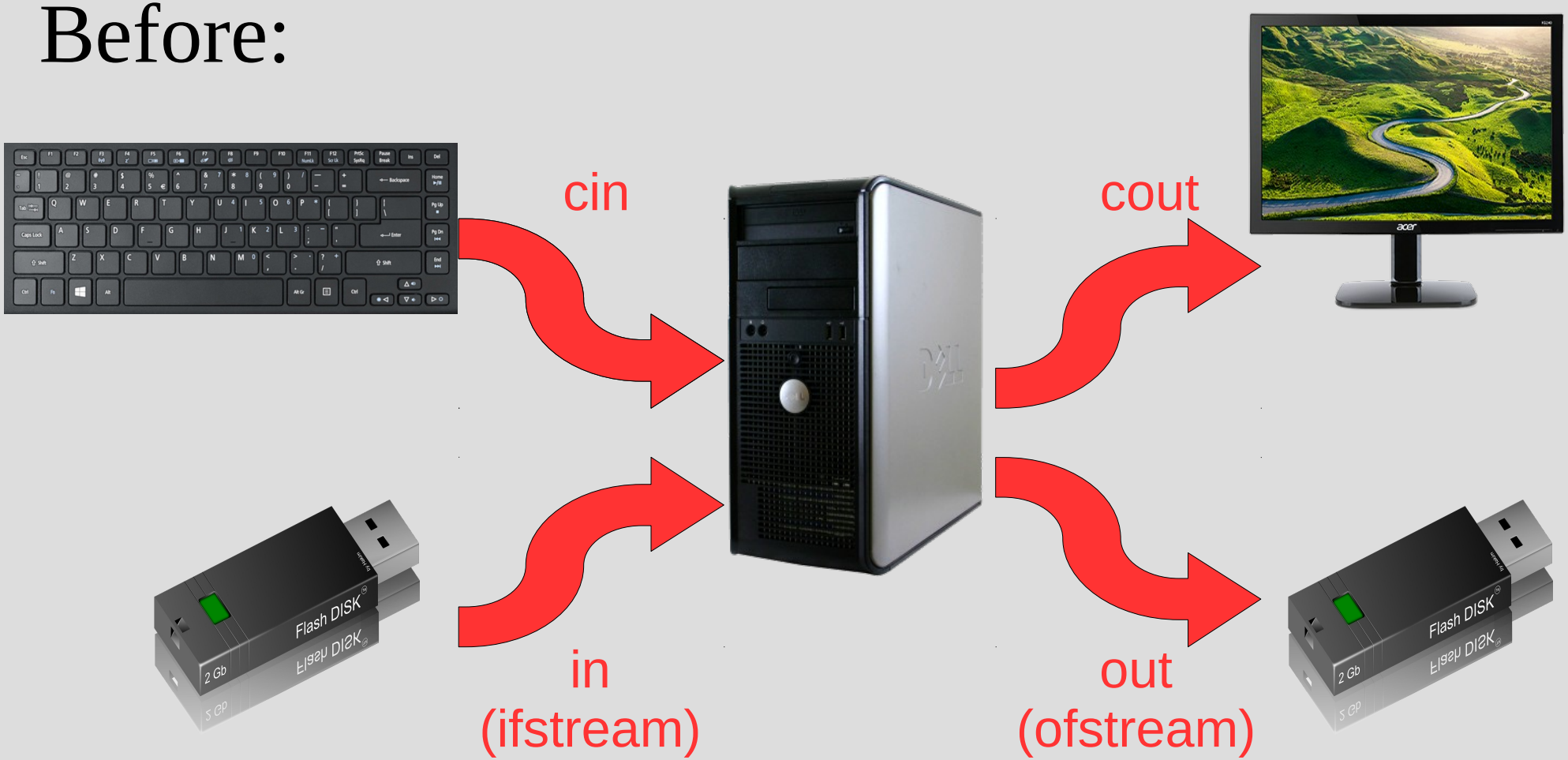
Before:



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 <u>end of file</u>

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)