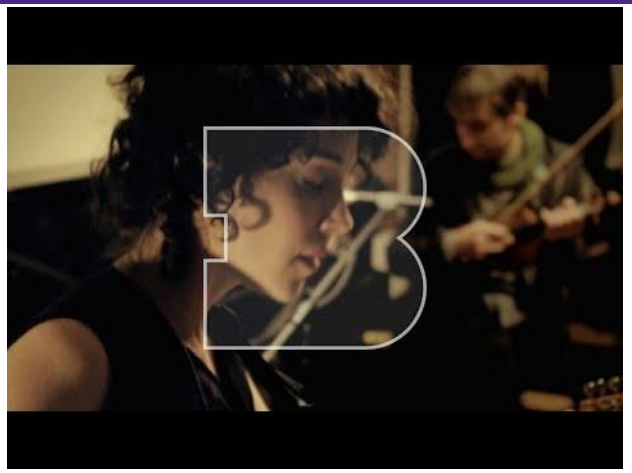
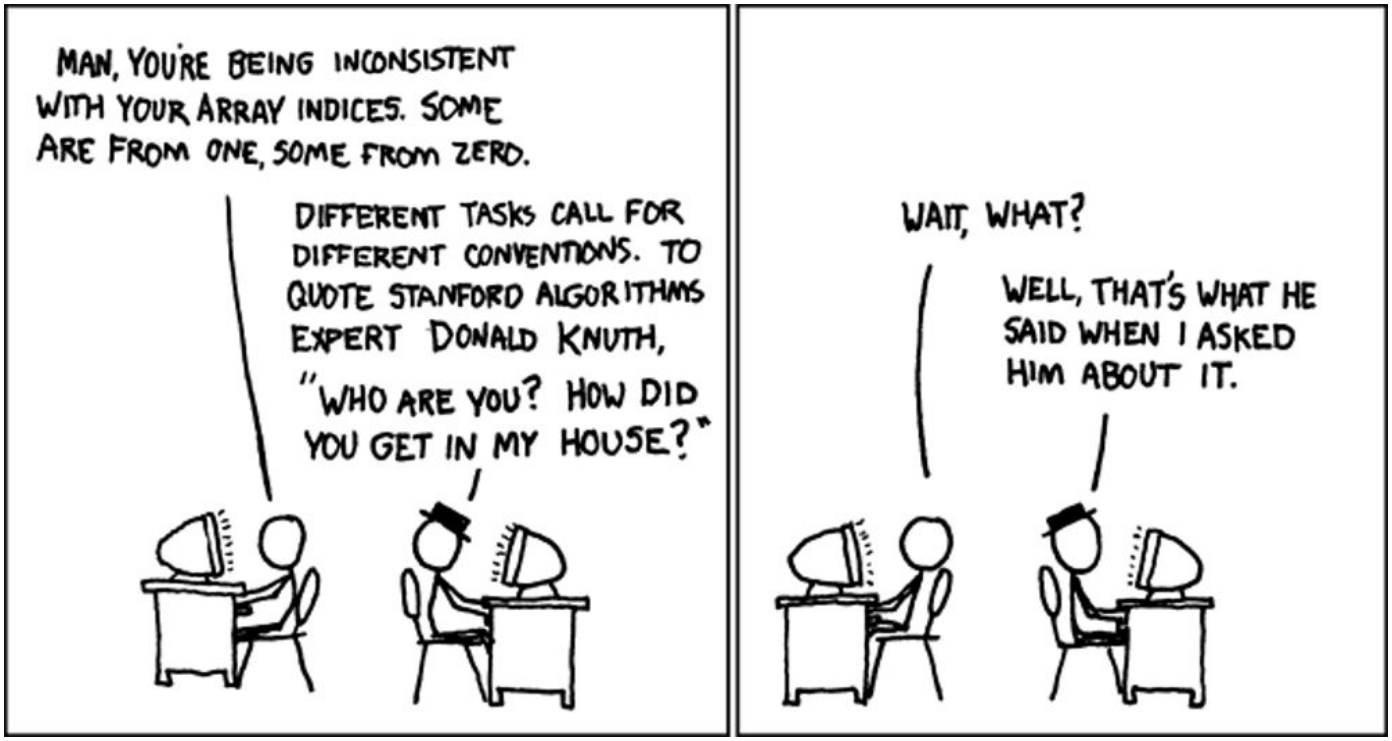
# Structs & Alignment

CSE 351 Summer 2021



**Instructor:**
Mara Kirdani-Ryan

**Teaching Assistants:**
Kashish Aggarwal
Nick Durand
Colton Jobs
Tim Mandzyuk




http://xkcd.com/163/

# Gentle, Loving Reminders

- hw12 due tonight! (7/23)
- hw13 due monday (7/26)

- Last lecture of Unit #2!
  - In-class critique on Wednesday (7/28)
  - Unit Summary #2 due on Monday, 8/2

- Lab 3 due next Friday (7/30)
  - You get to write some buffer overflow exploits!

# Unit Summary #1 Graded!

- Generally excellent
- Most S->N was because there wasn't any clear tie to socio-technical content
  - Regrades open tonight, if you're bugged
- In general, honest effort got full credit

# US #2

- Broadening scope!
  - Narratives *describing* the floor are ok, pictures are ok, any form that you'd like, as long as we can understand it
    - Narratives (stories) or pictures probably easiest
- I'm mostly looking for creative expression

# Mid-Quarter Survey

- Thanks for the feedback!
  - My pronouns are still they/them
  - If socio-technical content feels like a waste of time, I'd love to meet and listen
  - If unit summaries felt like a nightmare, or so stressful that you're dreading the next one, let's talk!
- If OH times don't work, send me an email -- I usually have time in the afternoon

# Sound good?
# Feeling ok?

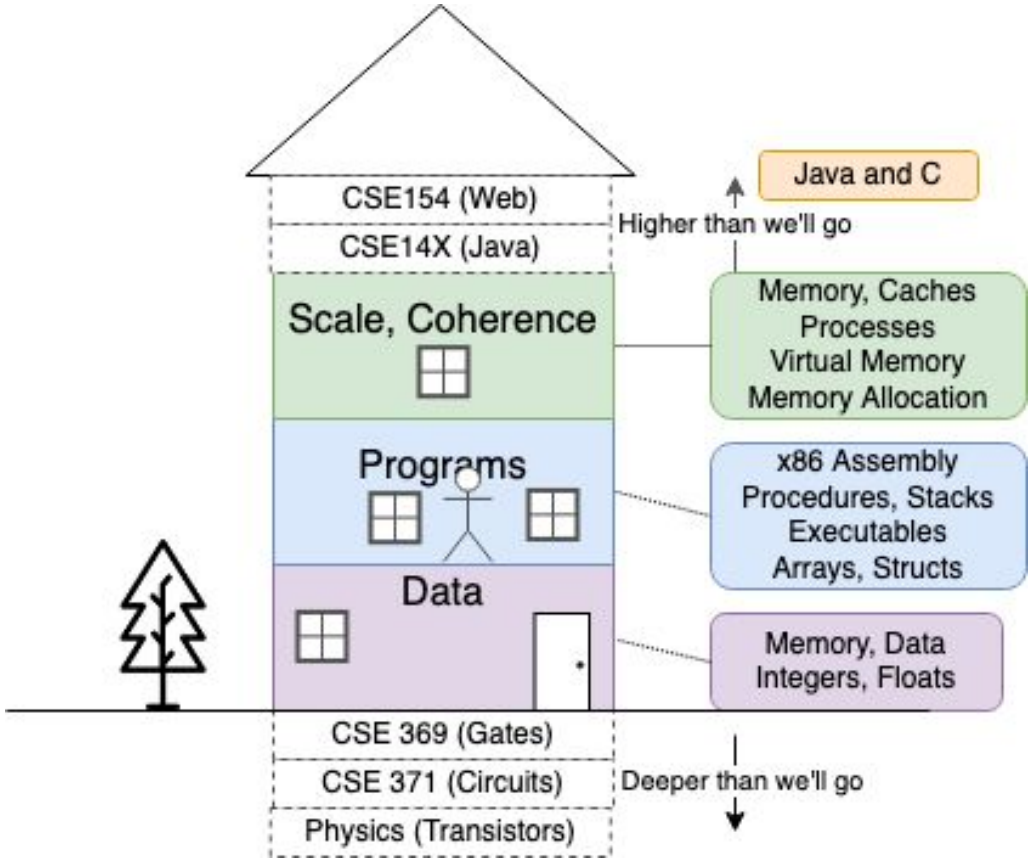
# Learning Objectives

Understanding this lecture means you can:

- Draw a representation of structs, laid out in memory, with alignment and padding
- Differentiate between internal & external fragmentation
- Rearrange fields of a struct to save space
- Explain the duality of insulation in computer science
- Give your own working definition of neurodiversity, and explain issues behind spectrum analogies and “high/low functioning” dichotomies

# Second Floor! Programs!

- Values in modern processors
- Critical Analysis
- Accessibility, agency and support
- Establishing and extending structures
- How programs are executed by a processor
- How data structures are stored in assembly



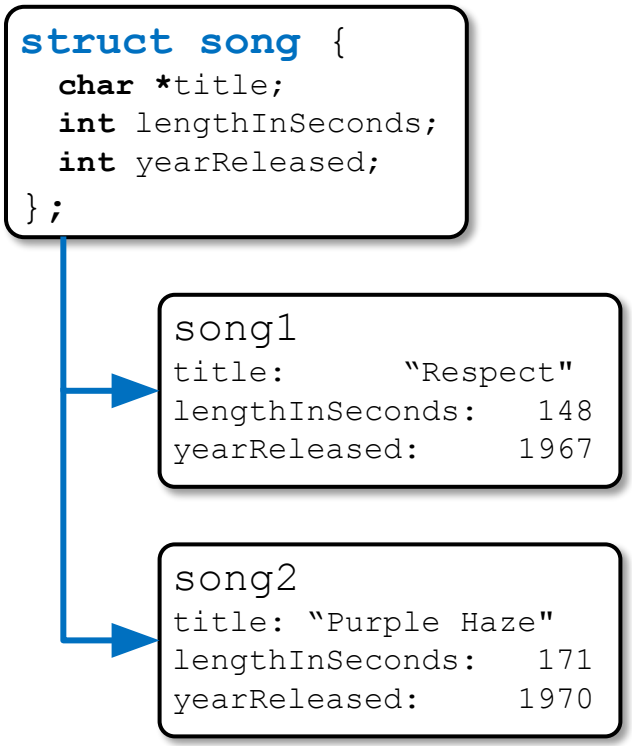
# Data Structures in Assembly

- Arrays
  - One-dimensional
  - Multi-dimensional (nested)
  - Multi-level
- **Structs**
  - **Alignment**
- ~~Unions~~

# Structs in C

- A structured group of variables, possibly including other structs
  - Way of defining compound data types

```
struct song {
  char *title;
  int lengthInSeconds;
  int yearReleased;
};

struct song song1;
song1.title = "Respect";
song1.lengthInSeconds = 148;
song1.yearReleased = 1967;

struct song song2;
song2.title = "Purple Haze";
song2.lengthInSeconds = 171;
song2.yearReleased = 1970;
```



# Struct Definitions

- Structure definition:
  - Does NOT declare a variable
  - Variable type is "**`struct name`**"

```
struct name {
  /* fields */
};
```

Easy to forget semicolon!

- Variable declarations like any other data type:

```
struct name name1;
struct name *pn;
struct name name_ar[3];
```

instance
pointer
array

# Scope of Struct Definition

- Placement of struct definition is important!
  - What actually happens when you declare a variable?
    - Creating space for it somewhere!
  - Program needs definition to determine space

```
struct data {
    int ar[4];
    long d;
};
```

Size = 24 bytes

Size = 32 bytes

```
struct rec {
    int a[4];
    long i;
    struct rec* next;
};
```

- Almost always define structs in global scope near the top of your C file
  - Struct definitions follow normal rules of scope

# Accessing Structure Members

- Given a struct instance, access member using the `.` operator:

```
struct rec r1;
r1.i = val;
```

```
struct rec {
    int a[4];
    long i;
    struct rec *next;
};
```

- Given a *pointer* to a struct:

```
struct rec *r;
r = &r1;  // or malloc space for r
```

We have two options:

- Use `*` and `.` operators: `(*r).i = val;`
- Use `->` operator for short: `r->i = val;`

- **Assembly:** register holds address of first byte
  - Access members with offsets

# Java connection

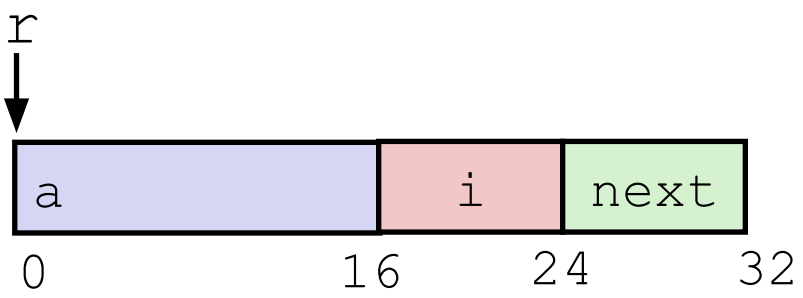
```
class Record { ... }
Record x = new Record();
```

- An instance of a class is like a *pointer to* a struct containing the fields
  - (Ignoring methods and subclassing for now)
  - So Java's `x.f` is like C's `x->f` or `(*x).f`

- In Java, almost everything is a pointer ("*reference*") to an object
  - Cannot declare variables or fields that are structs or arrays
  - Always a *pointer* to a struct or array
  - So every Java variable or field is ≤ 8 bytes (but can point to lots of data)

# Structure Representation

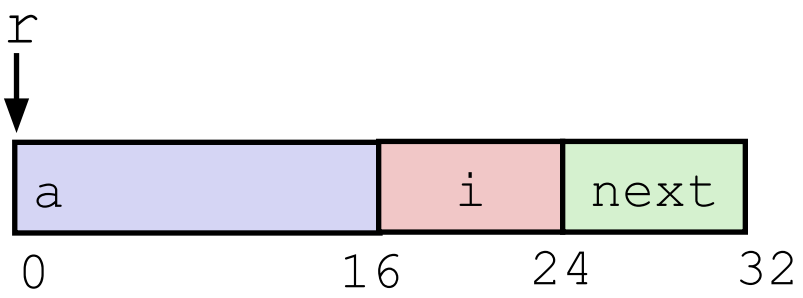
```
struct rec {
    int a[4];
    long i;
    struct rec *next;
};
struct rec st;
struct rec *r = &st;
```




- Characteristics
  - Contiguously-allocated region of memory
  - Refer to members within structure by names
  - Fields may be of different types

# Structure Representation

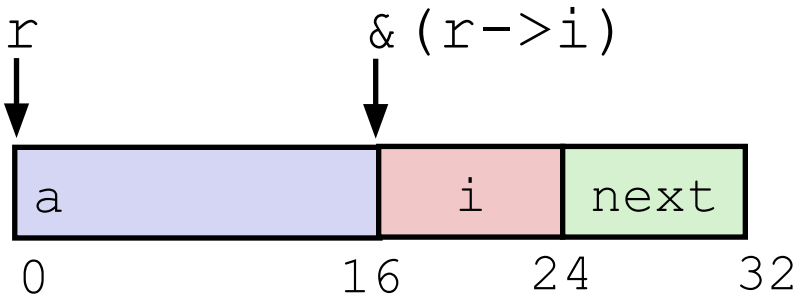
```
struct rec {
    int a[4];
    long i;
    struct rec *next;
};
struct rec st;
struct rec *r = &st;
```




- Structure represented as block of memory
  - Big enough to hold all of the fields
- Fields ordered according to declaration order
  - Even if another ordering would be more compact
- Compiler determines size + positions of fields
  - Machine-level program has no understanding of the structures in the source code

# Accessing a Structure Member

```
struct rec {
    int a[4];
    long i;
    struct rec *next;
};
struct rec st;
struct rec *r = &st;
```



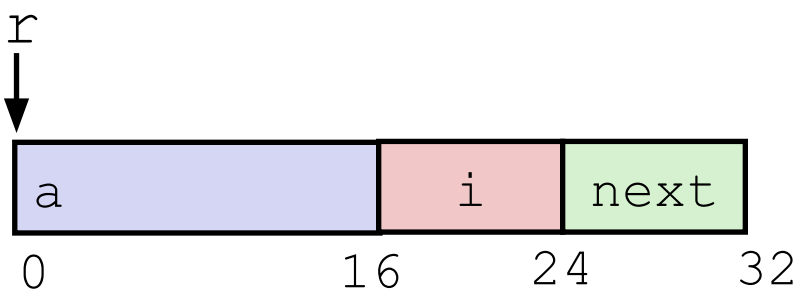

- Compiler knows the *offset* of each member within a struct
  - Compute as `*(r+offset)`
    - Referring to absolute offset, so no pointer arithmetic

```
long get_i(struct rec *r)
{
  return r->i;
}
```

```
  # r in %rdi
  movq  16(%rdi), %rax
  ret
```

# Pointer to Structure Member

```
struct rec {
    int a[4];
    long i;
    struct rec *next;
};
struct rec st;
struct rec *r = &st;
```




```
long* addr_of_i(struct rec *r)
{
  return &(r->i);
}
```

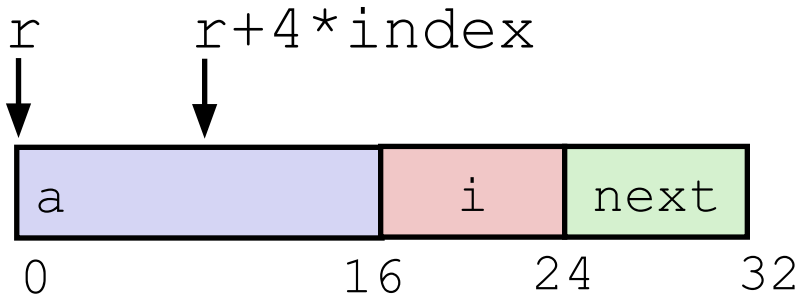
```
# r in %rdi
leaq  16(rdi),%rax
ret
```

```
struct rec** addr_of_next(struct rec *r)
{
  return &(r->next);
}
```

```
# r in %rdi
leaq 24(rdi),%rax
ret
```

# Generating Pointer to Array Element

```
struct rec {
    int a[4];
    long i;
    struct rec *next;
};
struct rec st;
struct rec *r = &st;
```




- Generating Pointer to Array Element
  - Offset of each structure member determined at compile time
  - Compute as: `r+4*index`

```
int* find_addr_of_array_elem
  (struct rec *r, long index)
{
  return &r->a[index];
}
```

`&(r->a[index])`

```
  # r in %rdi, index in %rsi
  leaq   (%rdi,%rsi,4), %rax
  ret
```

# Feelings check: Struct representation

# Review: Memory Alignment in x86-64

- *Aligned* means that any primitive object of $K$ bytes must have an address that is a multiple of $K$
- Aligned addresses for data types:

| | Type | Addresses |
|---|---|---|
| 1 | `char` | No restrictions |
| 2 | `short` | Lowest bit must be zero: $\ldots0_2$ |
| 4 | `int, float` | Lowest 2 bits zero: $\ldots00_2$ |
| 8 | `long, double, *` | Lowest 3 bits zero: $\ldots000_2$ |
| 16 | `long double` | Lowest 4 bits zero: $\ldots0000_2$ |

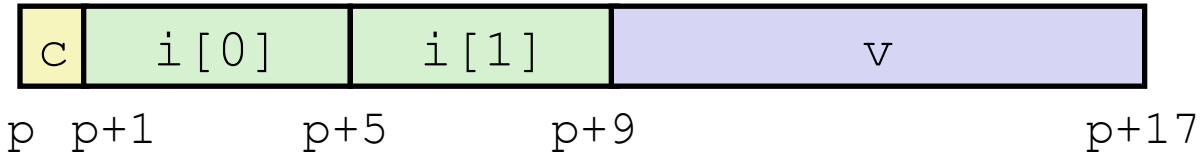
# Alignment Principles

- Aligned Data
  - Primitive data type requires $K$ bytes
  - Address must be multiple of $K$
  - Required on some machines; advised on x86-64

- Motivation for Aligning Data
  - Memory accessed by (aligned) chunks of bytes (width is system dependent)
    - Inefficient to load or store value that spans quad word boundaries
    - Virtual memory trickier when value spans 2 pages (more on this later)
  - Though x86-64 hardware will work regardless of alignment of data

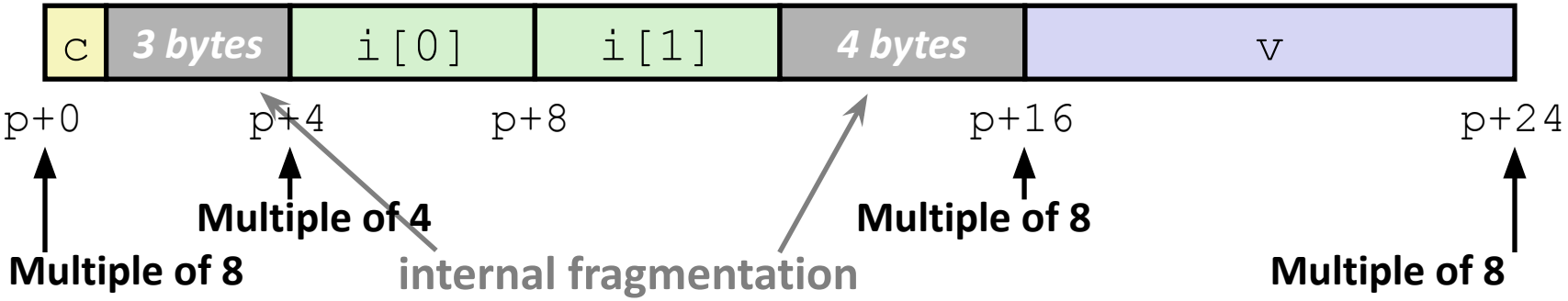
# Structures & Alignment

- Unaligned Data



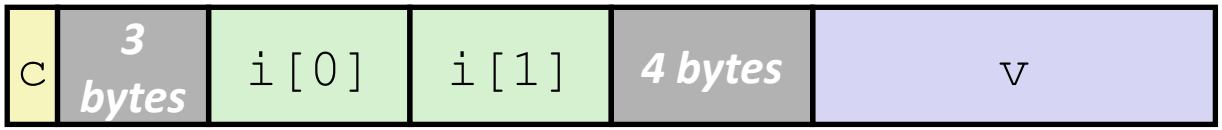

```
struct S1 {
  char c;
  int i[2];
  double v;
};
struct S1 st;
struct S1 *p = &st;
```

- Aligned Data
  - Primitive data type requires $K$ bytes
  - Address must be multiple of $K$



# Structs & Alignment: Fragmentation

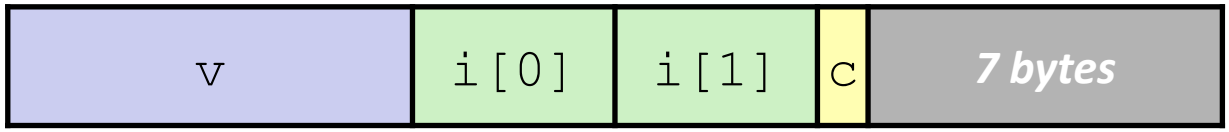
- Fragmentation occurs when there are unused portions of a struct
- Internal Fragmentation
  - Unused portion(s) occur *between*




p+0 p+4 p+8 p+16 p+24

```
struct S1 {
  char c;
  int i[2];
  double v;
};
```

- External Fragmentation
  - Unused portion at the end of the struct




p+0 p+8 p+12 p+16 p+24

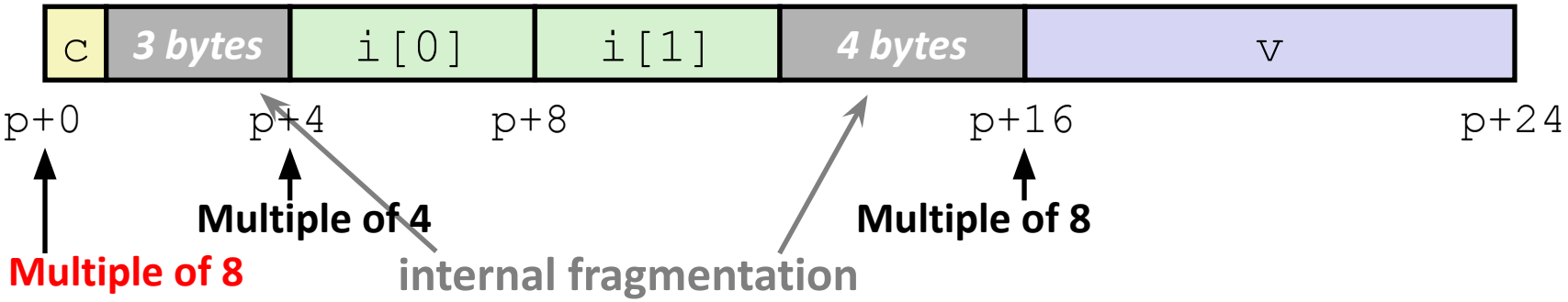
```
struct S2 {
  double v;
  int i[2];
  char c;
};
```

# Satisfying Alignment with Structs (1)

- Within structure:
  - Must satisfy each element's alignment requirement
- Overall structure placement
  - Each structure has alignment requirement $K_{\max}$
    - $K_{\max}$ = Largest alignment of any element
    - Counts array elements individually as elements

```
struct S1 {
  char c;
  int i[2];
  double v;
};
struct S1 st;
struct S1 *p = &st;
```

- Example:
  - $K_{\max}$ = 8, due to `double` element

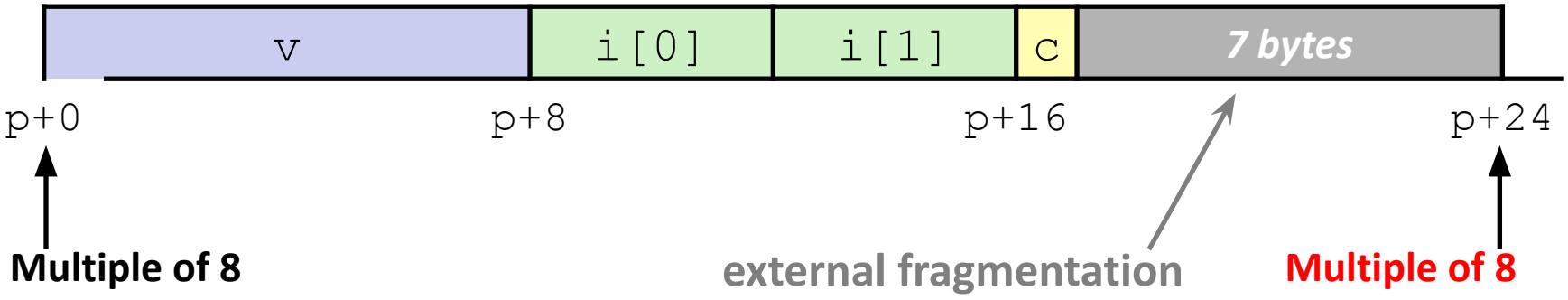


# Satisfying Alignment with Structs (2)

- Can find offset of individual fields using `offsetof()`
  - Need to `#include <stddef.h>`
  - *e.g.* `offsetof(struct S2, c)` returns 16

```
struct S2 {
  double v;
  int i[2];
  char c;
};
struct S2 st;
struct S2 *p = &st;
```

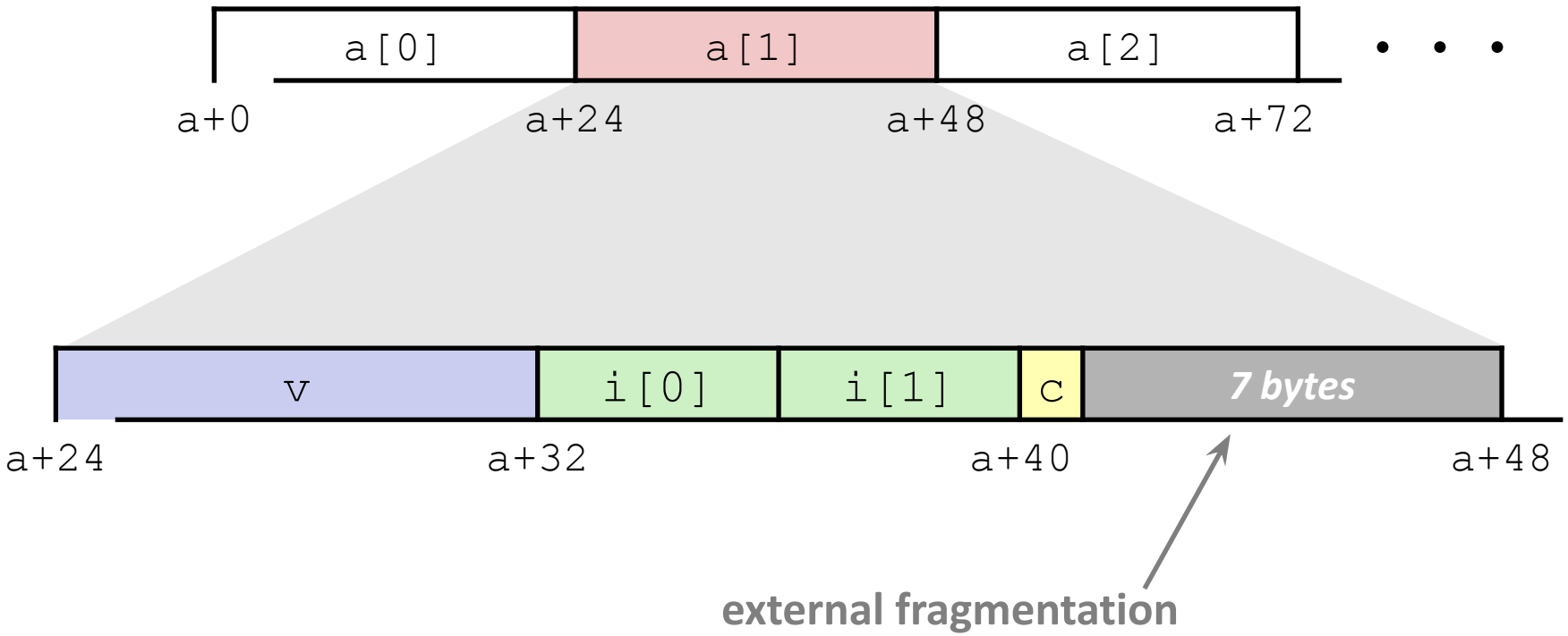
- For largest alignment requirement $K_{\max}$, overall structure size must be multiple of $K_{\max}$
  - Compiler will add padding at end of structure to meet overall structure alignment requirement



# Arrays of Structures

- Overall structure size multiple of $K_{max}$
- Satisfy alignment requirement for every element in array

```
struct S2 {
  double v;
  int i[2];
  char c;
};
struct S2 a[10];
```



# Alignment of Structs

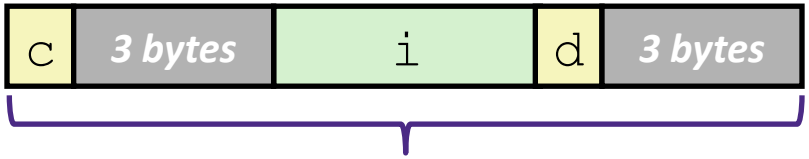
- Compiler will do the following:
  - Maintains declared *ordering* of fields in struct
  - Each ***field*** must be aligned *within* the struct *(may insert padding)*
    - `offsetof` can be used to get actual field offset
  - Overall struct must be ***aligned*** according to largest field
  - Total struct ***size*** must be multiple of its alignment *(may insert padding)*
    - `sizeof` should be used to get true size of structs

# How the Programmer Can Save Space

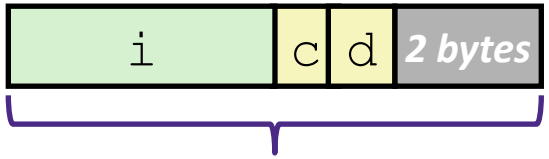
- Compiler must respect order elements are declared in
  - Sometimes the programmer can save space by declaring large data types first

```
struct S4 {
  char c;
  int i;
  char d;
};
struct S4 st;
```

```
struct S5 {
  int i;
  char c;
  char d;
};
struct S5 st;
```




12 bytes



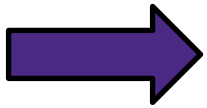

8 bytes

# Feelings check: Struct alignment

# Checking in: Structs

- Minimize the size of the struct by re-ordering the vars

```
struct old {
  int i;
  short s[3];
  char *c;
  float f;
};
```



```
struct new {
  int     i;
  _______ _______;
  _______ _______;
  _______ _______;
};
```

- How big is the old struct?

 16 bytes

 24 bytes

 Help!

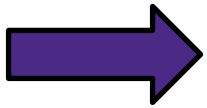
 28 bytes

 32 bytes

# Checking in: Structs

- Minimize the size of the struct by re-ordering the vars

```
struct old {
  int i;
  short s[3];
  char *c;
  float f;
};
```



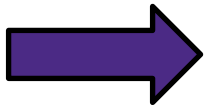
```
struct new {
  int     i;
  _______ _______;
  _______ _______;
  _______ _______;
};
```

- What's the layout for the new struct?

# Checking in: Structs

- Minimize the size of the struct by re-ordering the vars

```
struct old {
  int i;
  short s[3];
  char *c;
  float f;
};
```



```
struct new {
  int    i;
  float  f;
  char   *c;
  short  s[3];
};
```

- How big is the new struct?

 16 bytes

 24 bytes

 Help!

 28 bytes

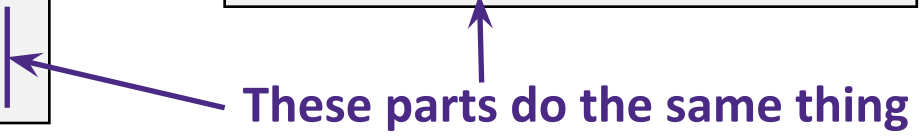
 32 bytes

# Aside: More Struct Definitions

- Can combine struct and instance definitions:

```
struct name {
  /* fields */
};
struct name st;
struct name *p = &st;
```

```
struct name {
  /* fields */
} st, *p = &st;
```




- Defines a struct type (`struct name`), an instance of that type (`st`), and a pointer to that type (`p`)
- This syntax is difficult to read
  - Mara doesn't like it in *most* situations because it conflates a type definition with an instance definition. But that's just their opinion…
  - We are showing it because you may see it in code in the future (and on the homework ☺)

# Aside: Typedef in C

- A way to create an *alias* for another data type:
  `typedef <data type> <alias>;`
  - After typedef, the alias can be used interchangeably with the original data type
  - *e.g.* `typedef unsigned long int uli;`
- Joint struct definition and typedef
  - Don't need to give struct a name in this case
  - `typedef` alone doesn't create an instance of the struct!

```
struct nm {
  /* fields */
};
typedef struct nm name;
name n1;
```

```
typedef struct {
  /* fields */
} name;
name n1;
```

# Summary

- Arrays in C
  - Aligned to satisfy every element's alignment requirement
- Structures
  - Allocate bytes for fields in order declared by programmer
  - Pad in middle to satisfy individual element alignment requirements
  - Pad at end to satisfy overall struct alignment requirement

# Last look around floor #2

- Arithmetic, conditionals, loops, functions, recursion at the assembly level
- How executables are created from programs
- Data structures in assembly (arrays & structs)
  - What can go wrong -- buffer overflows, inefficient memory usage with structs
- Neoliberalism & Processor values
  - Why we have so few companies..
- Accessibility, Agency, and Support

# How would we make C more accessible?

# Change the structure!

- Array Bounds Exceptions
- Floating point equality exceptions?
- Auto-reorder struct fields for efficiency?

- Structural changes for accessible spaces!

# But, what about the individual?

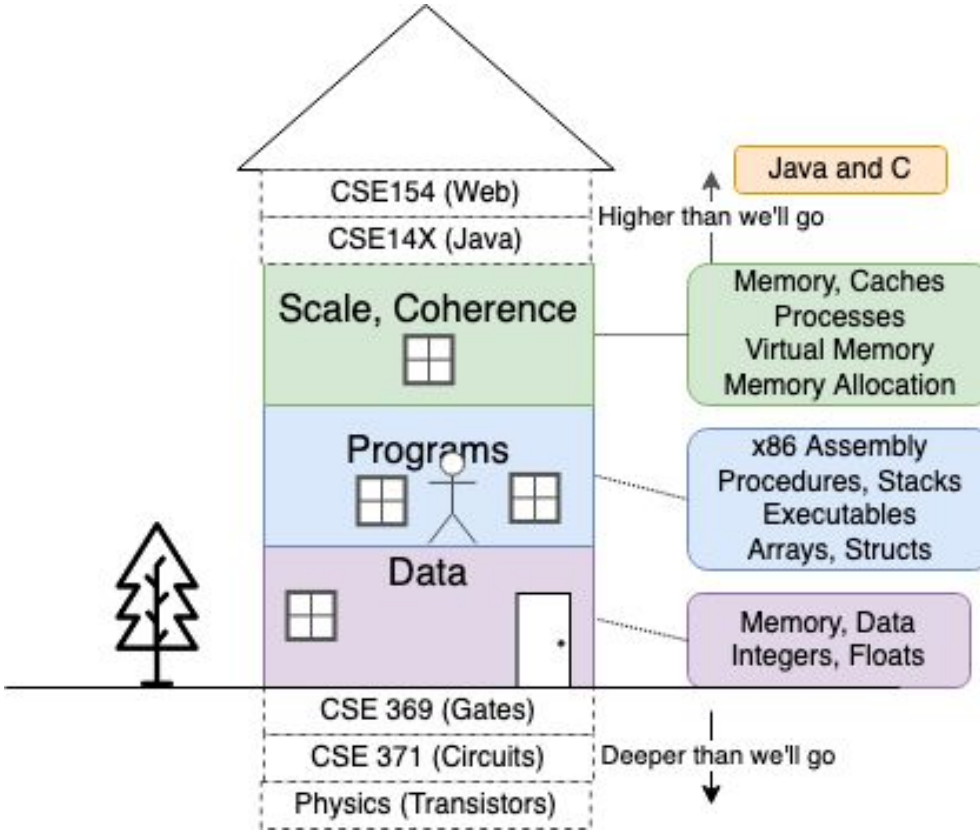
# Individual vs. Structure

- We've tried to emphasize structural changes over individual changes
  - i.e. "change C", instead of "don't do X when writing C"

- Largely reflects current rhetoric around justice
  - "Changing hearts & minds" -- hasn't worked
  - "Black faces in high places" -- hasn't worked
  - Structural oppression, structural justice

# *Structural Representation of a Discipline*

- What of this is accessible?
- Where can we find support?
- How much agency to residents have?
- What can we change?
- What do we need to look out for when we create change?
- How can we make change happen?



# So much needs to change!
## Why isn't change happening?

*Disclaimer: only one perspective, there are others

# Knowledge & Insulation

## (intended for L4, but we didn't have time)

# L4: Integers I, Bitwise Operators

- Operate on bit vectors
  - Operations applied bitwise
  - All of the properties of Boolean algebra apply

```
  01101001      01101001      01101001
& 01010101    | 01010101    ^ 01010101    ~ 01010101
  01000001      01111101      00111101      10101010
```

- Examples of useful operations:

$x \wedge x = 0$

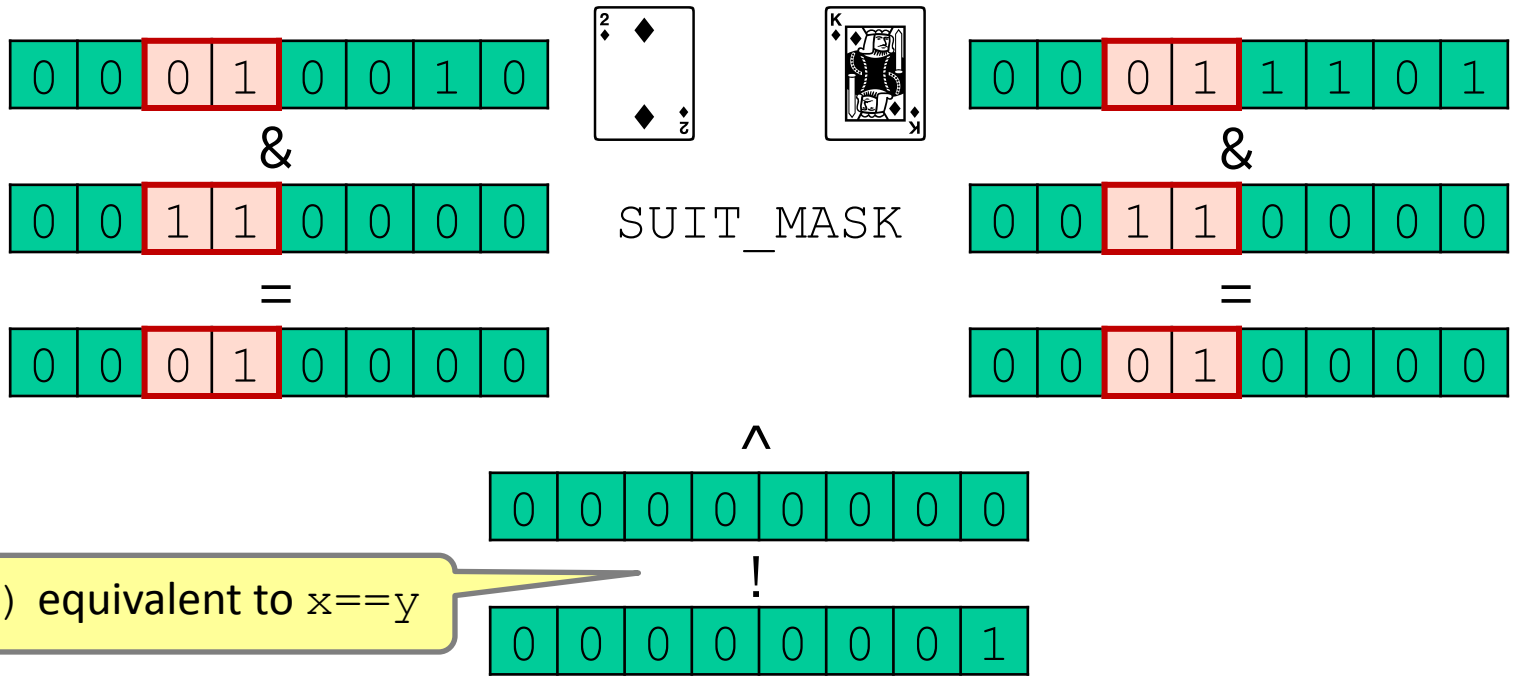
```
  01010101
^ 01010101
  00000000
```

$x \mid 1 = 1, \quad x \mid 0 = x$

```
  01010101
| 11110000
  11110101
```

# L4: Integers I, Bitwise Operators

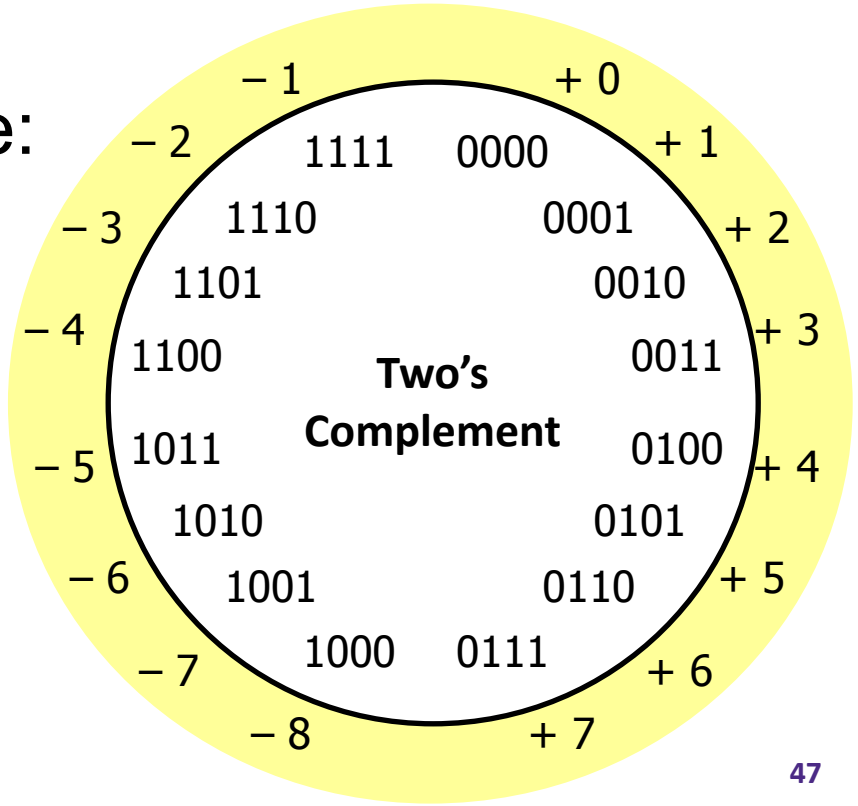
```
#define SUIT_MASK   0x30

int sameSuitP(char card1, char card2) {
  return (!((card1 & SUIT_MASK) ^ (card2 & SUIT_MASK)));
  //return (card1 & SUIT_MASK) == (card2 & SUIT_MASK);
}
```



# L4: Integers I, Bitwise Operators

- Roughly same number of (+) and (–) numbers
- Positive number encodings match unsigned
- Single zero encoding, all zeros

- Simple negation procedure:
  - Get negative representation of any integer by taking bitwise complement and then adding one!
    `( ~x + 1 == -x )`

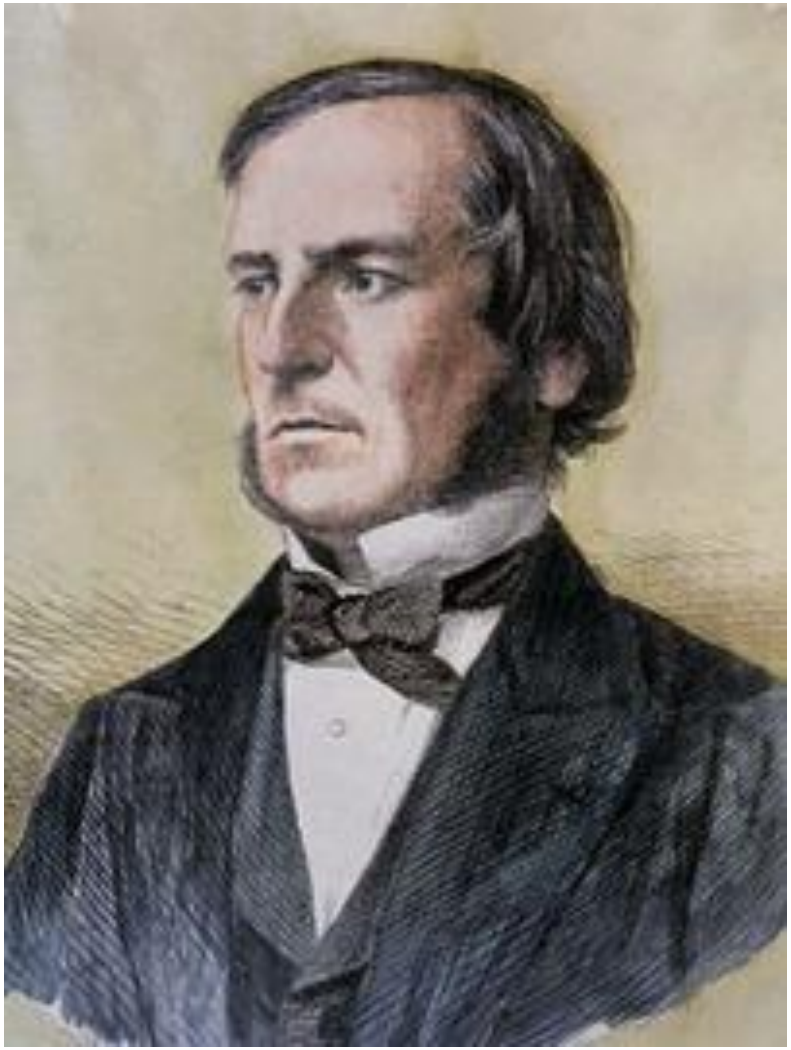


# Knowledge & Insulation

## (intended for L4, but we didn't have time)

# George Boole

- A committed educator!
  - Walked in the rain to teach, taught in wet clothes, wife wrapped him in wet blankets to cure him
- Philosophy, logic
  - Pseudo-religious doctrine of psychology
  - HEAVILY influenced by Indian systems of logic
    - Navya Logic!



# Colonialism in 19th Century

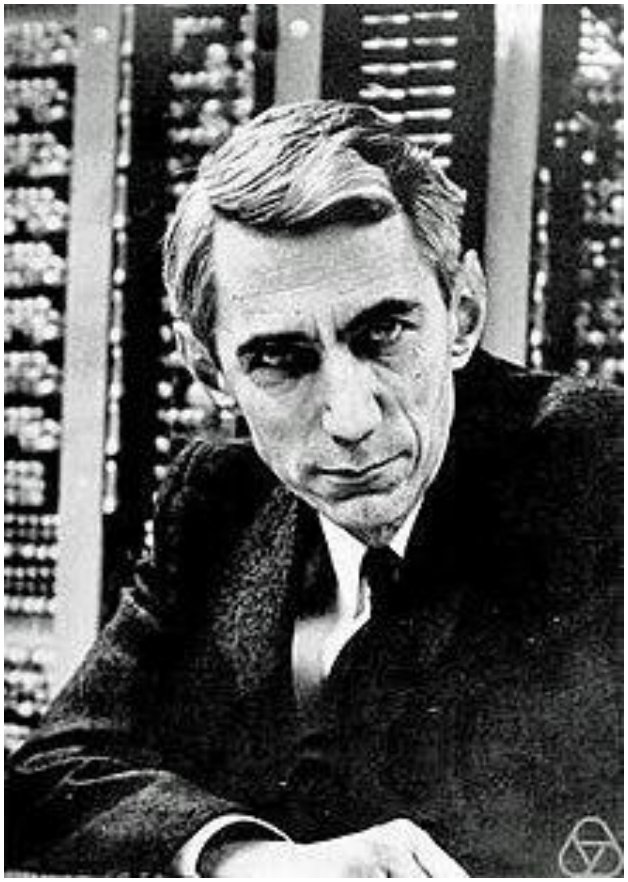
- 18th /19th century India under colonial rule
  - Re-education camps, the whole business

*"On examining this work I saw in it, not merely merit worthy of encouragement, but merit of a peculiar kind, the encouragement of which, as it appeared to me, was likely to promote native effort towards the restoration of the native mind in India."*

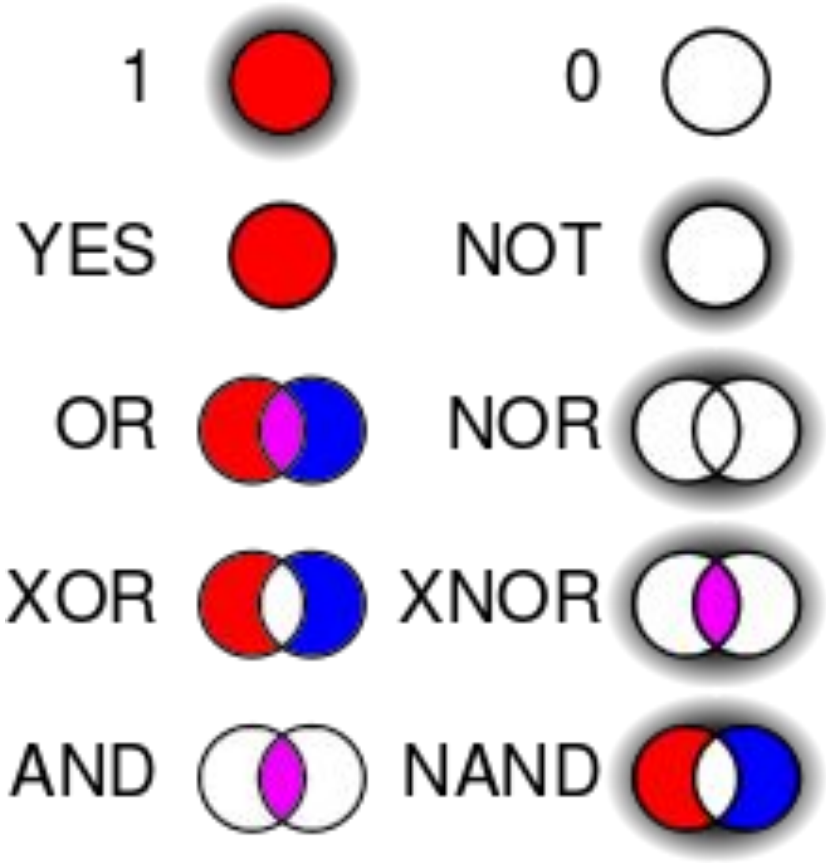
DeMorgan, prefacing: "A treatise on problems of maxima and minima, solved by algebra." by Ram Chundra, European Publication in 1859

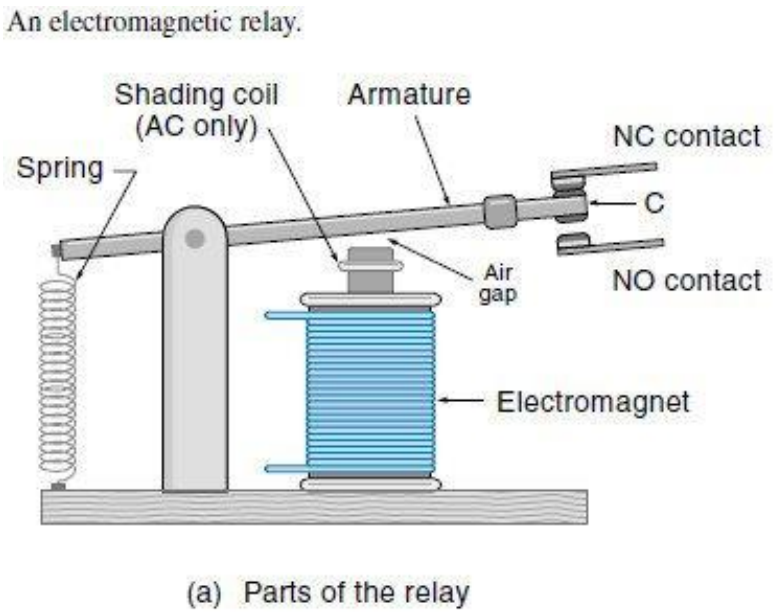
# Claude Shannon (1916-2001)

- Mathematician
  - Took a philosophy class! Not CS!
  - Learned Boole's algebra, realized relevance to digital logic
- 1937 Master's Thesis
  - "Founded" information theory



# I mean, it's a close connection




*Ba dum* ***dum***

# The “best” ideas come from outside!
## (Especially ideas for structural changes)

# Analogs: Depression, Neurodiversity

- I don't want to ask if any of y'all have experienced depression, anxiety, etc.
  - But, given statistics, I'd guess most of you?

- For me, I haven't realized how bad it was until I started asking for help
  - Including some pretty spooky suicidal spaces
  - I just assumed that everyone felt this way

# Analogs: Depression, Neurodiversity

- "I just assumed that everyone felt this way"
- For instance, everyone...
  - Has stretches where they don't leave their house
  - Gradually closes off contact with all support systems
  - Uses their work as a form of self-harm
  - Gets so focused on one thing they forget to eat
  - Can't distinguish sarcasm
  - Can't ignore the humming of fluorescents
  - Can't handle voices while they're falling asleep
  - Habitually avoids mirrors since puberty
    - *Lol, jk, this one's trans*

# Aside:

## Being trans/neurodiverse is so phenomenally amazing; I wouldn't change any of it

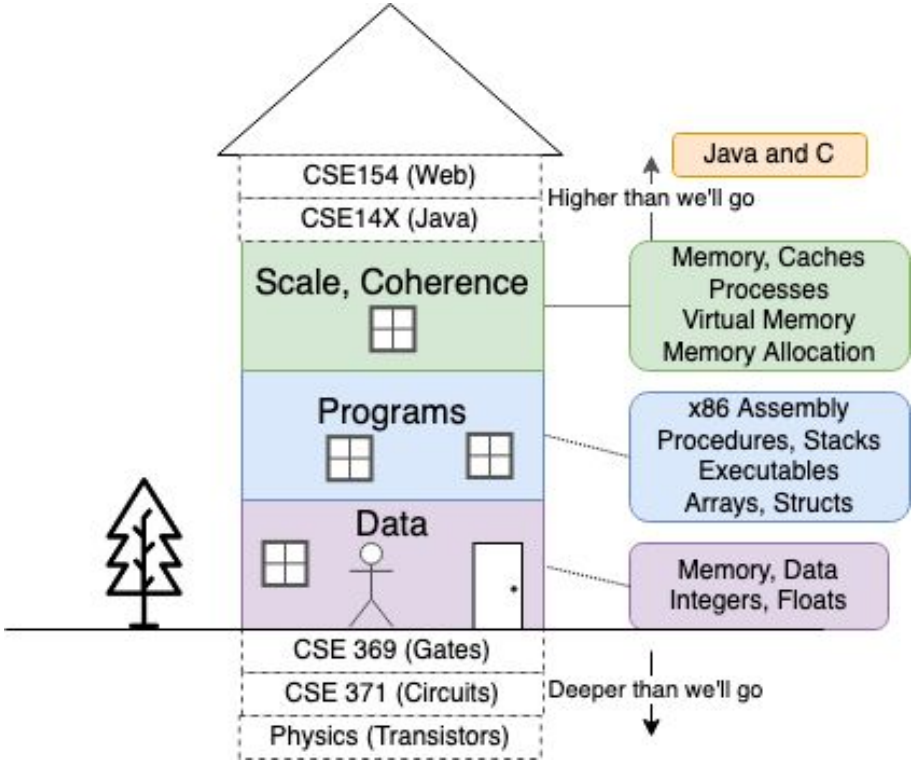
# Realizing what's happening

- I didn't understand until I "left the house"
  - Having someone tell me that this wasn't normal!
  - (In this case, the "house" was my own head, which is a phenomenally good space to isolate yourself)

- I realized three things:
  1. Not everyone felt this way
  2. Feeling this way wasn't sustainable (and matched a pretty high suicide rate)
  3. I could do something about it, even if that was just wearing earplugs
- I still isolate in my own head (grad student, remember), but it's become a much happier home

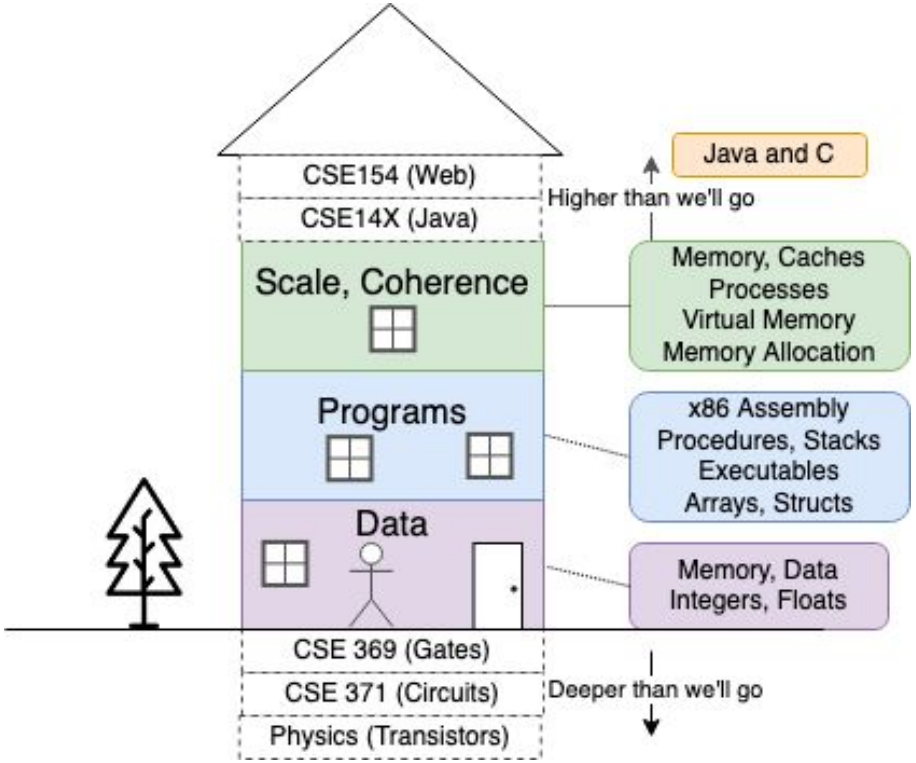
# I'd say that I "insulated" myself, as much as I could, from a world that didn't make space for me

# Insulation: a metaphor



# Insulation: a metaphor

- It keeps us warm!
  - It protects us from the outside!
- It can be harmful!
  - Asbestos, clearly
  - Fiberglass is still bad
  - Modern alternatives still harmful



# Insulation keeps us *comfortable and protected.*

# Insulation

- **George Boole**: European, described a concrete representation for thought & reasoning, influenced by Indian systems of logic
- **Claude Shannon:** Mathematician, takes a philosophy class, applies knowledge to switches to develop information theory
- **Me:** Built up personal insulation that would've killed me without external perspectives

**I think when you travel a lot, it's so chaotic, you're like "wow, it's amazing that everything doesn't go to hell". When you stay in one place, you think "why is everything going to hell".**

***Andrew Bird***

**It's hard to realize how bad things are, or how they could improve, without leaving insulated spaces.**

**It's hard to realize how bad things are, or how they could improve, without leaving insulated spaces.**

**So, why is CS still so insulated?**

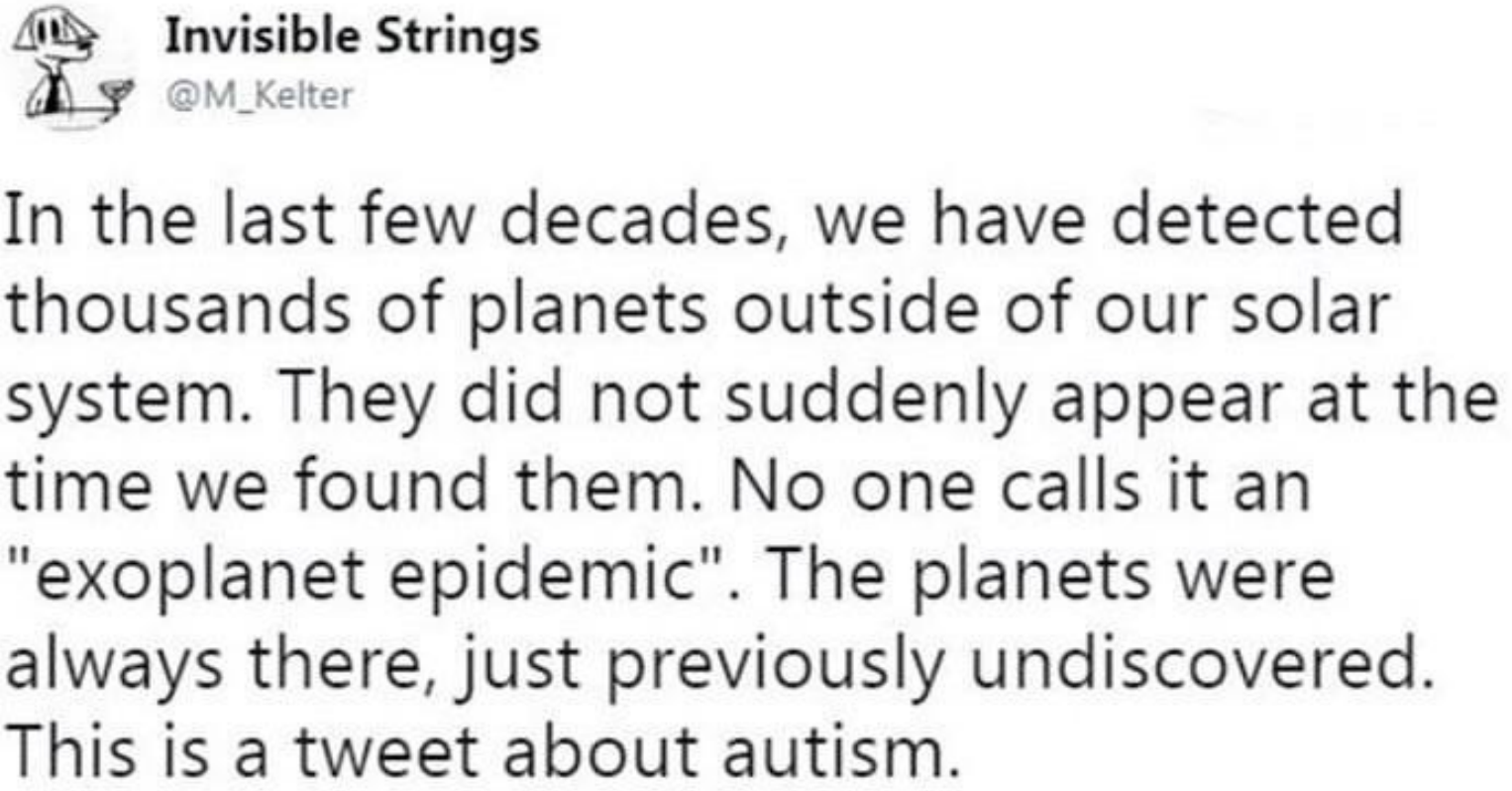
# Amy's Keynote (L4 reading)

- Briefly, CS has been a safe space for:
  - Women, whose jobs were automated by computers
  - Bullied kids that liked calculators
  - Wealthy "young geeks" (me)
  - Closeted trans kids (me)
  - Autistic people (me)
  - Queer folks trying to move to more progressive cities (SF/Seattle)
  - Those seeking social mobility, especially now

# Amy's Keynote (L4 reading)

- Briefly, CS has been a safe space for:
  - Women, whose jobs were automated by computers
  - Bullied kids that liked calculators
  - Wealthy "young geeks" (me)
  - Closeted trans kids (me)
  - **Autistic people (me)**
  - Queer folks trying to move to more progressive cities (SF/Seattle)
  - Those seeking social mobility, especially now

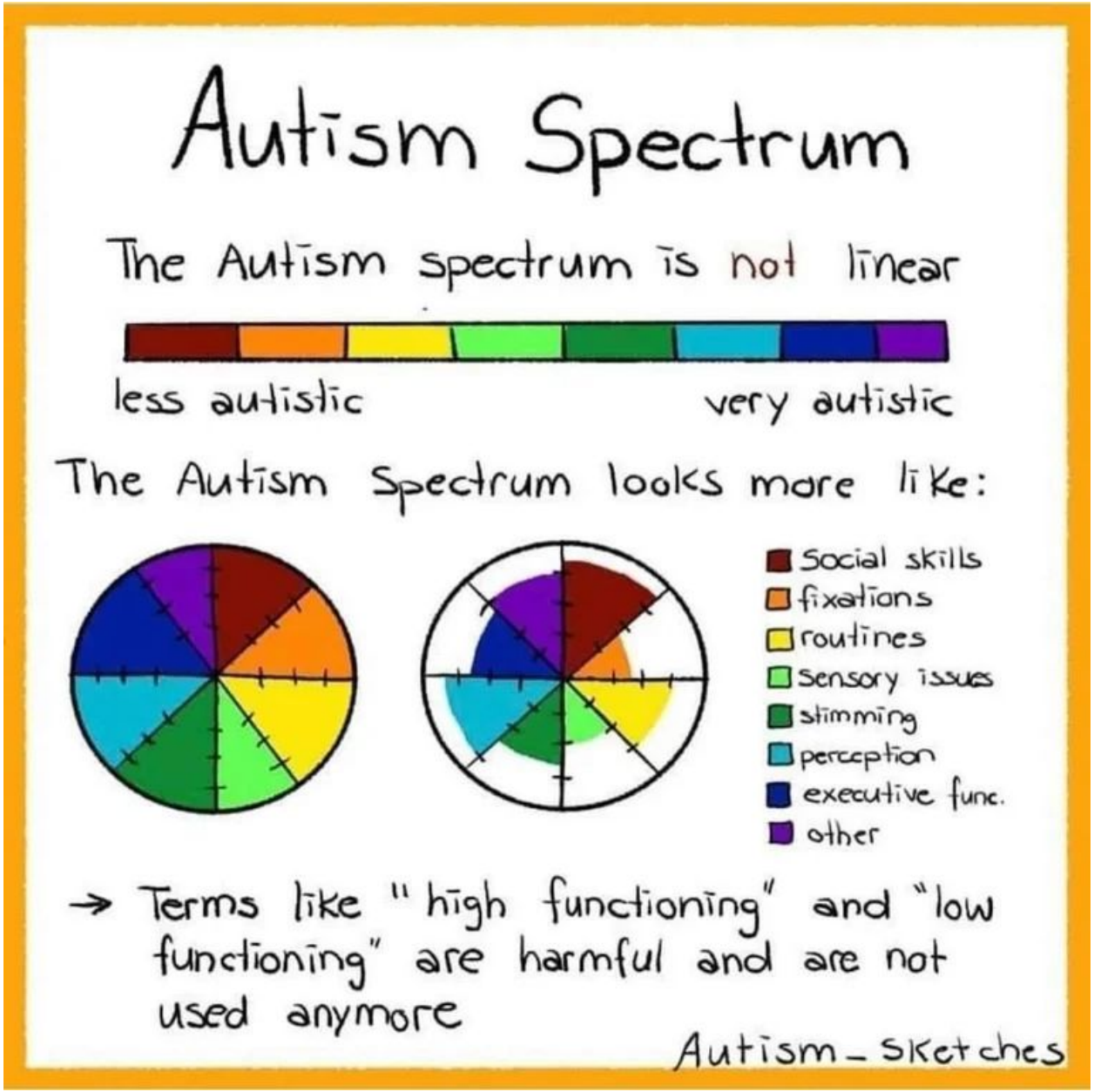
# Autism, briefly




More information @ https://autisticadvocacy.org/about-asan/about-autism/

# Autism, briefly

- Historically, as *Asperger's*
  - *Hans Asperger* was a Nazi, also Sukhareva's syndrome from Soviet Psychologist Grunya Sukhareva
- Historically viewed as a *spectrum*
  - Neurotypicality vs neurodiversity
  - "High-Functioning" vs "Low functioning"


Autism Spectrum
The Autism spectrum is not linear
less autistic
very autistic
The Autism Spectrum looks more like:
Social skills
fixations
routines
sensory issues
stimming
perception
executive func.
other
→ Terms like "high functioning" and "low functioning" are harmful and are not used anymore
Autism_Sketches

**Ellen Murray**
@ellenfromnowon

"High functioning" is used to deny support.

"Low functioning" is used to deny agency.

RETWEETS 187 LIKES 179

11:30 PM - 9 Mar 2016

Belfast, Northern Ireland

**Ellen Murray**
@ellenfromnowon

"High functioning" is used to deny support.

"Low functioning" is used to deny agency.

RETWEETS 187 LIKES 179

11:30 PM - 9 Mar 2016

Belfast, Northern Ireland

That's why we talked about agency/support last class!

# Autism, briefly

- Special interests: intense, passionate fixations on topics, “little professors”
  - Turing was “obsessed” with codes and ciphers, didn’t care about much else
- Theory of mind (some low, some high):
  - Understanding that the brains of others are different than yours (I struggle in some spaces)
- Empathy (some low, some high):
  - I have to be careful who I’m holding space for, others not so much
- Communication
  - Neurodiverse folks tend to communicate well with each other, less well with neurotypical folks

# Autism, and computing

- Attention to detail highly valued in CS
- Historically, CS is an intellectual space centering rules (theory) and routines (algorithms)
- Little social interaction required, "don't bother me" assumed
- Lots of space to pursue special interests without interruption!
- **Most of the world expect neurotypical conformity, most of CS emphasizes conformity along some* aspects of neurodiversity**
  - Again, UW is much better than my undergrad


WHAT YOU SEE
Detail oriented
Outgoing
Active
Super helpful
Hardworking
Performs well under pressure
Loyalty
VS
WHAT THEY'RE EXPERIENCING
Overthinking
People pleasing
Inability to slow down
Trouble saying no
Fear of failure
Procrastinating or overplanning
Poor boundaries

# Insulation and me

- I identify as neurodiverse (autistic), trans, enby
- CS ~~is~~ *was* a great closet…
  - Encouraged to be ”in my head”, outside my body
  - Focus on patterns, abstract concepts
  - Completely ignore society
- but, really, that’s no way to live
  - I promise, it would’ve killed me
- At some point, leave the house and go exploring!
  - Though camping can still be a stretch 😁

# I promise, I'm not alone

# I promise, I'm not alone

- Generally, transgender rate of 0.5% to 1%
- Autism Spectrum Disorder (ASD) around 1.5%
- 8-15% ASD among trans* folks!
- Higher rates of ASD in STEM than elsewhere
  - Most CS folks tend not to recognize neurodiversity
  - Especially with themselves, honestly

- I got to undergrad and realized that lots of folks were like me, realized later that most of them were autistic.
  - Though, everyone gets to pick their own labels

# Making Space in CS

- Foundations of computing:
  - Insulating for safety (i.e. being trans in Seattle, at UW)
  - Exploring new frontiers (even just philosophy)
- Both foundational!
  - Though, many focus on one or the other
  - Technical/Socio-technical; you need both!
- CS needs both to survive and thrive!
  - New ideas and safe spaces both necessary

# So, why is CS insulated?

# CS as a refuge

- It's been the only place that's safe for:
  - Women, whose jobs were automated by computers
  - Bullied kids that liked calculators
  - Wealthy "young geeks" (me)
  - Closeted trans kids (me)
  - Autistic people (me)
  - Queer folks trying to move to more progressive cities (SF/Seattle)
  - Those seeking social mobility, especially now

***Everyone*** **deserves support & agency.**

***Everyone*** **deserves a space that feels safe.**

# Remodeling the house of computing

- If you go in and start swinging hammers, you're going to scare some people
  - Fear sometimes manifests as aggression, especially among folks socialized as men
    - I only had access to anger for 15 years

- You might seem like an oppressor!
  - Even to those who've been oppressing others!
- There aren't many spaces where neurodiversity is valued!
  - There aren't many spaces that are safe!

# Insulation is deadly!

## But, going outside might seem even worse.

# Remodeling and Empathy

- Exploring into new spaces, especially for folks that resonate so strongly with routine and sameness, is a hard ask

- **I'm not asking you to excuse oppression, or make space for oppression!**
  - Just, please, if you're trying to change things, hold space for empathy, especially for those that might've hurt you.
  - You'll maintain the moral high ground!
  - You'll be more able to handle pushback!

# Loving-Kindness

1. Offer love and care to yourself, so wholeheartedly that it's like a waterfall
2. Offer love and care to others close to you, so wholeheartedly that it's like a waterfall
3. Offer the same love and care to strangers, people that you've never met
4. Offer the same love and care to those that have hurt you

- We're aiming for compassion for all beings here, it's a time-honored and quick way to justice

# Exploring Space, Making Space

- Goodness, if you can, take classes outside CS
  - Even better if they're non-technical!
  - *Philosophy*, Art, History, ***Education??***
- Claude Shannon, *Founder of Information Theory*
  - Go take a class because it looks interesting!
  - Maybe it'll be relevant to computing?
  - More likely, you'll learn something about yourself!

- This is (probably) the most intellectually diverse space that you'll be in for the rest of your life!
  - "Take a pottery class or something"

# CS is changing!

# CS is changing!

# CS is changing!
# What do you want to change?

# CS is still insulated!

# CS is still insulated!

## What can you bring to the house of computing? What do we need to know about that's outside the walls?

# We need insulation to survive, we'll die if we don't leave the house.

# We need insulation to survive, we'll die if we don't leave the house.

## Explore and make space, when you can!

# Data Structures in Assembly

This is extra (non-testable) material

- Arrays
  - One-dimensional
  - Multi-dimensional (nested)
  - Multi-level
- Structs
  - Alignment
- **Unions**

# Unions

This is extra (non-testable) material

- Only allocates enough space for the largest element in union
- Can only use one member at a time