Programming in C - Advanced Features

1.) Introduction of C

C is a language for small, fast programs
how do you run the program?
Two types of command

2.) Memory and Pointers

Using memory pointers
How do you pass a string to a function?
Array variables are like pointers…
Why pointers have types
Using pointers for data entry
scanf()
fgets() is an alternative to scanf()

3.) Strings: *String theory*

Create an array of arrays
Find strings containing the search text
Using the strstr() function
Array of arrays vs. array of pointers

4.) Creating Small Tools

Introducing the Standard Error
fprintf() prints to a data stream
Connect your input and output with a pipe
The bermuda tool
Creating your own data streams
There's more to main()

5.) Using Multiple Source Files

Data types
Type Casting
Creating your first header file
The shared code needs its own header file
Automate your builds with the make tool

6.) Structs, Unions, and Bitfields

Create your own structured data types with a struct
You need a pointer to the struct
(*t).age vs. *t.age
A union lets you reuse memory space
An enum variable stores a symbol
Bitfields store a custom number of bits

7.) Data Structures and Dynamic Memory

Linked lists are like chains of data
Linked lists allow inserts
Create a recursive structure
Create islands in C..
Inserting values into the list
Use the heap for dynamic storage
Give the memory back when you're done
Ask for memory with malloc()…
Let's fix the code using the strdup() function

8.) Advanced Functions

- Pass code to a function
- Every function name is a pointer to the function…
- How to create function pointers
- Get it sorted with the C Standard Library
- Use function pointers to set the order
- Create an array of function pointers

9.) Static and Dynamic Libraries

- Angle brackets are for standard headers
- Sharing .h header files
- Share .o object files by using the full pathname
- An archive contains .o files
- Create an archive with the ar command…
- Dynamic linking happens at runtime
- create an object file

10.) Processes and System Calls

- The exec() functions
- The array functions: execv(), execvp(), execve()
- Running a child process with fork() + exec()

11.) Inter-process Communication

- Redirecting input and output
- A look inside a typical process
- Redirection just replaces data streams
- fileno() tells you the descriptor
- Connect your processes with pipes
- Case study: opening stories in a browser
- Opening a web page in a browser
- The death of a process
- Catching signals and running your own code
- Rewriting the code to use a signal handler
- Use kill to send signals

12.) Sockets and Networking

- The Internet knock-knock server
- Knock-knock server overview
- BLAB: how servers talk to the Internet
- Reading from the client
- The server can only talk to one person at a time
- You can fork() a process for each client
- Writing a web client
- Clients are in charge
- Create a socket for an IP address
- getaddrinfo() gets addresses for domains

13.) Threads in C

- Create threads with pthread_create
- Use a mutex as a traffic signal