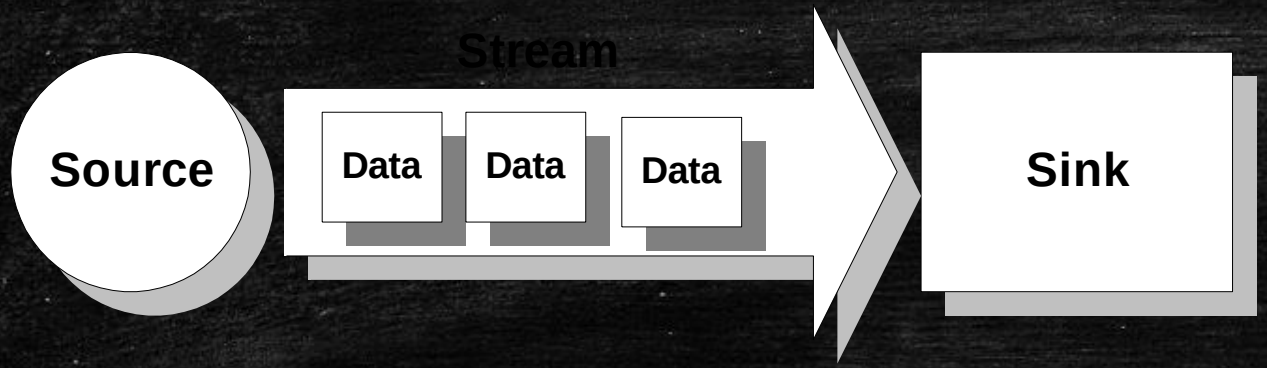
Streams & Filters

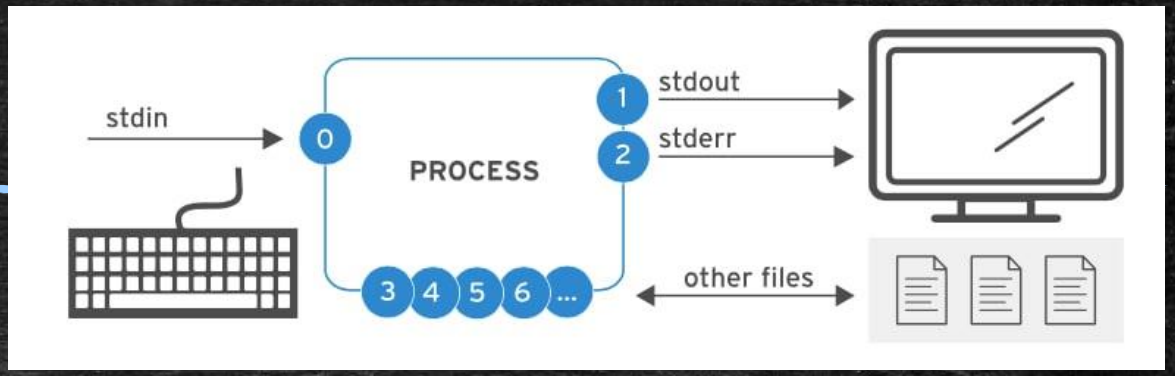
CS 150 – C++ Programming I
Lecture 11

What is a Stream?

- Streams are abstract data flows
  - Data flows into program from a source
  - Your program processes the data
  - Information flows out to a sink
- Sending info to a sink is called writing
- Getting info from a source is reading



Stream Classes



- Your operating system has three global stream objects: *stdin*, *stdout* and *stderror*
- C++ has specialized classes which do input and output
  - istream: objects which know how to read information
  - ostream: objects which know how to write information
- C++ automatically creates several global stream objects
  - cin, (input *istream* object wrapping *stdin*)
  - cout (wrapping *stdout*), cerr (wrapping *stderr*)

Raw Character I/O with Standard Streams

- *cin* and *cout* are objects so they have member functions
  - Use `cin.get(ch)` to read a single character from *stdin*
  - Use `cout.put(ch)` to write a single character to *stdout*
  - Use `cerr.put(ch)` to write a single character to *stderr*
  - More efficient than using insertion or extraction operators
- To echo every character from stdin, use a data loop:
  - ```
    char ch;
    while (cin.get(ch))
        cout.put(ch);
    ```
  - Stops when there is no more data to process
    ```

get() and put() Member Functions

- Both `get()` and `put()` return the stream used for I/O
- Reading a character: `istream& get(char& c);`
  - Output parameter. Must be a variable!!!
  - `while(cin.get(ch)) …`
- Writing a character: `ostream& put(char c);`
  - Input (value) parameter (which is convertible)
  - `cout.put('A');`
  - `cout.put(65);`

The Basic Text Echo Program

- The basic echo text filter program is:
  - ```
    char ch;
    while (cin.get(ch))
        cout.put(ch);
    ```
  - Reads a character and stores it in the variable `ch`
  - Returns the stream which is `false` when there is no data
- This is called a data loop; it stops when input is exhausted.
  - Console simulate with `Ctrl+D` (`Ctrl+Z` on Windows OS)
  - Use redirection to open and process files
    ```

Sources, Sinks & Redirection

- Operating system facility which connects *stdin*, *stdout* and *stderr* to different files, devices & processes
  - `cout` to a file (overwrite): `./exe > output.txt`
  - Append `cout` to a file: `./exe >> output.txt`
  - `cout` & `cerr` to files: `./exe 1> out.txt 2> err.txt`
  - Both to a single file: `./exe > combo.txt 2>&1`
  - Discard output: `./exe > /dev/null`
  - Program (pipe): `./exe | another-program`
  - Read input from a file: `./exe < input.txt`
- Exercise: modify `cecho` to read `alice.txt`

Different Kinds of Filters

- There are two general categories of filters
  - State filter: examine `ch` for changes
  - Process filter: modify `ch`
- Let's modify `cecho` to examine `ch` for changes
  - Want to print the chapter headings only
  - When we encounter a newline, followed by a `'C'` we'll continue printing until we encounter the next newline
- Notice, it doesn't quite do what we want
  - Nice if we could "look ahead" before we read the character!

Auxiliary Input Methods

- Look at next character in the stream
  - `int peek();` returns `EOF` if encountered
- Put a character back into a stream
  - Supports at least one character
  - Needn't put back same character read
    - `istream& putback(char c);`
    - Can also use `unget()`
- Read and discard unwanted input
  - `istream& ignore(max, delim);`
- Exercise: read from `alice.txt` to `toc.txt`