Misc compiler

- object file - compiled file with placeholder addresses for externally defined functions/objects
- clang++ (g++) -c -o output.o input.cpp input2.o ...
- export PATH="///:$PATH" - make sure added directory is the parent if includes have directory in path name
- Mac - linking to a dynamic library export DYLD_LIBRARY_PATH="///:$DYLD_LIBRARY_PATH" (LD_LIBRARY_PATH)
- -c - only run preprocess, compile, and assemble steps
- -F - add directory to framework include search path
- -I - add directory to include search path
- -L - pass dirrecotry to search for libraries to the linker (g++)
- -l - specify the name of a library to include (g++)
- -Wl - clang linker arguments
- -dynamiclib - mac command to create dynamic library,
- -shared - command to create dynamic libraries for other platforms
- ar ru - create static library with archiver tool
- ranlib - combined with ar to create static libraries - updates the library object files and index
- can use libtool in place of ar ru & ranlib
- use otool to display information about object files
- use -install_name on mac when creating a dylib to avoid having to export DYLD_LIBRARY_PATH
- clang++ -S -mllvm --x86-asm-syntax=intel test.cpp
- hexdump - displays hexadecimal files
- run C source code through preprocessor

```
cpp main.c main.i
clang -E main.c > main.i
```

- preprocessed C source code -> Assembly

```
gcc -S main.i
clang -S main.i
```

- assembly -> Relocatable object file

```
gcc -c main.s
clang -c main.s
```

- link object files and produce executable object file

```
gcc -o main main.o header.o
clang main.o header.o
```

- output the symbol table of an object file

```
readelf --symbols main.o
```

- output the ELF headers

```
readelf -h main.o
```

- if entry point address is 0x0, object file has no ‘main’

```
readelf -h main.o | grep "Entry point address"
```

- basic clang ast

```
clang -Xclang -ast-dump -fsyntax-only source_file.cpp
```

- list ast symbols

```
clang-check source_file.cpp -ast-list
```

- full dump of any file

```
clang-check source_file.cpp -ast-dump
```

- full dump and filter for symbols

```
clang-check source_file.cpp -ast-dump -ast-dump-filter Symbol_name
```

- emit ast as binary file

```
clang(++) -emit-ast source_file.cpp
```

- CMake generate compile commands for project

```
set(CMAKE_EXPORT_COMPILE_COMMANDS ON)
```

- set path for compile-commands database

```
clang-check -p <build-path>
```

- pretty print clang-check ast

```
-ast-print
```

Additional Commands

- tail -f /private/var/log/system.log (macOS)
- tail -f /var/system.log (macOS & Unix)
- ar
- ranlib
- nm
- libtool
- otool
    - tV - disassemble
    - -f Display the universal headers.
    - -h Display the Mach header.
    - -l Display the load commands.

- -L Display the names and version numbers of the shared libraries that the object file uses, as well as the shared library ID if the file is a shared library.
- -D Display just the install name of a shared library. See install_name_tool(1) for more info.
- -s segname sectname Display the contents of the section (segname,sectname). If the -v flag is specified, the section is displayed as its type, unless the type is zero (the section header flags). Also the sections (__OBJC,__protocol), (__OBJC,__string_object) and (__OBJC,__runtime_setup) are displayed symbolically if the -v flag is specified.
- -t Display the contents of the (__TEXT,__text) section. With the -v flag, this disassembles the text. With the -V flag, it also symbolically disassembles the oper- ands.
- -d Display the contents of the (__DATA,__data) section.
- -o Display the contents of the __OBJC segment used by the Objective-C run-time system.
- -r Display the relocation entries.
- -c Display the argument strings (argv[] and envp[]) from a core file.
- -I Display the indirect symbol table.
- -T Display the table of contents for a dynamically linked shared library.
- -R Display the reference table of a dynamically linked shared library.
- -M Display the module table of a dynamically linked shared library.
- -H Display the two-level namespace hints table.
- -G Display the data in code table.
- -C Display the linker optimization hints (-v for verbose mode can also be added).
- -P Print the info plist section, (__TEXT,__info_plist), as strings.
- -p name Used with the -t and -v or -V options to start the disassembly from symbol name and continue to the end of the (__TEXT,__text) section.
- -v Display verbosely (symbolically) when possible.
- -V Display the disassembled operands symbolically (this implies the -v option). This is useful with the -t option.
- -X Don't print leading addresses or headers with disassembly of sections.

    - -q Use the llvm disassembler when doing disassembly; this is available for the x86 and arm architectures. This is the default.

    - -mcpu=arg When doing disassembly using the llvm disassembler use the cpu arg.

    - -function_offsets When doing disassembly print the decimal offset from the last label printed.

    - -j When doing disassembly print the opcode bytes of the instructions.

    - -Q Use otool(1)'s disassembler when doing disassembly.

    - -arch arch_type Specifies the architecture, arch_type, of the file for otool(1) to operate on when the file is a universal file (aka a file with multiple architectures). (See arch(3) for the currently known arch_types.) The arch_type can be "all" to operate on all architectures in the file. The default is to display only the host archi- tecture, if the file contains it; otherwise, all architectures in the file are shown.

    - -m The object file names are not assumed to be in the archive(member) syntax, which allows file names containing parenthesis.

    - --version Print the otool(1) version information.

- size - list size of executable segments

    - -x -l -m

- leaks

- malloc_history

- c++filt - unmangle symbols

- nm - display symbols in executable

- file - list binary formats within an executable

Disassemblers

- Hopper - https://sites.fastspring.com/hopperapp/product/hopperdisassemblerv3
- objdump
- gdb
- ndisasm
- llvm-objdump
- llvm-mc
- otool
- capstone

```
llvm-dis -c .o file
```

Networking Commands

- nmap
- fping
- dig
- netstat
- nslookup
- ifconfig
- iperf
- whowatch

Misc Networking

- sample output of ifconfig

```
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
mtu 1500
inet 172.16.128.136
netmask 255.255.255.0
broadcast 172.16.128.255
inet6 fe80::20c:29ff:fecf:6b5f
prefixlen 64
scopeid 0x20<link>
ether 00:0c:29:cf:6b:5f
txqueuelen 1000  (Ethernet)
RX packets 103282  bytes 145463398 (138.7 MiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 48322  bytes 3426769 (3.2 MiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

###

en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST>
mtu 1500
ether 14:10:9f:d3:42:f1
inet 192.168.1.2
netmask 0xffffff00
broadcast 192.168.1.255
```

```
media: autoselect
status: active
```

- gateway - address of gateway (computer) to outside of LAN
    - only one ipv4 and one ipv6 gateway should be set and this is the point for communication beyond the LAN with other computers (starts at router, but includes NAT and firewall)
- broadcast - address used to communicate with all of the computers on the LAN
- name resolution order - /etc/nsswitc.conf, /etc/hosts, /etc/resolv.conf is used for DNS
- /etc/hosts
    - main purpose of /etc/hosts configuration file is to resolve hostnames that cannot be resolved any other way
    - can also be used to resolve hostnames on small networks with no DNS server
    - regardless of the type of network the computer is on, this file should contain a line specifying the IP address of the loopback device (127.0.0.1) as localhost.localdomain.
- /etc/resolv.conf
    - /etc/resolv.conf configuration file specifies the IP addresses of DNS servers and the search domain Unless configured to do otherwise, the network initialization scripts populate this file
- /etc/sysconfig/network
    - /etc/sysconfig/network configuration file specifies routing and host information for all network interfaces.
- /etc/sysconfig/network-scripts/ifcfg- (different for different flavors of Linux)
    - for each network interface, there is a corresponding interface configuration script
    - each of these files provide information specific to a particular network interface

Essential Tests for a System

- rebooting host (bare minimum)
- max out CPU
- take up memory and force garbage collection
- consume bandwidth to disk and cause wait for I/O
- network
    - dependency slow down
    - dependency disappears

    - packet loss
    - corruption

Interesting compiler-related links

- http://duartes.org/gustavo/blog/post/anatomy-of-a-program-in-memory/
- https://linux.die.net/man/5/elf
- https://linux.die.net/man/1/nm
- https://www.codeproject.com/Articles/1388/Calling-Conventions-Demystified
- https://blogs.msdn.microsoft.com/oldnewthing/20040102-00/?p=41213/
- https://www.technovelty.org/c/position-independent-code-and-x86-64-libraries.html
- http://www.lurklurk.org/linkers/linkers.html