LIBHEIF INSTALL

I wanted this tool for HEIC image conversions on my own system. Once I got started trying to figure it all out, I found a whole world out there doing cool stuff with it. I also found that my Rocky 8 load was gunna give me some challenges. Here is a install process that I hope overcomes those challenges, for you.

NEGATIVO17: https://negativo17.org

REPO: https://negativo17.org/repos

```
# ----- FIND A NICE PLACE TO WORK -----


cd /opt


# ----- ADD NEGATIVO17 REPO -----


dnf config-manager --add-repo=https://negativo17.org/repos/epel-multimedia.repo
dnf -y update


# ----- INSTALL LIBRARIES AND TOOLS -----


dnf -y install \
binutils-x86_64-linux-gnu brotli brotli-devel cmake cmake-data cmake-doc cmake-filesystem cmake-rpm-macros \
doxygen ffmpeg ffmpeg-devel ffmpeg-libs fontconfig-devel freeglut-devel freetype-devel fribidi-devel \
gcc-c++-x86_64-linux-gnu gcc-gfortran gcc-go gcc-objc gcc-objc++ gcc-x86_64-linux-gnu gdk-pixbuf2-devel \
giflib giflib-devel git git-core git-core-doc glibc-static gmp-c++ graphviz graphviz-devel harfbuzz-devel \
ilmbase-devel iso-codes jbigkit-devel kvazaar kvazaar-devel kvazaar-libs libaom-devel libavcodec-devel \
libavfilter-devel libavformat-devel libavif libavutil-devel libdav1d-devel libgs-devel libjpeg libjpeg-devel \
libjpeg-turbo-utils libjxl-devel libopenjph-devel libpng libpng-devel librsvg2-devel libswscale \
libswscale-devel libtiff libtiff-devel libwebp-devel libwmf libwmf-devel libwmf-lite libXext libXext-devel \
meson nasm ninja-build OpenEXR-devel openh264-devel openh264-libs pango-devel perl-devel pixman-devel \
qt5-qtbase-gui qtsinglecoreapplication-qt5 qtsinglecoreapplication-qt5-devel SDL2 SDL2-devel svt-vp9-devel \
turbojpeg turbojpeg-devel vvdec vvdec-devel vvdec-libs vvenc vvenc-devel vvenc-libs x264-devel x264-libs \
x265-devel x265-libs yasm


# ----- INSTALL CARGO/RUSTUP -----
```

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs -o rustup.sh
chmod +x rustup.sh
./rustup.sh --no-modify-path -y -q
rm -f rustup.sh


# ----- PRE-CREATE, CREATE, AND EXPORT MISSING PATHS -----


cat > /etc/ld.so.conf.d/local-lib.conf << "EOF"
/usr/local/lib
/usr/local/lib64
EOF


mkdir -p {/usr/local/lib64/pkgconfig,/usr/local/share/pkgconfig}
cat >> /root/.bashrc << "EOF"


PKGPATH=$(pkg-config --variable pc_path pkg-config);
PKG_CONFIG_PATH="$PKGPATH:/usr/local/lib64/pkgconfig:/usr/local/share/pkgconfig"
export PKG_CONFIG_PATH


PREPATH=$(echo $PATH);
CARPATH="$HOME/.cargo/bin"
PATH="$PREPATH:$CARPATH"
export PATH


EOF


# ----- CHECK YOUR BACHRC AND MAKE SURE THE NEW PATHS ARE GTG -----


cat /root/.bashrc


# ----- IF ALL IS GOOD, SOURCE AND LOAD -----


source /root/.bashrc
ldconfig
rustup update
```

The environment should be pretty well setup now. This next part is the meat and potatoes.   I am going to keep these builds around for awhile so I created scripts for each one, and will locate them inside a `codecs` directory. This is a bit out of the norm if you're reading the "libheif" documentation as well. But you're here, so here it goes...

```
# ----- CREATE CODECS DIRECTORY -----

mkdir -p /opt/codecs && cd /opt/codecs

# ----- CLEAN INSTALL OF GCC 11 COMPILER, CLANG, AND ALL ITS TOOLS -----

dnf -y remove clang* gcc-toolset*
dnf -y install gcc-toolset-11* clang

# ----- GENERATE THE SCRIPTS -----

cat > libde265.sh << "EOF"
#!/usr/bin/env bash

git clone https://github.com/strukturag/libde265.git

cd libde265
./autogen.sh && ./configure --enable-encoder
make
make install
ldconfig

EOF

cat > openjpg2.sh << "EOF"
#!/usr/bin/env bash

git clone -b v2.5.3 --depth 1 https://github.com/uclouvain/openjpeg.git

mkdir -p openjpeg/build
cd openjpeg/build

cmake .. -DCMAKE_BUILD_TYPE=Release -DBUILD_THIRDPARTY:BOOL=ON -DBUILD_CODEC:bool=on
make
make install
ldconfig

EOF

cat > aom.sh << "EOF"
```

```
#!/usr/bin/env bash
git clone -b v3.12.0 --depth 1 https://aomedia.googlesource.com/aom

cd aom
cmake -S . -B build.libavif -G Ninja \
 -DCMAKE_INSTALL_PREFIX="$(pwd)/dist" \
 -DCMAKE_BUILD_TYPE=Release \
 -DENABLE_DOCS=0 \
 -DENABLE_EXAMPLES=0 \
 -DENABLE_TESTDATA=0 \
 -DENABLE_TESTS=0 \
 -DENABLE_TOOLS=0 \
 -DCMAKE_POSITION_INDEPENDENT_CODE=ON

ninja -C build.libavif
ninja -C build.libavif install

cd ..

A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/aom/dist/lib64/pkgconfig"
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

cat > dav1d.sh << "EOF"
#!/usr/bin/env bash
git clone -b 1.5.0 --depth 1 https://code.videolan.org/videolan/dav1d.git

cd dav1d
meson build --default-library=static --buildtype release --prefix "$(pwd)/dist" $@
ninja -C build
ninja -C build install

cd ..

A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/dav1d/dist/lib64/pkgconfig"
```

```bash
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

cat > libwebp.sh << "EOF"
#!/usr/bin/env bash
git clone --single-branch https://chromium.googlesource.com/webm/libwebp

mkdir -p ./libwebp/build
cd ./libwebp/build

cmake -G Ninja -DCMAKE_INSTALL_PREFIX="$(pwd)/dist" -DBUILD_SHARED_LIBS=OFF -DCMAKE_BUILD_TYPE=Release ..
ninja sharpyuv
ninja install

cd ../..

A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/libwebp/build/dist/lib64/pkgconfig"
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

cat > rav1e.sh << "EOF"
#!/usr/bin/env bash
git clone -b v0.7.1 --depth 1 https://github.com/xiph/rav1e.git

cd rav1e

cargo update
cargo install cargo-c
cargo cinstall --crt-static --release --prefix="$(pwd)/dist" --library-type=staticlib

cd ..
```

```
A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/rav1e/dist/lib64/pkgconfig"
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

cat > svt.sh << "EOF"
#!/usr/bin/env bash
git clone -b v3.0.0 --depth 1 https://gitlab.com/AOMediaCodec/SVT-AV1.git

cd SVT-AV1/Build/linux

./build.sh release static no-apps disable-lto prefix=$(pwd)/install install

cd ../../..

A='#!/usr/bin/env bash'
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/SVT-AV1/Build/linux/install/lib64/pkgconfig"
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

cat > ultrahdr.sh << "EOF"
#!/usr/bin/env bash
git clone -b v1.4.0 --depth 1 https://github.com/google/libultrahdr.git ultrahdr

mkdir -p ./ultrahdr/build && cd ./ultrahdr/build

cmake -G Ninja -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX="$(pwd)/dist" -
DUHDR_BUILD_EXAMPLES=ON -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ ..
ninja
ninja install

cd ../..

A='#!/usr/bin/env bash'
```

```
B="PKG_CONFIG_PATH=\$PKG_CONFIG_PATH:$PWD/ultrahdr/build/dist/lib64/pkgconfig"
C='PKGCONFPATH'

sed -i "s%$A%$A\n$B%" $C

EOF

# ----- QUICK SCRIPT TO RUN ALL THE SCRIPTS -----

cat > allofum.sh << "EOF"
#!/usr/bin/env bash

# ----- RUN THESE TWO SCRIPTS FIRST SO THEY ARE INSTALLED TO THE SYSTEM -----
cd /opt/codecs && source libde265.sh
cd /opt/codecs && source openjpg2.sh

# ----- NOW RUN THROUGH THE REST -----
cd /opt/codecs && source aom.sh
cd /opt/codecs && source dav1d.sh
cd /opt/codecs && source libwebp.sh
cd /opt/codecs && source rav1e.sh
cd /opt/codecs && source svt.sh
cd /opt/codecs && source ultrahdr.sh

exit

EOF

chmod +x *.sh

# ----- PRE-CREATE A "PKGCONFPATH" FILE -----

cat > PKGCONFPATH << "EOF"
#!/usr/bin/env bash

export PKG_CONFIG_PATH
EOF

# ----- RUN ALL OF UM -----
```

```
cd /opt/codecs && source allofum.sh
```

At this point all dependencies are in place. Moving on to "libheif"! This will require jumping into a newer GCC toolset as Rocky 8 rolls with GCC 8.5. We will use version 11 which we loaded earlier.

```
# ----- GRAB LIBHEIF -----


cd /opt
git clone https://github.com/strukturag/libheif.git
cd libheif


# ----- SWITCH TO THE GCC 11 ENVIRONMENT -----


scl enable gcc-toolset-11 bash        # HAD CASES WHERE THIS COMMAND WOULD NOT SWITCH OVER, SOOO...
source /opt/rh/gcc-toolset-11/enable   # THROWING THIS ONE IN THE MIX AS WELL.


# ----- MAKE SURE YOUR GCC IS VERSION 11 -----


gcc --version


# ----- SOURCE AND LOAD OUR PATHS -----


source ~/.bashrc
source /opt/codecs/PKGCONFPATH
ldconfig


# ----- YOU CAN MIX AND MATCH OPTIONS - HERE ARE A COUPLE THAT I BUILT SUCCESSFULLY -----


## OPTION 1 ## ----- BUILD LIBHEIF RELEASE (SHARED) -----

  mkdir build && cd build
  cmake .. --preset=release \
   -DWITH_UVG266=OFF \
   -DENABLE_MULTITHREADING_SUPPORT=ON \
   -DENABLE_PARALLEL_TILE_DECODING=ON
  doxygen -u
  make -j$(nproc)


## OPTION 2 ## ----- BUILD LIBHEIF FUZZING (STATIC) -----
```

```
mkdir build && cd build
cmake .. --preset=fuzzing \
  -DWITH_UVG266=OFF \
  -DENABLE_MULTITHREADING_SUPPORT=ON \
  -DENABLE_PARALLEL_TILE_DECODING=ON
doxygen -u
make -j$(nproc)


# ----- I WILL BE USING THE "RELEASE" OPTION IN SOME OTHER WORK, SO I WILL INSTALL THAT ONE -----


make install


# ----- EXIT GCC TOOLKIT -----


exit
```

You should now have a monster libheif built on your system. It's a powerful app. Some of the tools to use it can be found in *usr/local/bin* (such as `heif-enc` and `heif-dec`).

I hope this helps! ;)