Some terminology

Last changed 4/5/2022

What we will learn in this set of slides

- Intuitively, what a *type* is
- What a *scope* is
- The C++ execution model

Types

- A *type* is the kind of something in a program
- The type of f in *float f;* is float.
    - The storage of the f is large enough to hold a float
    - Operations on f are operations on a float
- The type of fP in *float* fP;* is a pointer to a float.
    - The storage of fP is large enough to hold the address of a float value
    - Operations on fP are operations on a pointer
- *float g = f + 1;* uses different hardware than *float* fP2 = fP + 1;*
    - The type information is used by the compiler to determine the code to generate on variables of different types

;

We can define new types

```
typedef struct Point{
  int x;
  int y;
} Point_t;
Point_t aPoint;
enum wind_directions_t
   {NO_WIND, NORTH_WIND,
    SOUTH_WIND, EAST_WIND,
    WEST_WIND};
wind_directions_t wind_direction = NO_WIND;
wind_direction = 453; // doesn't work, compiler error
```

- *typedef* is used to define a new type
- A typedef can be used to
  - Create a shorter or simpler name for a type
  - Allow us to refer to a struct definition as a type
  - Allow us to control the range of values a variable can take on
- OO programming allows us to define new types (*classes)* have data and operators
  - *Objects* are storage that are of this type.

Scopes of symbols

- The scope of a symbol is the extent or range in a program in which the symbol is recognized, i.e., can be used.
- The scopes you worried about in C were primarily *file*, *function* and *block*

File scope

```
// file matrix.h
#ifndef MATRIX_H_
#define MATRIX_H_
int f1(int foo);
float mm(float a, float b, float c);
#endif /* MATRIX_H_ */
```

```
// file somefile.c
#include matrix.h
#include somefile.h
float f;

float f1(float f) {…}
float f2(float g) {…}
```

- The include file brings the declared functions in matrix.h into somefile.c file scope
- The variable *f,* declared in *float f,* has file scope
  - It is visible and useable within somefile.c
  - If a statement in another file tries to access it, it will not be found
- Functions *f1* and *f2* can only be accessed in somefile.c unless the other file include somefile.h that gives their headers.

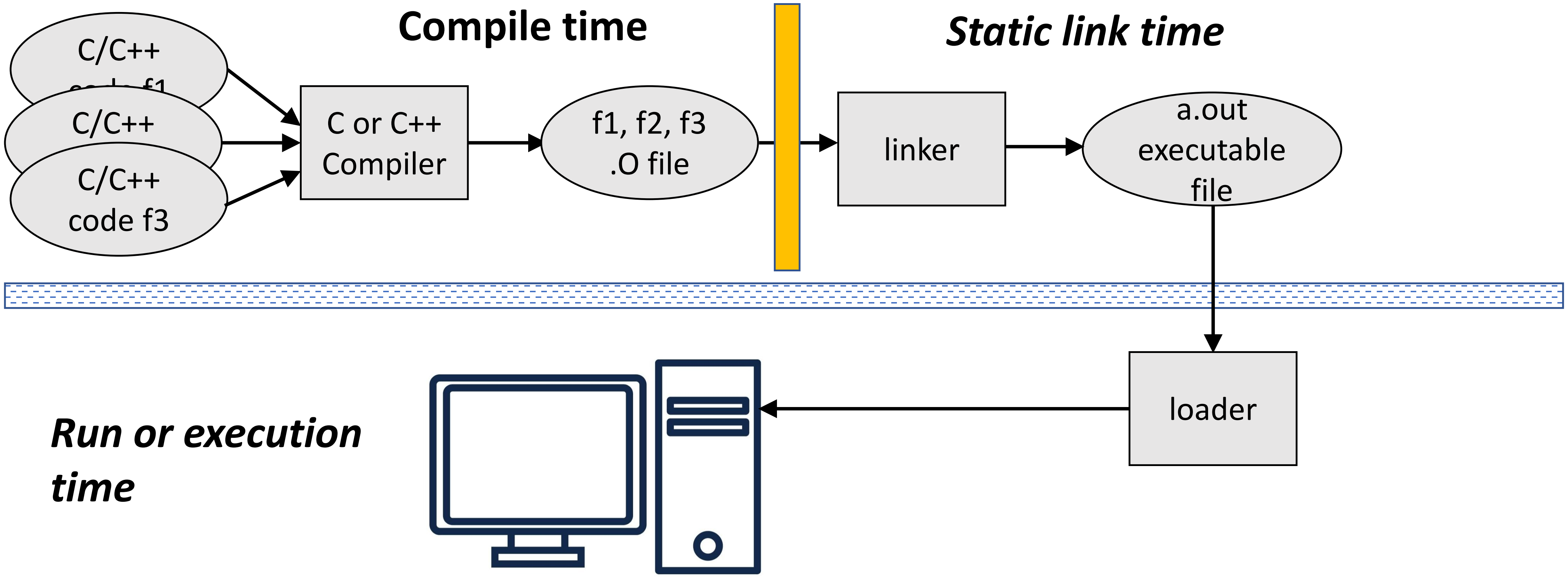
function scope

```
// file somefile.c
#include matrix.h
#include somefile.h
float f;

float f1(float f) {
  f = 4.
  return f;
}
void f2(float g) {
  float x = 1. + 2.;
  f = x;
}
```

- In *f1*, the parameter *f* hides the variable *f* declared at the top of the file
  - References to *f* in *f1* are to the parameter, not to the earlier declared *f*.
  - The parameter *f* is only accessible within the function *f1*.
- In *f2*, the parameter *g* and variable *x* have function scope and are only accessible within *f2.*
  - The *f* accessed is the file scope *f*

C/C++ compilation model


Compile time
Static link time
C/C++ code f1
C/C++
C/C++ code f3
C or C++ Compiler
f1, f2, f3 .O file
linker
a.out executable file
loader
Run or execution time