Arrays, Memory

CS 5006, 5007: C, Algorithms and Systems

Adrienne Slaughter, Joe Buck

Northeastern University

January 17, 2019

Questions?

1. Review of Week 1

2. More Formal Intro to C
   - What is C?
   - Intro to Variables
   - Introducing Arrays
   - Data Hierarchy

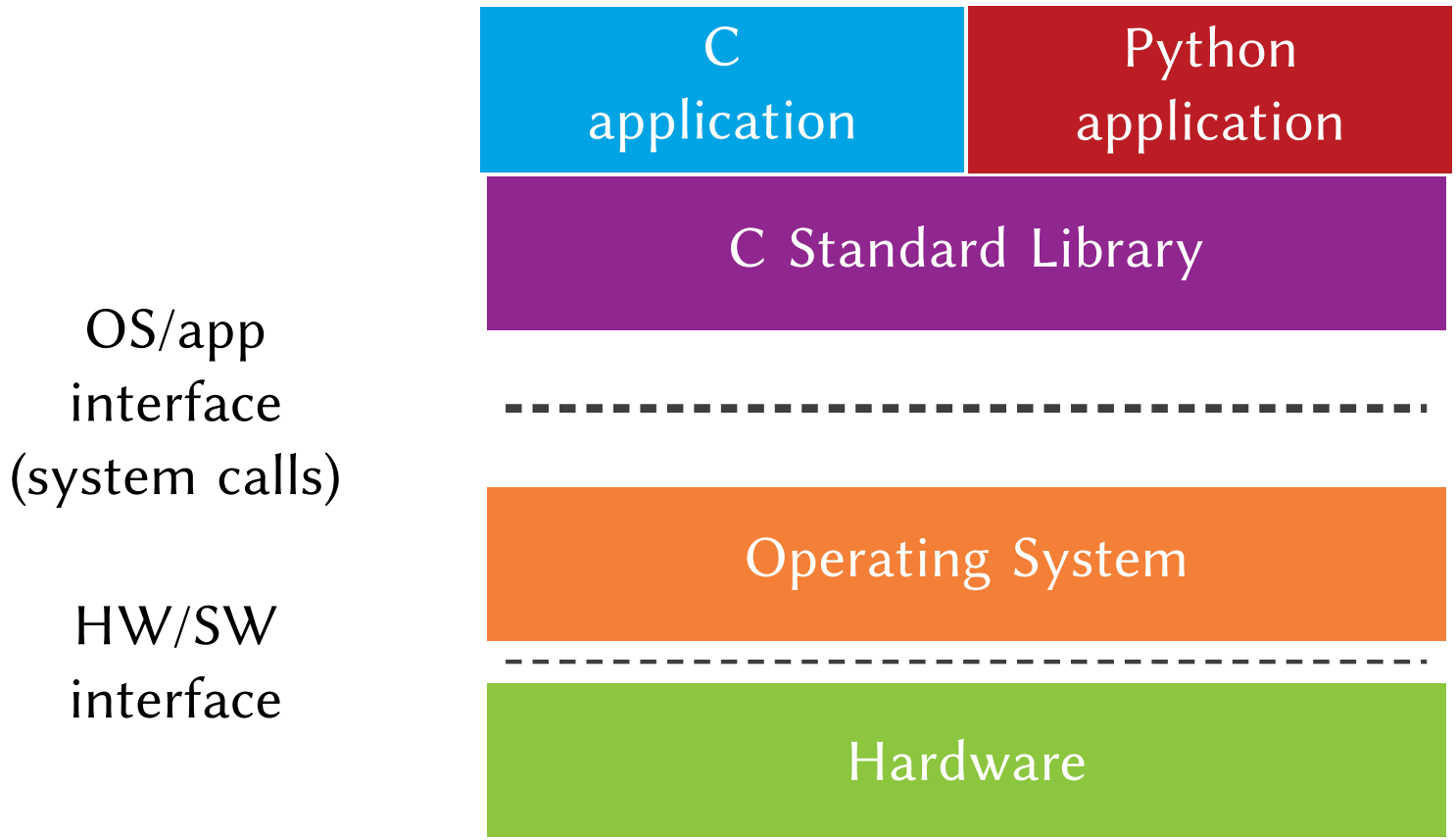
Section 1

Review of Week 1

Last Week

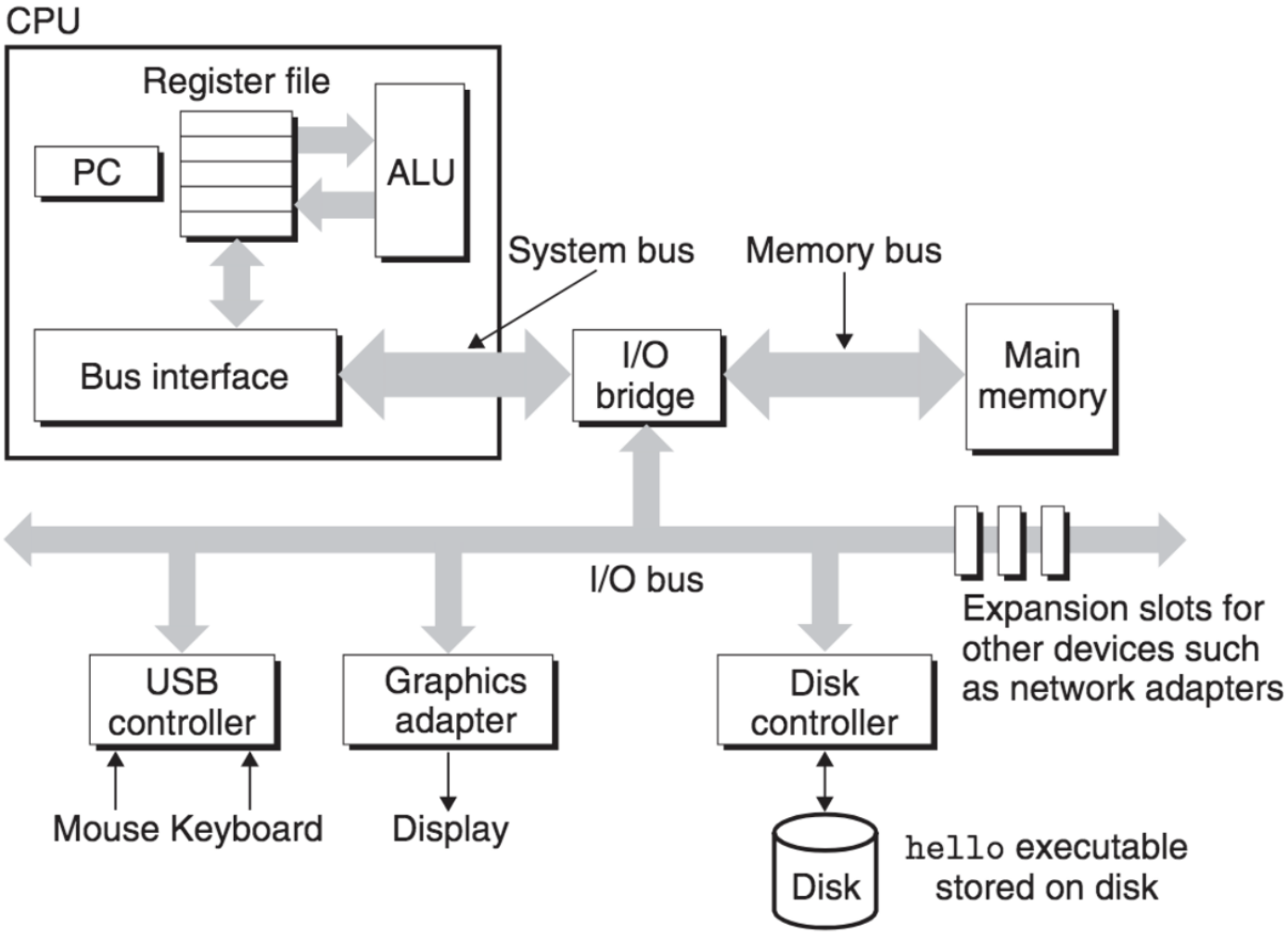
- What is a system?
- How do computers work?
- VirtualBox
- `gcc`
- `ssh`
- `make`

The Big Picture: What is a System?

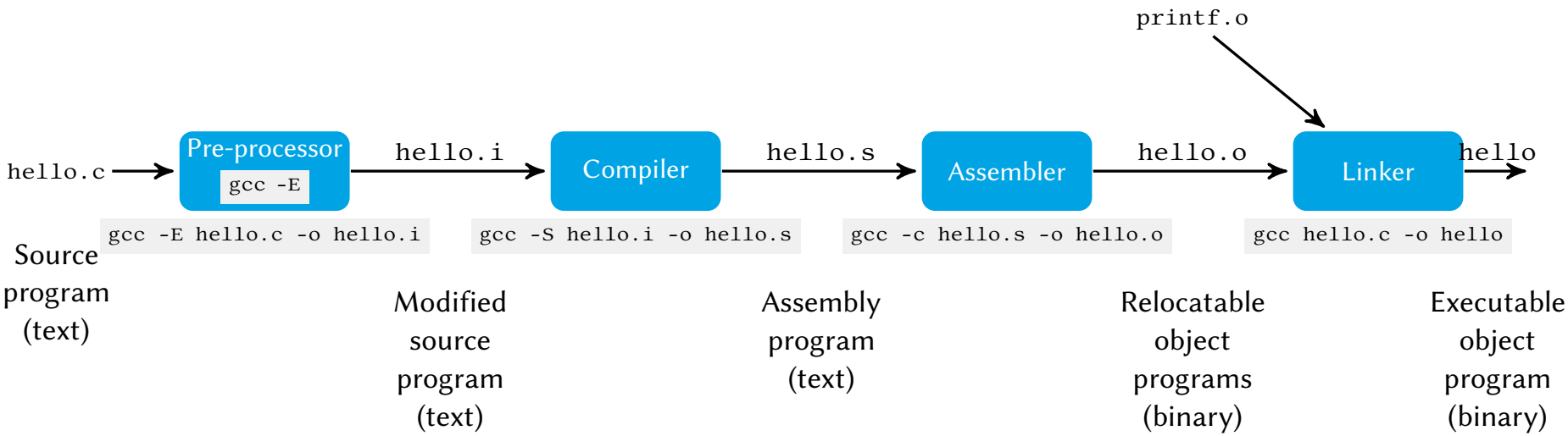


Computer Organization

What happens when we run our program?



What happens when your program gets compiled?



Agenda

- What is C?
- Types
- Intro to arrays
- Memory

Section 2

More Formal Intro to C

What is C?

- General purpose programming language

What is C?

- General purpose programming language
- UNIX (and most of the programs it uses) are written in C

What is C?

- General purpose programming language
- UNIX (and most of the programs it uses) are written in C
- Sometimes called a “systems programming language” because it’s used for writing compilers and OSs

What is C?

- General purpose programming language
- UNIX (and most of the programs it uses) are written in C
- Sometimes called a “systems programming language” because it’s used for writing compilers and OSs
- But it’s good to write many kinds of programs!

What is C?

- General purpose programming language
- UNIX (and most of the programs it uses) are written in C
- Sometimes called a "systems programming language" because it's used for writing compilers and OSs
- But it's good to write many kinds of programs!
- It's a relatively ***low-level*** language

What is C?

- General purpose programming language
- UNIX (and most of the programs it uses) are written in C
- Sometimes called a "systems programming language" because it's used for writing compilers and OSs
- But it's good to write many kinds of programs!
- It's a relatively ***low-level*** language
  - It uses the same kinds of objects that most machines do

What is C?

- General purpose programming language
- UNIX (and most of the programs it uses) are written in C
- Sometimes called a "systems programming language" because it's used for writing compilers and OSs
- But it's good to write many kinds of programs!
- It's a relatively ***low-level*** language
  - It uses the same kinds of objects that most machines do
- It doesn't include a lot of things that other programming languages do, but that keeps it very small.

What is C?

- General purpose programming language
- UNIX (and most of the programs it uses) are written in C
- Sometimes called a "systems programming language" because it's used for writing compilers and OSs
- But it's good to write many kinds of programs!
- It's a relatively ***low-level*** language
  - It uses the same kinds of objects that most machines do
- It doesn't include a lot of things that other programming languages do, but that keeps it very small.

What is C?

- It's a ***typed*** language

Let's Play!

Get started by coding up Hello World and running it.

```
[ahslaughter@adriennes-mbp:~]\$ ./hello
Hello, World!

```

Listing 1: Print Hello, World!

Now, make it two lines, with 2 lines of code.

```
1 [ahslaughter@adriennes-mbp:~]\$ ./hello
2 Hello,
3 World!
4
```

Listing 2: Make it two lines

Make the output two lines, but use one line of code.

```
1 [ahslaughter@adriennes-mbp:~]\$ ./hello
2 Hello,
3 World!
4
```

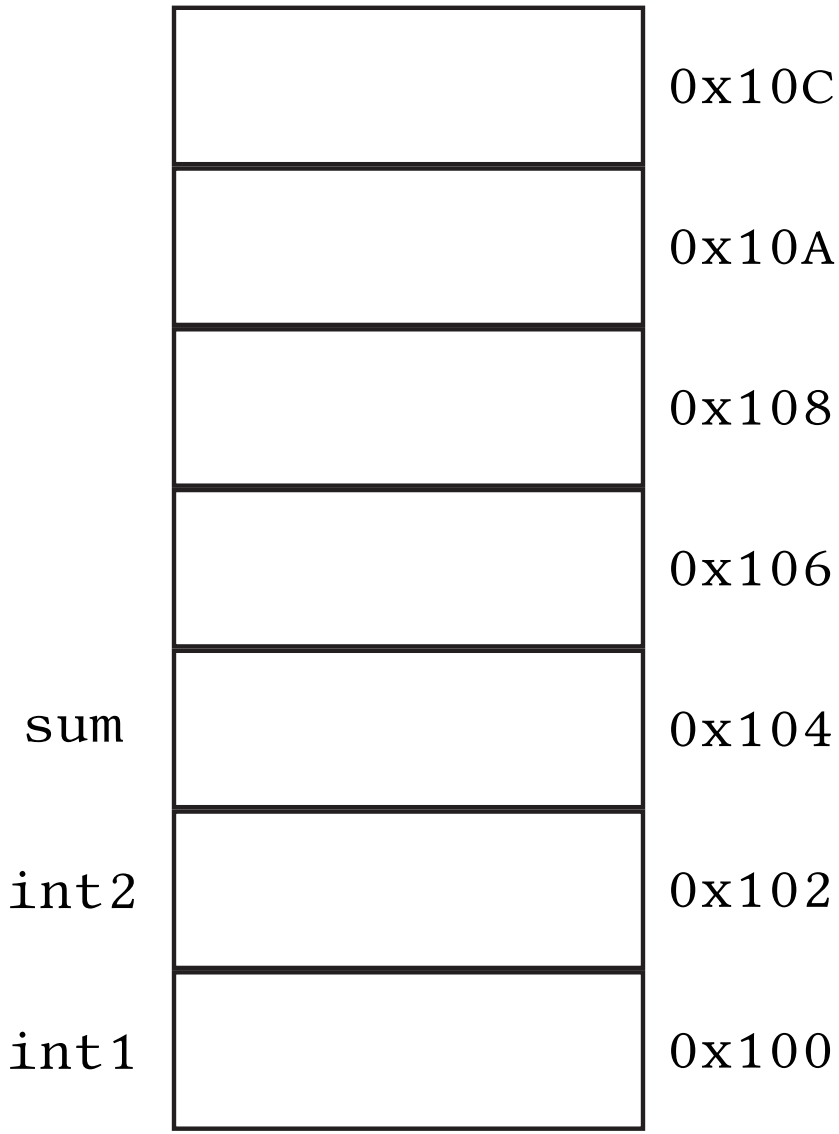
Listing 3: Make it two lines, but make your code one

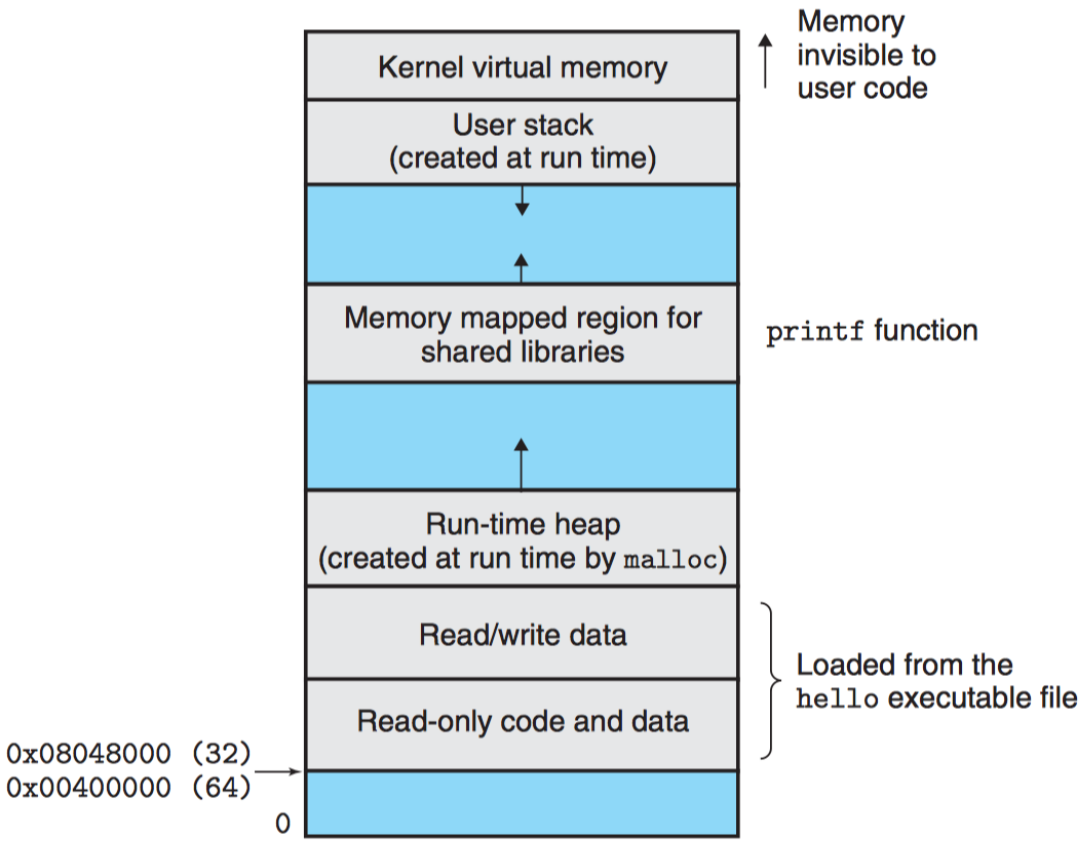
Example: Adding two numbers

```c
#include<stdio.h>

int main() {

  int int1, int2, sum;              // declaration

  printf("Enter the first integer: \n"); // prompt
  scanf("%d", &int1);             // read an int
  printf("Enter the second integer:\n");
  scanf("%d", &int2);
  sum = int1 + int2;
  printf("Sum is %d\n", sum);

  return 0;           // Program finished successfully
}
```

Listing 4: Input and output

Memory




Kernel virtual memory
User stack
(created at run time)
Memory mapped region for
shared libraries
Run-time heap
(created at run time by malloc)
Read/write data
Read-only code and data
0x08048000 (32)
0x00400000 (64)
0
Memory
invisible to
user code
printf function
Loaded from the
hello executable file

Things to remember about memory

- There is a limited amount on each machine
- A section is always dedicated to the OS
- A section is dedicated to shared libraries
- A section gets reserved for a program every time it runs
- When a program terminates, that memory get's de-allocated, but the OS does not clear the memory by default.

C Types

Basic types:

- int
- char
- float
- double

Derived types:

- array
- pointer
- structure
- union

And on its own: ***void***

Data Type Sizes

Type	Storage Size	Range
`char`	1 byte	0 to 255
`int`	2 or 3 bytes	-32,768 to 32,767
`short`	2 bytes	-32,768 to 32,767
`long`	4 bytes	-2,147,483,648 to 2,147,483,647
`float`	4 byte	1.2E-38 to 3.4E+38
`double`	8 byte	2.3E-308 to 1.7E+308
`long double`	10 byte	3.4E-4932 to 1.1E+4932

Data Type Sizes

Type	Storage Size	Range
`char`	1 byte	0 to 255
`int`	2 or 3 bytes	-32,768 to 32,767
`short`	2 bytes	-32,768 to 32,767
`long`	4 bytes	-2,147,483,648 to 2,147,483,647
`float`	4 byte	1.2E-38 to 3.4E+38
`double`	8 byte	2.3E-308 to 1.7E+308
`long double`	10 byte	3.4E-4932 to 1.1E+4932

You can always get the size of a type by calling `sizeof(<type>)`, such as `sizeof(int)`

Arithmetic Operators

Operation	Operator	Expression	C expression
Addition	`+`	$f + 7$	`f + 7`
Subtraction	`-`	$p - c$	`p - c`
Multiplication	`*`	bm	`b * m`
Division	`/`	x/y	`x / y`
Modulus	`%`	$r \mod s$	`r % s`

Example: Working with arrays

```c
#include<stdio.h>

int main() {

  int arr[10];

  for (int i=0; i<10; i++) {
    printf("%d\n", arr[i]);
  }
}
```

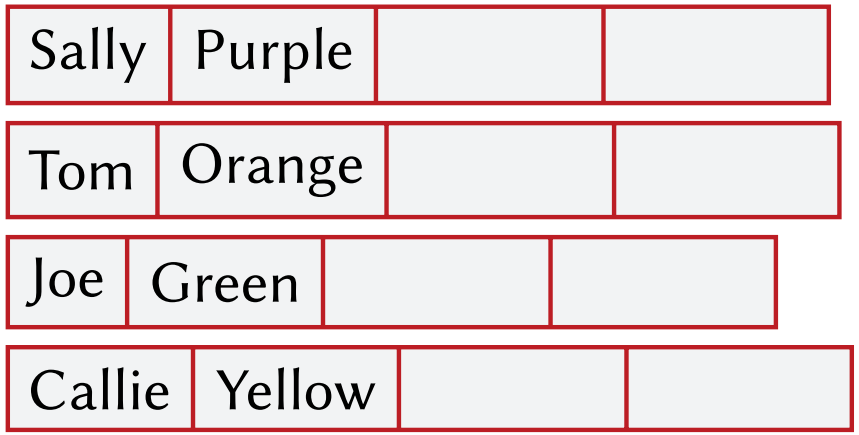
Listing 5: A first array

What are arrays?

- Contiguous chunk of memory
- Must hold the same type in every slot
- Must specify the size when you create it
- **YOU** must make sure you don't run off the end
- A 'string' is just an array of characters with a special character at the end (`\0`)

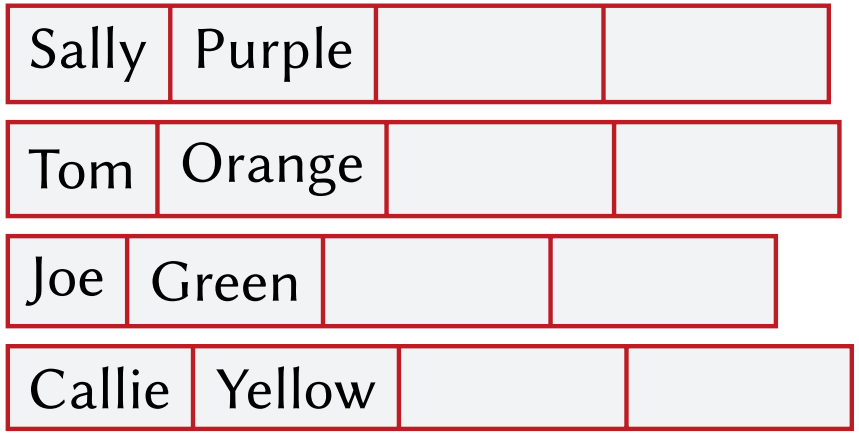
Data Hierarchy

A file



Data Hierarchy

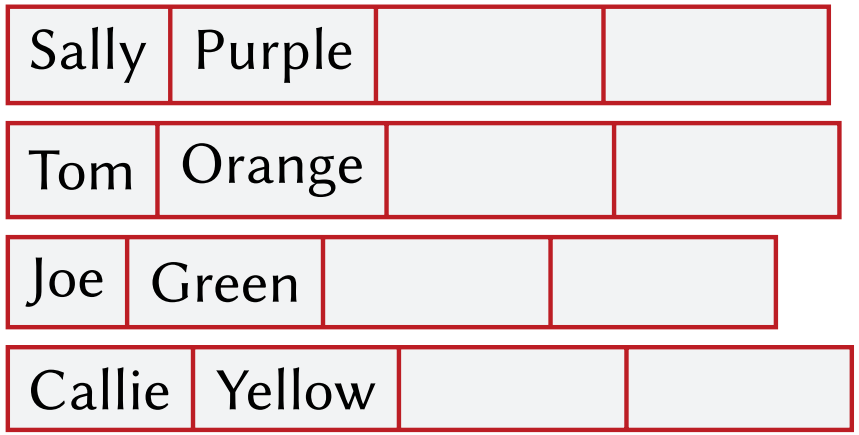
A file



A line/record of a file

Data Hierarchy

A file



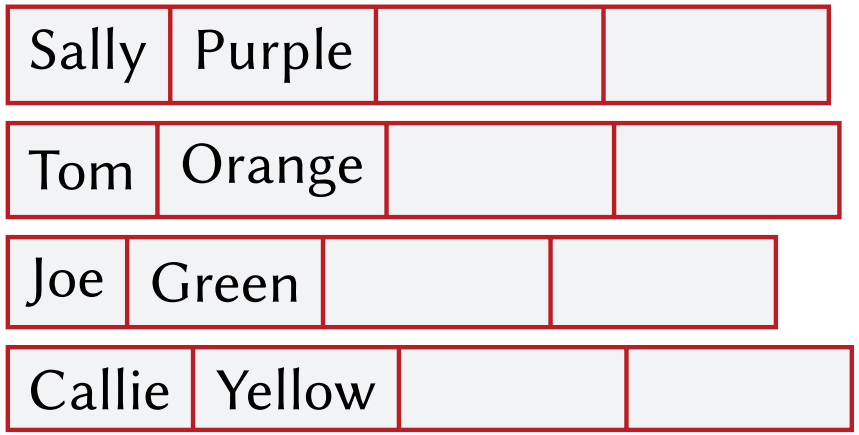
A line/record of a file

T, o, m

chars that make up a field in a record

Data Hierarchy

A file



A line/record of a file

T, o, m

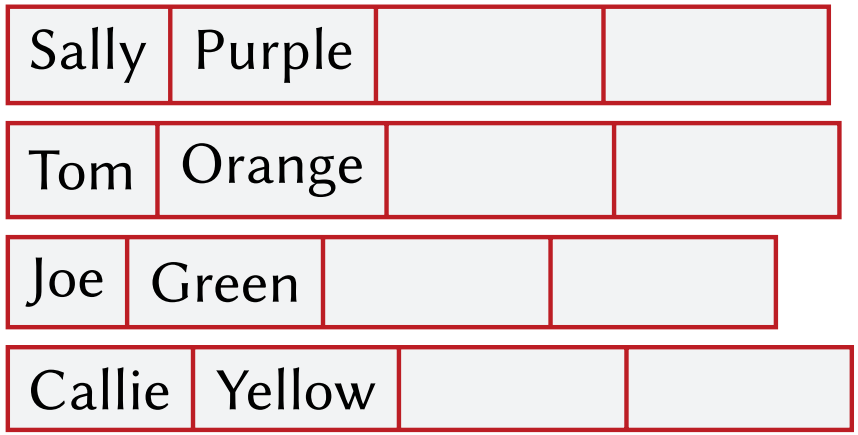
chars that make up a field in a record

01010100

a byte that represents a char in a field

Data Hierarchy

A file



A line/record of a file

T, o, m

chars that make up a field in a record

01010100

a byte that represents a char in a field

0

a bit in the byte

1. Review of Week 1

2. More Formal Intro to C
   - What is C?
   - Intro to Variables
   - Introducing Arrays
   - Data Hierarchy