CSE 13S 9/24/21 Lecture #1

TO DO:
VIRTUAL MACHINE running Ubuntu Linux
grammarly

memory
pointers
manipulating bits
actual editor
UNIX
tools like git, vim, make, cc, lldb etc.

cse13s.soe.ucsc.edu

design doc before code

Programs ~~are~~ assigned by Saturday night
Design docs due Thursday night pdf latex

P
Create ssh key
ssh: family of programs that let one computer communicate
with another
ssh - secure shell
scp - secure copy

public key: gitlab

```
mkdir cse13s
cd cse13s
git clone [illegible] git@git.ucsc.edu:cse13s/fall2021/jgu.git
```

git: source code control system
collab tool
Don't mess with contents of .git
commit and push regularly

Vi/Vim
Vi is the standard text editor found on Unix systems.
Vim is an acronym for "Vi IMproved" clone of Vi editor

commands are not obvious

```
# include <stdio.h>
int main
```

README.md
- what program does
- how to build it
- how to run it

DESIGN.pdf - how works, design, algorithms, problems solving, inputs, outputs,
WRITEUP.pdf - analysis of running your program, results

Design - shouldn't include C code pseudocode

```
git add
git commit -m "adding hello.c and READ.md"
git push

git add CHEATING.pdf
git commit -m " I accept
```

clang format

CSE 13S 9/21/21 Lecture #2

Derived from B, by Ken Thompson

Influenced by CPL and BCPL languages

PDP-11 processor

```
#include <stdio.h>   <- standard I/O package
int main (void)      no arguments
{
    printf("Hello, world!\n");   print
    return 0;
}
```

return int

output

0 = success

```
cc -o hello hello.c
./hello
```

right there   run

```
//    comment
#include <stdio.h>
int main(void){
 int fahr, celsius;
 int lower=0, upper=300, step=20;
  fahr=lower;
  while (fahr<=upper) {
      celsius = (5.0/9.0)*(fahr-32);
      printf("%3.0f%6.1f\n", fahr, celsius);
      fahr=fahr+step;
  }
  return 0;
}
```

%3.0f%6.1f

3 digits, 0 digits after decimal, float

pointer

```
char *s, c;
int i;
float f;
double d;
s= "This is a string";
```

((c = getchar()) != EOF
↑ end of file

↑
one char at a time

returns a character pointer * char
# define SYMBOL A 7 ← macro
↑
(compiler will see 7)

<< leftshift operation
SPREAD shift 1<<20   2^20

ASK

$$\begin{tikzcd}
\text{source code} \arrow[d, "\text{hello.c}"] & \\
\text{pre-processor} \arrow[d, "\text{hello.i}"] & \\
\text{compiler} \arrow[d, "\text{Assembly hello.s}"] & \\
\text{Assembler} \arrow[d, "\text{object code hello.o}"] & \\
\text{Linker} \arrow[d] & \text{Library} \arrow[l] \\
\text{executable hello} &
\end{tikzcd}$$

Stack
↓

↑
heap
uninitialized data

make automates ~~[illegible]~~

makefiles

gcc
- GNU C Compiler
- Default on Linux

cc
- unix / Linux environment variable that points to the default compiler

clang
- default on Mac and FreeBSD

CSE13S Lecture #3 9/29/21

$\wedge$	and	(&& in C)	conjuction
$\vee$	or	(|| in C)	disjunction
$\neg$	not	(! in C)	negation

exclusive or

$A \oplus B = (A \vee B) \wedge \neg (A \wedge B)$

$A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$

$A \oplus 0 = A$ $\quad$ $A \oplus 1 = \neg A$

$A \oplus A = 0$ $\quad$ $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

0 is false nothing else is false

logical expressions have type int

#include <stdbool.h> lets you have true and false

use {} with if statements always

```
scanf("%d", &n);   ← Read number
```

short-circuit evaluation

false && anything is false

true || anything is true

```
switch() {
  case  :
      break;
  case  :
      break;
  default;
}
```

```
goto        ← don't use
if (even)  goto even;
if (odd)   goto odd;
goto confused
```

```
for (int i=1; i<=10; i=i+1)

do {              ← [illegible] loop

    i=i+1;
} while(i<=10);

for (;;)
while(1)
```

CSE13S Lecture #3 9/29/21

$\wedge$	and	(&& in C)	conjuction
$\vee$	or	(|| in C)	disjunction
$\neg$	not	(! in C)	negation

exclusive or

$A \oplus B = (A \vee B) \wedge \neg (A \wedge B)$

$A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$

$A \oplus 0 = A$ $\quad A \oplus 1 = \neg A$

$A \oplus A = 0$ $\quad A \oplus (B \oplus C) = (A \oplus B) \oplus C$

0 is false nothing else is false

logical expressions have type int

#include <stdbool.h> lets you have true and false

use {} with if statements always

scanf ("%d", &n); ← Read number

short-circuit evaluation

false && anything is false

true || anything is true

```
switch() {
  case  ;
     break;
  case  ;
     break;
  default;
}
```

goto ← don't use

```
if (even)  goto even;
if (odd)   goto odd;
```

goto confused

```
for (int i=1; i<=10; i=i+1)

do {                bottom test loop

    i=i+1;
} while(i<=10);

for (;;)
while(1)
```

CSE 135 Lecture #4 10/1/81

In math function maps elements of a set called the domain onto a set called the range.

function in C is a block of code that performs a certain task.
- defined once, declared & call multiple

main() special, run when start, all other functions are subordinate

functions should:
- Define abstractions that are consistent and makes sense logically.
- Give names to those sequences of code.
- Hide implementation

```
return_type function_name (parameters)
{

}
```

$f(x) = x \log(x+1)$ and we write $f(2)$ sub 2 for x to get $2 \log(2+1)$

call ~~by name~~ rare

Formal parameters
    parameter that is used inside function body
Actual parameters
    - value of the value that is passed to the function
    - value can be copied to formal parameter

~~h = (x<1) ? 1 : x;~~

if h is less than 1, it is 1 otherwise x

* pointer
& address

put functions before main()
Macros in C operate through text replacement
#define PI 3.1415926
✓ sees 3.1415926 not PI
2 * PI * radius
return

pragma once - cleaner

ifndef - portable

static variable exists or persists across function calls

```
static inline bool even(int64_t n){
    return n%2 == 0
}
```