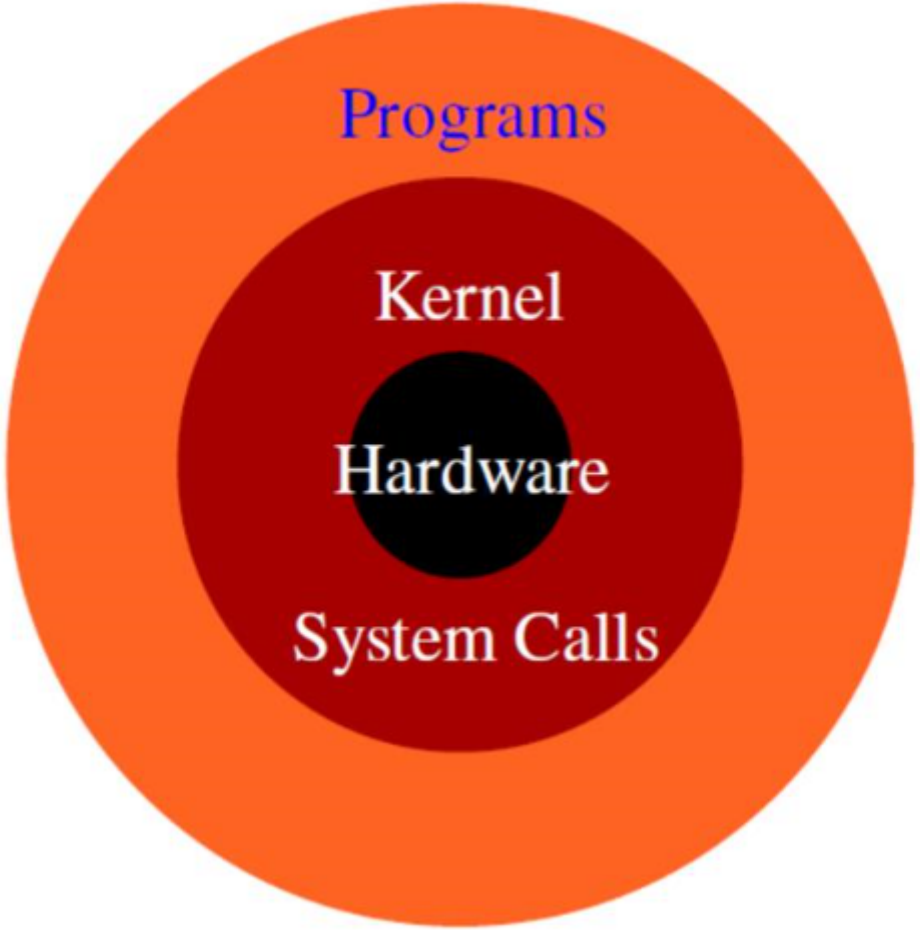
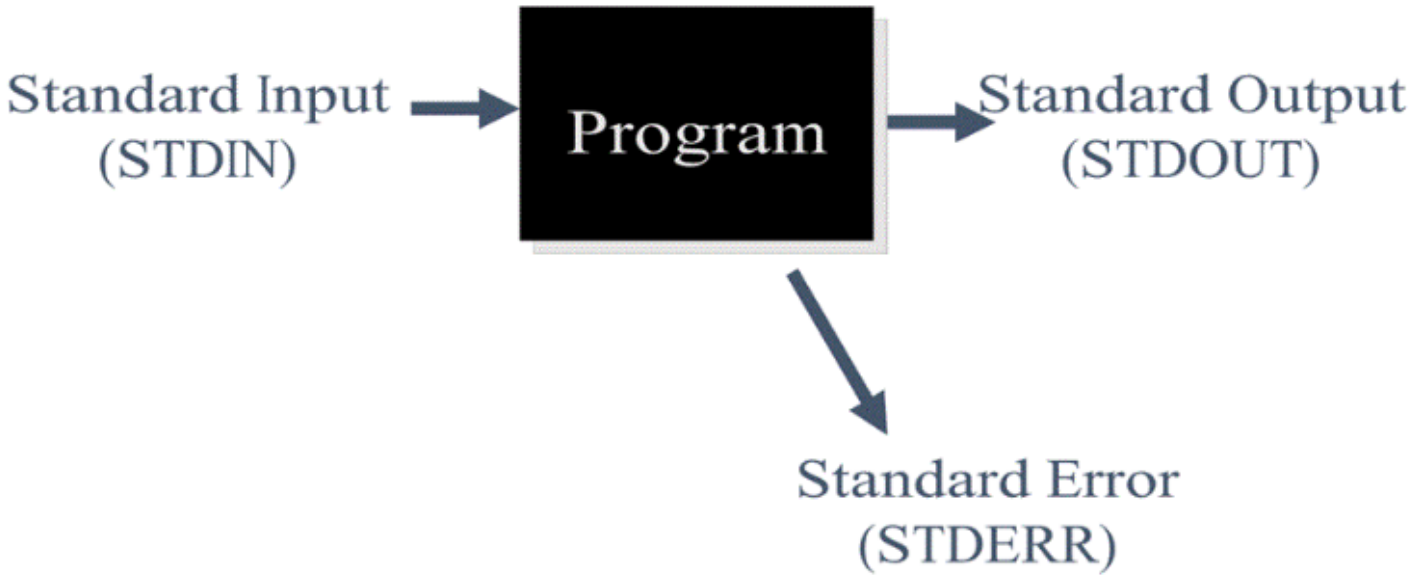
Course Summary (1)

- Your code requires help from many layers to get work done:
  - How are operating systems structured
  - How does the OS help us compile, execute, and run code. The fact that THIS REALLY MATTERS!
- Multiple ways to interact with the Operating System:
  - Shell and built-in programs
  - Our own code through C libraries
  - Our own code making system calls



Course Summary (2)

- To do interesting work, our programs need to take input and produce output
- Method 1 was standard input and output. Know how these work by default, and how to change them
- Method 2 was file IO. Text and binary modes important for different cases.



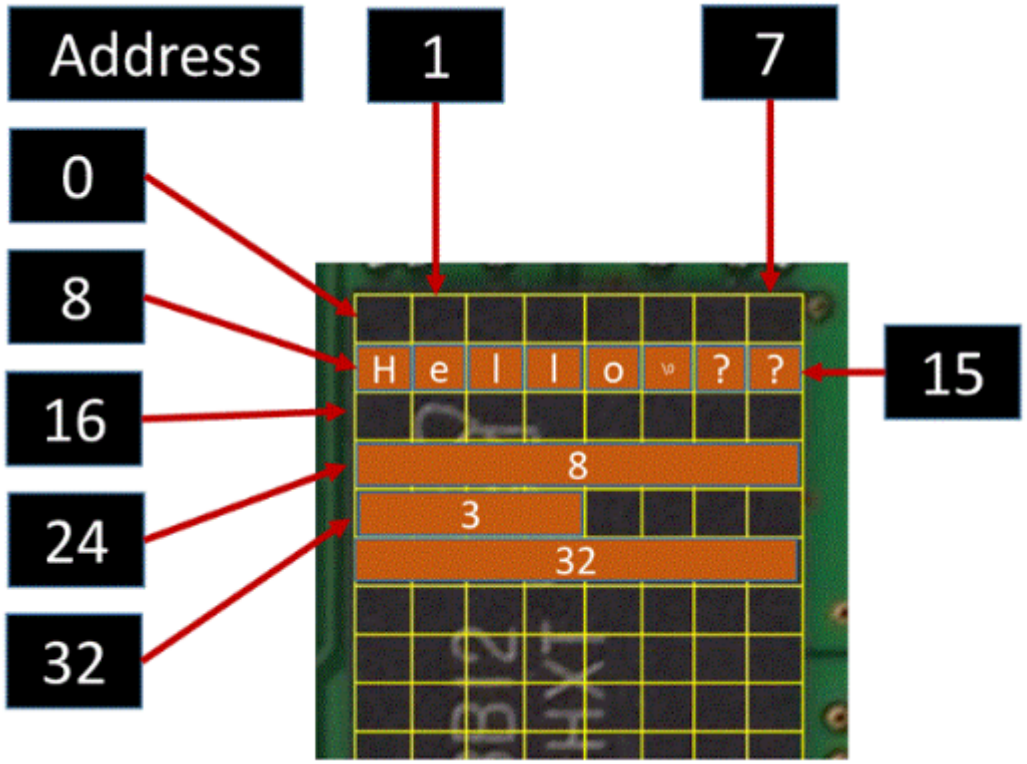
Course Summary (3)

- Data is essential to form useful systems. This data has natural types, but in C we can "mis-use" these.
- For example, packing many "colors" of an image into a 32-bit int
- Knowing these tricks and tools let us handle new data that arises (new user requests, new sensors)

Description	Type	Bits	Range
Integer	short int long	16 32 64	+/-32 thousand +/-2.1 billion +- 9.2×10^{18}
Floating point	float double long double	32 64 128	+/- 10^{38} +/- 10^{308} +/- 10^{4932}
Character	char unsigned char	8	-127 to 128 0 to 255
Pointer	char* int* (etc)	64	0 to 1.8×10^{19}

Course Summary (4)

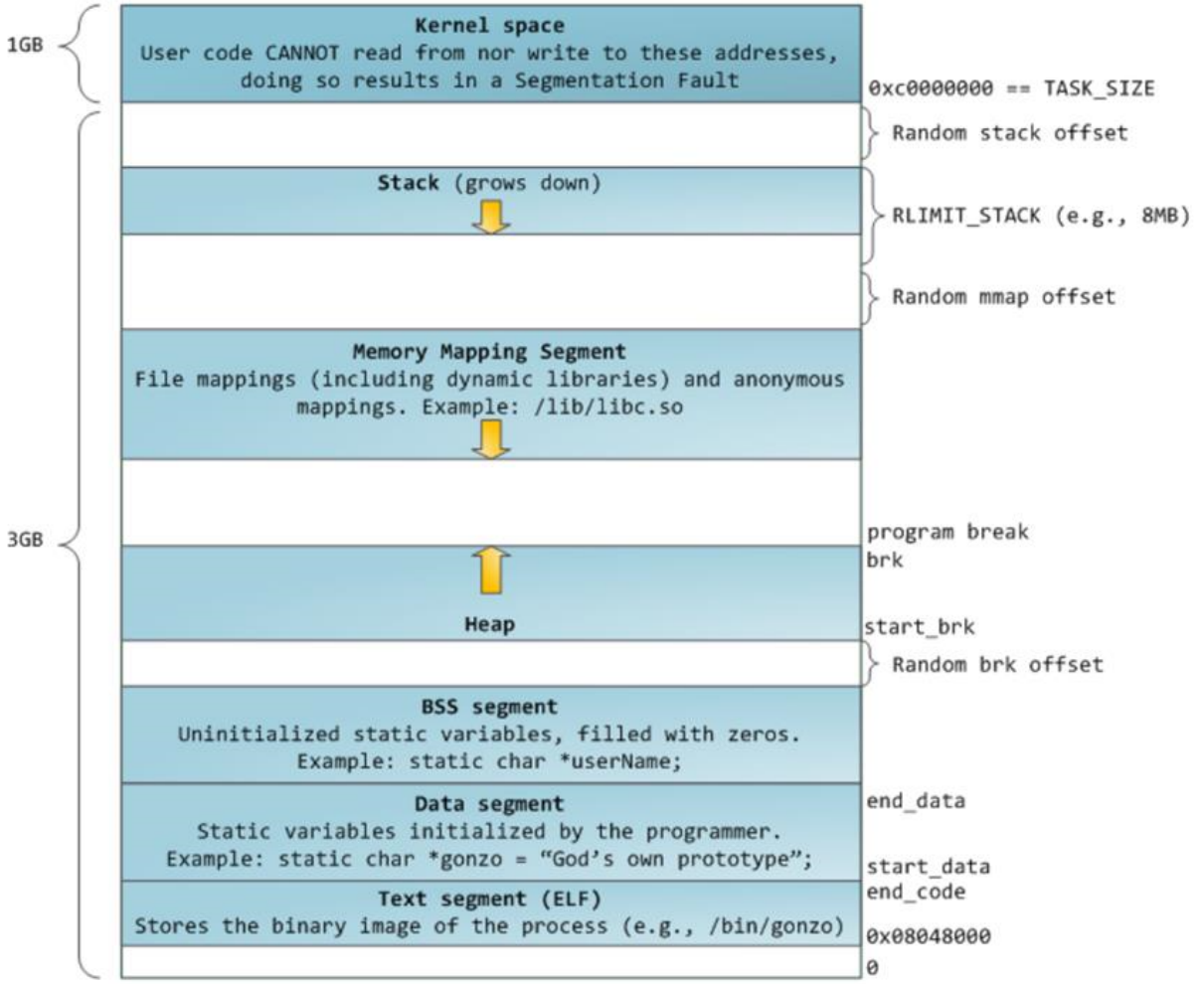
- C strings are character arrays
- C arrays and pointers are nearly equivalent:
  - Important to know the exceptions!
- Addresses to memory and sizes of variables become really important tools
- In C, pointers are used to deal directly with data (e.g., write()) and to make code re-usable (function pointers)



Course Summary (5)

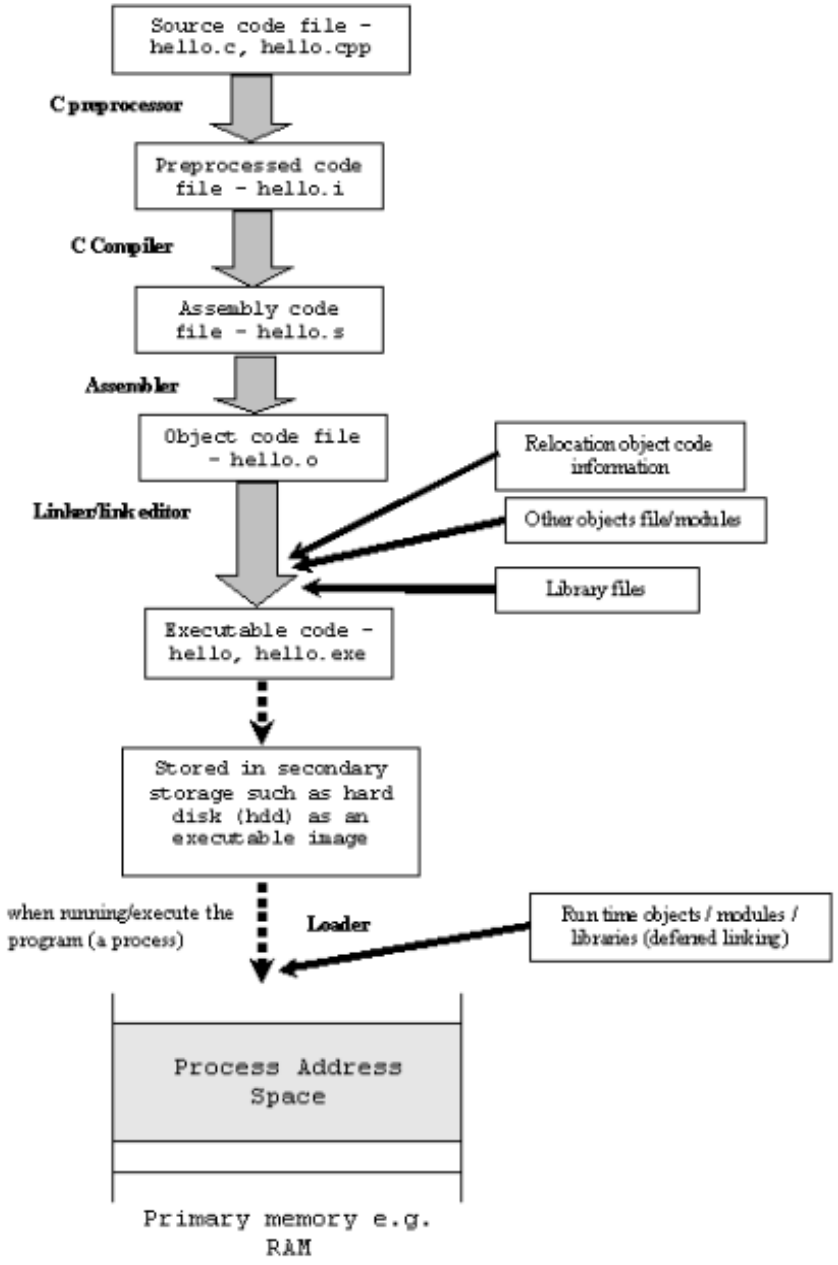
- The structure of a running process in memory has several impacts:
  - Stack memory is temporary
  - Heap memory has programmer-controlled lifetime
- Every process can access the kernel through system calls
- Compiled code is included

- Knowing this allows plug-ins (Adblock), hacking (buffer overflow), multi-process



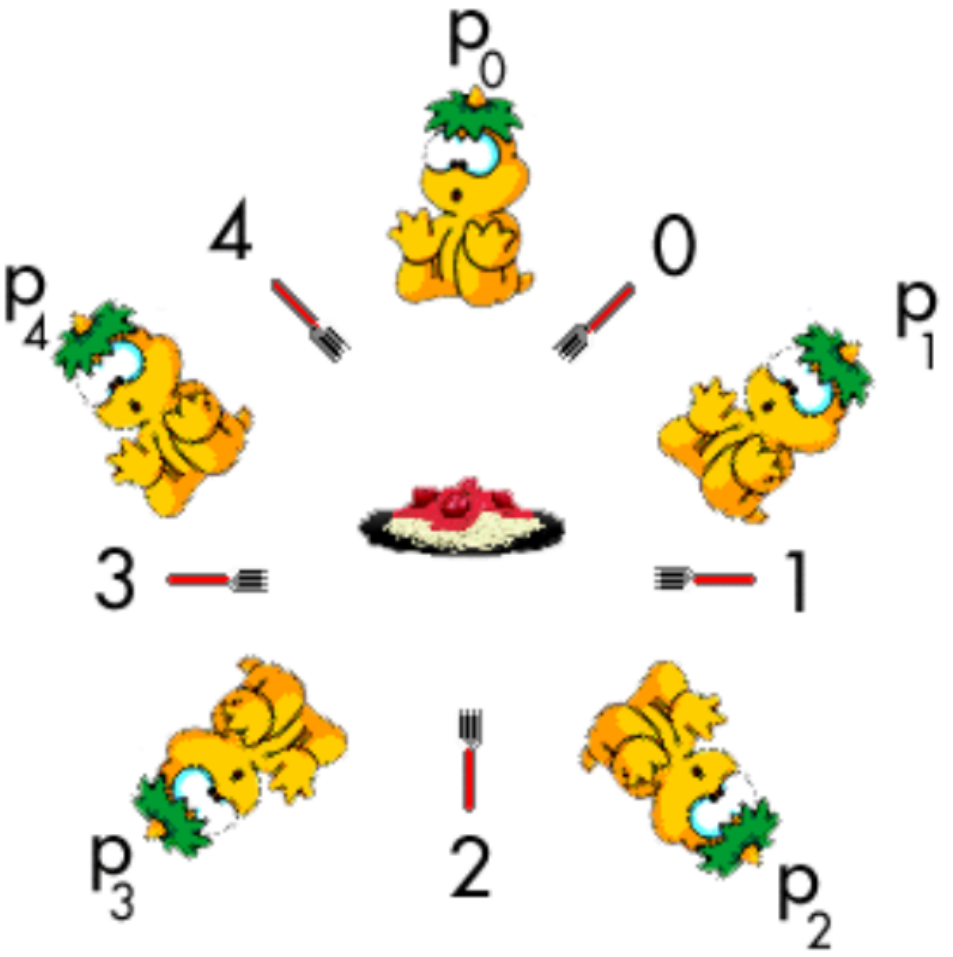
Course Summary (6)

- Compiling in C is not a single step. Knowing about the different phases gives us more opportunity to influence the final program:
  - Pre-processor debugging
  - Using shared libraries to customize code (or hack it)
- The process is usually automated with Makefiles (and even more high-level tools like Cmake, we did not cover this year)



Course Summary (7)

- Software systems are almost always distributed to include many processes, across many machines
- Dining Philosophers was one illustration of how this can go wrong
  - Semaphores are a proposed fix for this proposed by Dijkstra
- Two generals problem is about errorful comms and is impossible to solve perfectly:
  - Today, block-chain solves the Byzantine Generals problem, a variant about consensu in presence of traitors



Course Summary (8)

- The internet forms a world-wide software system:
  - IP includes addressing and builds networks
  - TCP includes ports and congestion control for fair usage by all
  - Each level wraps the previous in packets with headers (similar to BMP)
- The web sits on top of these, with HTTP request/responses defining the web client/server protocol