makefile

gcc

- what is gnu

- other c compilers

- compiling useful commands

- theory of compilers work

inbox

Makefile

Make – программа, запускается из командной строки, ищет файл с названием makefile в текущем каталоге. Чтобы использовать имя отличное от makefile

Можно ли файл makefile назвать по другому:

If you want to use a nonstandard name for your makefile, you can specify the makefile name with the `-f' or `--file' option. The arguments `-f name' or `--file=name' tell make to read the file name as the makefile. If you use more than one `-f' or `--file' option, you can specify several makefiles. All the makefiles are effectively concatenated in the order specified. The default makefile names `GNUmakefile', `makefile' and `Makefile' are not checked automatically if you specify `-f' or `--file'.

GCC

https://en.wikipedia.org/wiki/GNU\_Project

**GNU**

**Linux – OS Kernel. The Linux Kernel works with GNU(GNU is Not Unix) amd other projects to provide a fully functioning OS.**

**Calling each Linux distribution as LinuxOS is like calling all of the Windows Operating system the NTOS(windows uses NT Kernel) or Calling all of the macOS versions BSD(macOS uses BSD).**

But people just kept calling it Linux so noe Linux refers both to the Kernel and all the different operating systems that happens to use the Linux kernel. IMO Calling all of the Linux distros as Linux OS is misleading. Each distro can be quite distinct from others and often use completely different software packaging systems. It would be better if we treated each distro as it's very own OS.

GNU is the Software itself, but I think it also includes the people that makes it.

**The gnu project maintains a lot of different software projects.**

**https://www.gnu.org/software/#allgnupkgs**

The purpose of all these tools was to be part of a complete GNU OS.

They were planning an advanced (for the time) micro-kernel architecture called GNU/Hurd, but the kernel so far didn't achieve production quality and it is still being developed. Luckily, there was this Linus guy with a kernel that worked with the GNU tools.

**Most of the tools gnu provides have alternatives**...**but there isn't one project that provides all the functionality gnu has**, you usually have to take stuff from many different projects to get something close. **You can't have a working system without gnu or one of the alternatives. So lots of the tools gnu provides are the defacto standard on linux systems.**

A car doesn’t need just an engine to run. It also needs wheels, windows, exhaust, a battery, electrical wiring, lights, etc. Linux is the engine and GNU is the rest of the car. If we mention "Linux" you actually mean: GNU + Linux kernel + Mesa + Linux kernel modules. **GNU is an operating system that can be built with different kernels.** HURD is one, but there was also Debian GNU/kFreeBSD for some time. You can also run GNU on Windows with WSL (though WSL 2 runs GNU on Linux in a VM). Similarly, Linux can be used with different user-land software. Many Linux operating systems are not POSIX. For example, Android is a popular Linux operating system. There are also POSIX operating systems that use Linux and a user-lands other than GNU, such as Alpine. I’m talking about Unix-like systems, I'll usually refer to GNU/Linux to differentiate it from other Linux systems. GNU stands for "GNU is not unix" it is an operating system developed by Richard Stallman in 1984, it waited a couple od years to be published, until 1991 when Linus Torwalds made Linux kernel, and they merged it together to make a stable Os, since Linux is just a kernel, and gnu is operating system, Richard Stallman argues that is only fair to be called, gnu/Linux, look up for. The truth is that the term "operating system" is subjective. "Will Linux run POSIX applications?" The answer there is no, POSIX applications need interfaces beyond what Linux implements. GNU/Linux will run POSIX applications, though, so GNU/Linux is a (mostly) POSIX operating system. Linux is a kernel and gnu is operating system. The kernel is what interacts with/controls the system hardware and the operating system is everything that the user interacts with. together they form a full system. One part of this control and direction is interacting with the computer hardware - the operating system needs to be able to tell the physical computer what to do in order for things to work. This is where the kernel comes in - the kernel is actually just one part of a complete operating system. Linux or GNU/Linux or whatever you want to call it is a combination of the Linux kernel and various pieces of GNU software. Together, they form a complete operating system. People tend to call this complete operating system "Linux" because of the Linux kernel that it runs on. It's worth noting that kernels and "drivers", which you might be familiar with, accomplish similar tasks but are very different things. An operating system is a collection of software that is essential for the machine to work, that's why it should be called GNU+Linux and not Linux, because Linux is just a kernel (A software that speaks to the hardware), it cannot be an operating system without the necessary software (GNU), and GNU cannot be an OS without talking to the hardware, they mutually need, and can't be operating systems without the other.

**Linux is just an OS Kernel, not an OS. The Linux Kernel works with GNU(GNU is Not Unix) amd other projects to provide a fully functioning OS.**

Secondly, **when Linus created Linux, GNU was already available. So he simply used the GNU softwares to run his OS and called the whole system Linux.** Overtime that name became the de facto name while **RIchard Stallman** (creator/founder of GNU) in**sisted that every distro should use the name GNU/Linux**. In part this is proper appropriation, but more importantly, RS wanted people to recognize the free as in freedom philosophy behind this project that made the project even possible in the first place. **Linus has commented more than once that he is the engineer and RS is the philosopher. Linus tends to care less about the philosophical and social impact of the GNU/Linux project**. While, for RS, being a product of the 60s and 70s, he sees his free software movement as what really matters in the big scope of things. I think time has shown that RS' fears and concerns about how private software takes away our freedom, not just physically/economically but also social/psychological, is becoming truer everyday.

**The Free Software Foundation is the organization that builds Gnu.** (founded by Richard Stallman[6] on October 4, 1985.). The FSF was incorporated in Boston[9] where it is also based. The FSF holds the copyrights on many pieces of the GNU system, such as GNU Compiler Collection. As the holder of these copyrights, it has authority to enforce the copyleft requirements of the GNU General Public License (GPL).

What is gcc?

GNU Compiler Collection

GCC was originally written as the compiler for the GNU operating system. strive to provide regular, high quality releases. Sources are readily and freely available via Git and weekly snapshots.

**C compiler is called gcc and its C++ compiler is called g++, it does not contain C#**

Сборка простого main.c с помощью gcc

gcc –o Helloword Main.c

gcc Main.c -o Helloword

Работает и так и так

Helloword – имя application/x-executable файла на выходе

Что тут происходит?

https://www.gnu.org/software/c-intro-and-ref/manual/c-intro-and-ref.pdf

//There’s a **free GCC compiler intro booklet by Stallman**. And then Advanced C and C++ Compiling I a great in-depth book. Start with the stallmam book.

//yeah those are good books and i will add one "linkers and loader"

learn to write a compiler. you can do this as a human. t**ake 5 to 10 functions and write the assembly** to do the same thing. **become a human compiler**. then **think of how an assembler works**. then **think how an object file would be created**. then h**ow a linker will work**. take one part at a time

**how compilers work**

gcc hello.c

1. The first thing our compiler does is preprocess the code.

handles all our preprocessor directives so things like #include, #define, #pragma are all dealt with here. It also removes comments

2. compiling the code

Takes that C code after it has been preprocessed and creates assembly code. performance optimization and error checking

3. running the assembler

assembler takes this assembly file and creates an object file. That object file is the binary. the assembler as) translates hello.s into machine-language instructions, packages them in a form known as a relocatable object program , and stores the result in the object file hello.o.

4. linking

you have a big program that uses multiple libraries, compile them individually and then link them.

5. you're left with an executible file which you run and it does whatever your program does. executable object file (or simply executable ) that is ready to be loaded into memory and executed by the system.

**other info about compilers for c**

What compiler do you use for C programming?

- gcc and llvm, nothing else

**LLVM** stands for Low Level Virtual Machine. It is pretty much like GCC. It is Open Source and free software. It is mostly used and is a default compiler on most BSD OSes (incl. macOS), but not all. IIRC it also supports C#. But I'm not sure if it is mature in C# area. As it mostly runs on Unix-liked Systems. It also supports new languages like Rust, Swift, etc. **Its C compiler is called clang** and its C++ compiler is called clang++.

**other** **useful commands**

objdump

// disassembler to view an executable in assembly form

//https://metanit.com/assembler/tutorial/14.3.php **прочитать!**

//попробовать собрать hello world и изучить разбор исполняемого файла

ldd

//вывод списка разделяемых библиотек, используемых исполняемыми файлами или разделяемыми библиотеками

inbox

https://www.airs.com/blog/archives/38

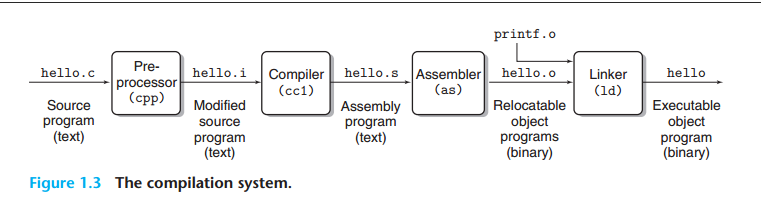
//Блог раскрывающие суть линковщиков, на реддите написали очень полезно

The C Companion by Allen I. Holub. A bit dated but excellent on general understanding of compiling, linking etc.

https://gcc.gnu.org/onlinedocs/gcc/index.html#SEC\_Contents

**Computer Systems A Programmer’s Perspective**

Programs Are Translated by Other Programs into Different Forms



1.6 Storage Devices Form a Hierarchy