Building DROPS HOWTO

Frank Mehnert Jork Löser Ronald Aigner
l4-hackers@os.inf.tu-dresden.de

March 2006

This document is also available in HTML format.

1 Quickstart

Although we recommend reading the few pages of this paper, here is the abstract:

1. Get the following components from `http://tudos.org/drops/download.html`: L4Env, L^{4}Linux, Fiasco.

2. Unpack the tar-balls in one directory. Memorize the directory.

3. Switch into `l4/` and enter

   `DROPS_STDDIR=<dir>/drops \`
   `make oldconfig`

   Substitute `<dir>` with the absolute pathname of the directory you kept in mind. Still in `l4/`, enter

   `make`

2 Involved Components

DROPS is a collection of packages using services of internal or external libraries and tools.

A DROPS package typically consists of a server, one or more client libraries and associated interface headers. Often there also exist one or more examples illustrating the usage of the package.

For C library support, DROPS utilizes one of two external C libraries dietlibc or uClibc. The two C libraries are included in the L4Env as packages. Some packages require the older OSKit libc. These packages are not actively maintained.

Some DROPS packages require L^{4}Linux, the L4 port of Linux. L^{4}Linux is available as another external module.

The internal tools include the BID makefile macros, the IDL compiler dice, a dependency generation tool, a code transformation tool, a configuration tool and others.

To build a DROPS package, all interface headers and libraries the packet depends on must be available, as well as the C library and the tools mentioned. For examples, the name server has the following dependencies:

- the resource manager `roottask` (part of DROPS)
- the logging server `log` (part of DROPS)
- to build it requires the packages `l4util`, `l4sys`, `roottask`, `sigma0`, `log`, `events`, and `crtx` (all part of DROPS)
- a C library (either dietlibc or uClibc)
- the BID make macros, and the tools `config`, `gendep`, `dice`, and the `bin` directory from `l4/tool`.

The building tools of DROPS allow for two locations of the required packets (rmgr, log...), C libraries and tools, depending on how you installed DROPS:

1. All packets and tools are available in source code. Their directories are specified relative to the `L4DIR`. Please see table 1 for the locations of the various components.

By executing `make` in `L4DIR/pkg`, all libraries are symlinked into subdirectories of `L4DIR/lib` and all interface headers and IDL files are symlinked into subdirectories at `L4DIR/include`. We also refer to `L4DIR` as *L4 directory*. All binary files are symlinked into subdirectories at `L4DIR/bin`.

2. The packets and tools are already compiled and installed at a central place at `DROPS_STDDIR`. Obviously, only the headers and IDL files of the packets are available in source code. We also refer to `DROPS_STDDIR` as *DROPS directory*.

component	default location
DROPS packets	`L4DIR/pkg`
DROPS Tools	`L4DIR/tool`
L^4Linux v2.2	`L4DIR/../linux22`
L^4Linux v2.6	`L4DIR/../l4linux-2.6`

Table 1: Locations of source code relative to `L4DIR`

The first method allows to inspect and to change the source code, e.g. for development or for debugging. The second method allows to use pre-compiled libraries from a central place without the need to download and compile the whole DROPS source tree. For instance, the students at the Dresden OS group can easily access the *DROPS directory* at `/home/drops` which is rebuild from the newest CVS version each night.

You can mix both methods of installation, i.e. some packets can be installed into the *DROPS directory* and some other packets can be compiled from sources within the *L4 directory*. If you have both versions of one packet, the one in `L4DIR/pkg` will be used when compiling other packets.

In the following section we show how to build a DROPS tree.

3 How to build DROPS

3.1 Downloading and Unpacking

The source code of the DROPS packages and the tools can be obtained from `http://tudos.org/drops/download.html` as tar-balls or via remote-CVS – see download instructions at this web page. Put all the tar-balls into one directory and unpack them.

If you are at the Dresden OS group, you can use the CVS repository at `/home/cvs`. You can also use the installed components at `/home/drops`. Check out the components you want to compile on your own. Additionally, you need the BID macros at `L4DIR/mk`, the Makefile at `L4DIR`. If you want to compile DROPS packages, you should also check out the Makefile at `L4DIR/pkg`. Before building a package you need to config your environment and target platform (see Section 3.3). For this, you need the `config` and `gendep` tools from the `L4DIR/tool` directory. Most packages use IDL to specify their interfaces. To compile the IDLs you need our IDL compiler `dice` also in the `L4DIR/tool` directory. We set up some CVS modules, please see the aforementioned download page for details.

3.2 OS requirements

The following external compilers and tools which are not part of the DROPS distribution are required:

tool	supported versions
GNU C	3.x
GNU make	$\geq$ 3.77
Binutils	$\geq$ 2.13.90
GNU bash	$\geq$ 2.05
GNU find	
awk	posix compatible
GNU sed	
GNU perl	$\geq$ 5.6.1
flex	
byacc / bison	
Doxygen (recommended)	$\geq$ 1.2.15
autoconf	$\geq$ 2.59
automake	$\geq$ 1.4
ncurses-dev(el)	
python-curses	

Table 2: Required build tools

We have tested DROPS with various versions of the GNU tools, but cannot give a guarantee that it works with the versions installed on your system. If you encounter a problem and cannot use the versions mentioned in table 2, contact us and we will try to solve the issue.

3.3 Configuring

Prior to compilation, you must configure the DROPS environment. Go into your `L4DIR` directory. If you uncompressed the tar-balls, this should be `l4/`. A

```
l4/> make config
```

starts an interactive configuration tool. You should at least adapt the `DROPS_STDDIR` setting at "Paths and Directories", as this is the directory where the components will be installed on a `make install`.

The default values meet most peoples requirements. Thus, instead of doing the interactive configuration, you can also configure non-interactively. Set the environment variable `DROPS_STDDIR` and use the make-target *oldconfig*:

```
l4/> DROPS_STDDIR=yourdir make oldconfig
```

If you are at Dresden OS group, a "`make oldconfig`" should do it.

3.4 Compilation

3.4.1 The Easy way

After configuring, go to the `L4DIR` directory and do a

```
l4/> make
```

This builds everything available as sources: the tools, C libraries, Fiasco micro-kernel and the DROPS packages. It also builds the documentation.

To install the things into `DROPS_STDDIR` (and obtain the same setup as we have at Dresden), do a

```
l4/> make install
```

3.4.2 Doing it separately

If you want to restart the build process, e.g. after editing some source files, you do not have to build everything again. Here is what the Makefile in `L4DIR` does.

1. DROPS tools in `L4DIR/tool`

   ```
   cd l4/tool ; make
   ```

2. L^{4}Linux v2.2 (if available)

   ```
   cd linux22 ; make -f Makefile.drops
   ```

3. Fiasco micro-kernel

   ```
   cd l4/kernel ; make
   ```

4. DROPS packages

   ```
   cd l4/pkg ; make
   ```

5. DROPS package and tool documentation

   ```
   cd l4/pkg ; make doc
   cd l4/doc/html ; make doc
   cd l4/tool; make doc
   ```

Generally, you can build the DROPS packages and tools separately, i.e., you can switch into the directory of a specific DROPS package or tool and issue the `make` command there. See Section 3.4.3 for details.

Note, although the makefile in `L4DIR` builds the documentation per default, the called submakefiles only build the documentation when `doc` is explicitly mentioned as a target on the cmdline.

3.4.3 Sub-makefiles and Sub-tree consistency

The DROPS build system allows to bring a subtree in a properly built state by issuing `make` at the top of that subtree. `Make` generally not builds things upwards in the directory hierarchy.

This is true for everything *below* `L4DIR`. An exception is L^{4}Linux, which has its own build system. `L4DIR`, serving as the project root directory, is the other exception, as it integrates the external components.

As a consequence, a `make` issued within a package X cannot help you building other packages X may depend on. Or, if you are within the `server/` directory of a package, `make` does not rebuild the `lib/` directory of that package. In these cases, you have to issue the `make` upwards the directory hierarchy.

As another consequence, you can be sure that `make` does not change your file-system outside the current subtree. Exceptions are the installation directories `L4DIR/bin`, `L4DIR/lib`, `L4DIR/include` and `L4DIR/doc`.

3.4.4 Dependencies

The DROPS build system automatically detects source code dependencies by intercepting the file system calls during compilation. This turned out the be both a robust and general solution.

It is generally not necessary to enforce rebuilding of DROPS source code, e.g. by doing a `make clean` followed by a `make`. A `make` rebuilds the depending files reliably, be it within one DROPS package, across DROPS packages or code of external components such as the C library.