

Programs with Multiple Files

- · How the code is usually split up
 - * Put main in its own file, with helper functions
 - acts like a driver
 - * Put each class declaration in a separate *.h file (called a header file)
 - * Put the implementation of each class (the member function definitions) in its own *.cpp file
 - * Each *.cpp file (including the driver) must #include the header file of each class that it uses or implements.

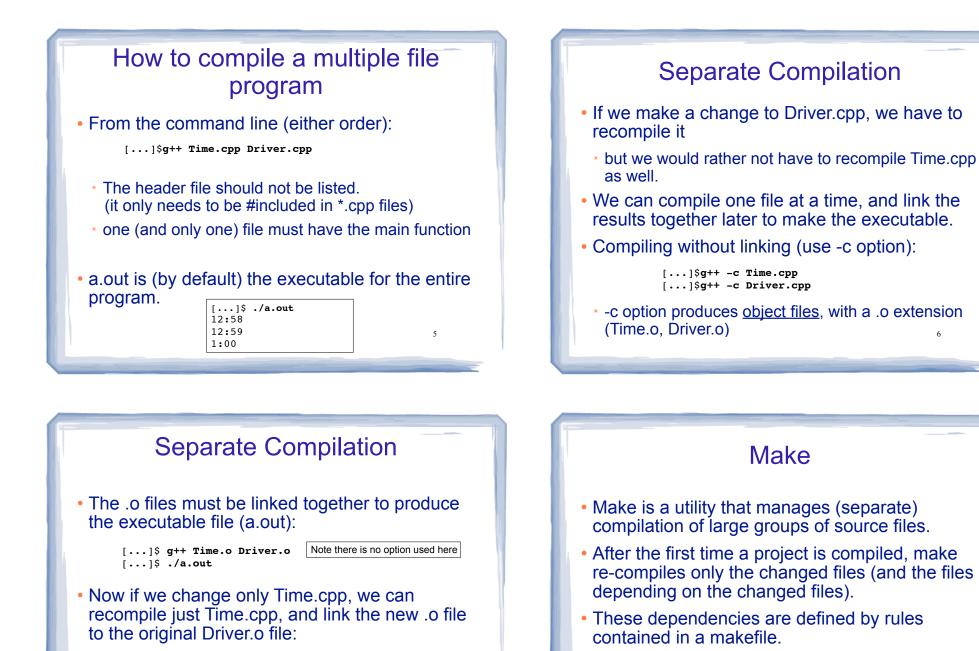
2

Time class, separate files

<pre>#include <string> using namespace std; // models a 12 hour clock class Time { int hour; int minute; void addHour(); void setHour(int); int getMinute() const; int getMinute() const; string display() const; int minute; void site (int); int getMinute() const; string display() const; int getMinute() const; string display() const; int getMinute() const; string display() const; int getMinute() const; int getMi</string></pre>		Driver.cpp
<pre>int main() { Time t; t.setHour(12); t.setHour(12); t.setHour(58); cout << t.display() <<endl; 0;="" <<="" const;="" cout="" display()="" display<="" endl;="" return="" string="" t.addminute();="" t.display()="" td=""><td>mespace std; s a 12 hour clock</td><td>#include<iostream> #include "Time.h"</iostream></td></endl;></pre>	mespace std; s a 12 hour clock	#include <iostream> #include "Time.h"</iostream>
<pre>private: int hour; int minute; void addHour(); public: void setHour(int); void setMinute(int); int getHour() const; int getMinute() const; int getMinute() const; string display() const;</pre> Time t; t.setHour(12); t.setMinute(58); cout << t.display() < <endl; t.addMinute(); cout << t.display() << endl; t.addMinute(); cout << t.display() << endl; return 0; } </endl; 		int main() (
<pre>int minute; void addHour(); public: void setHour(int); void setMinute(int); int getHour() const; string display() const;</pre> t.setMinute(58); cout << t.display() < <endl; t.addMinute(); cout << t.display() << endl; t.addMinute(); cout << t.display() << endl; return 0; }</endl; 		Time t;
<pre>void addHour(); public: void setHour(int); void setMinute(int); int getHour() const; string display() const;</pre> cout << t.display() << endl; t.addMinute(); cout << t.display() << endl; t.addMinute(); cout << t.display() << endl; t.addMinute(); cout << t.display() << endl; t.addMinute(); cout << t.display() << endl; return 0; }		
<pre>public: void setHour(int); void setMinute(int); int getHour() const; string display() const; cout << t.display() << endl; t.addMinute(); cout << t.display() << endl; return 0; string display() const;</pre>		<pre>cout << t.display() <<endl;< pre=""></endl;<></pre>
<pre>int getHour() const; return 0; int getMinute() const; } string display() const;</pre>	<pre>setHour(int);</pre>	<pre>cout << t.display() << endl;</pre>
<pre>string display() const;</pre>		1 1()
		}
<pre>void addMinute();</pre>		
};		

Time class, separate files

Time.cpp	
<pre>#include <iomanip></iomanip></pre>	<pre>void Time::addHour() {</pre>
#include <sstream></sstream>	if (hour == 12)
#include "Time.h"	hour = 1;
using namespace std;	else
	hour++;
<pre>void Time::setHour(int hr) {</pre>	}
hour = hr;	<pre>void Time::addMinute() {</pre>
}	if (minute == 59) {
	minute = 0;
<pre>void Time::setMinute(int min) {</pre>	addHour();
minute = min;	} else
}	minute++;
	}
int Time::getHour() const {	<pre>string Time::display() const {</pre>
return hour;	ostringstream sout;
}	<pre>sout.fill('0');</pre>
	sout << hour << ":"
<pre>int Time::getMinute() const {</pre>	<< setw(2) << minute;
return minute;	return sout.str();
}	}
	4



Produces new Time.o

making a new a.out

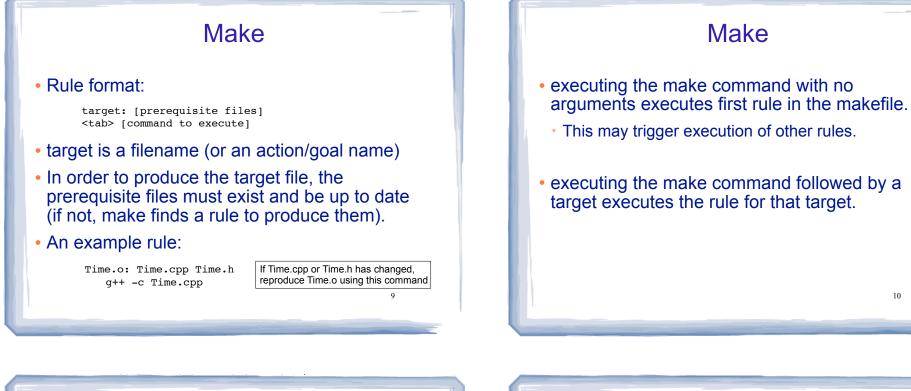
7

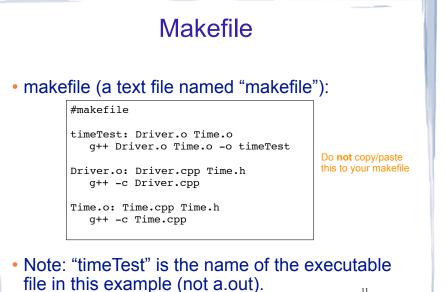
[...]\$g++ Time.o Driver.o Links new Time.o to old Driver.o,

[...]\$g++ -c Time.cpp

[...]\$./a.out

• The rules are defined and managed by humans (programmers).





Compile class + driver using make

- Make: [...]\$ make g++ -c Driver.cpp q++ -c Time.cpp q++ Driver.o Time.o -o timeTest
- Execute: [...]\$./timeTest 12:58 12:59 1:00
- Modify Driver.cpp, make again:

[...]\$ make g++ -c Driver.cpp q++ Driver.o Time.o -o timeTest 10