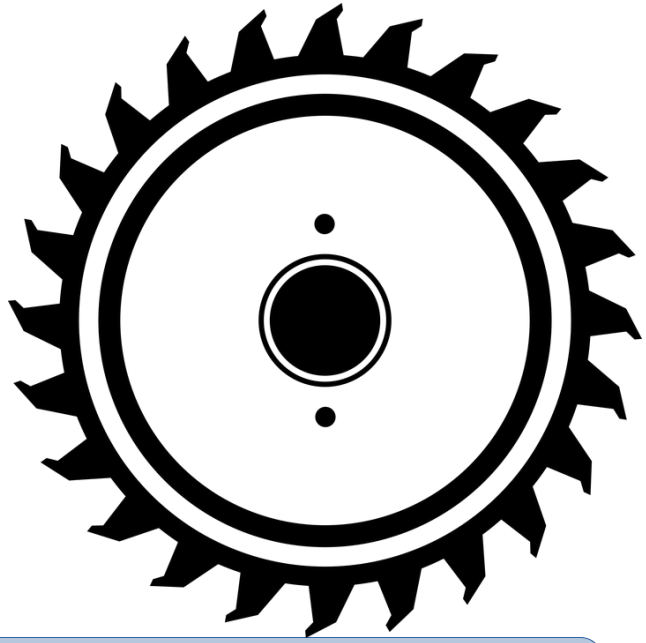
when you write 10 lines of code without searching on Google

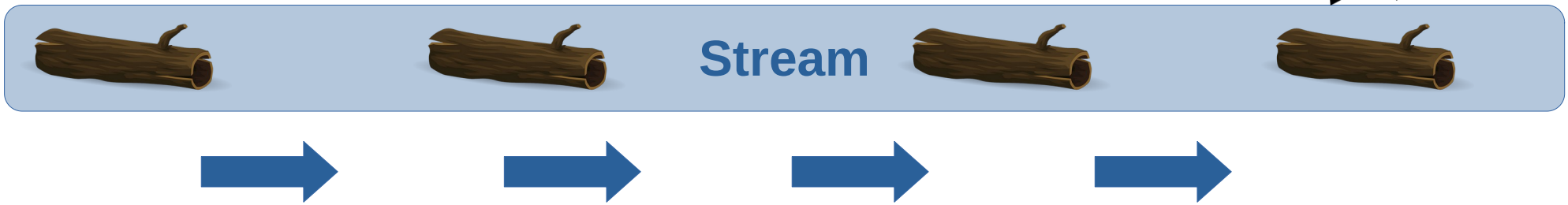
CS1428
Foundation of Computer Science

Lecture 3: Input/Output

Streams:

- Input devices and output devices all have dedicated memory that they write to and read from.
- Programs are not allowed to read or write each other’s memory, so a program reading the memory dedicated to a keyboard or writing to the memory dedicated to a monitor could be a hard problem.
- “Streams” are an abstraction that let us hide the actions taken by the operating system to move values from input devices to programs to output devices.




Stream

The Output Stream:

- **cout** connects your program to the console (the block of text that shows up on your monitor when you run a program).
- The stream insertion operator **<<** puts values into the stream. The arrows will always point into the stream.
- Multiple values can be sent to **cout** with the same line of code, but each value should be separated by another **<<**
- **endl** sends one line break to the console
- There are other output streams but you won’t be required to know them for this class.

Hello World:

```
#include<iostream>

using namespace std;

int main(){
    cout << “Hello, World.” << endl;
    return 0;
}
```

Hello World:

```
/home/gentry/Desktop/1428_testDir/junk
Hello, World.

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

The Input Stream:

- **cin** connects your program to the keyboard through the console.
- The stream extraction operator **>>** takes values out of the stream. The arrows will always point into the stream.
- Multiple values can be read from **cin** with the same line of code, but each variable should be separated by another **>>**
- White space will be read by **cin** as separating strings.

Hello User:

```
#include<iostream>

using namespace std;

int main(){
    string userName;
    cout << "What is your name: ";
    cin >> userName;
    cout << "Hello " << userName << endl;
    return 0;
}
```

Example Output

```
/home/gentry/Desktop/junk
What is your name: Gentry
Hello Gentry
Process returned 0 (0x0)   execution time : 2.773 s
Press ENTER to continue.
```

```
/home/gentry/Desktop/junk
What is your name: Gentry Atkinson
Hello Gentry
Process returned 0 (0x0)   execution time : 3.628 s
Press ENTER to continue.
```

cin With Multiple Inputs:

```
#include<iostream>

using namespace std;

int main(){
    string firstName, lastName;
    cout << "What is your name: ";
    cin >> firstName >>  lastName;
    cout << "Hello " << userName << endl;
    return 0;
}
```

Example Output

```
/home/gentry/Desktop/junk
What is your name: Gentry Atkinson
Hello Gentry Atkinson
Process returned 0 (0x0)   execution time : 3.448 s
Press ENTER to continue.
```

Libraries:

- Libraries are collections of tools that we can add into our programs.
- **cin**, **cout**, and **endl** come from the <**iostream>** library.
- Libraries are included using the **#include** directive
- Each library requires its own **#include**
- Some useful libraries:
  - **iostream**
  - **fstream**
  - **cmath**
  - **iomanip**
  - **string**

Program With Multiple Libraries:

```
#include<iostream>
#include<cmath>

using namespace std;

int main(){
    cout << "2 to the power of 3 is " << pow(2,3) << endl;
    return 0;
}
```

Example Output

```
/home/gentry/Desktop/junk
2 to the power of 3 is 8

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

cin Will Fit Input to the Variable:

```
#include<iostream>

using namespace std;

int main(){
    int a;
    char b;
    cout << "Enter one number and one character: ";
    cin >> a >> b;
    cout << "a is " << a << ", b is " << b << endl;
    return 0;
}
```

Example Output

```
/home/gentry/Desktop/junk
Enter one number and one character: 65g
a is 65, b is g

Process returned 0 (0x0)   execution time : 6.854 s
Press ENTER to continue.
```

```
/home/gentry/Desktop/junk
Enter one number and one character: c88
a is 0, b is y

Process returned 0 (0x0)   execution time : 4.038 s
Press ENTER to continue.
```

Manually Changing Data Types:

```
#include<iostream>

using namespace std;

int main(){
    float a = 7.8;
    cout << "a as a float: " << a << endl;
    cout << "a as an int: " << static_cast<int>(a) << endl;
    cout << "a as a bool: " << static_cast<bool>(a) << endl;
    return 0;
}
```

Example Output

```
/home/gentry/Desktop/junk

a as a float: 7.8
a as an int: 7
a as a bool: 1

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

File Streams:

- Rather than taking input from the keyboard and sending output to the screen, we can also read from and write to files.
- File I/O creates a permanent copy.
- We also use streams to read and write files, which come from **<fstream>**
- Input file streams are created like variables with the type **ifstream**. Output filestreams are created with **ofstream**.
- File streams have to be manually opened and closed.

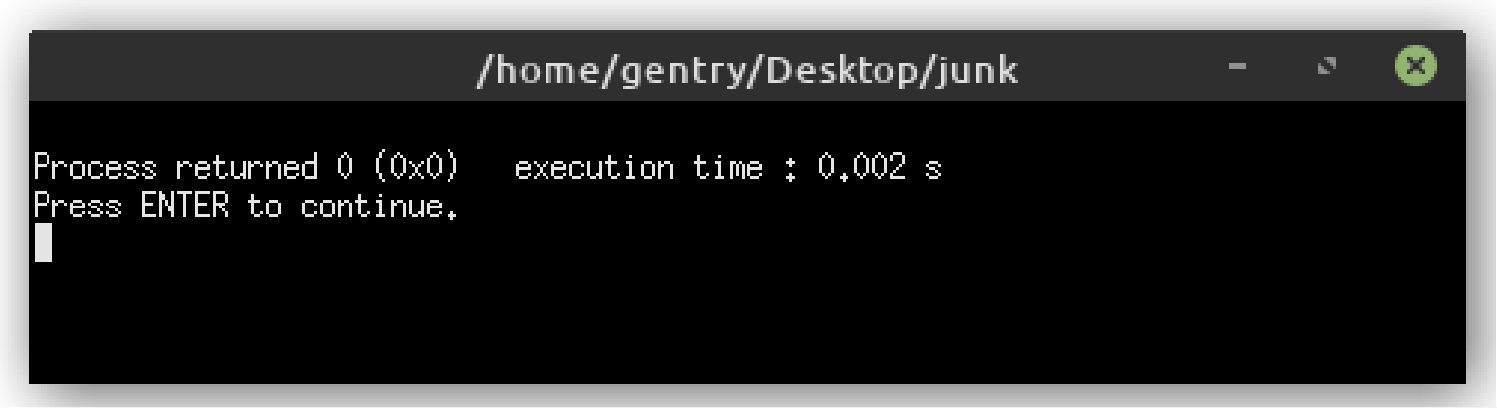
Hello File:

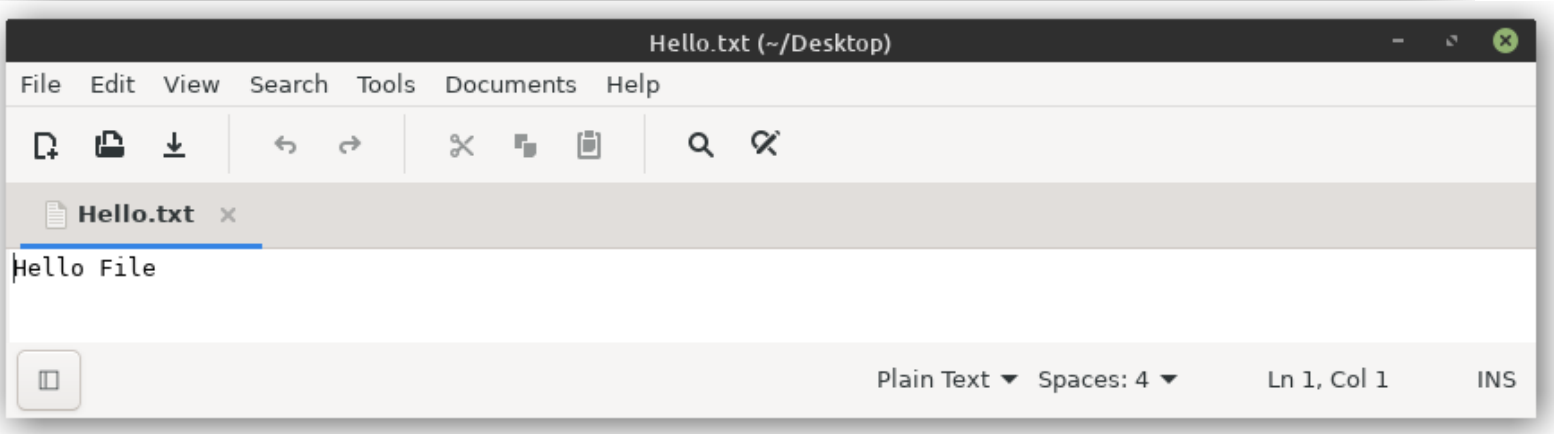
```
#include<fstream>

using namespace std;

int main(){
    ofstream fout;
    fout.open("Hello.txt");
    fout << "Hello File" << endl;
    fout.close();
    return 0;
}
```

Example Output





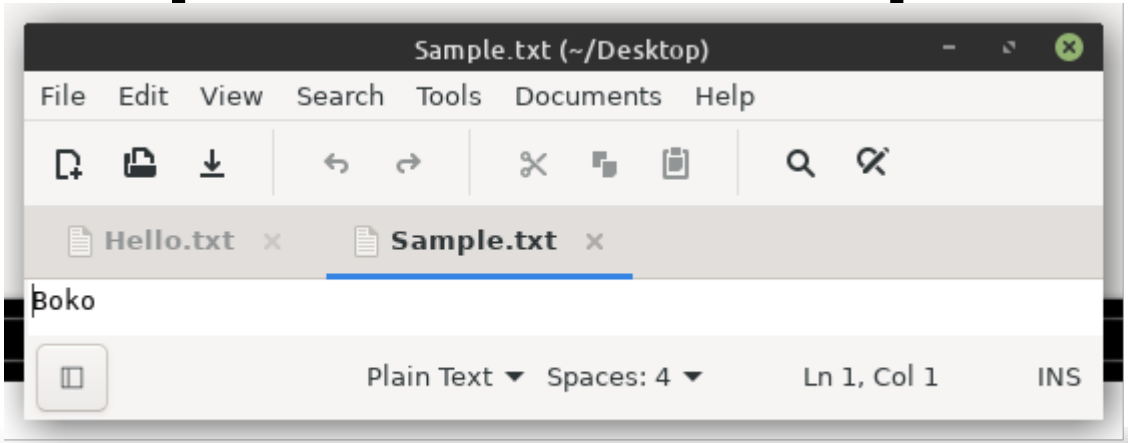
Reading a File:

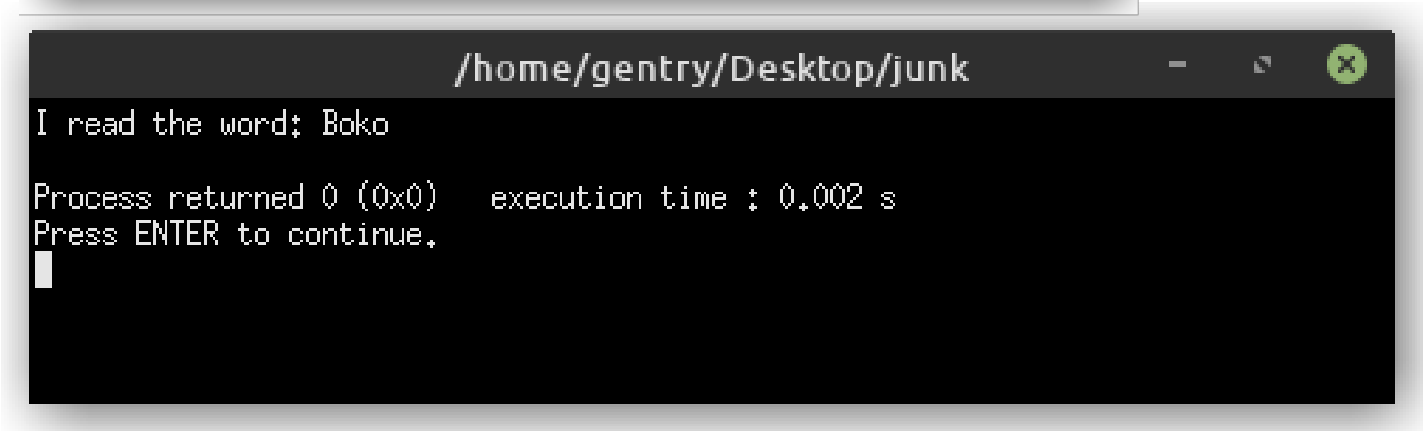
```
#include<fstream>
#include<iostream>

using namespace std;

int main(){
    ifstream fin;
    string input_string;
    fin.open("Sample.txt");
    fin >> input_string;
    cout << "I read the word: " << input_string << endl;
    return 0;
}
```

Example File and Output





Formatting I/O:

- The library **<iomanip>** provides many great tools for making your output easily readable.
- Well formatted text better convey the information your program generates.
- **<iomanip>** is used along with streams, either console or file.
- Tags to now:
  - **setprecision** and **fixed**
  - **setw**
  - **left** and **right**

Aligning Columns:

```
#include<iomanip>
#include<iostream>

using namespace std;

int main(){
    cout << left << setw(20) << "Name:" <<  "Lyndon Johnson" << endl;
    cout << left << setw(20) << "City:" <<  "Austin, TX" << endl;
    cout << left << setw(20) << "Age: " <<  112 << endl;
    return 0;
}
```

Example Output

```
/home/gentry/Desktop/1428_testDir/junk
Name:                Lyndon Johnson
City:                Austin, TX
Age:                 112

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

Numbers With Fixed Decimals:

```
#include<iomanip>
#include<iostream>

using namespace std;

//Important for Coding Projects!!!
int main(){
    float pi = 3.14159;
    cout << "Long number: " << pi << endl;
    cout << "Medium number: " << fixed << setprecision(3) << pi << endl;
    cout << "Short number: " << fixed << setprecision(1) << pi << endl;
    return 0;
}
```

Example Output

```
/home/gentry/Desktop/junk
Long number: 3.14159
Medium number: 3.142
Short number: 3.1

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

Default values:

- *Sometimes* a compiler will assign a default value to a variable, like automatically storing the number 0 in an **int**.
- *Sometimes* variables just start off holding whatever junk was left over in memory.
- Never assume that C++ will give you a safe default variable.

```
int main(){
    int x;
    cout << "Value of x: " << x << endl;
    //Bad Code!
    return 0;
}
```

Run Time vs. Compile Time:

- There are two times that a bug might show up in our programs.
- Errors in **syntax** are violations in the grammar of coding languages. Syntax errors are easy to find because the compiler will throw an error.
- Errors in **semantics** are grammatically correct but do the wrong thing. Semantics errors are hard to find because the program may still run, but produce erroneous output.

Style guide:

- Keep every line shorter than 80 characters.
- This makes code much more readable, especially with two files positioned side-by-side.

```
int main(){
    cout << "This line of text would be extremely hard to read and would require using my mouse
    cout << "You can break up cout statements "
        << " over several lines to make your "
        << " code easier to read." << endl;
    return 0;
}
```