Compiler Specific Topics

Kate Gregory

www.gregcons.com/kateblog

IDEs Can Make Starter Files

Create New Project

May create files you don't need

May put content in the files you don't need

- Often instructions are added as comments

Command Line Arguments

```
int main(int argc, char* argv[])
{
    return 0;
}
```

```
> Small.exe –g Hello 12
> clang++ a.cpp b.cpp
```

argv[]

char*

argc

- The main function can also be:
- Used when someone passes in parameters from the command prompt
- Not a common technique these days
- is an old (C-style) way to do a collection
- is an old (C-style) way to do a string
- is the count of how many arguments were passed

#pragma once

Prevents a file from being included multiple times

- Compiler errors about declaring things twice
- A includes B and C, they both include D

Not officially standard

- All the major compilers support it

Setting the Warning Level

Compilers can warn on every tiny thing, or be more tolerant

Most default warning levels are too tolerant

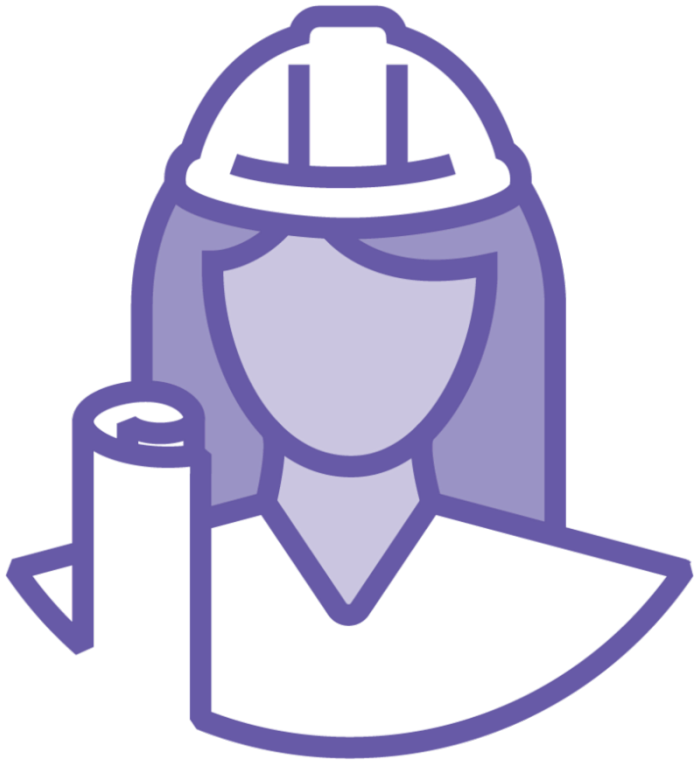
You can change the warning level

- Visual Studio: project options
- Command line: -Wall -Wextra -Wpedantic

You can treat warnings as errors, meaning the build fails if there are warnings

- Visual Studio: project options
- Command line: -Werror

Using Make



If you compile at the command line, you will probably use make

Make is actually a separate tool that calls your compiler

Controlled by makefiles

- Describes which files depend on each other
- Sets compiler options

Simplest Make File

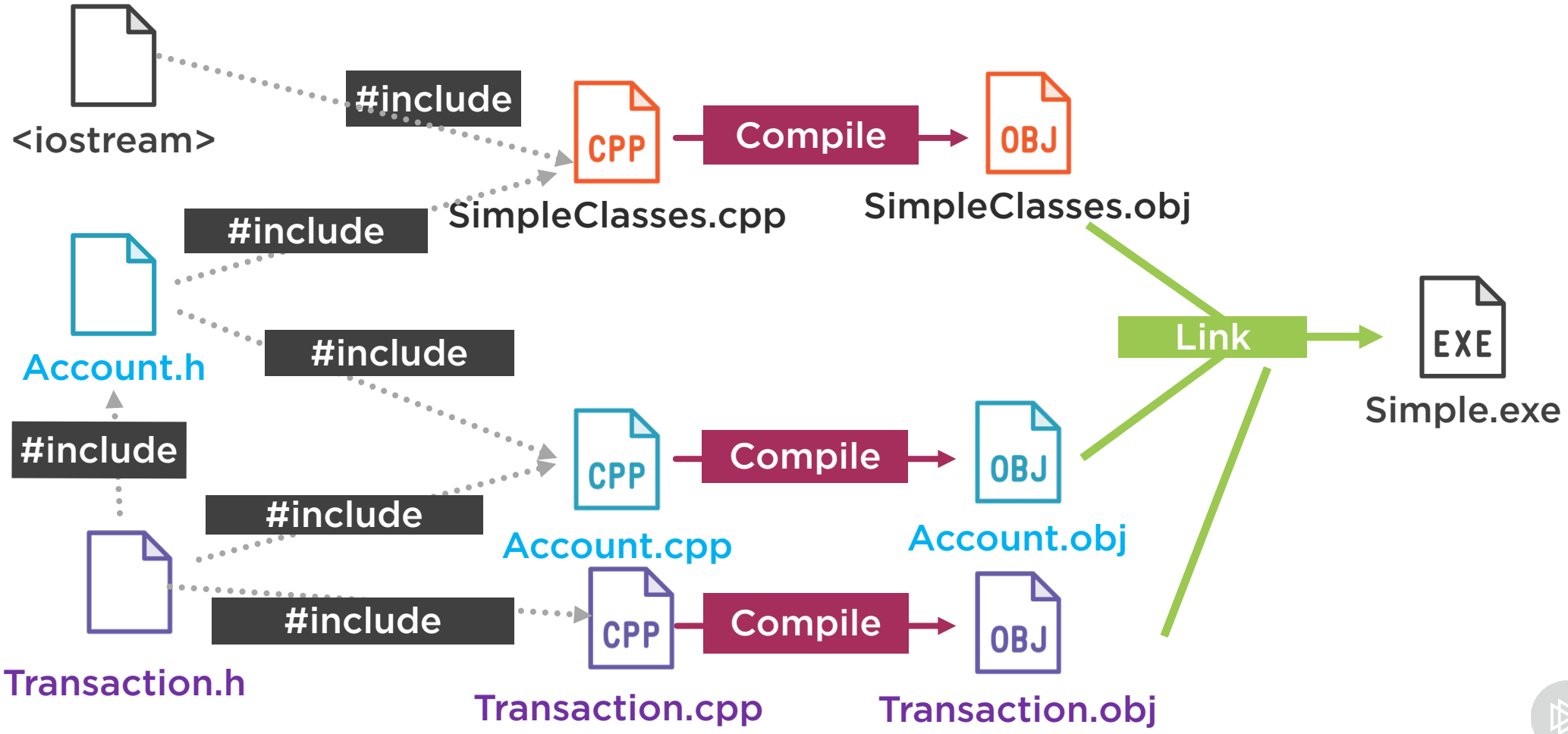
Replaces a long command line

makefile

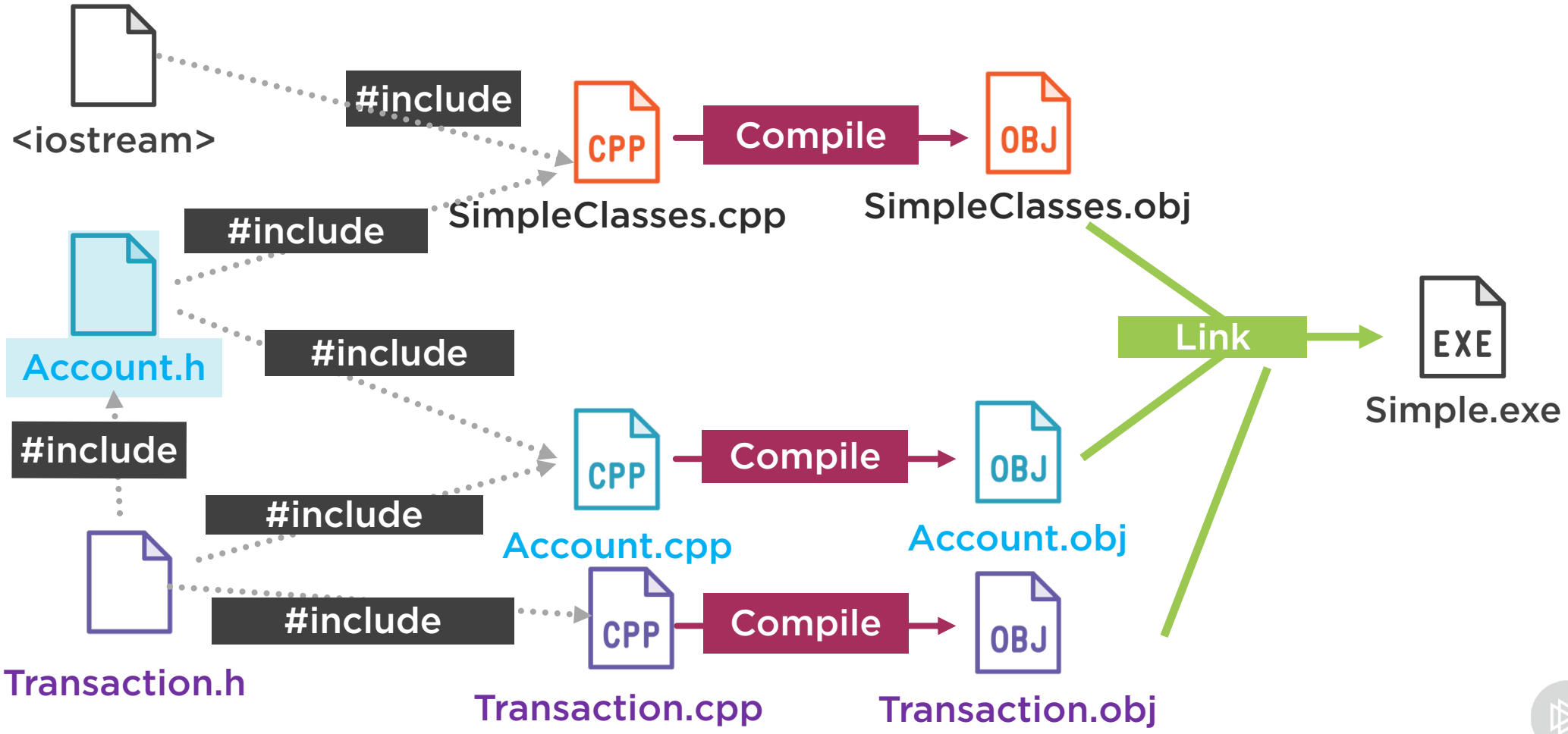
```
all:

	clang++ -std=c++11 -Wall -Wextra -Wpedantic -Werror Account.cpp Transaction.cpp SimpleClasses.cpp -o Simple
```

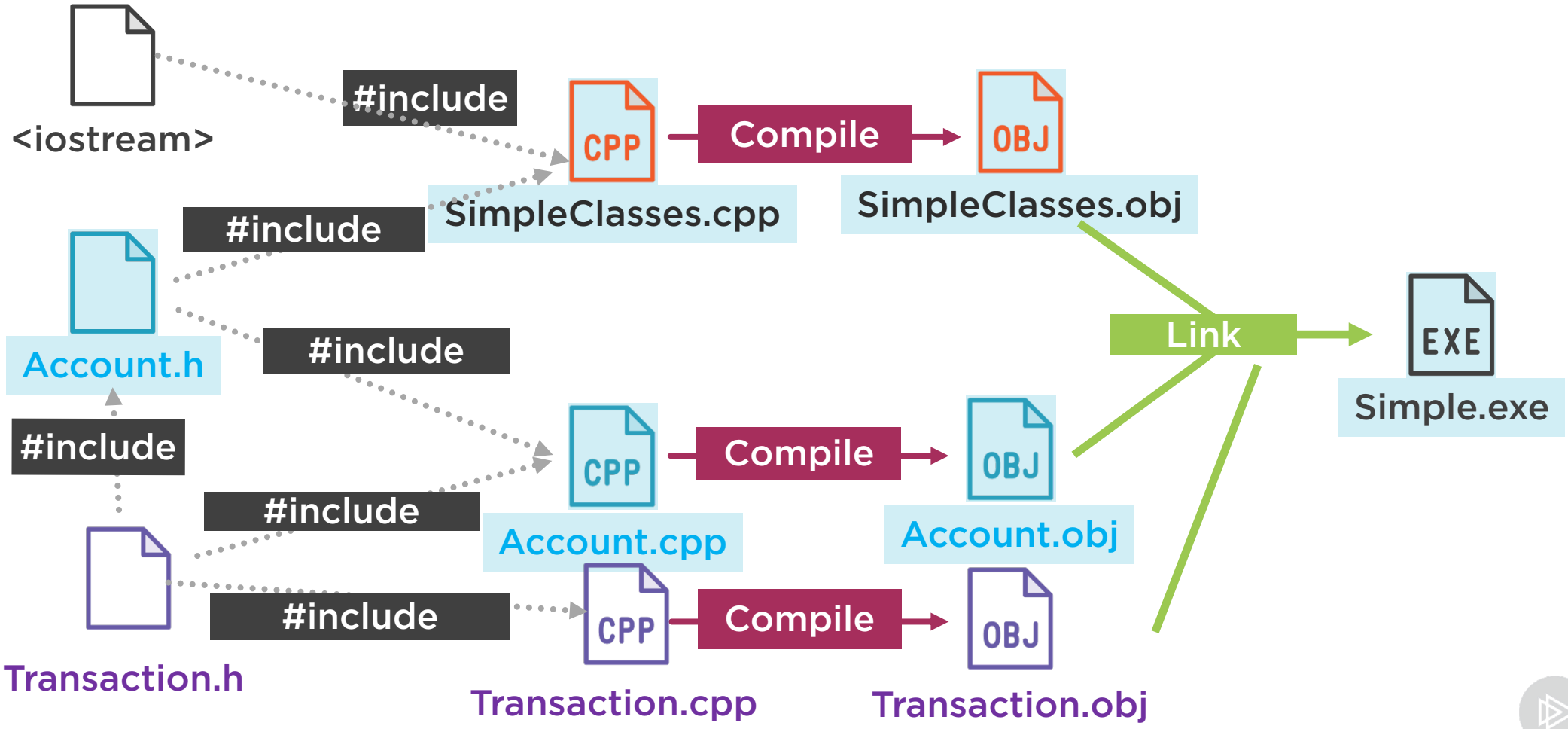
Building with Multiple Files



Building with Multiple Files



Building with Multiple Files



Shorten Build Times

IDEs track dependencies

Can use make for the same experience at the command line

Make Files

```
all: Simple

Simple: Account.o Transaction.o SimpleClasses.o
	clang++ Account.o Transaction.o SimpleClasses.o -o Simple

Account.o: Account.cpp Account.h Transaction.h
	clang++ -c -std=c++11 -Wall -Wextra -Wpedantic -Werror Account.cpp

Transaction.o: Transaction.cpp Transaction.h
	clang++ -c -std=c++11 -Wall -Wextra -Wpedantic -Werror Transaction.cpp

SimpleClasses.o: SimpleClasses.cpp Account.h
	clang++ -c -std=c++11 -Wall -Wextra -Wpedantic -Werror SimpleClasses.cpp
```

Summary

Tools that do things for you may not do exactly what you want

- But it's your code

You can shorten build times by using a tool that only recompiles files that changed, or depend on a changed file

Learning to use make will save you time at the command line