if you start the application from a terminal window command line where you will have more control to catch problems. All that said, the system builds can be useful in a university lab setting where there are possibly multiple users, or multiple versions.

There are two notable differences between *standard* views of CINELERRA-GG and this implementation for the system builds. Both of these can be configured during installation. The differences make it possible to have several different versions installed without having them *walk* on each other.

1. application name can be set during installation and defaults to: `cin`
2. the home configuration directory can also be set and defaults to: `$HOME/.bcast5`

To do a system build, you should read the file `README` that is at the top level after you get the source.

- You need about 6.0 GB of disk storage to operate a build and you need to have *git* installed.
- Obviously in order to install into the system, you must run as **root**.
- The *git:* step has to download many files (approx 130 MB) so allow time. When decompressed this will expand to about 530 MB.
- Run the following commands (this takes awhile):

```
# This is where you need the 6.0GB of disk space:
cd /<build_path>/
git clone --depth 1 "git://git.cinelerra-gg.org/goodguy/
    ↪ cinelerra.git" cinelerra5
# Change to the cloned directory:
cd cinelerra5/cinelerra-5.1
```

NOTE: if your system has never had CINELERRA-GG Infinity installed, you will have to make sure you have all of the compilers and libraries necessary. So on the very first build you should run:

```
./blds/bld_prepare.sh <os> # where <os> represents the
                           # Operating System of centos,
                           # fedora, suse, ubuntu, mint, debian.
./autogen.sh
```

```
./configure --prefix=/usr   # optional parameters can be added here
make 2>&1 | tee log         # make and log the build
```

`bld_prepare.sh` does not work for Arch Linux or Gentoo, so we have to install the dependencies manually. `README.arch` or `README.gentoo`, which contain the list of dependencies, can be found at:

https://cinelerra-gg.org/download/README.arch

https://cinelerra-gg.org/download/README.gentoo

- Check for obvious build errors:

```
grep "\*\*\*.*error" -ai log
```

If this reports errors and you need assistance or you think improvements can be made to the builds, email the log which is listed below to: cin@lists.cinelerra-gg.org

```
/<build_path>/cinelerra5/cinelerra-5.1/log
```

- If there are no build errors, finally just run:

```
make install
```

- If it all worked, you are all setup. Just click on the CINELERRA-GG desktop icon.

To do a single-user build, read the file `README` that is at the top level after you get the source.

1. You need at least 6 GB of disk storage to operate a build + you need to have `git` installed.
2. Recommend you build and run as **root**, just to avoid permission issues initially.
3. The *git* step has to download many files (approx 130 MB) so allow time.
4. Run the following commands (this takes awhile):

```
# This is where you need the 6GB of disk space
cd /<build_path>/
git clone --depth 1 "git://git.cinelerra-gg.org/goodguy/
   ↪ cinelerra.git" cinelerra5
```

```
# Toplevel directory:
cd cinelerra5/cinelerra-5.1
```

NOTE: if your system has never had CINELERRA-GG Infinity installed, you will have to make sure all the compilers and libraries necessary are installed. So on the very first build you should run as **root**:

```
./blds/bld_prepare.sh <os>
./autogen.sh
./configure --with-single-user
make 2>&1 | tee log
make install
```

Where <os> represents the Operating System supported by CINELERRA-GG, such as centos, fedora, suse, ubuntu, mint, debian. The "with-single-user" parameter makes it so. Check for errors before proceeding.

Then just start the application by keying in: `./cin` in the bin subdirectory OR add a desktop icon by using the appropriate directory to copy the files to, run as **root**, and edit to correct the directory path. Below are generic directions of how to do this.

```
cd /cinelerra_directory_path
cp -a image/cin.{svg,xpm} /usr/share/pixmaps/.
cp -a image/cin.desktop /usr/share/applications/cin.desktop
```

After you have followed the above, in the cin.desktop file, change the `Exec=cin` line to be `Exec=<your_directory_path>/bin/cin`.

The preceding directions for doing a single-user build have been meticulously followed to build and run on a newly installed ubuntu 15 system WITHOUT BEING ROOT except for the `bld_prepare.sh` and creating the desktop icon.

1.1.1 Notable Options and Caveats

These procedures and the CINELERRA-GG Infinity software have all been run as **root** on various home laptops and desktops. This provides the best chance to ensure all works correctly and also allows for handling errors, other problems and potential crashes with the most success. Included in this section are some of the build variations easily available for normal builds.

To see the full list of features use:

```
./configure --help
```

The default build is a system build which uses:

```
./configure --without-single-user
```

In the single-user build, the target directory is always `cin`. Because this is also the developer build, constant names are used throughout. However, you can rename files after the install is complete.

If your operating system has issues with the default install to `/usr/local`, you might have to change the location to `/usr` for a system build. Then you will have to use:

```
./configure --prefix=/usr
```

If you wish to change the default directory for a system build you will have to add the destination directory path on the `make install` line. For example:

```
make install DESTDIR=<your selected target directory path>
```

The application name can be set during installation, but defaults to `cin` so that the GG/Infinity build can coexist with other CINELERRA-GG builds if necessary. To override the default `cin` name, use:

```
./configure --with-exec-name=cinelerra
```

The home configuration directory can also be set, but default location is traditionally `$HOME/.bcast5`. For example:

```
./configure -with-config-dir=/myusername/.bcast5
```

NOTE: when you specify parameters to the configure program, it will create a `make` file as a consequence. Since in a `make` file, the $ is a special character, it must be escaped so in order to represent a $ as part of an input parameter, it has to be stuttered. That is, you will need $$ (2 dollar signs) to represent a single dollar sign.

It may be necessary on some distros which have missing or incomplete up-to-date libraries, to build CINELERRA-GG without Ladspa. To do so, use:

```
./configure --prefix=/usr --without-ladspa-build
```

Note that the with-ladspa-dir is the ladspa search path, and exists even if the ladspa build is not selected. This gives you the ability to specify an alternate ladspa system path by utilizing the `LADSPA_PATH` environment variable (that is, the default ladspa build is deselected).

Note for 32-bit 14.2 Slackware, Debian, Gentoo, Arch, FreeBSD, before running the configure, you will need to set up the following:

```
export ac_cv_header_xmmintrin_h=no
export FFMPEG_EXTRA_CFG=" --disable-vdpau"
```

1.1.2 Notes about Building from Git in your Customized Environment

Getting a build to work in a custom environment is not easy. If you have already installed libraries which are normally in the thirdparty build, getting them to be recognized means you have to install the *devel* version so the header files which match the library interfaces exist. Below is the list of thirdparty builds, but this list may have changed over time.

The *yes* means force build and *auto* means probe and use the system version if the build operation is not static. To get your customized build to work, you need to change the probe options for the conflicting libraries from *yes* to *auto*, or even rework the `configure.ac` script. There may be several libraries which need special treatment.

An example of a problem you might encounter with your customized installation is with `a52dec` which has probes line `(CHECK_LIB/CHECK_HEADERS)` in `configure.ac`, but `djbfft` does not. In this case, `djbfft` is only built because `a52dec` is built, so if your system has `a52dec`, set `a52dec` to auto and see if that problem is solved by retrying the build with:

```
./confgure --with-single-user -enable-a52dec=auto .
```

With persistence, you can get results, but it may take several tries to stabilize the build. If you need help, email the `log` and `config.log`, which is usually sufficient to determine why a build failed.