ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 31

Vince Weaver

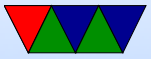
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

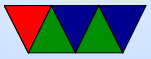
20 April 2026

Announcements

- HW#7 was announced
- Makeup lab this week, try to get any missing labs checked off
- Will monitor the need for lab prescence next week
- Reminder: no class Wednesday (the 22nd)
  Instead it's the senior project expo from noon-2pm
  Atrium outside the Hill Auditorium in Barrows Hall
  Will have something (quiz?) instead of attendance
- Title II warning



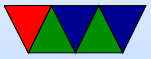
- Plan for rest of semester
- Student Evals



HW7 Update – Functions

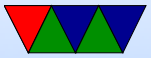
- Mostly functions, a bit long
- People mostly OK on this?
- One tricky thing is you can pass functions as arguments to other functions

```
result=foo( bar(), baz());
```



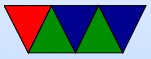
HW7 – Biggest Trouble (powerTo function)

- It wants you to write `powerTo()` but not using the math `exp()` function
- two arguments, double and an int
- Be sure if negative return 0
- Otherwise, use a loop. Be sure to start with a value of 1 (value to power 0 is 1) and then in a loop from 0 to the int parameter just multiply by the double parameter



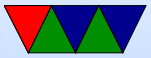
More File I/O

- Gave the most common stuff, there is more you can do
- Shame no lab that uses it
- Covered stuff you'll need in my later classes
- If you're trying to do a fully featured app there are some other things you might want to do



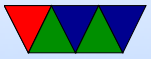
Advanced I/O

- This becomes more of a Linux issue than C issue
- Many of the Linux utils like `cp` (copy file), `mv` (move file), `rm` (remove file) were originally thin wrappers over these calls



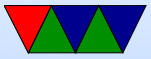
Creating a File

- `creat()`
- `open()` has special `O_CREAT`
- `fopen()` be careful, "w" means truncate or creat by default, if you want to write to existing file need "a" for append



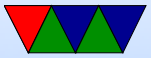
Erasing a File

- `unlink()`
- Why it's called this is complicated; on Linux multiple filenames can point to one file, so unlink() just removes the filename, the file doesn't totally go away until last link is gone
- It also doesn't go away until the last open file descriptor to it is closed. This is why erasing files might not clear up disk space



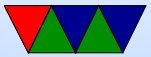
Moving/Renaming a file

- Isn't necessarily same as copy/erase
- On Linux the contents and name are separate, so renaming is as easy as just updating the name



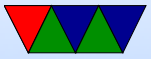
Creating/Removing directories

- `mkdir()`, `rmdir()`



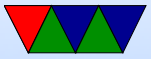
Scanning Directories for Files

- `opendir()`
- `readdir()`
- Way more complicated than you think. What if there are a lot of files so don't have room to read them all in? Just call again. But what if people have added/removed files since you read the first time?



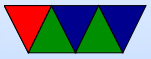
Finding size of files

- `stat()` family of functions
- `statvfs()` for filesystem (free disk space)



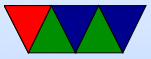
math.h

- For simple integer math we've already covered most of it and you don't need to do anything special
- Floating point doesn't need anything special for simple addition / subtraction, etc, or printing
- More advanced math though you might need to use the math library and include the `math.h` header



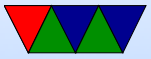
math library

- Back in the day when computers were smaller it seemed a waste to include a floating point library into the C library if only a subset used it
- (As an aside, back then floating point hardware was often an expensive add-on to a system, and emulating in software was slow)
- So there's a separate math library. If compiling yourself you have to put `-lm` as an option on the command line



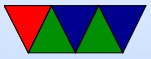
IEEE-754 standard floating point

- Probably implements IEEE-754 standard



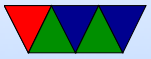
aside – why use C for math?

- Are there better alternatives?
- python?
- matlab?
- R?



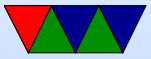
Useful Constants

- `M_E` – e
- `M_PI` – pi
- `M_PI_2` – pi/2
- `M_PI_4` – pi/4
- `M_2_PI` – 2/pi (*not* 2*pi)
- `M_SQRT2` – square root of 2
- `NAN` – Not a Number
- `INFINITY` – Infinity
- there are many more



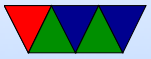
Error Returns

- If result is invalid e.g. `log(-1)` errno set to `EDOM`
- If result gets too big will return `HUGE_VAL` or maybe infinity and errno is `ERANGE`
- If result underflows (goes too close to 0) will return `0` and errno is `ERANGE`



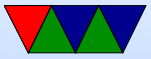
special values

- Not a Number (Nan)
- inf,-inf
- Note: NaN!=NaN
- Can detect with `isnan()`, `isinf()`
- quiet vs signalling NaNs



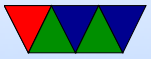
Floating Point Exceptions

- Maybe seen this already, with integer math if you divide by 0, or mod by 0, program will crash with "floating point exception" `SIGFPE`
  (we haven't talked about signals yet)
- If you divide a double or float by 0 though, you don't get an exception. `printf("%lf\n",128.0/0.0);` will giv you `inf`
- This is a soft
- To force floating point exceptions on Linux try
  `feenableexcept(FE_DIVBYZERO | FE_INVALID | FE_OVERFLOW );` (note



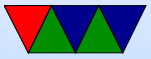
also need `#define _GNU_SOURCE` at start of file

- Also available `FE_UNDERFLOW` and `FE_INEXACT`. That last one is really common so not that useful
- Sometimes if doing parallel SIMD math, having to detect exceptions makes it run slower, `-fno-trapping-math`. Eternal tradeoff between speed and safety



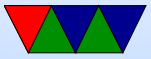
trig functions

- Take input in radians
- `sin()`, `cos()`, `tan()`
- hyperbolic: `sinh()`, `cosh()`, `tanh()`
- `sincos()` – both at same time
- `acos()`, `asin()`, `atan()`
- `atan2()` – two arguments x/y and based on signs can correctly determine which quadrant



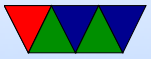
variants

- Starting with ISO/IEC 9899:1999 many functions have float and long float variants in addition to the default double
- `sin()` – returns double
- `sinf()` – returns float
- `sinl()` – returns long

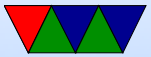


rounding

- Careful, some of these might work different than how you expect on negative numbers
- `ceil()` – ceiling, next highest integer
- `floor()` – floor, next lowest intger
- `trunc()` – nearest integer, toward zero
- `trunc()` – nearest integer, away from zero
- `nearbyint()` – use current round more `rint()` same but possibly raise exception
- `lrint()` return long int, not float

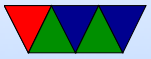


- `round()` – always round half-way case (0.5) away from zero



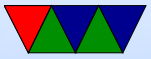
square root

- `sqrt()` return non-negative square root
- `cbrt()` cube root
- `hypot()` – calc hypotenuse (euclidian distance) sqrt(x*x+y*y)



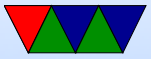
Powers

- `exp()` – returns e^x
- `pow()` – returns x^y
- `exp10()` – returns 10^x
- `exp2()` – returns 2^x
- `expm1()` – returns $e^x - 1$. Why special? ix x is small doing things separately might be inaccurate



Logarithms

- `log()` – natural logarithm
- `log10()` – log base 10
- `log2()` – log base 2
- `log1p()` – natural log 1+x (again, in case x is a small number)
- aside, do you remember how to convert between different log bases?

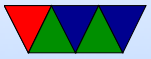


complex numbers

- C99 can do complex numbers
- include complex.h
- in that case I is imaginary number

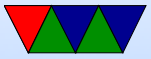
```
double pi = 4 * atan(1.0);
double complex z = cexp(I * pi);
printf("%f + %f * i\n", creal(z), cimag(z));
```

- `csin()`
- `csqrt()`
- cexp, cpow, clog
- cproj – project into Rieman sphere



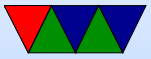
Matrix math

- C doesn't have built in vector or matrix types
- Arrays are similar but multiplying arrays is not going to give you matrix math
- Cluster computing class
- Probably want to use a BLAS library



arbitrary precision

- Floating point isn't exact
- Rounding issues
- Problem is it's binary
- Some values well behaved in decimal not in binary, for example 1/10 (.1) is a repeating fraction in binary
- Especially for things like money you want exact decimal values, not fp extractions. Special decimal math libaries/hardware to avoid rounding error
- Also might have numbers that lose precision in floating



point

Video games, get too far from origin (kerbal space program, minecraft)

- Custom libraries that can do exact values for very large numbers
- GNU MP (GMP) multi-precision library