goto;

GOTO
CHICAGO 2023

#GOTOchgo

goto;

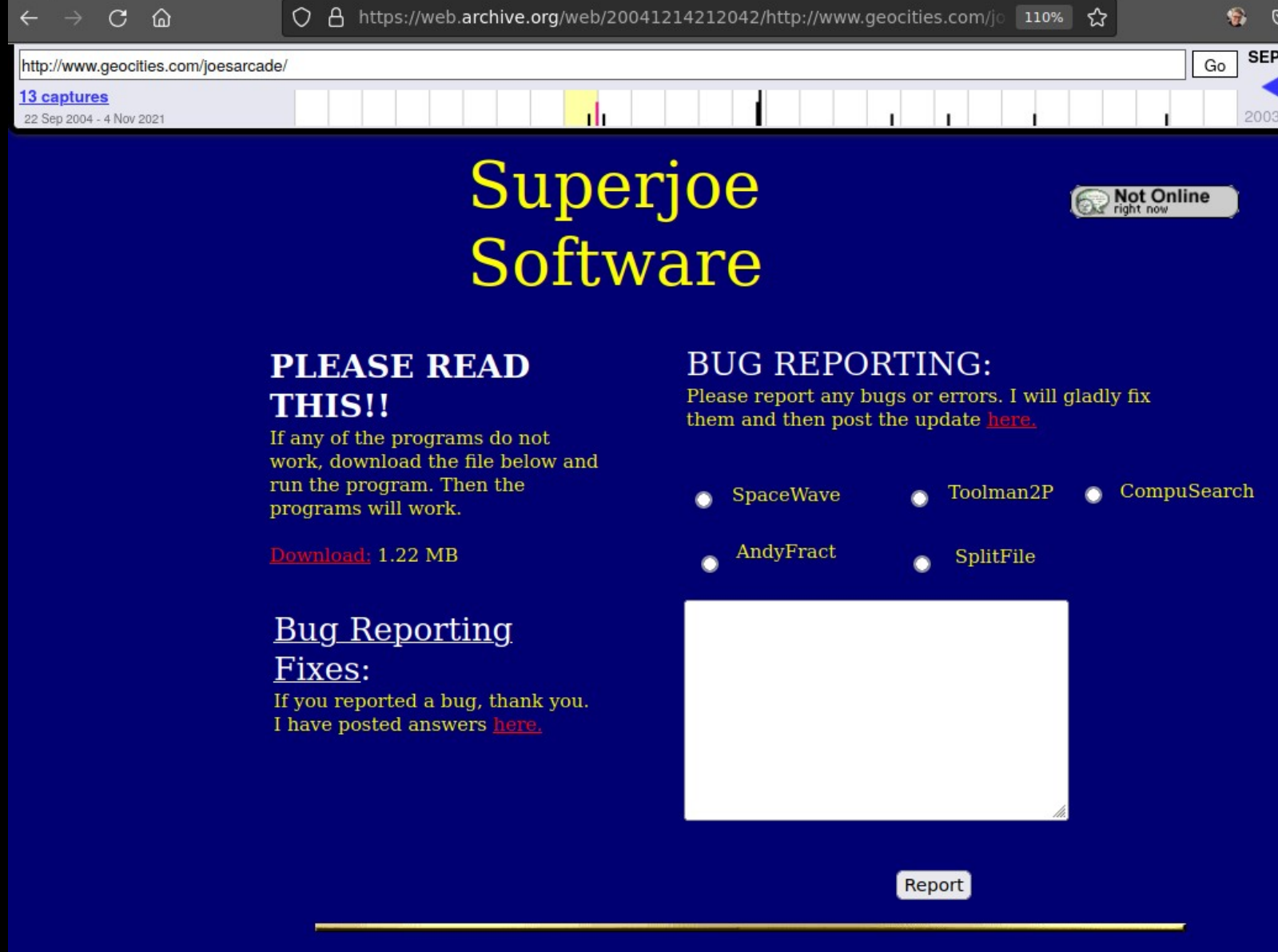
How To Build Software From Source

Andrew Kelley

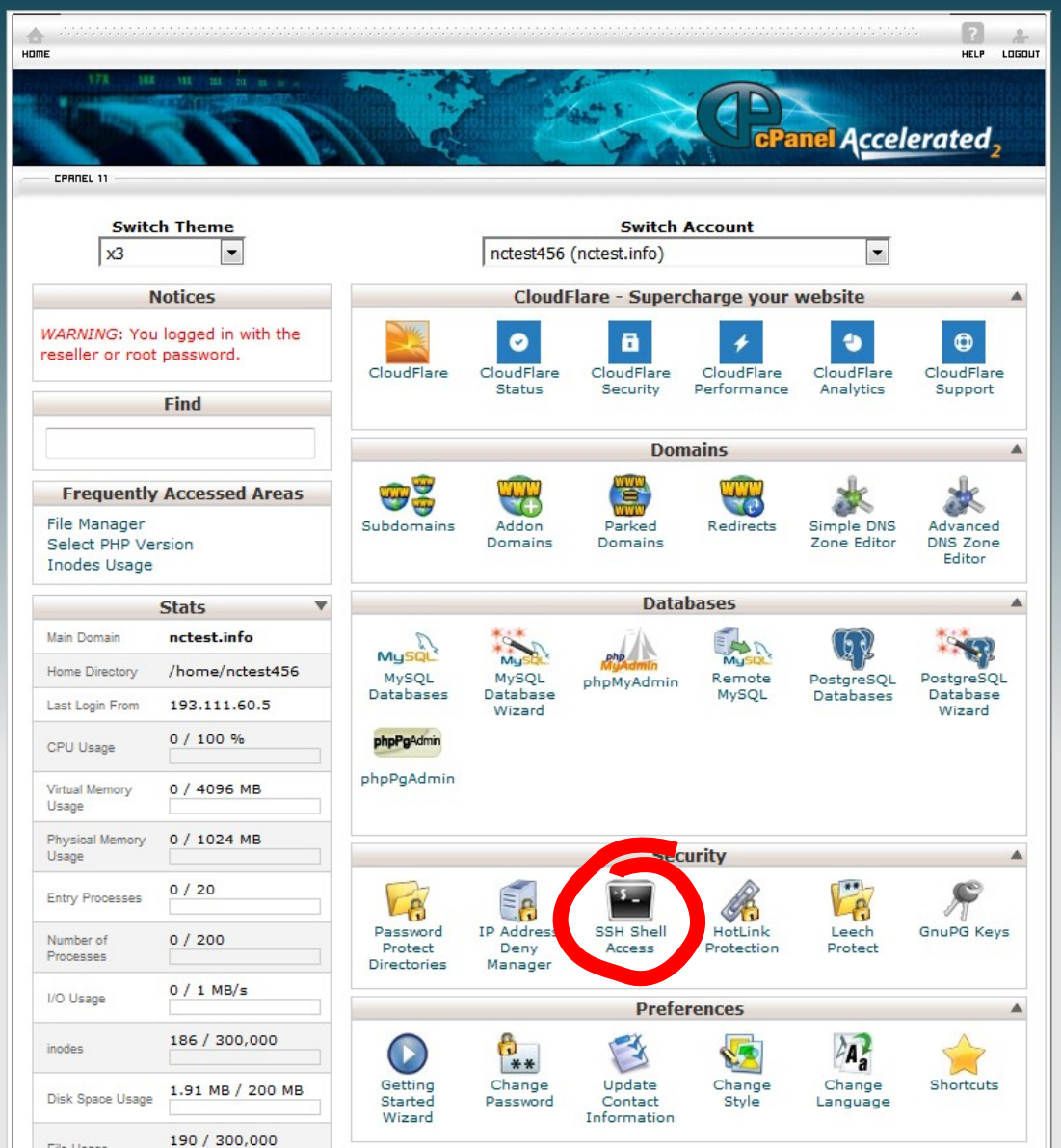
Hello!

- Creator of Zig programming language
- President of Zig Software Foundation
- 22 years of programming experience
- open source software

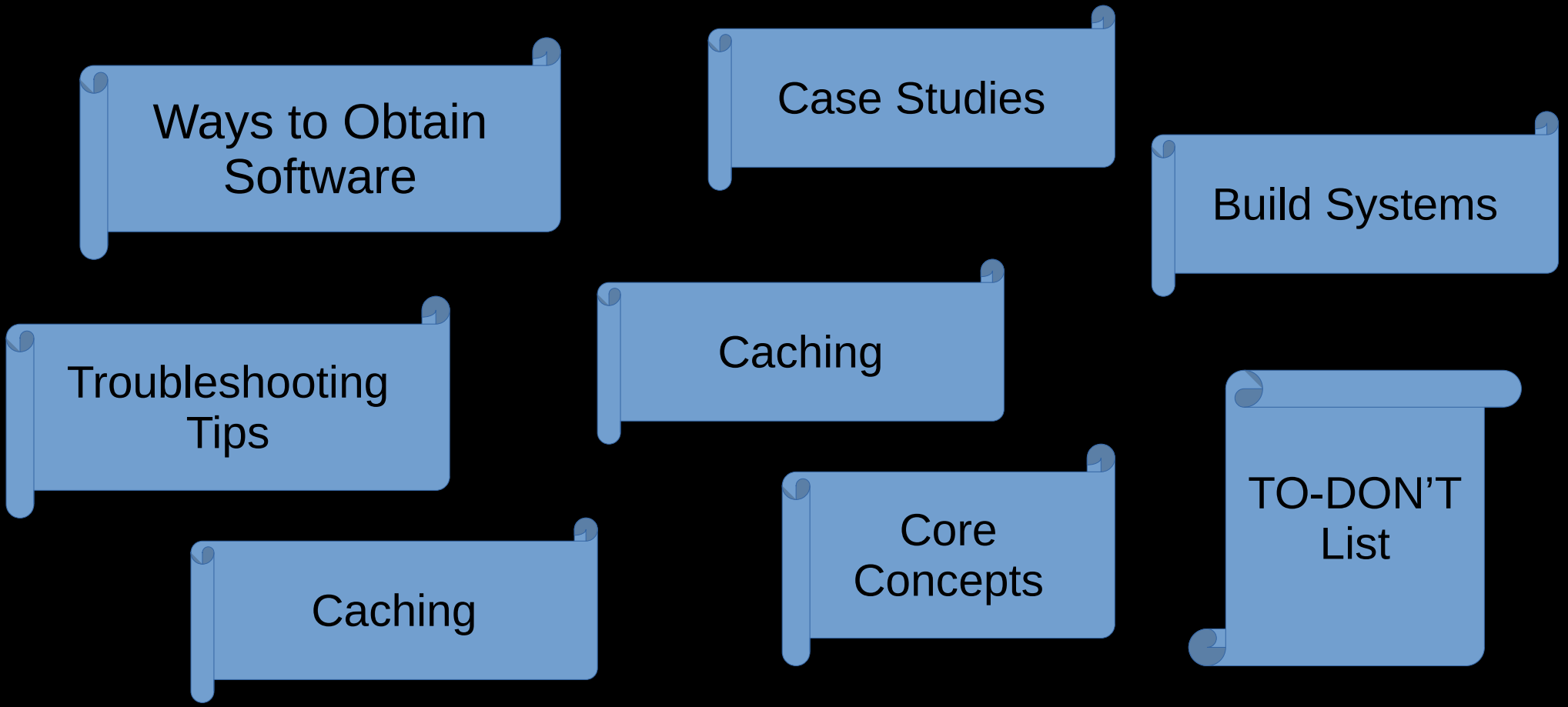
Remember GeoCities?



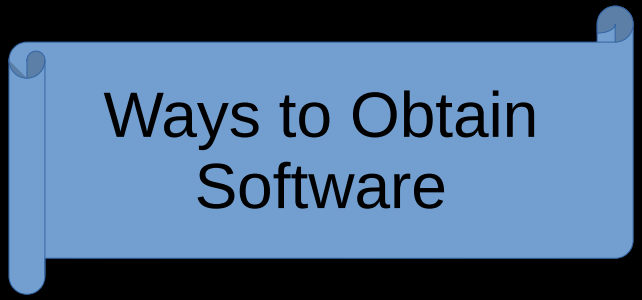
Upgrading to HostMonster for $5/month

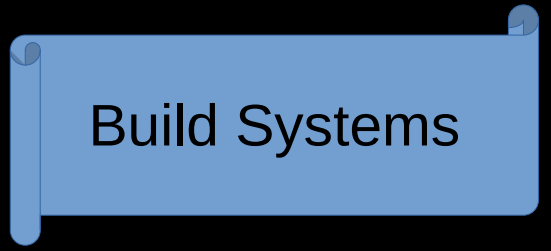


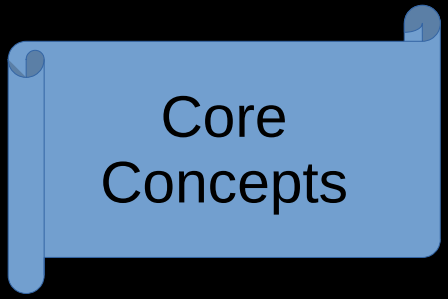
How to Build Software from Source



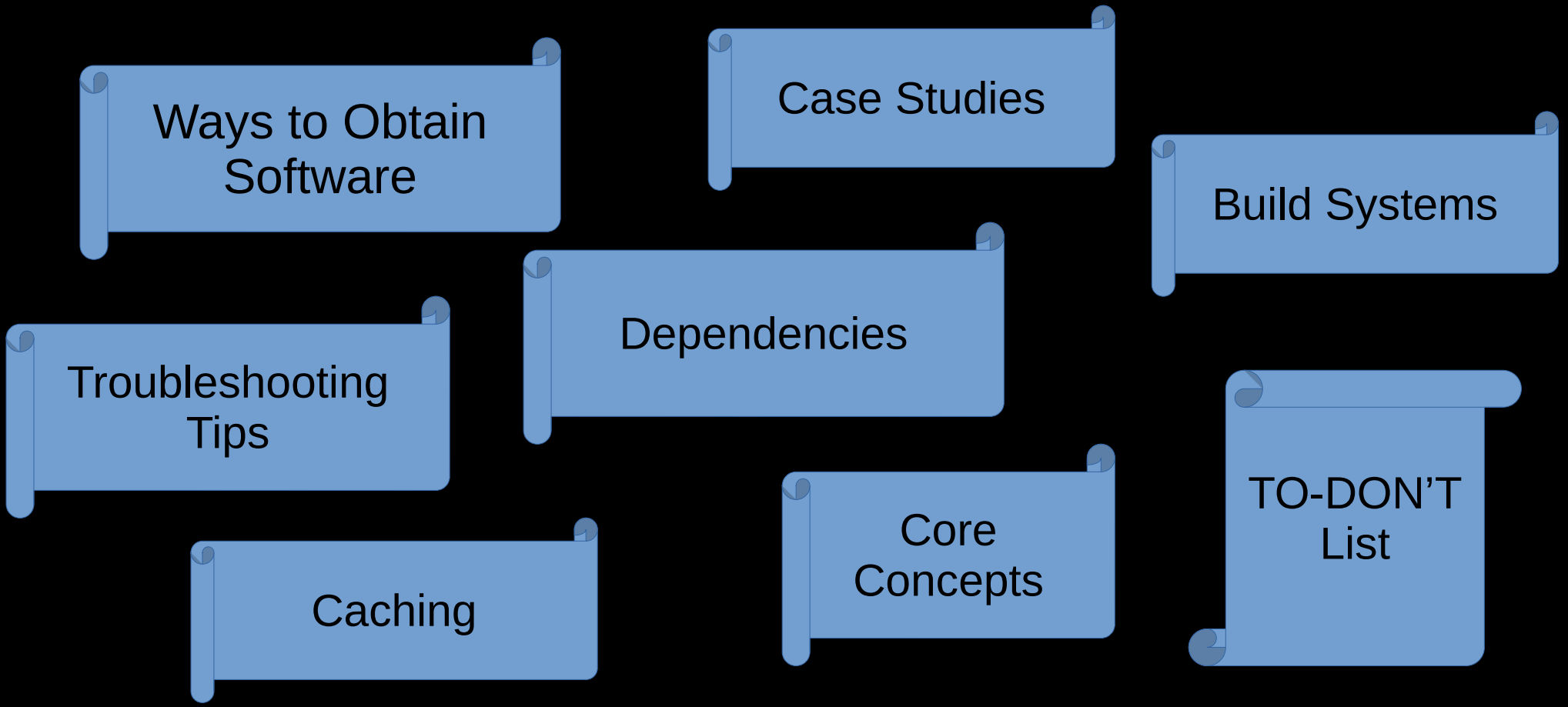
How to Build Software from Source







How to Build Software from Source



You can learn a lot about someone based on how they obtain their software

Credit: @kcgreenn

Ways to Obtain Software

Upstream Binary Distributions

Ways to Obtain Software

Upstream Binary Distributions

intended experience from the application developer's perspective

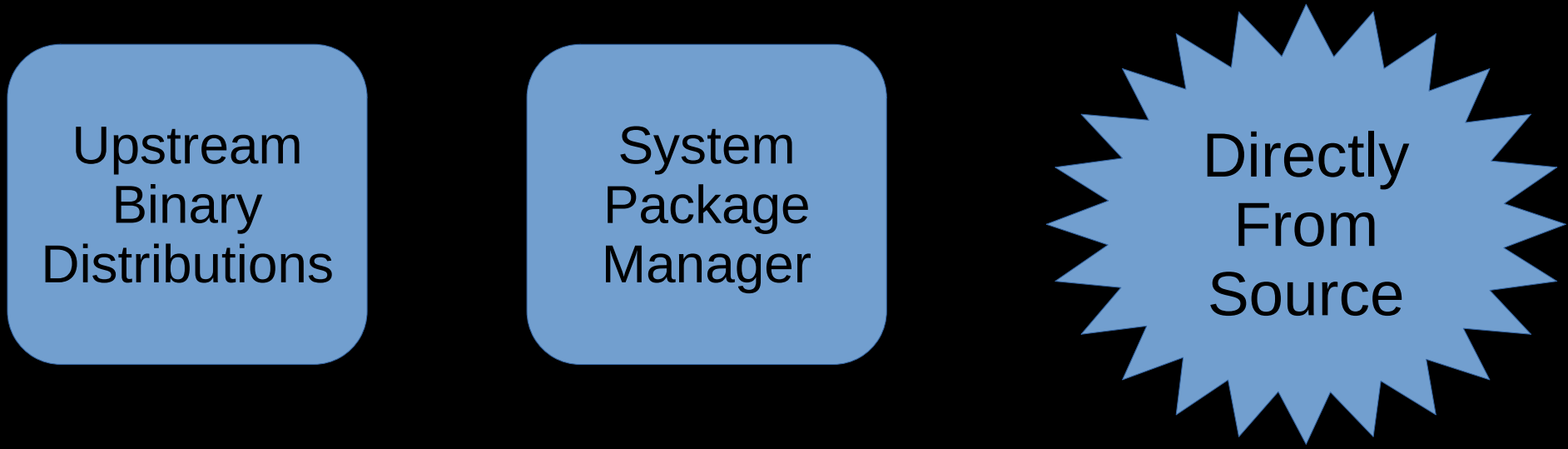
Ways to Obtain Software

Upstream Binary Distributions

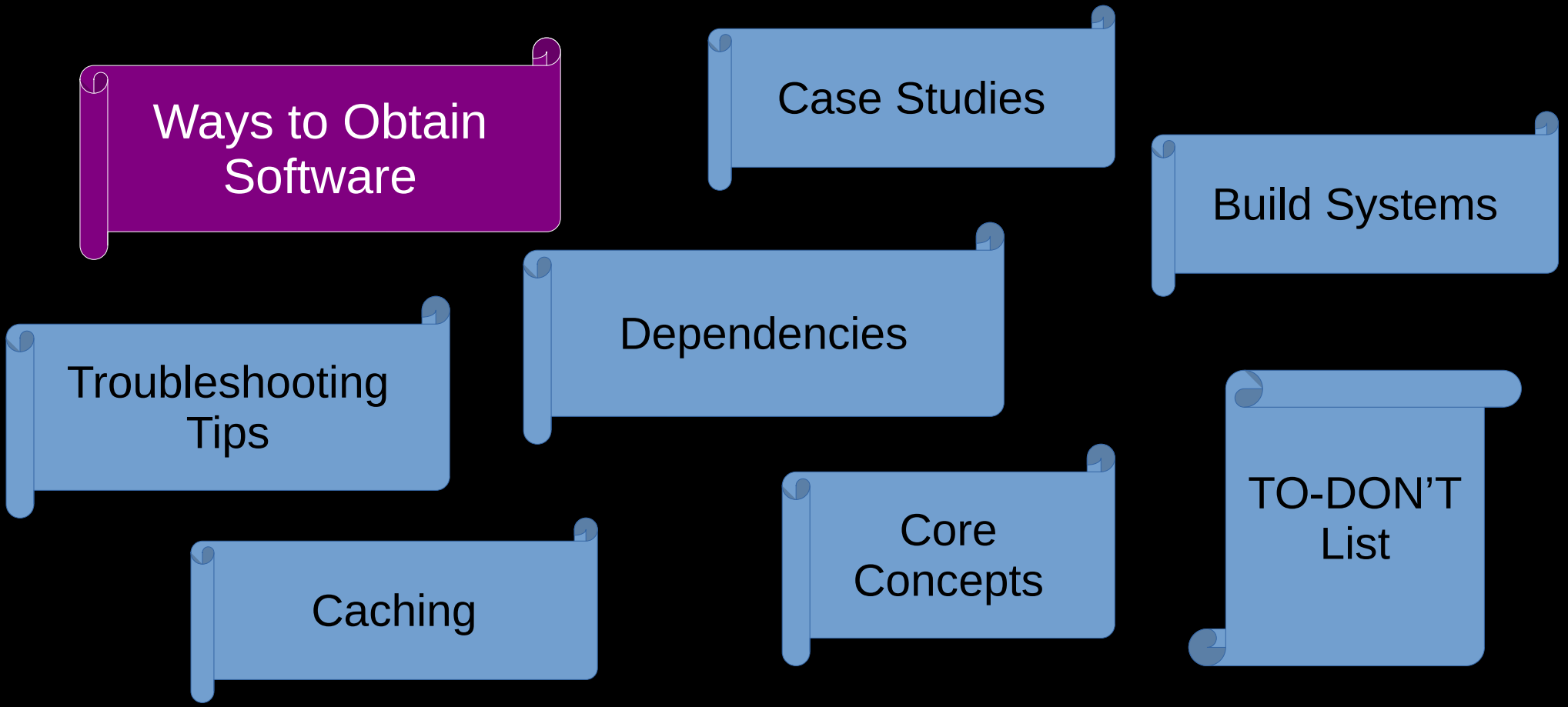
System Package Manager

intended experience from the system distribution maintainer's perspective

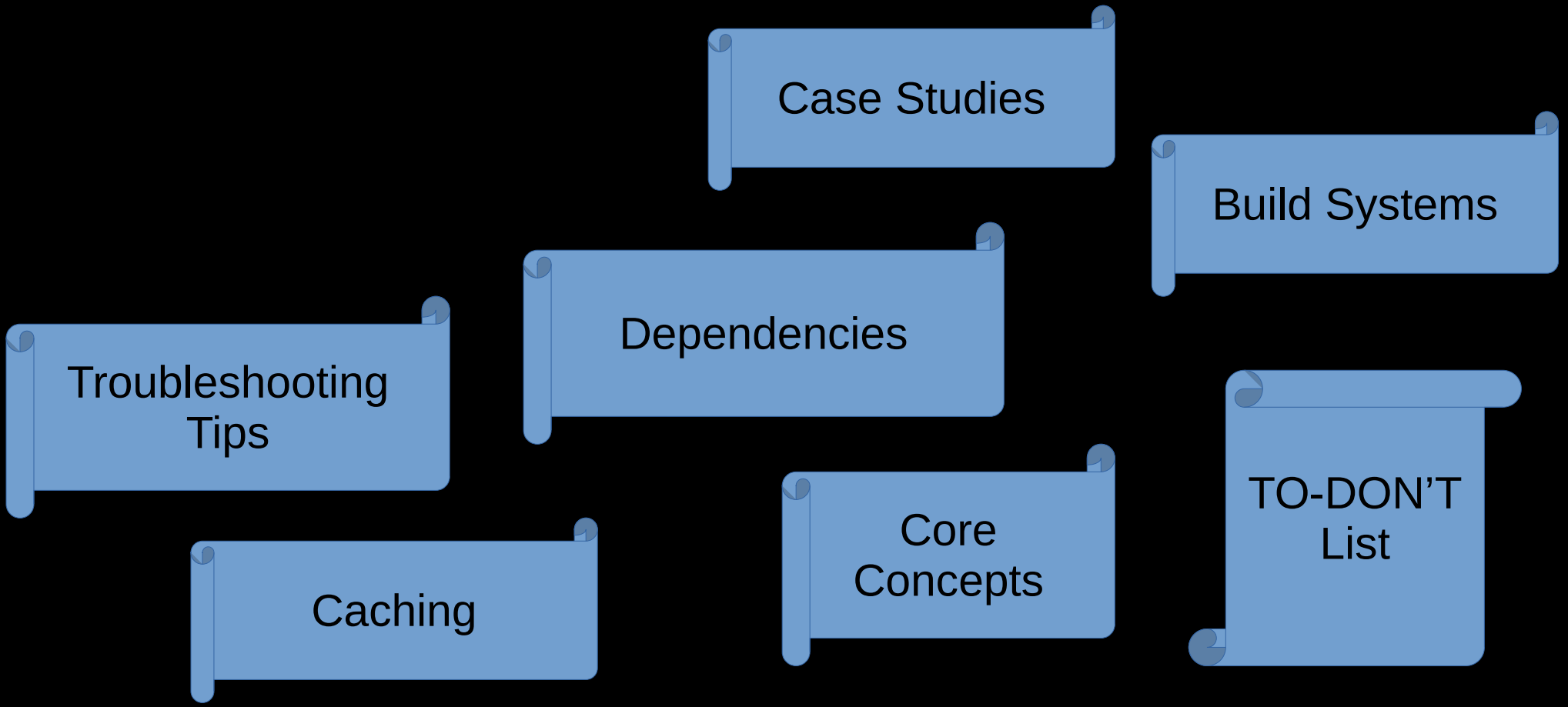
Ways to Obtain Software



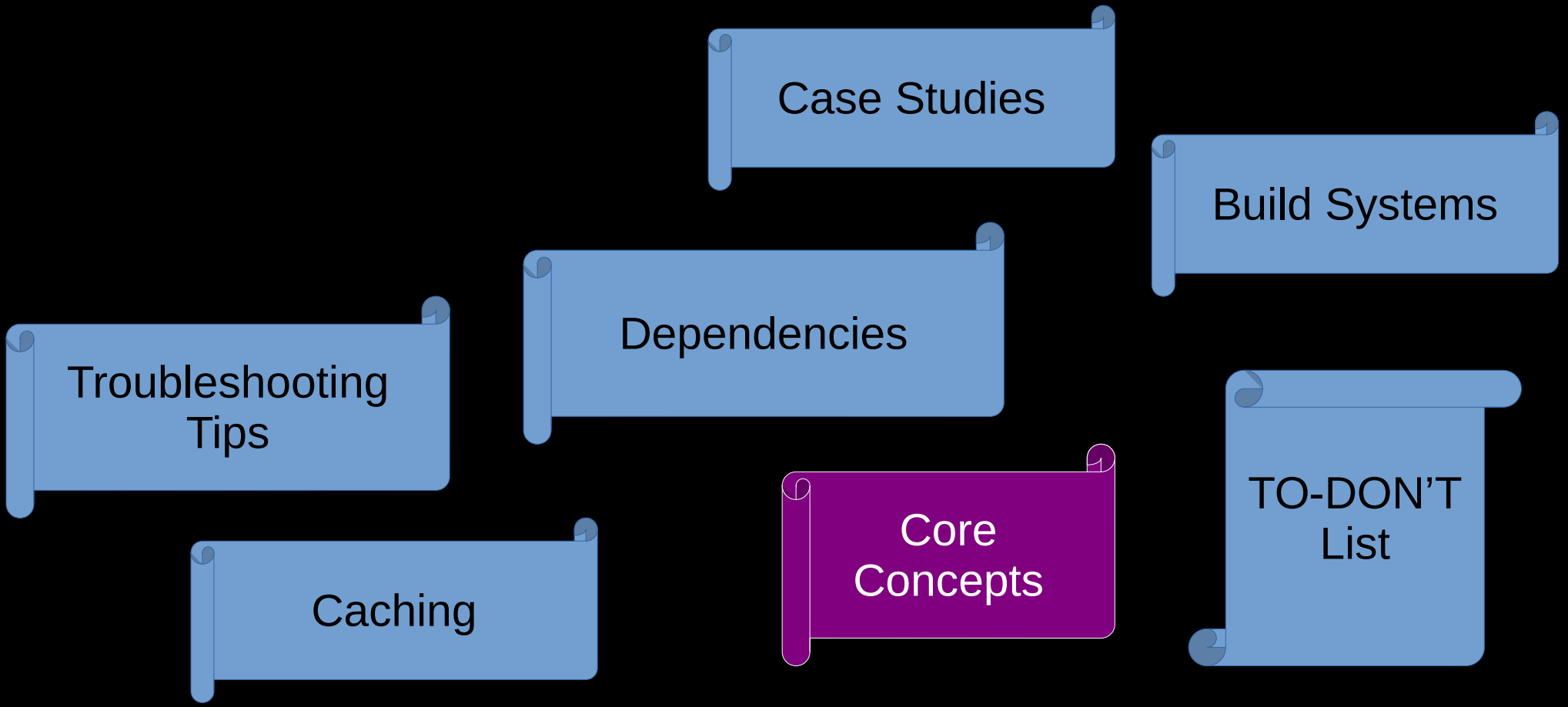
How to Build Software from Source



How to Build Software from Source



How to Build Software from Source



Core Concepts – configure phase

```
[nix-shell:~/Downloads/ffmpeg-n6.0]$ ./configure --help
Usage: configure [options]
Options: [defaults in brackets after descriptions]

Help options:
  --help                   print this message
  --quiet                  Suppress showing informative output
  --list-decoders          show all available decoders
  --list-encoders          show all available encoders
  --list-hwaccels          show all available hardware accelerators
  --list-demuxers          show all available demuxers
  --list-muxers            show all available muxers
  --list-parsers           show all available parsers
  --list-protocols         show all available protocols
  --list-bsfs              show all available bitstream filters
  --list-indevs            show all available input devices
  --list-outdevs           show all available output devices
  --list-filters           show all available filters

Standard options:
  --logfile=FILE           log tests and output to FILE [ffbuild/config.log]
  --disable-logging        do not log configure debug information
  --fatal-warnings         fail if any configure warning is generated
  --prefix=PREFIX          install in PREFIX [/usr/local]
  --bindir=DIR             install binaries in DIR [PREFIX/bin]
  --datadir=DIR            install data files in DIR [PREFIX/share/ffmpeg]
  --docdir=DIR             install documentation in DIR [PREFIX/share/doc/ffmpeg]
```

Core Concepts – configure phase

```
...
Configuration options:
  --disable-static         do not build static libraries [no]
  --enable-shared          build shared libraries [no]
  --enable-small           optimize for size instead of speed
  --disable-runtime-cpudetect disable detecting CPU capabilities at runtime (smaller
binary)
  --enable-gray            enable full grayscale support (slower color)
  --disable-swscale-alpha  disable alpha channel support in swscale
  --disable-all            disable building components, libraries and programs
  --disable-autodetect     disable automatically detected external libraries [no]

Program options:
  --disable-programs       do not build command line programs
  --disable-ffmpeg         disable ffmpeg build
  --disable-ffplay         disable ffplay build
  --disable-ffprobe        disable ffprobe build

Documentation options:
  --disable-doc            do not build documentation
  --disable-htmlpages      do not build HTML documentation pages
  --disable-manpages       do not build man documentation pages
  --disable-podpages       do not build POD documentation pages
  --disable-txtpages       do not build text documentation pages

Component options:
  --disable-avdevice       disable libavdevice build
```

Core Concepts – configure phase

```
External library support:

  Using any of the following switches will allow FFmpeg to link to the
  corresponding external library. All the components depending on that library
  will become enabled, if all their other dependencies are met and they are not
  explicitly disabled. E.g. --enable-libopus will enable linking to
  libopus and allow the libopus encoder to be built, unless it is
  specifically disabled with --disable-encoder=libopus.

  Note that only the system libraries are auto-detected. All the other external
  libraries must be explicitly enabled.

  Also note that the following help text describes the purpose of the libraries
  themselves, not all their features will necessarily be usable by FFmpeg.

  --disable-alsa           disable ALSA support [autodetect]
  --disable-appkit         disable Apple AppKit framework [autodetect]
  --disable-avfoundation   disable Apple AVFoundation framework [autodetect]
  --enable-avisynth        enable reading of AviSynth script files [no]
  --disable-bzlib          disable bzlib [autodetect]
  --disable-coreimage      disable Apple CoreImage framework [autodetect]
  --enable-chromaprint     enable audio fingerprinting with chromaprint [no]
  --enable-frei0r          enable frei0r video filtering [no]
  --enable-gcrypt          enable gcrypt, needed for rtmp(t)e support
                           if openssl, librtmp or gmp is not used [no]
  --enable-gmp             enable gmp, needed for rtmp(t)e support
                           if openssl or librtmp is not used [no]
```

Core Concepts – configure phase

```
[nix-shell:~/Downloads/ffmpeg-n6.0]$ ./configure --prefix=$HOME/local/ffmpeg --disable-x86asm
install prefix            /home/andy/local/ffmpeg
source path               .
C compiler                gcc
C library                 glibc
ARCH                      x86 (generic)
big-endian                no
runtime cpu detection     yes
standalone assembly       no
x86 assembler             nasm
MMX enabled               yes
MMXEXT enabled            yes
3DNow! enabled            yes
3DNow! extended enabled   yes
SSE enabled               yes
SSSE3 enabled             yes
AESNI enabled             yes
AVX enabled               yes
AVX2 enabled              yes
AVX-512 enabled           yes
...
debug symbols             yes
strip symbols             yes
optimize for size         no
optimizations             yes
static                    yes
shared                    no
```

Core Concepts – configure phase

```
Enabled decoders:
aac                     cri                     mp1float                sami
aac_fixed               cscd                    mp2                     sanm
aac_latm                cyuv                    mp2float                sbc
aasc                    dca                     mp3                     scpr
ac3                     dds                     mp3adu                  sdx2_dpcm
ac3_fixed               derf_dpcm               mp3adufloat             sga
acelp_kelvin            dfa                     mp3float                sgi
adpcm_4xm               dfpwm                   mp3on4                  sgirle
adpcm_adx               dirac                   mp3on4float             sheervideo
adpcm_afc               dnxhd                   mpc7                    shorten
adpcm_agm               dolby_e                 mpc8                    simbiosis_imx
adpcm_aica              dpx                     mpeg1_v4l2m2m           sipr
adpcm_argo              dsd_lsbf                mpeg1video              siren
adpcm_ct                dsd_lsbf_planar         mpeg2_v4l2m2m           smackaud
adpcm_dtk               dsd_msbf                mpeg2video              smacker
adpcm_ea                dsd_msbf_planar         mpeg4                   smc
adpcm_ea_maxis_xa       dsicinaudio             mpeg4_v4l2m2m           smvjpeg
adpcm_ea_r1             dsicinvideo             mpegvideo               snow
adpcm_ea_r2             dss_sp                  mpl2                    sol_dpcm
adpcm_ea_r3             dst                     msa1                    sonic
adpcm_ea_xas            dvaudio                 msmpeg4v1               sp5x
adpcm_g722              dvbsub                  msmpeg4v2               speedhq
adpcm_g726              dvdsub                  msmpeg4v3               speex
adpcm_g726le            dvvideo                 msnsiren                srt
adpcm_ima_acorn         dxtory                  msp2                    ssa
adpcm_ima_alp           dxv                     msrle                   stl
```

Core Concepts – configure phase

```
...
Enabled bsfs:
aac_adtstoasc           filter_units            mov2textsub             setts
av1_frame_merge         h264_metadata           mp3_header_decompress   text2movsub
av1_frame_split         h264_mp4toannexb        mpeg2_metadata          trace_headers
av1_metadata            h264_redundant_pps      mpeg4_unpack_bframes    truehd_core
chomp                   hapqa_extract           noise                   vp9_metadata
dca_core                hevc_metadata           null                    vp9_raw_reorder
dts2pts                 hevc_mp4toannexb        opus_metadata           vp9_superframe
dump_extradata          imx_dump_header         pcm_rechunk             vp9_superframe_split
dv_error_marker         media100_to_mjpegb      pgs_frame_merge
eac3_core               mjpeg2jpeg              prores_metadata
extract_extradata       mjpega_dump_header      remove_extradata

Enabled indevs:
fbdev                   lavfi                   oss                     v4l2

Enabled outdevs:
fbdev                   oss                     v4l2

License: LGPL version 2.1 or later

WARNING: pkg-config not found, library detection may fail.

[nix-shell:~/Downloads/ffmpeg-n6.0]$
```

Core Concepts – install prefix

```
[nix-shell:~/Downloads/ffmpeg-n6.0]$ ./configure --help
...
Standard options:
  --logfile=FILE           log tests and output to FILE [ffbuild/config.log]
  --disable-logging        do not log configure debug information
  --fatal-warnings         fail if any configure warning is generated
  --prefix=PREFIX          install in PREFIX [/usr/local]
  --bindir=DIR             install binaries in DIR [PREFIX/bin]
  --datadir=DIR            install data files in DIR [PREFIX/share/ffmpeg]
  --docdir=DIR             install documentation in DIR [PREFIX/share/doc/ffmpeg]
```

Core Concepts – install prefix

- This is both where *dependencies* are found, as well as where the output files will be installed to.
- `$prefix/bin/*` - executables
- `$prefix/lib/*` - libraries (machine code)
- `$prefix/include/*` - C/C++ header files
- ***Never, ever use the default prefix***

Core Concepts – install prefix

Core Concepts – install prefix

Your Application is Consistent Across Operating Systems

Windows Users' Experience is Consistent Across their Apps

Core Concepts – build phase

`make`

Core Concepts – build phase

~~`make`~~

`make -j`

Core Concepts – build phase

~~`make`~~

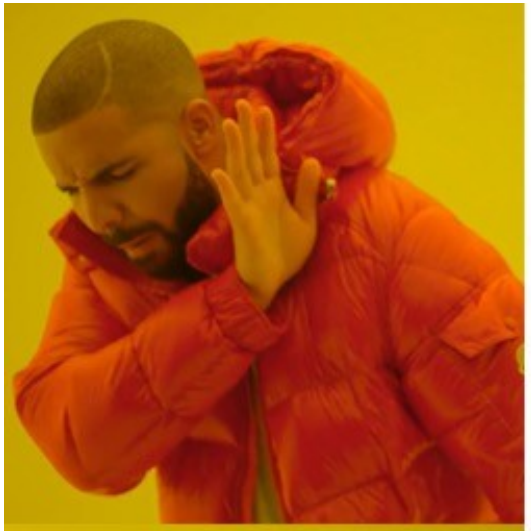
~~`make -j`~~

`make -j<N>`

Core Concepts – build phase

```
sudo make install
```

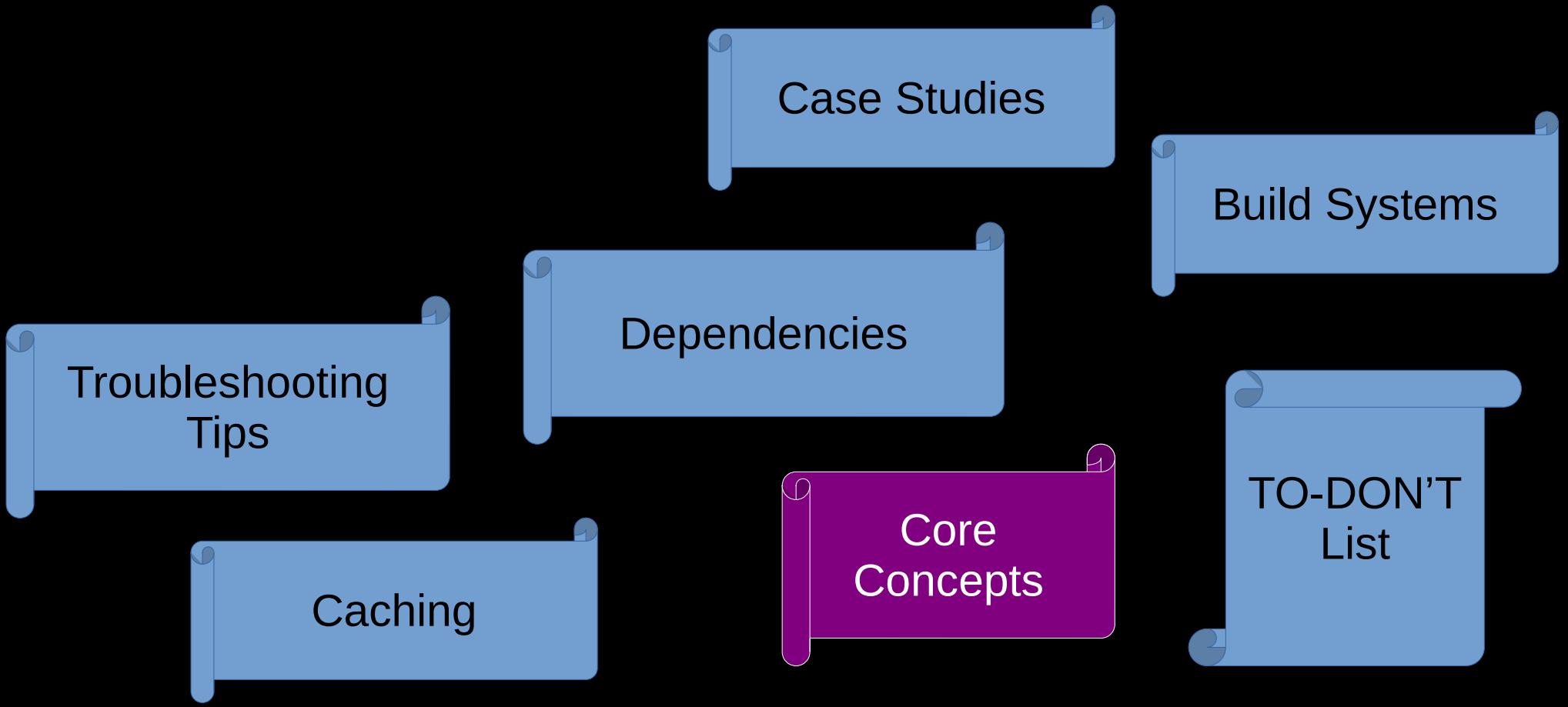
Core Concepts – build phase



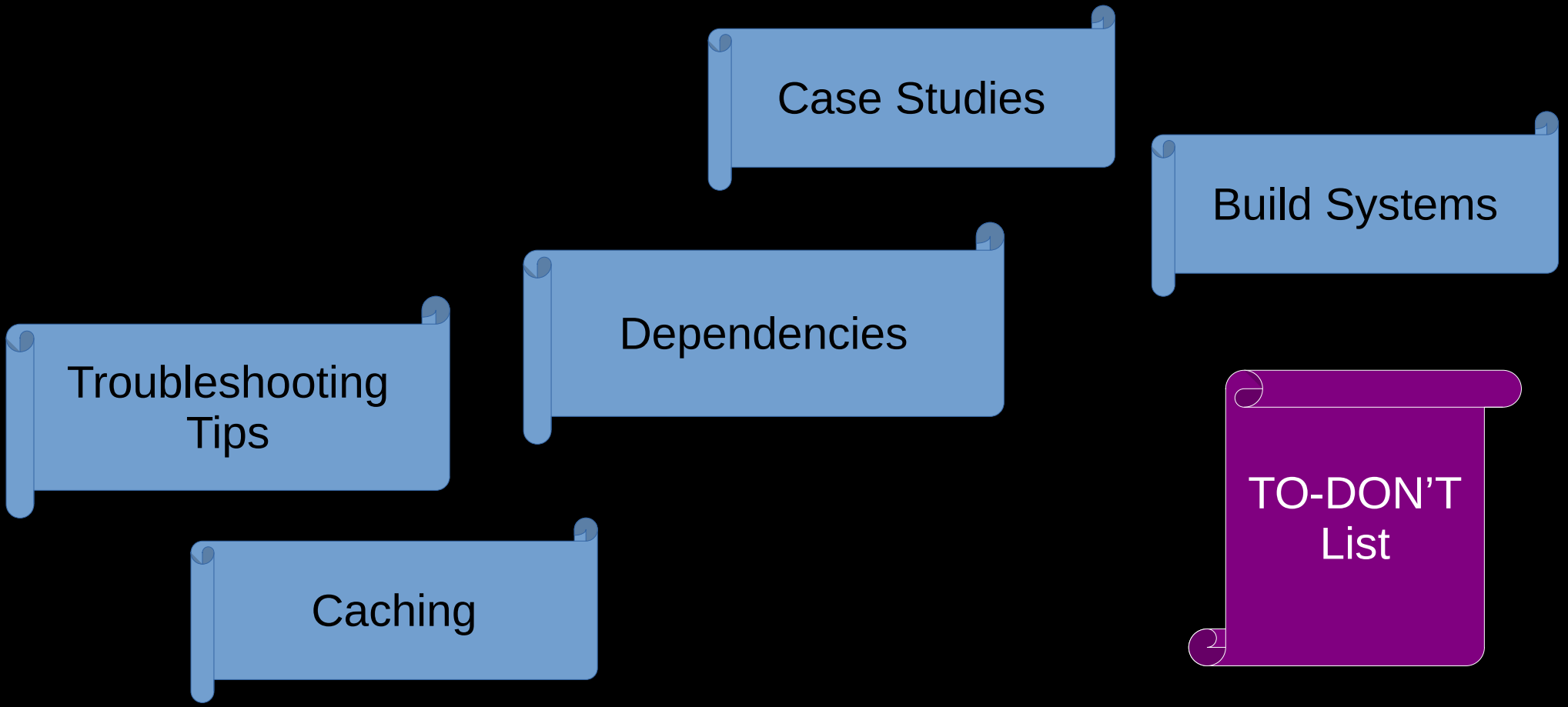
```
make && sudo make install
```


```
make install
```

How to Build Software from Source



How to Build Software from Source



Insane Things I've Seen People Do While Failing to Build from Source

6. Use the dev branch instead of a tagged release

Insane Things I've Seen People Do While Failing to Build from Source

5. Use a different compiler than their system toolchain

Insane Things I’ve Seen People Do While Failing to Build from Source

4. Use Docker or a virtual machine

Insane Things I've Seen People Do While Failing to Build from Source

3. ssh into a different machine and start copying files around haphazardly

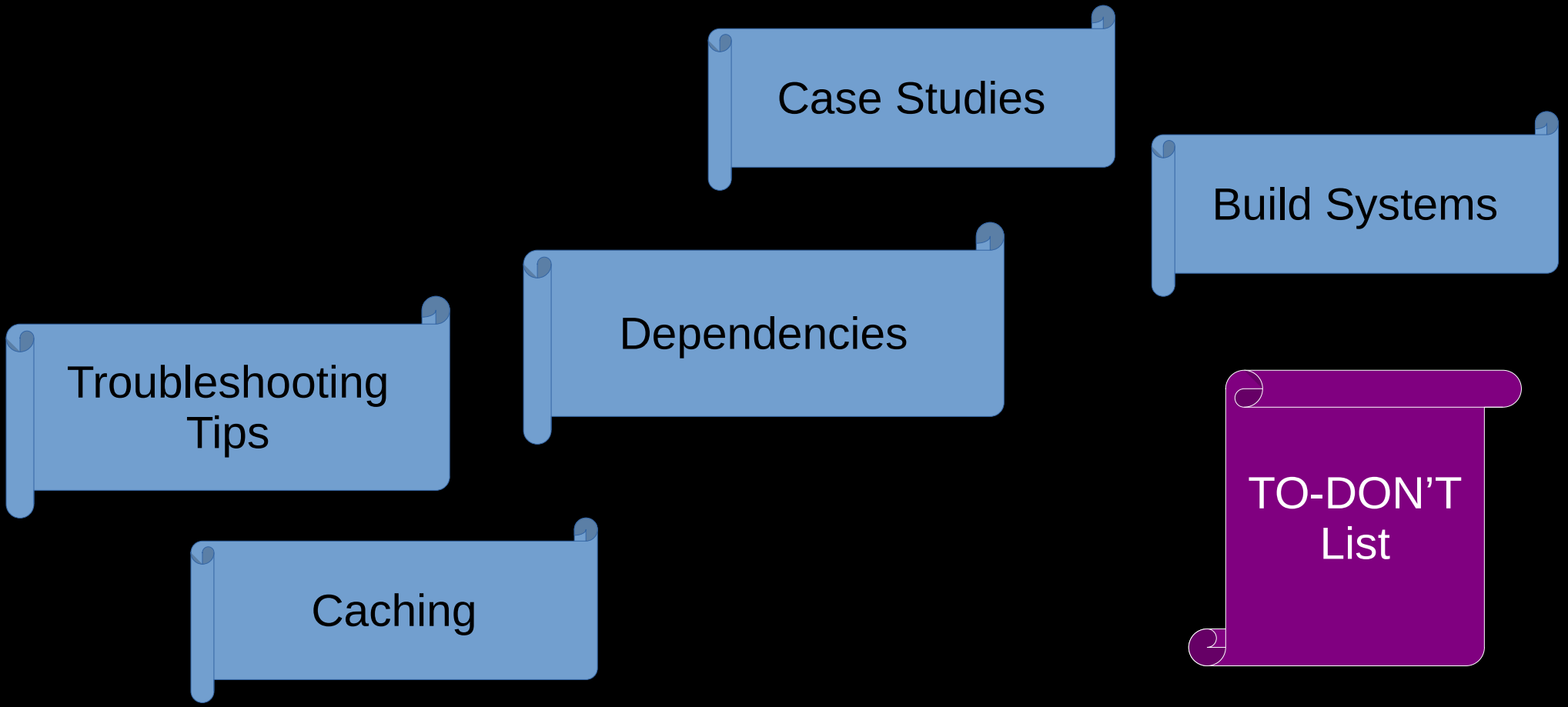
Insane Things I've Seen People Do While Failing to Build from Source

2. try a bunch of weird configure options that also don't work, then only report a bug from this second attempt

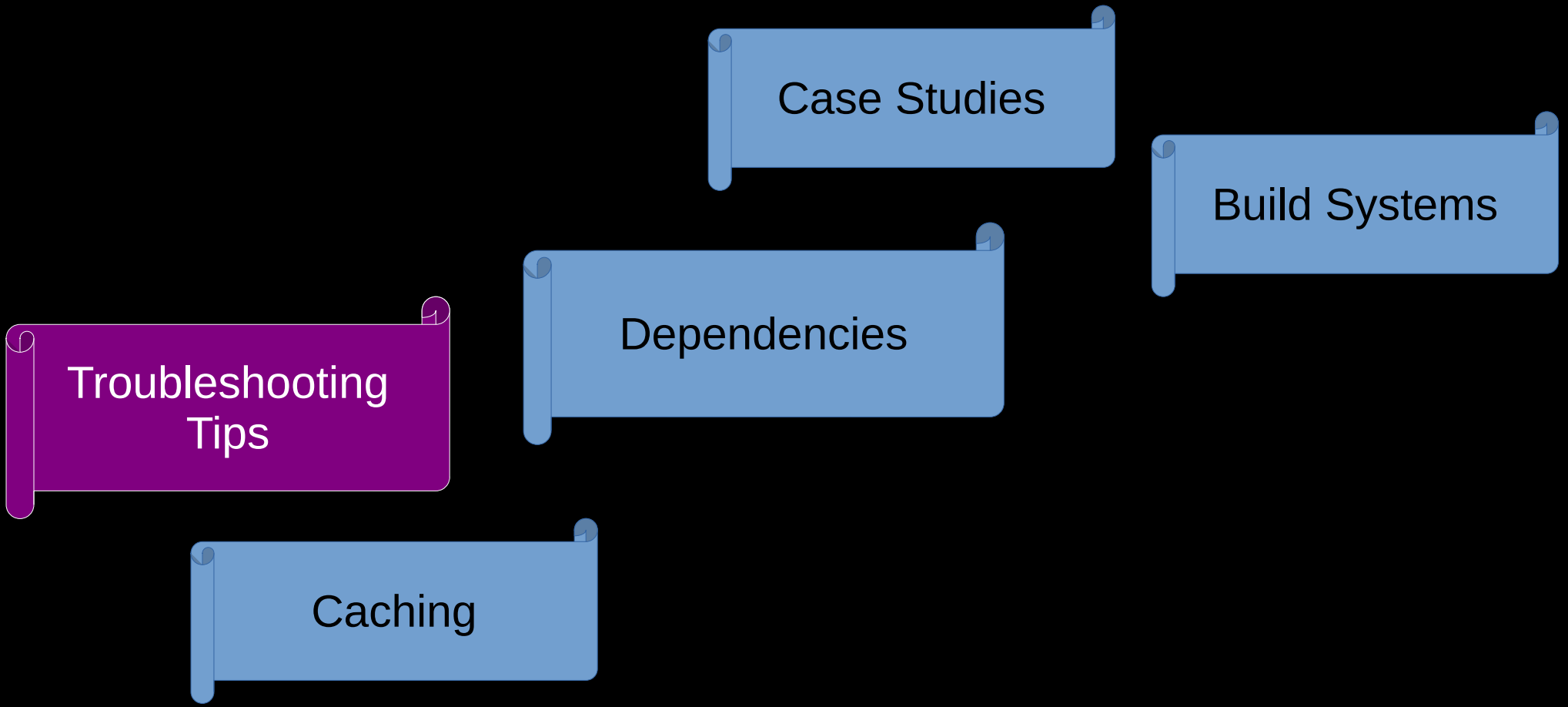
Insane Things I’ve Seen People Do While Failing to Build from Source

1. start patching the code haphazardly

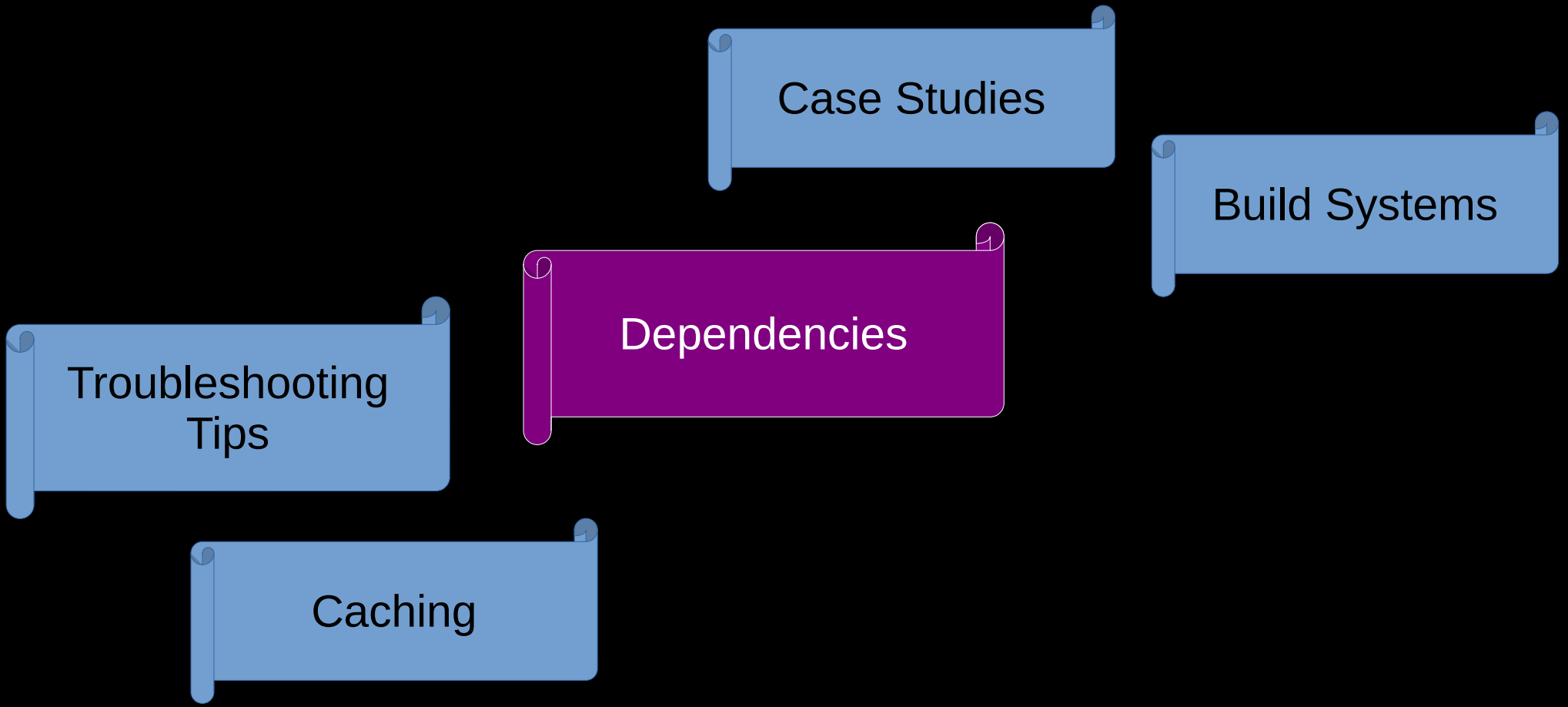
How to Build Software from Source



How to Build Software from Source



How to Build Software from Source



Dependencies

- Note the versions
- System-provided or also build the dependencies from source?
- Optional dependencies
- Organizing into prefixes

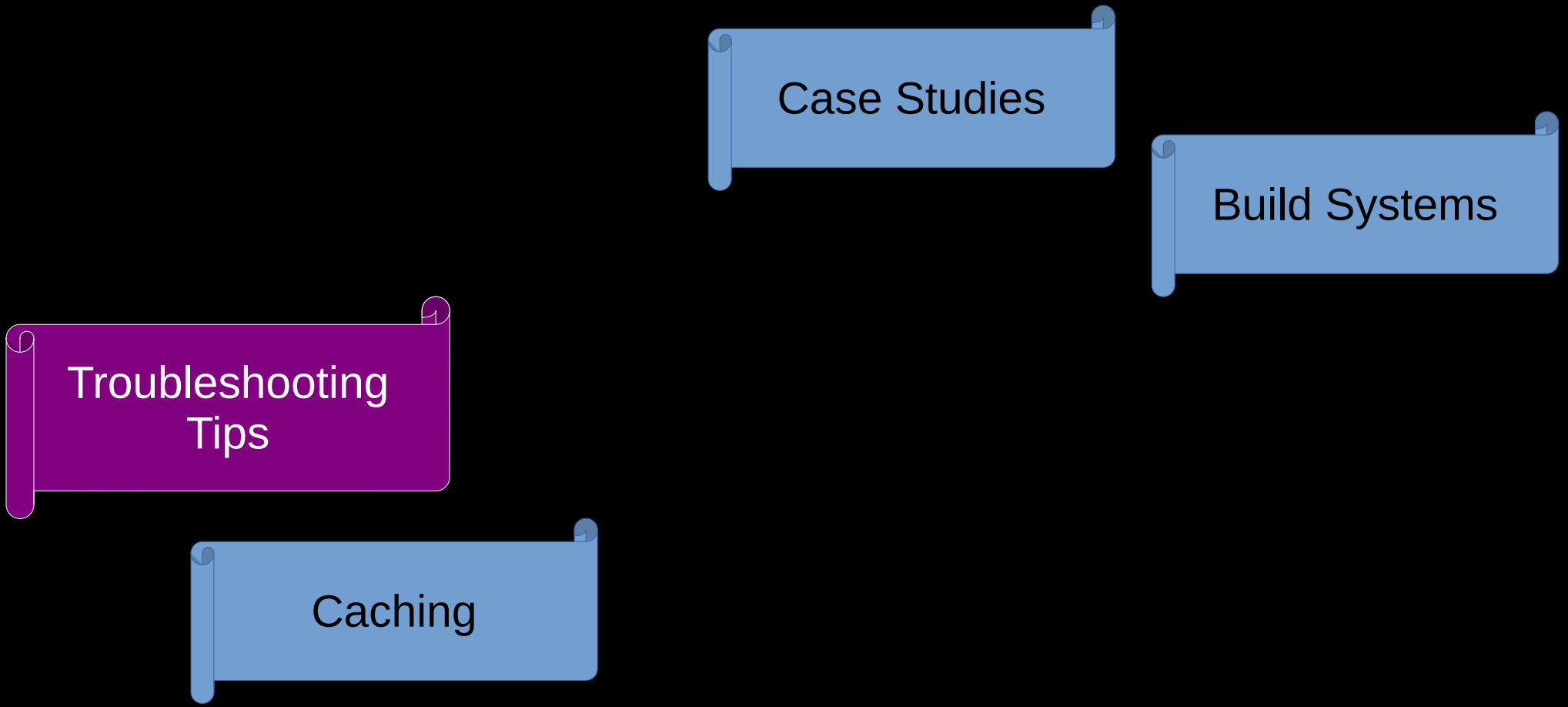
Troubleshooting Tips

- Use compatible versions of software
- Use tagged releases
- Use the latest bug fix release
- Make helpful bug reports!
  - Include all the error messages
  - Mention the versions of everything (OS, compiler, dependencies, the source code)

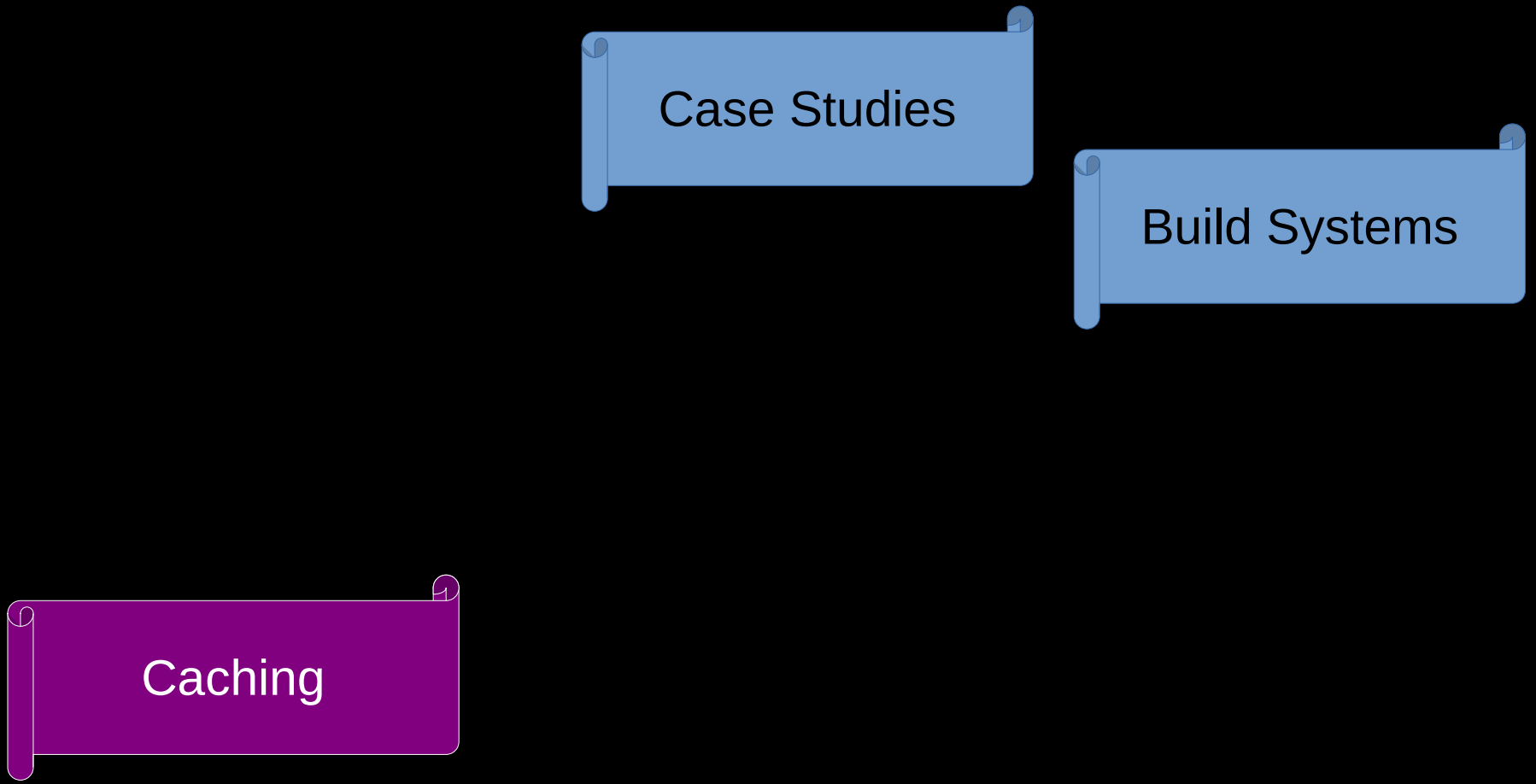
Common Issues

- Wrong version of dependency
- System dependencies are configured incorrectly
- Missing dependencies
  - Be sure to scrutinize configure phase output!
- Combination of OS and compiler version not tested together yet by the project maintainers
  - Usually fixed with small build system patch to modify a flag

How to Build Software from Source



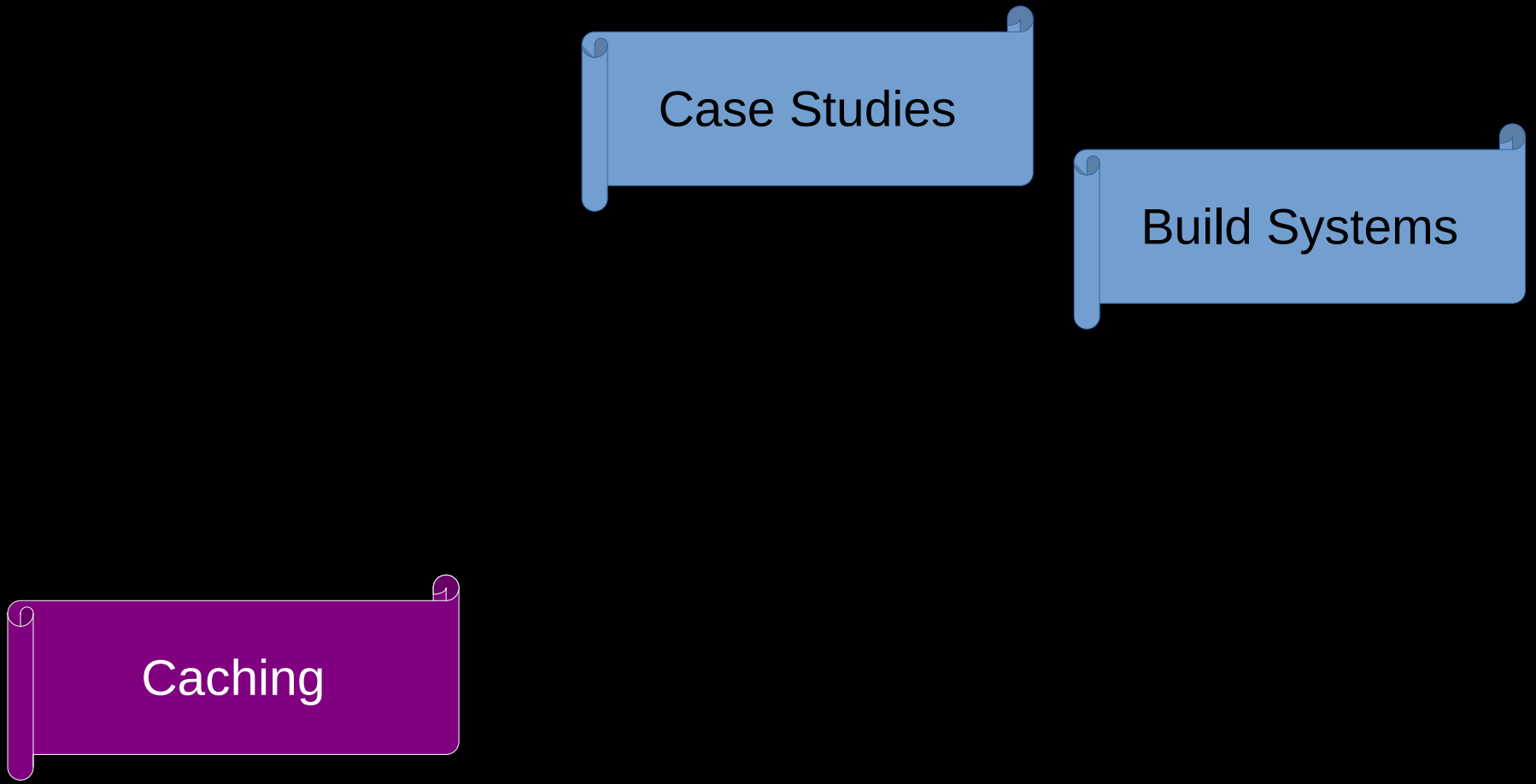
How to Build Software from Source



Caching

- mtime-based caching is totally busted
  - https://apenwarr.ca/log/20181113
- Depfiles
- How can we fix the mtime-based caching?
  - inode, size, hash
  - mtime granularity still an issue…

How to Build Software from Source



How to Build Software from Source

Case Studies

Build Systems

Build Systems

- make vs ninja
- cmake vs autotools
- Key feature: being easily (or already!) installed everywhere
- Key feature: portability

Build System

Zig Build System

- One dependency to rule them all

Filename	Signature	Kind	Size
zig-0.10.1.tar.xz	minisig	Source	14.4MiB
zig-bootstrap-0.10.1.tar.xz	minisig	Source	41.9MiB
zig-windows-x86_64-0.10.1.zip	minisig	Binary	69.9MiB
zig-windows-aarch64-0.10.1.zip	minisig	Binary	66.2MiB
zig-macos-aarch64-0.10.1.tar.xz	minisig	Binary	38.6MiB
zig-macos-x86_64-0.10.1.tar.xz	minisig	Binary	43.0MiB
zig-linux-x86_64-0.10.1.tar.xz	minisig	Binary	42.0MiB

Zig Build System

- Compiles C/C++/Zig code without any dependency on system toolchain

https://andrewkelley.me/post/zig-cc-powerful-drop-in-replacement-gcc-clang.html

Andrew Kelley

RSS Feed
GitHub

`zig cc`: a Powerful Drop-In Replacement for GCC/Clang

If you have heard of Zig before, you may know it as a promising new programming language which is ambitiously trying to overthrow C as the de-facto systems language. But did you know that it also can straight up compile C code?

This has been possible for a while, and you can see some examples of this on the home page. What's new is that the `zig cc` sub-command is available, and it supports

Zig Build System

- Write your build script in a Real Programming Language

```zig
const std = @import("std");

pub fn build(b: *std.Build) void {
    const target = b.standardTargetOptions(.{});
    const optimize = b.standardOptimizeOption(.{});

    const exe = b.addExecutable(.{
        .name = "abc",
        .root_source_file = .{ .path = "src/main.zig" },
        .target = target,
        .optimize = optimize,
    });

    b.installArtifact(exe);

    const unit_tests = b.addTest(.{
        .root_source_file = .{ .path = "src/main.zig" },
        .target = target,
        .optimize = optimize,
    });

    const run_unit_tests = b.addRunArtifact(unit_tests);

    const test_step = b.step("test", "Run unit tests");
    test_step.dependOn(&run_unit_tests.step);
}
```

Zig Build System

- Cross-compiles to Windows, macOS, and Linux with the flick of a switch

```
andy@ark ~/t/abc> zig build test -Dtarget=aarch64-linux -fqemu -fsummary
Build Summary: 3/3 steps succeeded; 1/1 tests passed
test success
└─ run test 1 passed 67ms MaxRSS:13M
   └─ zig test Debug aarch64-linux success 2s MaxRSS:229M
andy@ark ~/t/abc>
```

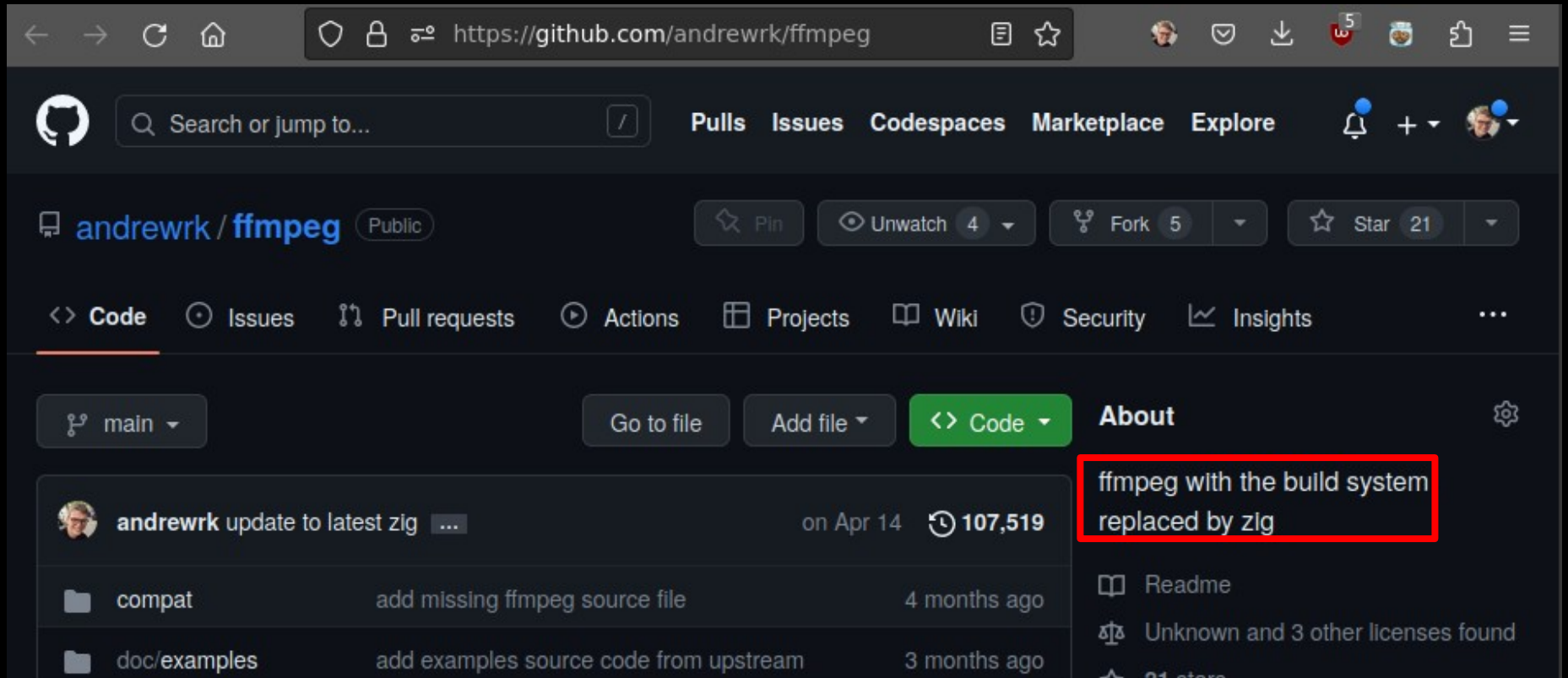
```
andy@ark ~/d/2Pew (main)> uname
Linux
andy@ark ~/d/2Pew (main)> zig build -Dtarget=x86_64-windows
andy@ark ~/d/2Pew (main)> ls zig-out
data  pew.exe
andy@ark ~/d/2Pew (main)>
```

Zig Build System

- Parallelism & Caching

Zig Build System

- Parallelism & Caching



Zig Build System

- Parallelism & Caching

```
$ time ./configure
14s
$ time make -j16 install
1m31s
```

upstream build system

```
$ time zig build
1m21s
```

zig build system

Zig Build System - Cache System

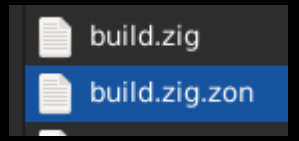
`zig-cache/h/0a9f142b72cfae28525913fa81754869.txt`

`<inode> <mtime.tv_sec> <mtime.tv_nsec> <hash> <path>`

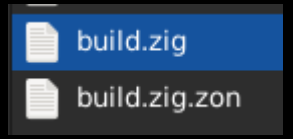
```
10422 8809030 1673140923570305355 70edd0d8832f57cdda9bbf1a6c9f114a 0 /home/andy/dev/ffmpeg/libavformat/rtpdec_qdm2.c
19459 48245006 1000000000 ddad51c30bd8562ab7e02c4cd04448bd 0 /.../glibc-2.35-224-dev/include/string.h
4286 48244619 1000000000 b314a1ad1f8c2a04fb9002f305457fe5 0 /.../glibc-2.35-224-dev/include/bits/libc-header-start.h
18116 48244864 1000000000 849f6bb6037446b8f5491b3eebcfa164 0 /.../glibc-2.35-224-dev/include/features.h
1409 48244863 1000000000 f70fbb235fabf08537e401ba250e5544 0 /nix/store/.../include/features-time64.h
...
```

- Build Summary

```
andy@ark ~/d/ffmpeg (main)> time zig build -fsummary
Build Summary: 311/311 steps succeeded
install success
├─ install ffmpeg success
│  └─ zig build-lib ffmpeg Debug native success 51s MaxRSS:508M
│     ├─ zig build-lib z Debug native-native success 861ms MaxRSS:102M
│     ├─ install zconf.h to zconf.h success
│     ├─ install zlib.h to zlib.h success
│     ├─ zig build-lib mp3lame Debug native-native success 2s MaxRSS:107M
│     │  └─ configure autoconf header config.h.in to config.h success
│     ├─ install include/ success
│     ├─ zig build-lib vorbis Debug native-native success 2s MaxRSS:128M
│     │  ├─ zig build-lib ogg Debug native-native success 530ms MaxRSS:88M
│     │  └─ install include/ogg/ success
│     ├─ install include/vorbis/ success
│     ├─ zig build-lib ogg Debug native-native (reused)
│     ├─ install include/ogg/ (reused)
│     ├─ configure blank header to libavutil/avconfig.h success
│     ├─ configure blank header to config.h success
│     ├─ run nasm (resample.o) success 422ms MaxRSS:18M
│     │  ├─ zig build-exe nasm ReleaseFast native success 9s MaxRSS:261M
│     │  │  ├─ configure blank header to version.h success
│     │  │  └─ configure autoconf header config/config.h.in to config/config.h success
│     │  └─ configure nasm header to config.asm success
│     ├─ run nasm (audio_convert.o) success 1s MaxRSS:31M
│     │  ├─ zig build-exe nasm ReleaseFast native (+2 more reused dependencies)
│     │  └─ configure nasm header to config.asm (reused)
│     ├─ run nasm (rematrix.o) success 202ms MaxRSS:11M
│     │  ├─ zig build-exe nasm ReleaseFast native (+2 more reused dependencies)
│     │  └─ configure nasm header to config.asm (reused)
│     ├─ run nasm (input.o) success 1s MaxRSS:36M
│     │  ├─ zig build-exe nasm ReleaseFast native (+2 more reused dependencies)
```



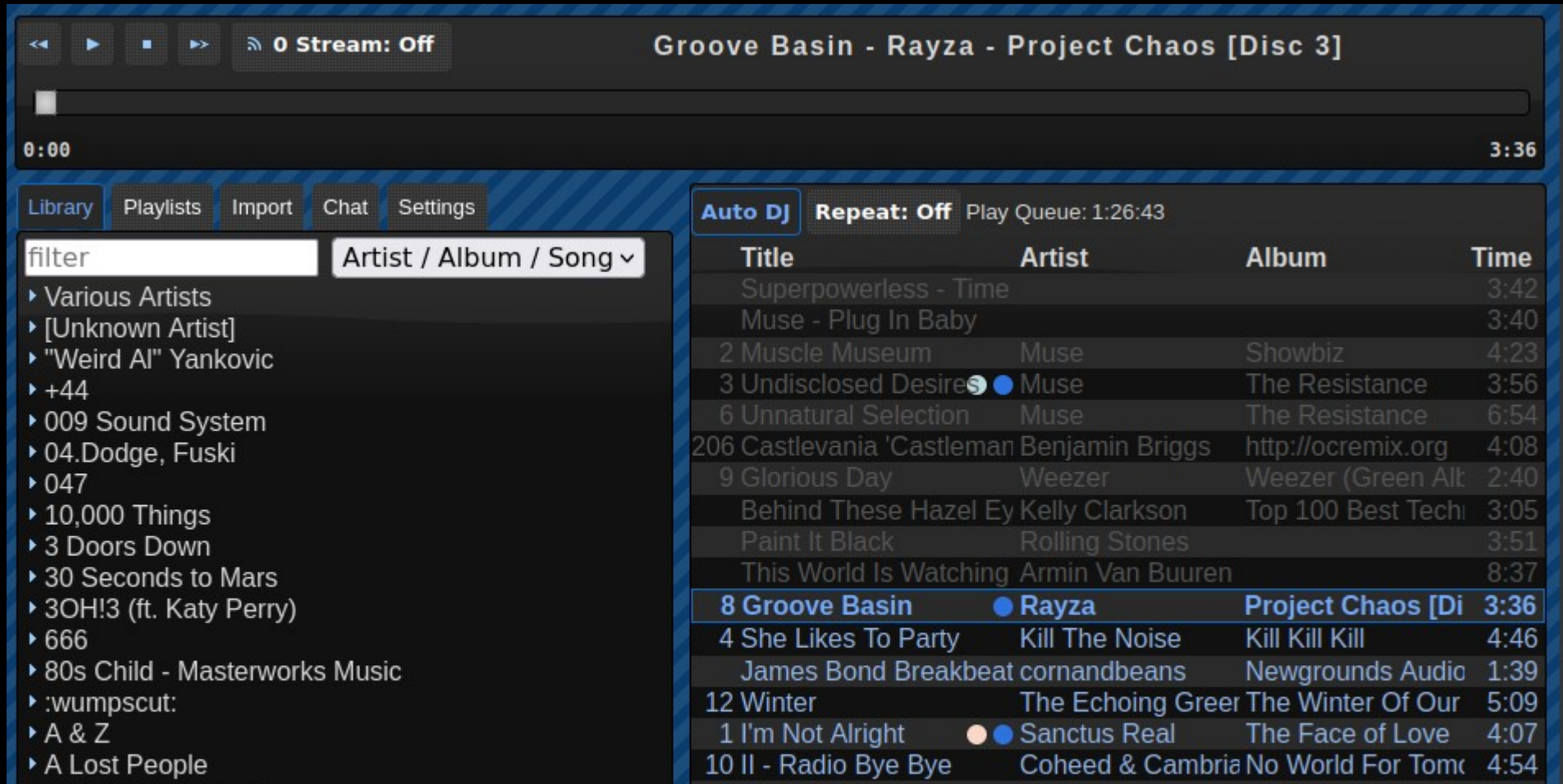

```
.{
    .name = "libffmpeg",
    .version = "5.1.2",

    .dependencies = .{
        .libz = .{
            .url = "https://github.com/andrewrk/libz/archive/8bf2a1c61b87773131bee2086737b5f02b7c0888.tar.gz",
            .hash = "12209e851f7e2c6ba2f01de3e11b1771f03e49666065320abd8414aac152bfa75fae",
        },

        .libmp3lame = .{
            .url = "https://github.com/andrewrk/libmp3lame/archive/b8408fa9dc751baee2a03cd430a996920ca2c5bc.tar.gz",
            .hash = "122025fb57739eb67edbbafed2b270479089ee7395cf4b0849001ab95c16a2bae0d9",
        },
        .libvorbis = .{
            .url = "https://github.com/andrewrk/libvorbis/archive/a1dc13ff76e262007eea91894dd16d74c2147619.tar.gz",
            .hash = "1220d6c5bf73bee37589a086147ab8f7e855ee98becd3f2e26e58a920ca8f766105f",
        },
        .libogg = .{
            .url = "https://github.com/andrewrk/libogg/archive/9f514ae46e28589e47d25017385eb0d6ff4c7e9a.tar.gz",
            .hash = "1220f96a4eaae5bad95ab9391431f125b7cc32edbd6d17397ce066d498f8fc9b63c2",
        },

        // This is used to compile some assembly files into object files for x86.
        // Without this, ffmpeg considers the build "crippled".
        .nasm = .{
            .url = "https://github.com/andrewrk/nasm/archive/5ef3ed029e4917d07c88f265c7b12fcfd81bd6ab.tar.gz",
            .hash = "122032bc8d97d857b7c2f71252da293e4f293a4ea0d162909fb0705ba17c40ae2a87",
        },
    },
}
```




```zig
const std = @import("std");

pub fn build(b: *std.build.Builder) void {
    const target = b.standardTargetOptions(.{});
    const optimize = b.standardOptimizeOption(.{});

    const libz_dep = b.dependency("libz", .{
        .target = target,
        .optimize = optimize,
    });
    const libmp3lame_dep = b.dependency("libmp3lame", .{
        .target = target,
        .optimize = optimize,
    });
    const libvorbis_dep = b.dependency("libvorbis", .{
        .target = target,
        .optimize = optimize,
    });
    const libogg_dep = b.dependency("libogg", .{
        .target = target,
        .optimize = optimize,
    });

    const lib = b.addStaticLibrary(.{
        .name = "ffmpeg",
        .target = target,
        .optimize = optimize,
    });
    lib.linkLibrary(libz_dep.artifact("z"));
    lib.linkLibrary(libmp3lame_dep.artifact("mp3lame"));
    lib.linkLibrary(libvorbis_dep.artifact("vorbis"));
    lib.linkLibrary(libogg_dep.artifact("ogg"));
    lib.linkLibC();
    lib.addIncludePath(".");
```

So what?

So what?


0 Stream: Off
Groove Basin - Rayza - Project Chaos [Disc 3]
0:00
3:36
Library
Playlists
Import
Chat
Settings
filter
Artist / Album / Song
Various Artists
[Unknown Artist]
"Weird Al" Yankovic
+44
009 Sound System
04.Dodge, Fuski
047
10,000 Things
3 Doors Down
30 Seconds to Mars
3OH!3 (ft. Katy Perry)
666
80s Child - Masterworks Music
:wumpscut:
A & Z
A Lost People
Auto DJ
Repeat: Off
Play Queue: 1:26:43
Title
Artist
Album
Time
Superpowerless - Time 3:42
Muse - Plug In Baby 3:40
2 Muscle Museum Muse Showbiz 4:23
3 Undisclosed Desires Muse The Resistance 3:56
6 Unnatural Selection Muse The Resistance 6:54
206 Castlevania 'Castleman Benjamin Briggs http://ocremix.org 4:08
9 Glorious Day Weezer Weezer (Green Alb 2:40
Behind These Hazel Ey Kelly Clarkson Top 100 Best Tech 3:05
Paint It Black Rolling Stones 3:51
This World Is Watching Armin Van Buuren 8:37
8 Groove Basin Rayza Project Chaos [Di 3:36
4 She Likes To Party Kill The Noise Kill Kill Kill 4:46
James Bond Breakbeat cornandbeans Newgrounds Audio 1:39
12 Winter The Echoing Green The Winter Of Our 5:09
1 I'm Not Alright Sanctus Real The Face of Love 4:07
10 II - Radio Bye Bye Coheed & Cambria No World For Tomo 4:54

So what?

Installation

This project is being actively developed, so the installation instructions are the same as the development instructions.

First, download zig master branch. Then you can use this command to build and run the project:

```
zig build run
```

Have a look at `zig build --help` for more options.

Build System

Available in the upcoming Zig 0.11.0 release

How to Build Software from Source

Case Studies

Build Systems

How to Build Software from Source

How to Build Software from Source

let’s recap

How to Build Software from Source

- What’s the best way to obtain the software? Maybe getting it from your system or from upstream would be better.

How to Build Software from Source

- Installing from source is easy!
  - pay attention to the configure step
  - manage your install prefixes carefully

How to Build Software from Source

- Installing from source is easy!
  - pay attention to the configure step
  - manage your install prefixes carefully
  - please, for the love of god, don’t use root

How to Build Software from Source

- Installing from source is easy!
  - pay attention to the configure step
  - manage your install prefixes carefully
  - please, for the love of god, don’t use root
- **Zig build system is cool, give it a try**

Zig Software Foundation

- We’re a 501(c)(3) non-profit
- Consider using your company-sponsored donation matching to keep us funded!
- ziglang.org/zsf

thanks for listening, everybody!

goto;

Don't forget to
vote for this session
in the **GOTO Guide app**